Post-Quantum Digital Signature Standardization Project

Elena Kirshanova[1] Nikita Kolesnikov[1] Ekaterina Malygina[1]
Semyon Novoselov[1]

[1]I. Kant Baltic Federal University, Kaliningrad, Russia
{EKirshanova, NiKolesnikov1, EMalygina, SNovoselov}@kantiana.ru

Translated and edited by Maria Schopp

Abstract

In this paper we propose an algebraic lattice-based signature scheme. The design of the proposal follows the Fiat-Shamir paradigm. Our scheme is proved secure in the quantum random oracle model and achieves security against UF-sCMA adversaries. The concrete parameters to instantiate the scheme that achieves 100 bits of security are proposed. Thanks to the algebraic structure of the construction, the scheme is flexible in security levels so that we can achieve trade-offs between speed and security. Our proposal may serve as the basis for a standard of lattice-based schemes.

1 Introduction

Lattice cryptographic primitives are one of the most promising trends in modern cryptography, not only because of the resistance of these primitives to attacks on a quantum computer, but due to the large range of constructions (homomorphic encryption, electronic voting, different types of signatures), as well as their reliability against classical attacks. Cryptographic structures on lattices are not only elegant in theory, but significant in practice, and therefore will be standardized soon enough.[1]

In this article, we propose a digital signature scheme based on algebraic lattices. The proposed design satisfies the following basic properties:

1. schema security is based on "average" tasks, namely LWR (Learning with Rounding) and SIS (Shortest Integer Solution) are classic hard problems on lattices, defined which are given in Section 2),
2. for the efficiency of the scheme, we use the so-called modular version of the tasks, namely, module-LWR, module-SIS [14], which allows not only to reduce the size of the circuit parameters and operation time, but also makes it possible to easily vary the security levels of the scheme,
3. the stability of the circuit is proved in the quantum model QROM (Quantum Random Oracle Model) for A "strong" attacker, namely, for an attack of the UF-sCMA (see Section 2),
4. in the process of generating keys and signatures, instead of normal distribution, we use equal number distribution from the interval, which reduces the risk of third-party attacks,
5. we propose a specific set of parameters for a scheme with a bit estimate of the complexity of attacks on set parameters (see Section 5).

The scheme presented here is based on the Fiat-Shamir paradigm [12, 15] and the ideology continues a series of works on proposing specific signature schemes [5, 6, 11]. The main difference between our scheme and those previously proposed is that the security of keys based on LWR task (not LWE task, Learning with Errors). We believe that our approach simplifies description and potentially speeds up calculations.

[1]One trial version of New Hope key exchange have already been tested in TLS connections for the Google browser Chrome [4]. The post-quantum standardization process is available at `https://csrc.nist.gov/project/post-quantum-cryptography`.

2 Preliminaries

2.1 Symbols

We will denote $\mathbb{Z}/q\mathbb{Z}$ - the ring of integers modulo q, the result $z \mod q$ is represented in the integral in the interval $[0, q-1]$. Further, we denote by R, R_q, and R_p the polynomial rings $\mathbb{Z}[x]/(x^n+1)$, $\mathbb{Z}/q\mathbb{Z}[x]/(x^n+1)$, and $\mathbb{Z}/p\mathbb{Z}[x]/(x^n+1)$, respectively. We will denote vectors in bold lowercase letters (for example, measures, $\mathbf{x}$), matrices in uppercase, for example, $\mathbf{A}$), and constants in ordinary lowercase. We denote the identity matrix as $\mathbb{I}$. Elements of the ring $\mathbb{Z}[x]/(x^n+1)$ will be understood as vectors of polynomial coefficients. Vectors are column vectors by default. The Euclidean (or ℓ_2) norm vector x is defined as $\|\mathbf{x}\| = \|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$, and the ℓ_∞-norm as $\|\mathbf{x}\|_\infty = \mathsf{max}_i |x_i|$.

We associate polynomials from the ring R with vector coefficients of length n, therefore the product of the vectors $\mathbf{x} \cdot \mathbf{y}$ should be understood as the product of the corresponding polynomials. An element $\mathbf{a} \in R_q$ corresponding to the matrix $\mathsf{rot}(\mathbf{a}) \in (\mathbb{Z}/q\mathbb{Z})^{n \times n}$, the i-th row of which are the coefficents of the polynomial $x^{i-1} \cdot \mathbf{a}$. Such a matrix defines the product of any element from R_q by a polynomial $\mathbf{a}$.

For a finite set S, the notation $s \leftarrow S$ means that s is drawn from the uniform distribution on S. For S_β^ℓ, denote the set of vectors of length ℓ, each whose coefficient is taken in accordance with the uniform distribution from the set $[-\beta, \beta]$.

For any $x \in \mathbb{Q}$, writing $\mathsf{Round}(x) \in \mathbb{Z}$ means taking the nearest integer, where $1/2$ is up to 1. For an integer $x \in \mathbb{Z}/q\mathbb{Z}$, in the binary representation of which $\log q$ bits, the function $\mathsf{MSB}(x, d)$ (or $\mathsf{LSB}(x, d)$) means taking the largest (or smallest) bits. Everything extends to operations on vectors of coefficients.

In our scheme, we will use two modules $q = 2^\nu$ and $p = 2^\mu$. Converting an element $x \in \mathbb{Z}/q\mathbb{Z}$ to $x' \in \mathbb{Z}/p\mathbb{Z}$ follows the rule $x' = \mathsf{Round}(x \cdot \frac{p}{q})$. Since our modules are degree 2, the same result can be obtained by adding x to the constant $h = 2^{\nu-\mu-1}$ and taking μ significant bits: $x' = \mathsf{MSB}(x + h, \mu)$. This representation of the Round operation is used, for example, in [10]. A vector, each coordinate of which is equal to h, will be denoted by $\mathbf{h}$.

For any integer $w > 0$, let $B_w = \{x \in R | \|x\|_\infty = 1, x = \sqrt{w}\} \subseteq R$.

2.2 Syntax and security models for digital signatures

Definition 1: A digital signature is a primitive consisting of three algorithms:

- a probabilistic algorithm for generating a key pair, $\mathsf{KeyGen}(\mathsf{par})$, which returns a secret key (sk) and verification key (vk),
- a probabalistic signature generation algorithm $\mathsf{Sign}(\mathsf{sk}, m)$, which for a message $m \in \mathcal{M}$ rotates the signature σ,
- a deterministic algorithm $\mathsf{Verify}(m, \sigma, \mathsf{vk})$, which returns either "Accept" (signature σ is correct for (m, vk), or "Reject" (signature σ is not correct for (m, vk)).

A digital signature is correct with a fraction of error ϵ if for all pairs $(\mathsf{sk}, \mathsf{vk}) \in \mathsf{KeyGen}(\mathsf{par})$ and all messages $m \in \mathcal{M}$, we have $Pr[\mathsf{Verify}(m, \mathsf{Sign}(\mathsf{sk}, m), \mathsf{vk}) = \text{"}Accept\text{"}] \geq 1 - \epsilon$.

To prove the security of the signature scheme, we need three models: the "weak" model UF-NMA (unforgeability against no-message attack), in which the attacker does not have access to the signature (the dying oracle); model UF-CMA (unforgeability against chosen-message attack), where the attacker, having access to the signing oracle, must form a signature for a new message. The "strong" UF model is sCMA (strong unforgeability against chosen-message attack), where there is a well-formed message for which the attacker already knows the signature.

2.3 Complex problems on lattices

The security of our signature relies on two "hard on average" tasks. The first is the task Learning with Rounding (LWR) [7], the deterministic version of the Learning with Errors task (LWE [16]). At the heart of security signature keys lie in the difficulty of this modular version of the problem over the quotient ring R_q

[14]. Everything calculations are performed in the quotient ring R_q, matrix A is formed as a block matrix of $k \cdot \ell$ elements from R_q, where each block is the matrix $\mathsf{rot}(a)$.

For our scheme, in contrast to the classical problems LWR and LWE, where the matrix A is taken as in a certain way from $R_q^{k\times l}$, we will require that at least one of the $k \cdot \ell$ polynomials be invertible in R_q. A random element from R_q for $q = 2\nu$ will be reversible with negligible probability, therefore, one of the blocks of the matrix $\mathbf{A}$ is constructed not from a random polynomial, but using the polynomial on a special kind, the reversibility of which follows from the following fact proved in [17]. We believe that the special form of one block of the matrix $\mathbf{A}$ does not contradict the assumptions difficulties.

Fact 1: Invertibility of an element in a factor ring,[17, Theorem 11.1]
Let $\mathbb{Z}/q\mathbb{Z}$ be a ring. Then $f = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ is a unit in R_q if and only if a_0 is invertible in $\mathbb{Z}/q\mathbb{Z}$, and $a_1, \ldots, a_{n-1}$ are nilpotents in $\mathbb{Z}/q\mathbb{Z}$.

In the ring $\mathbb{Z}/q\mathbb{Z}$, where $q = 2^{\nu}$, even elements other than 0 are nilpotent. Using Fact 1, we can easily construct an invertible polynomial in R_q. Moreover, there are quite a lot of invertible polynomials in R_q of order $2^{\Omega(n)}$.

Definition 2: Learning With Rounding (MLWR) problem
Let $q \geq p \geq 1, k, l \geq 1$ be integers. MLWR distribution for vector $s \leftarrow R_q^{\ell}$ is many pairs of the form $(\mathbf{A}, \mathsf{Round}(\frac{p}{q} \cdot \mathbf{A} \cdot \mathbf{s})))$, where $\tilde{\mathbf{A}} \leftarrow R_q^{k\times l}$.

Search problem: A vector s can be recovered from an arbitrarily large number of samples from the MLWR distribution, $s \leftarrow R_q^{\ell}$.

The problem of distingushing distributions: For a given arbitrarily large number of a set of $\tilde{R}_q^{k\times \ell}$ determine if they are uniformly distributed or MLWR-distributed for the vectors $s \leftarrow R_q^{\ell}$. the probability of success of the algorithm solving the MLWR problem with the parameters introduced above is denoted by $\mathsf{Adv}_{k,\ell,p,q}^{\mathsf{MLWR}}$.

Both versions of the problem are equivalent (that is, having an oracle solving one problem, you can solve the other in polynomial time in n)[8]. In proving the security of the signature scheme, we need a second version.

However, signature security is based on the problem of finding a Short Integer Solutions (Short Integer Solution Problem, SIS)[1]. We need a modular version of this task.

Definition 3: Short Integer Solution (MSIS) problem
Let us fix $b \in \mathbb{N}$ and let $A \leftarrow R_q^{k\times l}$. The modular problem of finding a short integer solution, parameterized by $b > 0$, consists in finding a "short" non-zero preimage $y \leftarrow R_q^{k\times l}$ in the lattice defined by $\mathbf{A}$, i.e.,

$$y \neq 0, \quad [\mathbb{I}|\mathbf{A}] \cdot y = 0, \quad \text{and} \quad \|y\|_\infty \leq b.$$

The probability of success for an algorithm solving the MSIS problem with the parameters introduced above is denoted by $Adv_{k,l,q,b}^{MSIS}$. To prove the safety of our scheme, we need a variant of the SIS problem, so called SelfTargetSIS, as proposed in [13]. The same work describes a reduction from SIS to SelfTargetSIS.

Definition 4: Self Target (SIS) problem
Let $H : \{0,1\}^* \to Bw$ be a cryptographic hash function. Let $\mathbf{A} \leftarrow R_q^{k\times \ell}$, and let the quantum random oracle be denoted $\mathcal{H}(\cdot)$. Then, the SelfTargetSIS task is reduced to walking

$$\mathbf{y} = [\mathbf{r}, \mathbf{c}]^T, \quad \text{where} \quad 0 \leq \|y\|_\infty \leq \gamma, \quad \mathcal{H}([\mathbb{A}|\mathbb{I}] \cdot \mathbf{y}, M) = \mathbf{c},$$

Here, $M \in \{0,1\}^*$ is the original message. Probability of success of the algorithm solving the SelfTarget-SIS problem with the parameters introduced above is denoted as $\mathsf{Adv}_{k,l,q,b}^{\mathsf{SelfTargetSIS}}$.

3 Description of the circuit

The digital signature will depend on the following parameters: $q = 2\nu$, $p = 2\mu$, $\nu > \mu$. We will use the cryptographic hash function $\mathcal{H} : \{0,1\}^* \to B_w$(see [11] for constructing hash functions with range B_w).

For resistance to quantum attacks [9], must are fulfilled $|B_w| \geq 2^{256}$, which is achieved for $w \geq 60$. The parameters k and l are responsible for the dimension of keys. The parameters s, γ define the intervals for the coefficients of the polynomials during the generation keys or signatures, parameters d, β are responsible for the correctness and security of the scheme. Signature generated for messages $M \in \{0,1\}^*$. Specific parameter values are given in Section 5.

Algorithm 3.1 Key generation

```
    Inputs: $k > \ell > 1$, $q > p$, $s$
    Outputs: $\mathbf{A}$, $\mathbf{t}$

1: $\mathbf{A} \leftarrow R_q^{k\times\ell}$
2: $s \leftarrow S_s^{\ell}$
3: $t = \mathsf{Round}(\frac{p}{q} \cdot \mathbf{As})$        $\triangleright \|\mathbf{t} - \mathbf{As}\|_\infty \leq 2^{\nu-\mu}$
4: return $\mathsf{sk} = \mathbf{s}$, $\mathsf{vk} = (\mathbf{A}, \mathbf{t})$
```

Algorithm 3.2 Signature generation

```
    Inputs: $q = 2^\nu$, $p = 2^\mu$, $\ell > 1$, $M$, $\mathbf{A}$, $\mathbf{t}$, $\mathbf{s}$, $d$, $\mathcal{H}$, $\beta$, $\gamma$, $w$
    Outputs: $(\mathbf{z}, \mathbf{c})$

1: $y \leftarrow S_{\gamma-1}^{\ell}$
2: $\mathbf{c} = \mathcal{H}(\mathsf{MSB}(\mathbf{A}\cdot\mathbf{y}, d), M)$
3: $\mathbf{z} = \mathbf{y} + \mathbf{sc}$
4: $\mathbf{w} = \mathbf{Az} - \mathbf{t}\cdot 2^{\nu-\mu}\cdot\mathbf{c}$
5: if $(\|LSB(\pm\mathbf{w}, \nu - d)\|_\infty \geq 2^{\nu-d} - w\cdot 2^{\nu-\mu+1})$ or $(\|\mathbf{z}\|_\infty \geq \gamma - \beta)$ then
6:     restart
7: else
8:     return $(\mathbf{z}, \mathbf{c})$
```

Algorithm 3.3 Signature verification

```
    Inputs: $M$, $\mathbf{z}$, $\mathbf{c}$, $\mathbf{A}$, $\mathbf{t}$, $d$, $\mathcal{H}$, $\beta$, $\gamma$
    Outputs: Accept or Reject

1: $\mathbf{w} = \mathbf{Az} - \mathbf{t}\cdot 2^{\nu-\mu}\cdot\mathbf{c}$
2: $\mathbf{c}' = \mathcal{H}(\mathsf{MSB}(\mathbf{w}, d)), M)$
3: if $\mathbf{c}' == \mathbf{c}$ and $\|\mathbf{z}\|_\infty \leq \gamma - \beta$ then
4:     return "Accept"
5: else
6:     return "Reject"
```

3.1 Correctness

Let us prove the correctness of our scheme. Since $\mathbf{w} = \mathbf{A}\cdot\mathbf{z} - \mathbf{t}\cdot 2^{\nu-\mu}\cdot\mathbf{c}$, $\mathbf{z} = \mathbf{y} + \mathbf{s}\cdot\mathbf{c}$, and $\mathbf{t} = \mathsf{Round}\left(\frac{p}{q}\cdot\mathbf{As}\right)$, then

$$\mathbf{w} = \mathbf{A}\cdot(y + \mathbf{s}\cdot\mathbf{c}) - \mathbf{c}\cdot 2^{\nu-\mu}\cdot\mathsf{Round}\left(\frac{p}{q}\cdot\mathbf{As}\right) = \mathbf{Ay}\cdot\mathbf{Asc} - \mathbf{c}\cdot 2^{\nu-\mu}\cdot\mathsf{Round}\left(\frac{p}{q}\cdot\mathbf{As}\right)$$

According to the notation we introduced, $\mathsf{Round}\left(\frac{p}{q}\cdot\mathbf{As}\right) = \mathsf{MSB}(\mathbf{As} + \mathbf{h}, \mu)$, where $\mathbf{h}$ is a vector, with each coordinate equal to $h = 2^{\nu-\mu-1}$. Then,

$$\mathbf{w} = \mathbf{Ay} + \mathbf{Asc} - \mathbf{c}\cdot 2^{\nu-\mu}\cdot\mathsf{MSB}(\mathbf{As} + \mathbf{h}, \mu) = \mathbf{Ay} + \mathbf{Asc} - \mathbf{c}\cdot(\mathbf{As} + \mathbf{h} + \mathsf{LSB}(\mathbf{As} + \mathbf{h}, \nu - \mu))$$

Expanding the parentheses, we finally get

$$w = \mathbf{A}\mathbf{y} - \mathbf{c} + (\mathbf{h} + \mathsf{LSB}(\mathbf{A}\mathbf{s} + \mathbf{h}, \nu - \mu)),$$

where $\|\mathbf{c} \cdot (\mathbf{h} + \mathsf{LSB}(\mathbf{A}\mathbf{s} + \mathbf{h}, \nu - \mu))\|_\infty < w \cdot 2^{\nu-\mu+1}$, since $\mathbf{c} \in B_w$ and $\|\mathsf{LSB}(\mathbf{A}\mathbf{s}, \nu - \mu)\|_\infty < 2^{\nu-\mu}$.

Considering $\mathsf{LSB}(\mathbf{w}, \nu - d)$ in Algorithm 3.2 at step 5, and taking into account the error $\mathbf{c} \cdot (\mathbf{h} + \mathsf{LSB}(\mathbf{A}\mathbf{s} + \mathbf{h}, \nu - \mu))$, we obtain that for $\|\mathsf{LSB}(\mathbf{w}, \nu - d)\|_\infty > 2^{\nu-d} - w \cdot 2^{\nu-\mu+1}$, the algorithm rejects the value $\mathbf{w}$.

Since $\mathbf{c} \cdot (\mathbf{h} + \mathsf{LSB}(\mathbf{A}\mathbf{s} + \mathbf{h}, \nu - \mu))$ is a small error vector, it is obvious from the equality above that $\mathsf{MSB}(\mathbf{w}, d) = \mathsf{MSB}(\mathbf{A}\mathbf{y}, d)$. Therefore, the computation of $\mathbf{c}'$ at step 2 of Algorithm 3.3 coincides with the value of the vector $\mathbf{c}$ at step 2 of Algorithm 3.2.

3.2 On The number of iterations in the sign algorithm

In the process of calculating the signature, Algorithm 3.2 at step 5 checks whether the coefficients of the vector of the torus $\mathbf{z}$ in the interval $[-(\gamma - \beta - 1), \gamma - \beta - 1]$. For a fixed key s, the probability of this event depends on the $\|\mathbf{y}\|_\infty$ selected in step 1. Let's calculate this probability.

Let $\mathbf{z} = \mathbf{y} + \mathbf{v}$ such that $\mathbf{z} \in S^\ell_{\gamma-\beta-1}$. We set $\beta = \|\mathbf{cs}\|_\infty$. Since $\|\mathbf{s}\|_\infty \leq s$ and $\mathbf{c} \in B_w$, then $\beta < ws$. Hence, $\|\mathbf{v}\|_\infty \leq \beta$. For each coefficient $\mathbf{v}_i$ of vector $\mathbf{v}$, the corresponding coefficient $\mathbf{z}_i$ lies in the interval $[-(\gamma - \beta - 1), \gamma - \beta - 1]$. Since $\mathbf{y} = \mathbf{z} - \mathbf{v}$, then $\mathbf{y} \in S^\ell_{\gamma-1}$, and the corresponding coefficient $\mathbf{y}_i$ lies in the interval $[-(\gamma - 1), \gamma - 1]$. Therefore,

$$p_1 = \Pr_{\mathbf{y} \leftarrow S^\ell_{\gamma-1}} [\|\mathbf{z}\|_\infty < \gamma - \beta] = \frac{|S^\ell_{\gamma-\beta-1}|}{|S^\ell_{\gamma-1}|} = \left(\frac{2\gamma - 2\beta - 1}{2\gamma - 1}\right)^{n\cdot\ell} = \left(1 - \frac{\beta}{\gamma - \frac{1}{2}}\right)^{n\cdot\ell} \approx \mathsf{exp}\left(-\frac{\beta n \ell}{\gamma}\right)$$

Algorithm 3.2 at step 5 also checks when the coefficients of the vector $\mathsf{LSB}(\mathbf{w}, \nu - d)$ are not within the interval $[0, 2^{\nu-d} - w \cdot 2^{\nu-\mu+1} - 1]$. The probability of this event obvously depends on the sign of the error vector $\mathbf{c} \cdot (\mathbf{h} + \mathsf{LSB}(\mathbf{A}\mathbf{s} + \mathbf{h}, \nu - \mu))$, which arises when simplifying the expression $\mathbf{w} = \mathbf{A} \cdot \mathbf{z} - \mathbf{t} \cdot 2^{\nu-\mu} \cdot \mathbf{c}$ at step 4. Calculate this probability.

As shown above, each coefficient of the error vector $\mathbf{c} \cdot (\mathbf{h} + \mathsf{LSB}(\mathbf{A}\mathbf{s} + \mathbf{h}, \nu - \mu))$ lies in the interval $[0, w \cdot 2^{\nu-\mu+1} - 1]$. For each such coefficient, the corresponding coefficient of the vector $\mathsf{LSB}(\mathbf{w}, \nu - d)$ falls into the interval $[0, 2^{\nu-d} - 1]$. Given the (heuristically) uniform nature of the distributions, as a result we get

$$p_2 = \Pr_{\mathbf{w} \in S^k_{2^{\nu-d}-1}} [\|LSB(\mathbf{w}, \nu - d)\|_\infty < 2^{\nu-d} - w \cdot 2^{\nu-\mu}] = \left(\frac{2^{\nu-d} - w \cdot 2^{\nu-\mu+2} - 1}{2^{\nu-d+1} - 1}\right)^{n\cdot k}$$
$$= \left(1 - \frac{w 2^{\nu-\mu+2}}{2^{\nu-d} - 1}\right)^{n\cdot k} \approx \mathsf{exp}\left(-nk \cdot \frac{w 2^{\nu-\mu+2}}{2^{\nu-d} - 1}\right)$$

Thus, the expected number of repetitions of the sign function of algorithm 3.2 is given by

$$\mathbb{E}[\#\mathsf{iterations}] = (p_1 \cdot p_2)^{-1} \tag{1}$$

4 Proof of safety

The proof of the safet of our scheme is an adaptation of the proof from [11, 13] for giving MLWR and for another choice of module. In the Dilithium circuit, $q \equiv 1 \mod n$ is used, while here, q is a power of two. Our signature is built on the Fiat-Shamir paradigm[2] (more precisely, the Fiat-Shamir paradigm with interruptions [15]). Proof of security in the QROM (Quantum Random Oracle Model, or quantum model of a random oracle)[3] consists of two stages.

[2]Fiat-Shamir proposed a universal method for constructing a signature scheme from a cryptographic hash functions and identification schemes with some security properties.

[3]In QROM, in contrast to the usual ROM (classical model of a random oracle), the attacker has quantum access to a random oracle, that is, can query oracle values for messages in quantum superposition.

At the first stage, we show that there is an algorithm (Algorithm 4.1) that simulates the non-transformation of the signature $\mathsf{Sign}()$ (Algorithm 3.2) so that the result of the simulator is statistically indistinguishable from the output of the real procedure. In the language of the Fiat-Shamir paradigm, this means that the signature is based on a zero-knowledge protocol, namely the Honest Verifier Zero Knowledge (zero-knowledge with an honest reviewer, see [13]). The proof is exactly the same. This is the proof for the Dilithium scheme; for completeness, we repeat it in Lemma 1.

Algorithm 4.1 Signature simulator

```
    Inputs: $pk = (\mathbf{A}, \mathbf{t}) \in R_q^{k\times\ell} \times R_p^k$
    Outputs: $(\mathbf{z}, \mathbf{c}) \in R_q^\ell \times \mathcal{C}$

1: With probability $1 - \frac{|S^\ell_{\gamma-\beta-1}|}{|S^\ell_{\gamma-1}|}$ restart
2: $\mathbf{z} \leftarrow S^\ell_{\gamma-\beta-1}$
3: if $\|\mathsf{LSB}(\mathbf{Az} - 2^{\nu-\mu}\cdot\mathbf{tc}, \nu-d)\|_\infty > 2^{\nu-d} - c\cdot 2^{\nu-\mu+1}$ then
4:     restart
5: else
6:     return $(\mathbf{z}, \mathbf{c})$
```

Lemma 1. *If $\beta \geq \max_{\mathbf{c}\in\mathcal{C},\mathbf{s}\in\mathcal{S}} \|\mathbf{c}\cdot\mathbf{s}\|_\infty$, then the outputs of Algorithm 3.2 and Algorithm 4.1 have the same total distribution.*

Proof. For any $\mathbf{z} \in S^\ell_{\gamma-\beta-1}$ and for any $\mathbf{c} \in \mathcal{C}$:

$$\Pr_{\mathbf{y}\leftarrow S^\ell_{\gamma-1}}[\mathbf{y} + \mathbf{cs} = \mathbf{z}] = \Pr_{\mathbf{y}\leftarrow S^\ell_{\gamma-1}}[\mathbf{y} = \mathbf{z} - \mathbf{cs}]$$

Since we let $\|\mathbf{c}\cdot\mathbf{s}\| \leq \beta$, then $\mathbf{z} - \mathbf{cs} \in S^\ell_{\gamma-1}$. Therefore, $\Pr_{\mathbf{y}\leftarrow S^\ell_{\gamma-1}}[\mathbf{y} = \mathbf{z} - \mathbf{ct}] = \frac{1}{|S^\ell_{\gamma-1}|}$, and therefore, each element $\mathbf{z} \in S^\ell_{\gamma-\beta-1}$ has the same chance of being generated by Algorithm 3.2, which means that Step 2 of Algorithm 4.1 ideally simulates vector $\mathbf{z}$. In addition, the restart probability in Algorithm 3.2 at step 6 due to the fulfillment of $\|z\|_\infty \geq \gamma - \beta$ is equal to $1 - \frac{S^\ell_{\gamma-\beta-1}}{S^\ell_{\gamma-1}}$. This is exactly the restart probability at the first step of Algorithm 4.1. The remaining steps of Algorithms 3.2 and 4.1 are identical. □

Next, we will show that in the process of generating a signature, a sufficient amount of entropy is supplied to the input hash function $\mathcal{H}$. A similar proof in [13, Appendix C] amounts to obtaining the lower bound for the probability that a random element in R_q is invertible. In the case of a prime q, the rate of obtaining a reversible element from a randomly selected one is at least $(1-n/q)$, If q is a power of two, this probability is rather small. Therefore, to form an open key $\mathbf{A} \in R_q^{k\times\ell}$, we generate a known reversible element using Fact 1. This is an effective procedure and the number of reversible elements in R_q is exponential in n. Further reasoning is similar to the proof in [13, Appendix C], so here we only provide it in our notation.

Lemma 2. *For $\mathbf{A} \leftarrow R_q^{k\times\ell}$ such that at least one of the $k\cdot\ell$ polynomials forming $\mathbf{A}$ is invertible, and for all $\mathbf{w} \in R_q^\ell$:*

$$\Pr_{\mathbf{y}\leftarrow S^\ell_{\gamma-1}}[\mathsf{MSB}(\mathbf{Ay}, d) = \mathbf{w}] < \left(\frac{d+1}{2\gamma-1}\right)^n$$

Combining the results of Lemmas 1 and 2, we can assert that in order to prove UF-CMA signature scheme strength, it is enough to show only UF-NMA strength (in UF-NMA, the attacker does not have access to the signing oracle). This is classic result of a lattice reduction [13]. Therfore, in the second step of the proof, we will show that using complexity of MLWR and SelfTargetSIS tasks, our signature is persistent in the UF-NMA model. This step again is an adaptation of the proof for the Dilithium scheme [13, Lemma 4.10].

Lemma 3. *For any quantum algorithm A that successfull attacks the signature scheme, calculated in Algorithms 3.1-3.3, in the UF-NMA model using the oracle H, no more than Q_H times, there exist quantum algorithms B, C such that:*

$$\mathsf{Adv}^{\mathsf{UF-NMA}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{MLWR}}_{\mathsf{PARAMS}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathsf{PARAMS}}(\mathcal{C}),$$

with operating time $T(\mathcal{B}) = T(\mathcal{C}) = T(\mathcal{A}) + Q_{\mathcal{H}}$.

Proof. Suppose that algorithm $\mathcal{C}$, which attacks SelfTargetMSIS, receives the matrix $\mathbf{A}' = [\mathbf{A}|\mathbf{t}'|\mathbb{I}] \in R_q^{k\times(\ell+1+k)}$. Further, the algorithm $\mathcal{C}$ sets $\mathbf{t} = \mathsf{Round}\left(\mathbf{t}'\frac{p}{q}\right) = \mathsf{MSB}(\mathbf{t}+\mathbf{h} \mod q, \mu)$ and uses the pair $(\mathbf{A}, \mathbf{t})$ as the public key for Algorithm $\mathcal{A}$. Regarding the assumption complexity of the MLWR problem, such a pair is indistinguishable from the real public key returned by Algorithm 3.1. In this case, Algorithm $\mathcal{A}$ returns a signature $(\mathbf{z}, \mathbf{c})$ which for some message M passes the verification algorithm: $\|\mathbf{z}\|_\infty < \gamma - \beta$ and

$$c = \mathcal{H}(\mathsf{MSB}(\mathbf{w}, d), M),$$

where $\mathbf{w} = \mathbf{Az} - 2^{\nu-\mu}\mathbf{t}\cdot\mathbf{c}$. Since $\mathsf{MSB}(\mathbf{w}, d)2^{\nu-d} = \mathbf{w} - \mathsf{LSB}(\mathbf{w}, \nu - d)$ and $\mathbf{t}' = 2^{\nu-\mu}\mathbf{t} + \mathsf{LSB}(\mathbf{t}', \nu-\mu)$, verification equation is equivalent to

$$c = \mathcal{H}(\mathbf{Az} - 2^{\nu-\mu}\mathbf{t}\cdot\mathbf{c} - \mathsf{LSB}(\mathbf{w}, \nu-d), M) = \mathcal{H}(\mathbf{Az} - \mathbf{t}'\mathbf{c} + \mathbf{e}, M),$$

where $\mathbf{e} = -\mathsf{LSB}(\mathbf{w}, \nu-d) - \mathsf{LSB}(\mathbf{t}', \nu-\mu)\cdot\mathbf{c}$. Moreover, $\mathbf{e} : \|\mathbf{e}\|_\infty < 2^{\nu-d} + w\cdot 2^{\nu-\mu}$. Hence, we get SelfTargetSIS problem solution

$$c = \mathcal{H}\left([\mathbf{A}|\mathbf{t}'|\mathbb{I}]\begin{bmatrix}\mathbf{z}\\ \mathbf{c}\\ \mathbf{e}\end{bmatrix}, M\right),$$

where $\|[\mathbf{z}|\mathbf{c}|\mathbf{e}]\|_\infty < \max\{\gamma-\beta, w, 2^{\nu-d} + w\cdot 2^{\nu-\mu}\}$. □

As a result, the robustness of the scheme is based on three objectives: MLWR, MSIS, and SelfTargetMSIS. Intuitive but, key security is based on the MLWR task (i.e., the key recovery task is reduced to MLWR solution), obtaining a fake signature is reduced to the SelfTargetMSIS task, and MSIS needs to achieve security in the sUF-CMA model. Combining all the previous statements, we get the main security theorem for our signature scheme.

Theorem 4. *For any quantum algorithm $\mathcal{A}$ successfully attacking the signature scheme, calculated in Algorithms 3.1-3.3, in the sUF-CMA model using the oracle $\mathcal{H}$ at most $Q_{\mathcal{H}}$ times, there are quantum algorithms $\mathcal{B}, \mathcal{C}, \mathcal{D}$ with running time $T(\mathcal{D}) = T(\mathcal{B}) = T(\mathcal{C}) = T(\mathcal{A}) + Q_{\mathcal{H}}$, such that:*

$$\mathsf{Adv}^{\mathsf{sUF-CMA}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{MLWR}}_{k,\ell,p,q}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{k,\ell,q,B}(\mathcal{C}) + \mathsf{Adv}^{\mathsf{MSIS}}_{k,\ell,q,B'}(\mathcal{D}) + \left(\frac{d+1}{2\gamma-1}\right)^n,$$

where $B = \max\{\gamma-\beta, 2^{\nu-d} + w2^{\nu-\mu}\}$, $B' = \max\{2^{\nu-d+1}, 2(\gamma-\beta)\}$.

Proof. The proof of the circuit's robustness in the UF-CMA model, as follows from Lemmas 1, 2, and 3, relies on the complexity of MLWR and SelfTargetSIS tasks. To obtain evidence for the stronger model, sUF-CMA, we use the complexity of the MSIS problem. Recall that the difference between two models is that in sUF-CMA, an attacker is successful if he generates a new signature $\sigma' = (\mathbf{z}, \mathbf{c})$ of some message M, for which the attacker could already know another correct signature $\sigma \neq \sigma'$.

Suppose $\mathcal{D}$ finds a vector $\mathbf{z}'$ such that the verification algorithm takes the pair $(\mathbf{z}', \mathbf{c})$ as a correct signature for some message M. Then, $\mathsf{MSB}(\mathbf{w}', d) = \mathsf{MSB}(\mathbf{w}, d)$, where $\mathbf{w}' = \mathbf{Az}' - 2^{\nu-\mu}\mathbf{tc}$ and $\mathbf{w} = \mathbf{Az} - 2^{\nu-\mu}\mathbf{tc}$. Since $\mathbf{w}' = 2^{\nu-d}\mathsf{MSB}(\mathbf{w}', d) - \mathsf{LSB}(\mathbf{w}', \nu-d)$ and $\|\mathsf{LSB}(\mathbf{w}', \nu-d)\|_\infty \le 2^{\nu-d} - 1$ (similarly for $\mathbf{w}$), we obtain:

$$\|\mathbf{Az} - 2^{\nu-\mu}\mathbf{tc} - 2^{\nu-d}\mathsf{MSB}(\mathbf{w}, d)\|_\infty \le 2^{\nu-d} - 1$$

$$\|\mathbf{Az}' - 2^{\nu-\mu}\mathbf{tc} - 2^{\nu-d}\mathsf{MSB}(\mathbf{w}', d)\|_\infty \le 2^{\nu-d} - 1$$

Summing up both inequalities, we obtain $\mathbf{A}(\mathbf{z} - \mathbf{z}') + \mathbf{e} = 0$, where $\|\mathbf{e}\|_\infty \le 2^{\nu-d+1} - 2$ and $\|\mathbf{z} - \mathbf{z}'\|_\infty \le 2^{\gamma-\beta}$. Thus, we have a solution for the MSIS problem. □

5 Attacks and parameter selection

The security of our siganture scheme is based on two classic lattice problems - MLWR and MSIS. We will determine the specific parameters of the circuit based on the complexity attack on these tasks.

We work with modular lattices defined over a ring of whole cyclotomic extension, namely, $R = \mathbb{Z}[x]/(x^{256}+1)$, that is, $n = 256$. Basic parameters that determine the complexity of MLWR and MSIS are k (specifies the rank of the matrices) and l (specifies the size of the secret vector $\mathbf{s}$).

Solving the MLWR problem is reduced to finding a short vector in a q-ary lattice of rank d:

$$\Lambda_{\mathsf{MLWR}} = \{\mathbf{x} \in \mathbb{Z}^d | [\mathsf{rot}(\mathbf{A})|\mathbb{I}|\mathbf{t}]\mathbf{x} = 0 \mod q\},$$

where $d \le n(\ell + k) + 1$. We use the $\le$ sign, since the optimal attack may not use some rows of the matrix $[\mathsf{rot}(\mathbf{A})|\mathbb{I}|\mathbf{t}]$. The vector $\mathbf{x} \in \Lambda_{\mathsf{MLWR}}$ we need is the vector $\mathbf{x}_{\mathsf{short}} = [\mathsf{rot}(s)| - \mathbf{t}_{\mathsf{low}}| - 1]$, where $\mathbf{t}_{\mathsf{low}} = \mathbf{A}\mathbf{s} - \mathbf{t}$ and $\|\mathbf{t}_{\mathsf{low}}\|_\infty \le 2^{\nu-\mu}$. This is a "short" vector in the lattice Λ_{MLWR}, so how much shorter it is, $\sqrt{d}q^{\frac{1}{nk}}$, are the Minkowski bounds for Λ_{MLWR}.

This is a classic "primal" attack on LWR, the difficulty of which depends on the running time algorithm BKZ to find a vector of length $\|\mathbf{x}_{\mathsf{short}}\|$. Estimate the specific opening hours of BKZ is a non-trivial task. To obtain a value of 104 in Table 1 – a conservative estimate of the time change of work BKZ to solve the LWR problem – we relied on work [2] and software code [3]. We do not give an estimate for the so-called "dual" attack on LWR, since the "primal" method for our parameters turned out to be much more efficient.

Let us now consider the complexity of the MSIS task (since the SelfTargetSIS task is reduced to MSIS, and for our parameters, attacks on MSIS work much more efficiently, the determining factor is the complexity of MSIS). The most effective of all knowna attacks against MSIS is finding a short vector in the lattice

$$\Lambda_{\mathsf{MSIS}} = \{\mathbf{x} \in \mathbb{Z}^d | [\mathsf{rot}(\mathbf{A})|\mathbb{I}]\mathbf{x} = \mathbf{0} \mod q\}.$$

In contrast to the MLWR attack, the optimal algorithm for the MSIS problem can omit some columns of the matrix $[\mathsf{rot}(\mathbf{A}|\mathbb{I})]$. A short vector $\mathbf{x} \in \Lambda_{\mathsf{MSIS}}$ normalizes $\|\mathbf{x}\|_\infty \le \mathsf{max}\{2^{\nu-d+1}, 2(\gamma - \beta)\}$. For the parameters shown in Table 1, these two values roughly coincide.

To obtain the specific complexity of the MSIS attack, we used the strategy described in [11, Appendix C].[4]

n	k	ℓ	w	ν	μ	s	d	β	γ	$\mathbb{E}[\#\mathsf{iterations}]$	MSIS (BKZ-b)	MLWR (BKZ-b)
256	4	3	60	23	19	4	3	240	1048096	20	93 (320)	104 (357)

Table 1: Proposed digital signature parameters and their security level. The last two parameters (93 and 104, respectively) correspond to the bit complexity of the attack on MSIS (with the optimal block size in the BKZ algorithm equal to 320) and MLWR (with optimal block size 357), respectively. In both calculations, we assume (conservatively) that the complexity of finding the short vector in the lattice of dimension d equals $2^{0.292d}$, which asymptotically corresponds to the complexity of the sieving algorithm. The parameter β according to the definition $\beta = \|\mathbf{cs}\|_\infty \le w\|\mathbf{s}\|_\infty = 240$.

6 Implementation features

An experimental implementation of the digital signature scheme in C++ is in the public domain, licensed under the GNU GPL v3. The process of debugging and testing was performed on OC Ubuntu Linux 04/18/4 LTS x64, g++ compiler from GCC 9.2.1 package. Implementation uses separate parts of the code (library) of the Crystals-Dilithium digital signature project, which are in the public domain. In this section, we will describe the technical features of the implementation and give the costing of computing resources in time.[5]

6.1 Storing the public key

Storing the public key explicitly requires significant memory consumption to represent the matrix $\mathbf{A}$. It consists of $k \cdot \ell$ polynomials in R_q. Thus, the size of the occupied matrix $\mathbf{A}$ memory is $k \cdot \ell \cdot n \cdot \nu$ bits, while matrix generation is performed using a pseudo-random generator from an initialization vector of a

[4]The script with which you can get Table 1 is available at `https://crypto-kantiana.com/elena.kirshanova/#research`.

[5]`https://github.com/ElenaKirshanova/pqc_LWR_signature`

fixed length of 512 bits. Therefore, we store the value of the initialization vector and calculate the matrix $\mathbf{A}$ before each generation or signature verification. In order to ensure the reversibility of at least one of the $k \times \ell$ polynomials in R_q, we fix one polynomial, for example, located at position $\mathbf{A}[0,0]$, and perform the following actions on it:

1. All odd coefficients, except for 0, are reduced by one.
2. If the 0th coefficient is even, add one to it.

Thus, at the position $\mathbf{A}[0,0]$ there will be a polynomial satisfying the conditions given in [17, Theorem 11.1], i.e., invertible in R_q, and it is this polynomial that will be obtained from the initialization vector in a unique way.

6.2 Generating a uniform distribution

Secret vectors $\mathbf{s} \leftarrow S_s^\ell$ (private key) and $\mathbf{y} \leftarrow S_{\gamma-1}^\ell$ (participates in the signature algorithm) must be generated randomly from a uniform distribution. For this we generate a pseudo-random sequence of bytes using the SHAKE-128 (SHA3) algorithm, then as output from this sequence, we choose u-bit numbers, choosing the minimum value as u, where $2^u > s$ and $2^u > \gamma - 1$ for vectors $\mathbf{s}$ and $\mathbf{y}$, respectively. Each number is checked for the appropriateness of the intervals from which we sample. Thus, from a uniform distribution on the interval $[0 \ldots 2^u - 1]$, we obtain a uniform distribution on the intervals $[-s \ldots s]$ and $[-(\gamma-1) \ldots (\gamma-1)]$ for elements of vectors $\mathbf{s}$ and $\mathbf{y}$, respectively.

The sequence of bytes at the output of SHAKE-128 is generated from a random initialization vector. The length of the output sequence is chosen so that the probability of a successful guess of the sample from it is at least $1 - 10^5$.

6.3 Memory and time efficiency

Testing of key pair generation, signature, and verification time was done on the CPU Intel Xeon (R) E-214G 3.50GHz x 12, operating time is indicated by the number of clock cycles in Table 2. Mesasurement of the average time value was calculated using the number of tests $N = 1000$, under the same conditions used in Dilithium [11].

	|sk|	|vk|	|sig|	KeyGen	Sign	Verify
Our signature	3K	2.4K	1.9K	2.08M	24.6M	2.6M
Dilithium	2.05K	1.2K	2.0K	133K	700K	150K

Table 2: Sizes of keys and signature in bytes for parameters taken from Table 1 in comparison with one set of Dilithium signature parameters [11], comparable in terms of security. The running times of the KeyGen, Sign, and Verify algorithms are expressed in the number os clock cycles.

References

[1] M. Ajtai. Generating hard instances of lattice problems (extended abstract). *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC'96*, pages 99–108, 1996.

[2] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to lwe. *Advances in Cryptology - ASIACRYPT 2017*, pages 297–322, 2017.

[3] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the lwe, ntru schemes! *Security and Cryptography for Networks*, page 351–367, 2018.

[4] Erdem Alkim, Leo Ducas, Thomas Poppelmann, and Peter Schwabe. Post-quantum key exchange: A new hope. *Proceedings of the 25th USENIX Conference on Security Symposium, SEC'16*, pages 327–343, 2016.

[5] Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Kramer, and Filip Pawlega. Revisiting tesla in the quantum random oracle model. *Post-Quantum Cryptography*, pages 143–162, 2017.

[6] Shi Bai and Steven D. Galbraith. An improvised compression technique for signautres based on learning with errors. *Topics in Cryptology - CS-RSA 2014*, pages 28–47, 2014.

[7] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. *Annual International Confernce on the Theory and Applications of Gryptographic Techniques*, pages 719–737, 2012.

[8] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. *Theory of Cryptography*, pages 209–224, 2016.

[9] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. 305:53–74, 2002.

[10] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. *Progress in Cryptology - AFRICACRYPT 2018*, pages 282–305, 2018.

[11] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb 2018.

[12] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. *Advances in Cryptology - CRYPTO '86*, pages 186–194, 1987.

[13] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. *Advances in Cryptology - EUROCRYPT 2018*, pages 552–586, 2018.

[14] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des Codes Cryptography*, 75(3):565–599, Jun 2015.

[15] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. *Advances in Cryptology - ASIACRYPT 2009*, pages 596–616, 2009.

[16] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

[17] John J. Watkins. *Topics in cummutative ring theory.* Princeton University Press, 2007.