Lattice-based Problems for Post-Quantum Security

Tejas Pujari

November 2023

1 Why Lattice-Based Cryptography

Lattice-based cryptographic schemes base their security on the difficulty of certain problems regarding lattices. Lattice-based schemes offer three major advantages over currently used schemes:

1. They are secure against attacks from quantum adversaries
2. They are just as secure in the worst and average case
3. The only known fully homomorphic encryption schemes are lattice-based

1.1 Average/worst case reductions

For many problems, the difficulty of solving certain instances of the problem is very different from solving the average instance of the problem for example RSA the difficulty of factorizing integer n with 2 large prime factors is way more than that of factoring any random no n that is why it is taken to be product of large primes in RSA.
But, for many classes of lattice problems, it has been shown that the difficulty of solving the average case of the problem is just as difficult as solving the worst case of the problem. This is done by first assuming that an algorithm that efficiently solves the average case of the problem that exists, and then shows that this algorithm can be used as a subroutine in another algorithm to efficiently solve the worst case of that problem. Thus, if a lattice-based scheme is based on a problem that falls into one of these classes, the worst-case hardness of the problem will guarantee the security of the implementation of the scheme, even if these implementations use an average case version of the problem.

2 Lattice

The idea of a lattice is very intuitive. The mental image that you may have of a repeating pattern of points is an accurate depiction of a 2-dimensional lattice.

More formally, a lattice is a discrete additive subgroup of Rn. A basis of a lattice is a set of vectors in the lattice such that any lattice vector can be written as a linear combination of the basis vectors.

2.1 Lattice definition

Given n linearly independent vectors $b_1, b_2, \ldots, b_n \in \mathbb{R}^m$ (column vectors), the lattice generated by them is defined as:

$$\Lambda = \left\{ \sum_{i=1}^{n} a_i b_i \;\middle|\; a_i \in \mathbb{Z} \right\}$$

where a_i are integer coefficients.
If $n = m$, then the lattice is said to be full rank.

Example 1: The lattice generated by the vectors $(1, 0)$ and $(0, 1)$ in $\mathbb{R}^2$ is a full rank lattice, denoted as $\mathbb{Z}^2$.

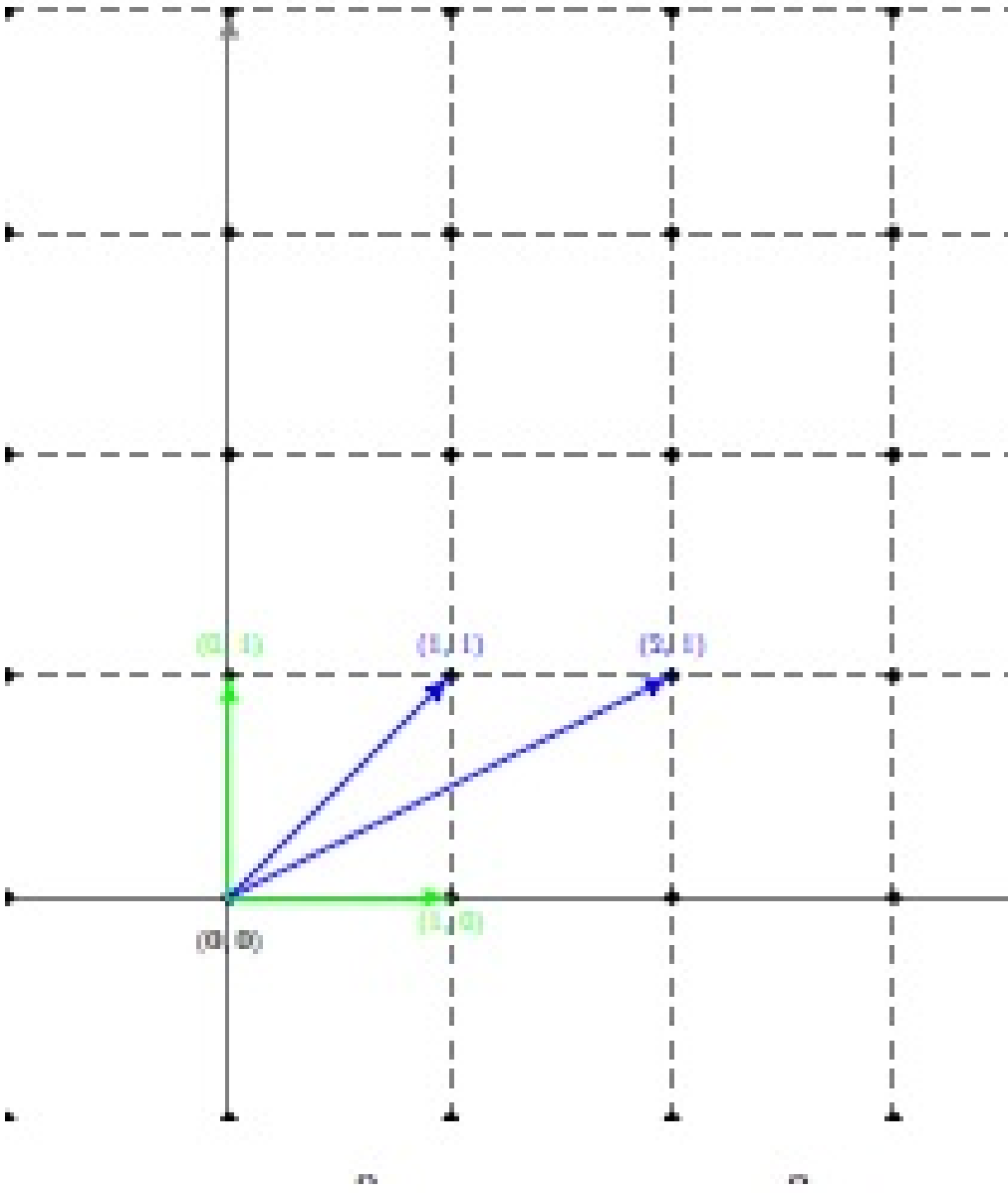
Example 2: Consider the lattice generated by the vectors $(1, 1)$ and $(2, 0)$. Does this generate $\mathbb{Z}^2$?

Proof. No. Since,

$$\alpha(1, 1) + \beta(2, 0) = (0, 1)$$

implies $\alpha = 1$ and $\beta = \frac{-1}{2}$. However, both α and β are not integers. Therefore, the lattice generated by $(1, 1)$ and $(2, 0)$ does not generate the full-rank lattice $\mathbb{Z}^2$. □

Figure 1: An example image



3 Fundamental Region

Definition: A set $F \subset \mathbb{R}^n$ is a fundamental region of a lattice L if its translates $x + F := \{x + y : y \in F\}$, taken over all $x \in L$, form a partition of $\mathbb{R}^n$.

Note: The translate of the lattice L by the set F partitions the space $\mathbb{R}^n$ into disjoint regions.

Example: The interval $[0, 1)$ is a fundamental region of the lattice $\mathbb{Z}$ over $\mathbb{R}$.

The fundamental parallelepiped of a lattice basis B is defined as

$$P(B) = \{B.c : c \in [0, 1)^n\} = \left\{ \sum_{i=1}^{n} c_i b_i : c_i \in [0, 1) \right\}.$$

Notation: If $B = \{b_1, b_2, \ldots, b_n\}$ is a basis, then the lattice generated by B is denoted by $L(B)$.

$$\text{span of a lattice } L(B) = \text{span}(L(B)) = \{B\mathbf{y} : \mathbf{y} \in \mathbb{R}^n\},$$

i.e., the subspace of $\mathbb{R}^m$ spanned by B.

3.1 Lattice problems

Although there are many lattice problems almost all of the ones relevant to cryptography have something to do with the shortest vector problem(SVP)

SVP Problem is to find the shortest vector of a lattice given some arbitrary basis of the lattice. This may at first glance seem simple. However, the fact that we are given an arbitrary basis for the lattice and not a nice one will make this problem difficult. A good basis would be one with short basis vectors that are "as orthogonal as possible". On the other hand, a bad basis may have long vectors that point in almost the same direction. A variant of this problem is called GapSVP(β) Informally, the GapSVP(β)problem is to decide if the shortest vector of a lattice is long or short (relative to β).

4 Learning with errors

4.1 Problem definition

The Learning With Errors problem is parameterized by some positive integer modulus q and some error distribution χ over the ring $\mathbb{Z}_q^n$. Suppose that there is some **secret vector** $\mathbf{s} \in \mathbb{Z}_q^n$ that is sampled uniformly randomly from $\mathbb{Z}_q^n$. Then pick many a_i uniformly randomly from $\mathbb{Z}_q^n$ and $e_i \leftarrow \chi$. For each a_i and e_i, define $b_i = \mathbf{s} \cdot a_i + e_i$. Then make the pairs (a_i, b_i). The LWE(q, χ) problem is to figure out $\mathbf{s}$ given the many pairs.

Given

- $(a_1, b_1 = a_1 \cdot s + e_1)$
- $(a_2, b_2 = a_2 \cdot s + e_2)$
- $(a_3, b_3 = a_3 \cdot s + e_3)$

$$\vdots$$

$(a_n, b_n = a_n \cdot s + e_n)$

Find

$$\mathbf{s} \in \mathbb{Z}_q^n$$

Note : The above problem is known as the **search LWE** problem as we are finding the vector s. There are variants of LWE problems for example **Decision LWE** problem which is to decide given many pairs they are of the form LWE or if they are just totally random.

4.2 Lattice problem to LWE reduction

Not all instances of the search LWE problem are computationally infeasible. For instance, if χ is always 0, then one can simply find s through Gaussian elimination. However, it has been shown that for certain constraints on the parameters, the search LWE problem is quantum and classically "difficult".

In his paper, Regev establishes the computational infeasibility of the search LWE problem by showing that if there exists an algorithm to efficiently solve the search LWE problem, then that algorithm can be used to solve the GapSVP problem. More precisely, he shows a quantum algorithm to solve GapSVP that contains an algorithm that solves search LWE as a subroutine. The complexity of the GapSVP algorithm depends on the complexity of the LWE algorithm that it uses. If the LWE algorithm is efficient, then the GapSVP algorithm will be as well. Thus, if GapSVP is quantum computationally infeasible, then there must not exist an efficient algorithm (quantum or classical) for solving the search LWE problem.

5 LWE based cryptosystem

Let n be the security parameter of the cryptosystem. This means that the higher n is, the more computationally difficult it will be to crack the cryptosystem. We pick q, the integer modulus, to be some prime number between n^2 and $2n^2$. We pick m, the number of LWE samples in the public key, to be $(1+\varepsilon)(n+1)\log q$, where $\varepsilon > 0$ is some arbitrary constant. The error distribution χ is taken to be a discrete Gaussian distribution over $\mathbb{Z}_q^n$ whose standard deviation is $\sigma(n) = \mathcal{O}\left(\frac{1}{\sqrt{n}\cdot\log n}\right)$. All of the operations that follow are performed mod q.

- Private Key: Choose some uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$ for the secret key.
- Public Key: For $i = 1, \ldots, m$, pick a_i uniformly randomly from $\mathbb{Z}_n^q$ and e_i from χ. Then we let $b_i = a_i \cdot \mathbf{s} + e_i$. The public key is the set of LWE samples $\{(a_i, b_i)\}_{i=1}^m$.
- Encryption: Pick a random $S \in \{0,1\}^n$. Then let $(a, b) = \left(\sum_{i\in S} a_i, \sum_{i\in S} b_i\right)$. To encrypt a bit $x \in \{0,1\}$, let the encryption be $(a_0, b_0) = (a, b + \lfloor \frac{q}{2} \rfloor \cdot x)$.
- Decryption: To decrypt a pair (a_0, b_0), consider $b_0 - a_0 \cdot \mathbf{s}$. If this value is closer to 0 than $\lfloor \frac{q}{2} \rfloor$, the decryption is 0, otherwise, it is 1.

Note : The above scheme satisfies the correctness property because the error distribution we chose earlier outputs the error vector$e$ which is close to zero.

5.1 Seurity of the scheme

In order for the scheme to be secure, an adversary should not be able to distinguish encryptions of 0 and encryptions of 1. Regev shows that if there is some polynomial time algorithm that can differentiate between encryptions of 0 and 1, then there is some algorithm that can distinguish between LWE samples and uniformly random samples with a significant chance of success. Since we assume that decision LWE is computationally infeasible, this would imply that there does not exist a polynomial time algorithm that distinguishes between encryptions of 0 and 1.

5.2 Key size

In order for a cryptosystem to be viable in the real world, the key sizes must be. We see that the size of the public key is $\mathcal{O}(m.n.log(q))$. We define $\mathcal{O}'$ to be the complexity ignoring all logarithmic factors. Thus, given the definition of m, the size of the public key is $\mathcal{O}'(n^2)$ This is a fairly large key size, as the security factor will often be in the many hundreds.

Note: Hence the LWE-based scheme described above is not efficient.

6 Ring learning with errors (RLWE)

As we see the above LWE description, however, it does not allow the construction of schemes to be efficient enough for practical purposes. For this reason, an interesting variant of LWE called Ring-LWE has been developed which significantly decreases the amount of storage needed and speeds up the computations.

Informally speaking, vectors of bits in $\mathbb{Z}_q^n$ are seen as coefficients of polynomials in $R_q = \mathbb{Z}_q[x]/(f)$ where f is a polynomial of degree n in $\mathbb{Z}_q[x]$. Hence, one element of the ring R_q can stand for n elements of $\mathbb{Z}_q$, thus reducing public and private keys by a factor of n.

6.1 RLWE based Cryptosystem

A basic public-key scheme based on Ring-LWE works as follows. Let $R = \mathbb{Z}[x]/\left(x^n + 1\right)$ where n is a power 2 and $R_q = R/qR$. Both the secret key s and the error e are chosen according to an error distribution χ, hence $s, e \xleftarrow{\chi} R$. The public key is $(a, b = a \cdot s + e) \in R_q^2$ where $a \xleftarrow{U} R_q$. An n-bit message $z \in \{0,1\}^n$ is then seen as an element of R via the following transformation.

$$(z_1, \ldots, z_n) \mapsto \sum_{i=1}^{n} z_i x^{i-1}$$

Then, three "small" random elements $r, e_1, e_2 \in R$ are chosen according to χ and are used to compute the ciphertext $(u, v) \in R_q^2$ as follows.

$$\begin{aligned} u &= a \cdot r + e_1 \pmod{q} \\ v &= b \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil \cdot z \pmod{q} \end{aligned}$$

The decryption works as follows (the $(\mathrm{mod}\, q)$ is implicit).

$$\begin{aligned} v - u \cdot s &= b \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil \cdot z - a \cdot s \cdot r - e_1 \cdot s \\ &= a \cdot s \cdot r + e \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil \cdot z - a \cdot r \cdot s - e_1 \cdot s \\ &= (e \cdot r + e_2 - e_1 \cdot s) + \left\lfloor \frac{q}{2} \right\rceil \cdot z \\ &= E + \left\lfloor \frac{q}{2} \right\rceil \cdot z \end{aligned}$$

For a suitable choice of the parameters, $E \in R$ is a polynomial whose coefficients have magnitude less than $q/4$ (this is the reason why $r, e_1, e_2 \in R$ were "small"), so that the bits of z can be recovered by rounding each coefficient of $v - u \cdot s$ back to either 0 or $\lfloor q/2 \rceil$.

7 Learning with rounding (LWR)

7.1 Introduction

The learning with rounding (LWR) problem, introduced by Banerjee, Peikert, and Rosen is a variant of learning with errors (LWE), where one replaces random errors with deterministic rounding. The LWR problem was shown to be as hard as LWE for a setting of parameters where the modulus and modulus-to-error ratio are super-polynomial. In particular, a smaller modulus gives us greater efficiency and a smaller modulus-to-error ratio gives us greater security, which now follows from the worst-case hardness of GapSVP with polynomial (rather than super-polynomial) approximation factors.

7.2 Idea

Instead of adding a random small error to various samples $\langle a, s\rangle \in \mathbb{Z}_q$ so as to hide their exact value, we release a deterministically rounded version of $\langle a, s\rangle$. In particular, for some $p < q$, we divide up the elements of $\mathbb{Z}_q$ into p contiguous intervals of roughly $\frac{q}{p}$ elements each and define the rounding function $\lfloor \ \rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p$ that maps $x \in \mathbb{Z}_q$ into the index of the interval that x belongs to. For example, if q and p are both powers of 2, then this could correspond to outputting the $\log(p)$ most significant bits of x. We can extend the rounding function to vectors by applying it componentwise.

7.3 LWR assumption

The learning with rounding assumption says that

$$(A, \lfloor A \cdot s \rfloor_p) \text{is computationally indistinguishable from } (A, \lfloor u \rfloor_p)$$

The work in the paper shows a beautifully simple reduction proving the hardness of the LWR problem under the LWE assumption for some range of parameters. In particular, they observe that if the error size β is sufficiently small and the ratio q/p is sufficiently big, then $\lfloor \langle a, s\rangle \rfloor_p = \lfloor \langle a, s\rangle + e \rfloor_p$

with overwhelming probability over random $a \in \mathbb{Z}_q$ and $e \xleftarrow{\$} \chi$.

In particular, the only way that the two values differ is if $\langle a, s\rangle$ ends up within a distance of $|e|$ from a boundary between two different intervals; but since the intervals are of size q/p and the ball around the boundary is only of size $2|e|$, this is unlikely to happen when q/p is super-polynomially bigger than $2|e|$. Therefore, one can show that:

$$(A, \lfloor A \cdot s \rfloor_p) \approx (A, \lfloor A \cdot s + e \rfloor_p) \approx (A, \lfloor u \rfloor_p)$$

where the first modification is statistically close and the second follows immediately from the hardness of LWE.

Unfortunately, the argument only goes through when (q/p) is bigger than the error size β by a super-polynomial factor. In fact, if we want statistical

distance $2^{-\lambda}$, we would need to set $q \geq 2^{\lambda}\beta p$, where λ is a security parameter. This has three important consequences:

1. the modulus q has to be super-polynomial, which makes all of the computations less efficient

2. the modulus-to-error ratio $\frac{q}{\beta}$ is super-polynomial, which makes the LWE problem easier and only gives us a reduction if we assume the hardness of the lattice problem GapSVP with super-polynomial approximation factors (a stronger assumption)

3. the ratio of the input-to-output modulus q/p is super-polynomial, meaning that we must "throw away" a lot of information when rounding and therefore get fewer bits of output per LWR sample.

Let's define the Learning with error problem formally, it is defined by the following rounding function for integers $q \geq p \geq 2$

$$\lfloor . \rceil_p : \mathbb{Z}_q \longrightarrow \mathbb{Z}_p : x \longrightarrow \lfloor p/q \cdot x \rceil$$

where We naturally identify elements of $\mathbb{Z}_k$ with integers in the interval $(0, 1, ..., k-1)$ Also $\lfloor . \rceil_p$ partitions $\mathbb{Z}_q$ into intervals of length $\frac{q}{p}$ which it maps to the same image.

8 LWR based scheme

Lattice-based Ring-LWE (LWR) can be directly employed in the construction of a deterministic encryption scheme. Such a cryptosystem consistently produces the same ciphertext for a given plaintext and key, even across separate executions of the encryption algorithm. While this property may raise security concerns in certain scenarios, it is a desired feature in applications like searchable databases.

The parameters involved in the LWR scheme are denoted as n, m, q, and p. Initially, a matrix $A \in \mathbb{Z}_q^{m \times n}$, statistically close to uniform, is chosen as the public key. The secret key is represented by a trapdoor function T. The existence of an algorithm denoted as Invert is proved in the original paper. This algorithm returns a secret $s \in \mathbb{Z}_q^n$ given A, T, and the LWE sample $c = As + e \in \mathbb{Z}_q^m$.

The encryption of an n-bit message $s \in \{0,1\}^n$ is expressed as $c = \lfloor As \rfloor_p$. The decryption algorithm takes the ciphertext c and the secret key T as input, working in two steps:

1. Transform c from an LWR sample to an LWE one by applying $\lfloor \frac{q}{p} \cdot c \rfloor \in \mathbb{Z}_q^m$. This step involves the following calculation:

$$\begin{aligned} \lfloor \frac{q}{p} \cdot c \rfloor &= \lfloor \frac{q}{p} \cdot \lfloor As \rfloor_p \rfloor \\ &= \lfloor \frac{q}{p} \cdot \lfloor \frac{p}{q} As \rfloor \rfloor \\ &= \lfloor \frac{q}{p} \cdot \left(\frac{p}{q} As + e' \right) \rfloor \\ &= As + e. \end{aligned}$$

2. Apply the Invert algorithm to A, T, and $\lfloor \frac{q}{p} \cdot c \rfloor$.

9 Learning parity with noise

The search version of the learning parity with noise problem with parameters $\ell \in \mathbb{N}$ (the length of the secret), $\tau \in \mathbb{R}$ where $0 < \tau < 0.5$ (the noise rate) and $q \in \mathbb{N}$ (the numbers of samples) asks to find a fixed random ℓ bit secret $\mathrm{s} \in \mathbb{Z}_2^\ell$ from q samples of the form $\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle \oplus e$ where $\mathbf{a} \in \mathbb{Z}_2^\ell$ is random and $e \in \mathbb{Z}_2$ has Bernoulli distribution with parameter τ (we denote this distribution with Ber_τ), i.e. $\Pr[e = 1] = \tau$. The decisional LPN problem is defined similarly, except that we require that one cannot even distinguish noisy inner products from random ones. The distinction between the search and the decisional version of a problem is often made for problems used in cryptography. Typically, assuming the decisional version of a problem allows for much simpler and more efficient constructions of cryptosystems, whereas the search version is a weaker assumption, and thus constructions based on it require less "faith" in the presumed hardness of the assumption 1 Interestingly, for the LPN problem one can show that the distinction between the search and the decisional version is irrelevant, more on this below.

9.1 The Basic LPN Problem

Definition 1 (search/decisional LPN Problem). For $\tau \in]0, 1/2[, \ell \in \mathbb{N}$, the decisional $\mathrm{LPN}_{\tau,\ell}$ problem is (q, t, ϵ)-hard if for every distinguisher D running in time t

$$|\Pr_{\mathbf{s},\mathbf{A},\mathbf{e}}[\mathrm{D}(\mathbf{A}, \mathbf{A}.\mathbf{s} \oplus \mathbf{e}) = 1] - \Pr_{\mathbf{r},\mathbf{A}}[\mathrm{D}(\mathbf{A}, \mathbf{r}) = 1]| \leq \epsilon$$

Let $s \xleftarrow{\$} \mathbb{Z}_2$, $A \xleftarrow{\$} \mathbb{Z}_2^{q \times l}$, $e \xleftarrow{\$} \mathrm{Ber}_\tau^q$, and $r \xleftarrow{\$} \mathbb{Z}_2^q$. The search $\mathrm{LPN}(\tau, \epsilon)$ problem is (q, t, ε)-hard if for every D running in time t, is (q, t, ϵ)-hard if for every D running in time t

$$\Pr_{\mathbf{s},\mathbf{A},\mathrm{e}}[\mathrm{D}(\mathbf{A}, \mathbf{A}.\mathbf{s} \oplus \mathbf{e}) = \mathbf{s}] \leq \epsilon$$

9.2 Reductions of LPN to other problems

With the current state in complexity theory, we cannot expect to prove that there exists no efficient adversary who breaks the LPN problem, as this would imply $P \neq NP$. The search LPN problem can be stated as the NP-complete problem of decoding random linear codes Think of A as the generator matrix and s as the message. The decoding problem then asks to recover the message s from the noisy codeword $A.s \oplus e$, which is exactly the search LPN. The LPN problem has been extensively studied in learning theory, as an efficient algorithm for LPN would allow to learning of several important concept classes like 2-DNF formulas, juntas, and any function with a sparse Fourier spectrum

LPN based Cryptosystem:

Key Generation:

$$\text{Public Parameters: } n, r, d, t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

Let $C : \mathbb{Z}_2^r \to \mathbb{Z}_2^n$ be an $[n, r, d]$ error-correcting code with correction capacity t. The code C is publicly known.

Generate a secret key matrix M of size $k \times n$.

Encryption:

For a r-bit vector $x \in \mathbb{Z}_2^r$,

Draw $a \xleftarrow{\$} \mathbb{Z}_2^k$ and compute

$y = C(x) \oplus (aM) \oplus \nu, \quad$ where $\nu \xleftarrow{\$}$ Bernoulli(τ).

The ciphertext is (a, y).

Decryption:

For a ciphertext (a, y), the receiver computes

$x = (y \oplus (aM)) \oplus C^{-1}(y \oplus \nu)$,

where $C^{-1}(\cdot)$ is the decoding of $C(\cdot)$.

If $\text{HW}(\nu) \leq t$, output x; otherwise, output $\perp$. The Hamming weight $\text{HW}(\nu)$ is the number of non-zero bits in ν.

References

1. Oded Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography."
   - Available at: `https://link.springer.com/content/pdf/10.1007/0-387-24274-8_32.pdf`
2. Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz, "Property Testing and Learning Parity with Noise."
   - Available at: `https://www.math.ias.edu/m̃osh/publications/parity.pdf`
3. Vadim Lyubashevsky, Chris Peikert, and Oded Regev, "On Ideal Lattices and Learning with Errors Over Rings."
   - Available at: `https://eprint.iacr.org/2010/190.pdf`
4. Stephen Harrigan, "Lattice-Based Cryptography and the Learning with Errors Problem."