Quantum-Resistant Forward-Secure Digital Signature Scheme Based on q-ary Lattices

Mariusz Jurkiewicz

Military University of Technology, Warsaw, Poland

https://doi.org/10.26636/jtit.2024.2.1581

Abstract — In this paper, we design and consider a new digital signature scheme with an evolving secret key, using random q-ary lattices as its domain. It is proved that, in addition to offering classic eu-cma security, the scheme is existentially forward unforgeable under an adaptive chosen message attack (fu-cma). We also prove that the secret keys are updated without revealing anything about any of the keys from the prior periods. Therefore, we design a polynomial-time reduction and use it to show that the ability to create a forgery leads to a feasible method of solving the well-known small integer solution (SIS) problem. Since the security of the scheme is based on computational hardness of a SIS problem, it turns out to be resistant to both classic and quantum methods. In addition, the scheme is based on the "Fiat-Shamir with aborts" approach that foils a transcript attack. As for the key-updating mechanism, it is based on selected properties of binary trees, with the number of leaves being the same as the number of time periods in the scheme. Forward security is gained under the assumption that one out of two hash functions is modeled as a random oracle.

Keywords — digital signature scheme, forward security, q-ary lattices, random-oracle model, SIS problem

1. Introduction

This paper shows that the concepts and results from [1] may be adapted to a post-quantum domain. To this end, we design and analyze a new quantum-safe key evolving digital signature scheme and prove its forward security characteristics.

It is well known that the security of asymmetric crypto schemes is based on two computationally hard problems, namely factorization and DLP in finite groups of prime order. However, it should be noticed that the hardness of these problems is based solely on a classical computational model and, due to the famous Shor's algorithm (also the Kaye-Zalka's algorithm), both problems turn out to be of polynomial complexity in the quantum model.

These facts, in conjunction with the acceleration of research leading to the creation of a quantum computer with a sufficiently large quantum register, compelled the US National Security Agency (NSA) to publish, in 2015, a report indicating the need to increase the length of all Suite-B scheme keys and to urgently work out solutions that are resistant to those threats.

Soon thereafter, the National Institute of Standards and Technology (NIST) organized a competition to update their standards to include post-quantum cryptography. NIST pointed out five domains that were assumed to be a substrate for quantum-safe problems. All this triggered an immediate response of the crypto community and a lot of effort has been invested into the research on a wide variety of post-quantum aspects.

After years of intensive research, with a particular focus being on breaking SIDH and Rainbow, it turned out that lattices seemed to be the most promising and flexible of all the suggested domains, serving as a substrate for modern asymmetric crypto primitives. These are the reasons why we have focused our considerations on the theory of lattices.

Signature schemes with an evolving private key [2], [3] are characterized by the fact that the entire lifetime of the public key is split into a certain number of sub-periods to which different secret keys are assigned. More precisely, in the first step, both the public key and the initial secret key are generated and assigned to the first period, and then the initial key is updated to the next period and so on, until the last period is reached.

Natural security requirements associated with these digital schemes are such that even if the secret key, assigned with a specific sub-period, has been reveled, then it is impossible to conduct a forgery for any of the previous sub-periods. This creates a formal security model known as forward security [2].

It should be noted that for any scheme of this sort, the updating mechanism is of crucial importance in terms of providing the required security needs. Therefore, in addition to the obvious explanation of unforgeability within separated time frames, it must be proven, above all, that the disclosure of a certain secret key reveals nothing about the past periods.

In other words, this mechanism must ensure a non-trivial property that a secret key associated with a given time frame must store nothing but data required for making current signatures and for generating a secret key for the next period.

The presented scheme is based on the so-called "Fiat-Shamir with aborts" approach [4], [5], proposed by Lyubashevsky [5], and is based on an idea derived from statistics and known as rejection sampling.

Despite the fact that the design itself has been inspired by many papers concerning different aspects of the lattice theory, [5]–[7] need to be considered as the key sources of the techniques behind our considerations. We were able to implement these different ideas to create a state-of-the-art and quantum-safe forward secure digital signature scheme.

2. Preliminaries

For a positive integer k, let $[k] := \{1, \ldots, k\}$, and $[k]_0 := \{0, 1, \ldots, k\}$. Let us denote ℓ_2 and ℓ_∞ norm by $\|\cdot\|$ and $\|\cdot\|_\infty$, respectively.

Vectors are presented in a column form and are denoted by lower-case letters in bold print (e.g. $\mathbf{x}$). We view a matrix as a set of its column vectors and denote it by capital letters in bold print (e.g. $\mathbf{A}$). The i-th entry of a vector $\mathbf{x}$ is denoted by x_i, and the j-th column of a matrix $\mathbf{A}$ is denoted $\mathbf{a}_j$ or $\mathbf{A}[j]$. We identify a matrix $\mathbf{A}$ with an ordered set $\{\mathbf{a}_j\}$ of its column vector and define the norm of matrix $\mathbf{A}$ as the norm of its longest column, i.e., $\|\mathbf{A}\| = \max_j \|\mathbf{a}_j\|$. If the columns of $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$ are linearly independent, then $\mathbf{A}^* = \{\mathbf{a}_1^*, \ldots, \mathbf{a}_k^*\}$ denotes the Gram-Schmidt orthogonalization of vectors $\mathbf{a}_1, \ldots, \mathbf{a}_k$ taken in that order.

For $\mathbf{A} \in \mathbb{R}^{n\times m_1}$ and $\mathbf{B} \in \mathbb{R}^{n\times m_2}$, having an equal number of rows, $[\mathbf{A}|\mathbf{B}] \in \mathbb{R}^{n\times(m_1+m_2)}$ denotes the concatenation of the columns of $\mathbf{A}$ followed by the columns of $\mathbf{B}$. Likewise, for $\mathbf{A} \in \mathbb{R}^{n_1\times m}$ and $\mathbf{B} \in \mathbb{R}^{n_2\times m'}$, having an equal number of columns, $[\mathbf{A};\mathbf{B}] \in \mathbb{R}^{(n_1+n_2)\times m}$ is the concatenation of the rows of $\mathbf{A}$ and the rows of $\mathbf{B}$.

Let I be a countable set and let $\{X_n\}_{n\in I}$, $\{Y_n\}_{n\in I}$ be two families of random variables, such that X_n, Y_n take values in a finite set $\mathcal{R}_n$. We call $\{X_n\}_{n\in I}$ and $\{Y_n\}_{n\in I}$ *statistically close* if their statistical distance is negligible, where the statistical distance between $\{X_n\}_{n\in I}$ and $\{Y_n\}_{n\in I}$ is defined as the following function [8]:

$$\Delta(X_n, Y_n) = \frac{1}{2} \sum_{r\in\mathcal{R}_n} \big|\Pr[X_n = r] - \Pr[Y_n = r]\big| .$$

We conclude this section with a very useful and nontrivial result, called the forking lemma. A general version of this assertion is presented, and the basic forking lemma may be found in [9].

Lemma 1. General forking lemma [10]. Fix an integer ϱ and set H of size $h \geqslant 2$. Let A a randomized algorithm returning, based on an input of $x, h_1, \ldots, h_\varrho$ a pair, with the first element thereof being is an integer in the range $0, \ldots, \varrho$ and the other element being referred to as a side output. Let IG be a randomized algorithm that may be called an input generator. The accepting probability of A, denoted acc, is defined as the probability that $J \geqslant 1$ in the experiment.

$$x \xleftarrow{\$} \mathsf{IG};\ h_1, \ldots, h_\varrho \xleftarrow{\$} H;\ (J, \sigma) \xleftarrow{\$} \mathsf{A}(x, h_1, \ldots, h_\varrho) .$$

The forking algorithm $\mathsf{F_A}$ associated with A is a randomized algorithm processes and input x in the following manner:

Algorithm 1. $\mathsf{F_A}$.

```
Input $x$
1. $h_1, \ldots, h_\varrho \xleftarrow{\$} H$
2. $(I, \sigma) \xleftarrow{\$} \mathsf{A}(x, h_1, \ldots, h_\varrho; \rho)$        $\triangleright$ $\rho$ is a tape
3. If $I = 0$ then
4.     Return $(0, \varepsilon, \varepsilon)$
5. End If
6. $\widehat{h}_I, \ldots, \widehat{h}_\varrho \xleftarrow{\$} H$
7. $(\widehat{I}, \widehat{\sigma}) \xleftarrow{\$} \mathsf{A}(x, h_1, \ldots, h_{I-1}, \widehat{h}_I, \ldots, \widehat{h}_\varrho; \rho)$
8. If $I = \widehat{I}$ and $h_I \neq \widehat{h}_I$ then
9.     Return $(1, \sigma, \widehat{\sigma})$
10. Else
11.     Return $(0, \varepsilon, \varepsilon)$
12. End If
```

Let

$$\mathsf{frk} = \Pr\left[b = 1 : x \xleftarrow{\$} \mathsf{IG};\ (b, \sigma, \widehat{\sigma}) \leftarrow \mathsf{F_A}(x)\right] .$$

Then

$$\mathsf{frk} \geqslant \mathsf{acc} \cdot \left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right) .$$

2.1. Lattices

Here, lattices are defined and some of their basic properties are presented. A comprehensive introduction to this theory may be found in [11].

Any discrete additive subgroup $\mathcal{L}$ of $\mathbb{R}^m$ is called an m-dimensional lattice. By definition, a lattice $\mathcal{L} \subset \mathbb{R}^m$ induces a quotient group $\mathbb{R}^m / \mathcal{L}$ of cosets

$$\mathbf{x} + \mathcal{L} = \{\mathbf{x} + \mathbf{v} \mid \mathbf{v} \in \mathcal{L}\}, \quad \mathbf{x} \in \mathbb{R}^m,$$

with respect to the addition operation $(\mathbf{x} + \mathcal{L}) + (\mathbf{y} + \mathcal{L}) = (\mathbf{x} + \mathbf{y}) + \mathcal{L}$.

It turns out that for every lattice $\mathcal{L}$, there is a set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ such that any lattice point is an integer linear combinations of vectors form $\mathbf{B}$, i.e.:

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) = \left\{\sum_{i=1}^{k} \alpha_i \mathbf{b}_i \mid \alpha_i \in \mathbb{Z}\right\}.$$

Set $\mathbf{B}$ is called a *basis* of $\mathcal{L}$, whereas cardinality $k = \#\mathbf{B}$ is called the *rank* of $\mathcal{L}$. If $k = m$ then we say that $\mathcal{L}$ is a *full-rank* lattice. Lattice basis $\mathbf{B}$ is not unique, namely for any unimodular matrix $\mathbf{U} \in \mathbb{Z}^{m\times m}$ (i.e. $|\det \mathbf{U}| = 1$), $\mathbf{B}\cdot\mathbf{U}$ is also a basis of $\mathcal{L}(\mathbf{B})$.

A fundamental domain of $\mathcal{L}$ is a connected set $\mathcal{F} \subset \mathbb{R}^m$ such that $\mathbf{0} \in \mathcal{F}$ and it contains exactly one representative $\bar{\mathbf{x}}$ of every coset $\mathbf{x} + \mathcal{L}$. For a lattice $\mathcal{L}$ having basis $\mathbf{B}$, a commonly used fundamental domain is the origin-centered fundamental parallelepiped $\mathcal{P}(\mathbf{B}) = \mathbf{B} \cdot \left(-\frac{1}{2}, \frac{1}{2}\right]^m$, where a coset $\mathbf{x} + \mathcal{L}$ has a representative $\mathbf{x} - \mathbf{B} \cdot \lfloor \mathbf{B}^{-1} \cdot \mathbf{x} \rceil$.

The measure of fundamental parallelepiped can be easily computed as $\operatorname{vol}\mathcal{P}(\mathbf{B}) = \left(\det(\mathbf{B}\cdot\mathbf{B}^T)\right)^{1/2}$. Additionally, this number does not depend on the choice of bases of the lattice, i.e. if $\mathcal{L} = \mathcal{L}(\mathbf{B}_1) = \mathcal{L}(\mathbf{B}_2)$ then $\operatorname{vol}\mathcal{P}(\mathbf{B}_1) = \operatorname{vol}\mathcal{P}(\mathbf{B}_2)$. Hence, the measure of the fundamental parallelepipeds is invariant of the lattice, is called the *determinant* of the lattice $\mathcal{L}$ and is denoted by $\det \mathcal{L}$.

Let $B_m(\mathbf{0}, r) := \{\mathbf{x} \in \mathbb{R}^m \mid \|\mathbf{x}\| < r\}$ be m-dimensional open sphere of radius r centered at $\mathbf{0}$. For any m-dimensional lattice $\mathcal{L}$, the i-th minimum $\lambda_i(\mathcal{L})$ is the shortest radius $r > 0$ such that $B_m(\mathbf{0}, r)$ contains i linearly independent lattice vectors:

$$\lambda_i(\mathcal{L}) := \inf \{r > 0 \mid \dim(\operatorname{span}(\mathcal{L} \cap B_m(\mathbf{0}, r))) \geqslant i\}.$$

For any lattice $\mathcal{L}$ of rank k, $\lambda_1(\mathcal{L}) \leqslant \ldots \leqslant \lambda_k(\mathcal{L})$ are constants, and $\lambda_1(\mathcal{L})$ is the length of the shortest vector in $\mathcal{L}$, i.e. $\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L}/\{\mathbf{0}\}} \|\mathbf{v}\|$.

There are several natural computational problems relating to lattices, and for an evaluation of the complexity of different types of lattice problems, see [11], [12] for example. Due to the overall concept underpinning the paper, we recall the notion of *approximate shortest independent vector problem* (SIVP).

Definition 2. For a given basis $\mathbf{B}$ of an m-dimensional lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ of rank k, the goal of the *approximate shortest independent vector problem* (SIVP_γ) is to find a set of k linearly independent lattice vectors $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \subset \mathcal{L}$, such that $\|\mathbf{V}\| \leqslant \gamma(m) \cdot \lambda_k(\mathcal{L})$, where the approximation factor $\gamma = \gamma(m)$ is a function of dimension m.

We point out that the most famous and commonly used version of SIVP_γ problem concerns the case of full-rank lattices, i.e. when $k = m$.

A full rank lattice $\mathcal{L}$ is called an *integer lattice*, if $\mathcal{L} \subset \mathbb{Z}^m$, an integer lattice is called a *q-ary lattice*, if $q\mathbb{Z}^m \subset \mathcal{L} \subset \mathbb{Z}^m$, where $q \in \mathbb{Z}_{\geqslant 1}$. Additionally, $\mathbb{Z}^m / \mathcal{L}$ is a finite group and $|\mathbb{Z}^m / \mathcal{L}| = \det \mathcal{L}$.

Let $n, m, q \in \mathbb{Z}_{q \geqslant 1}$, $n < m$ and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a full-rank matrix. Below we define a q-ary lattice that is the main object of this paper.

$$\mathcal{L}_q^{\perp}(\mathbf{A}) = \left\{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod q\right\} \\ \left(= \ker\left(\mathbf{A} : \mathbb{Z}^m \to \mathbb{Z}_q^n\right)\right).$$

By definition, $\mathcal{L}_q^{\perp}(\mathbf{A}) \subset \mathbb{Z}^m$, and for any $\mathbf{x} \in q\mathbb{Z}^m$, $\mathbf{A}\mathbf{x} \equiv 0 \pmod q$, so we get $q\mathbb{Z}^m \subset \mathcal{L}_q^{\perp}(A) \subset \mathbb{Z}^m$. This means that $\mathcal{L}_q^{\perp}(\mathbf{A})$ is indeed a full-rank q-ary lattice of dimension m. It is worth mentioning that matrix $\mathbf{A}$ which induces $\mathcal{L} = \mathcal{L}_q^{\perp}(\mathbf{A})$ is called a *parity matrix* and it is not a base of $\mathcal{L}$. Moreover, it turns out that if q is a prime, then $|\det \mathcal{L}_q^{\perp}(\mathbf{A})| = |\mathbb{Z}^m / \mathcal{L}_q^{\perp}(\mathbf{A})| = q^n$.

For a given vector $\mathbf{u} \in \mathbb{Z}_q^n$, a lattice $\mathcal{L}_q^{\perp}(\mathbf{A})$ is associated with another lattice that is defined by:

$$\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} \equiv \mathbf{u} \pmod q\}.$$

If $\mathbf{v} \in \mathcal{L}_q^{\mathbf{u}}(\mathbf{A})$, then the lattices $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A})$ and $\mathcal{L}_q^{\perp}(\mathbf{A})$ are tailored by the following relation $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}) = \mathbf{v} + \mathcal{L}_q^{\perp}(\mathbf{A})$.

2.2. Discrete Gaussian

For any real $s > 0$ and $\mathbf{c} \in \mathbb{R}^m$, the Gaussian function $\rho_{s,\mathbf{c}}$ centered on $\mathbf{c}$ with parameter s is defined as:

$$\rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\frac{\pi}{s^2}\|\mathbf{x}-\mathbf{c}\|^2}, \quad \mathbf{x} \in \mathbb{R}^m,$$

and

$$\rho_s = \rho_{s,\mathbf{0}}, \quad \rho = \rho_1, \text{ i.e. } \rho(\mathbf{x}) = e^{-\pi\|\mathbf{x}\|^2}.$$

From the definition, we have $\rho_s(\mathbf{x}) = \rho\left(s^{-1}\mathbf{x}\right)$.

Let $\mathcal{L} \subseteq \mathbb{Z}^m$ be a lattice and let $\rho_{s,\mathbf{c}}(\mathcal{L}) = \sum_{\mathbf{x} \in \mathcal{L}} \rho_{s,\mathbf{c}}(\mathbf{x})$. Let us define discrete Gaussian distribution over $\mathcal{L}$ with center $\mathbf{c}$, and parameter s as:

$$\mathcal{D}_{\mathcal{L},s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\mathcal{L})}, \quad \mathbf{x} \in \mathcal{L}.$$

For notational convenience:

$$\mathcal{D}_{\mathcal{L},s} = \mathcal{D}_{\mathcal{L},s,\mathbf{0}},\ \mathcal{D}_{s,\mathbf{c}}^m = \mathcal{D}_{\mathbb{Z}^m,s,\mathbf{c}},\ \mathcal{D}_s^m = \mathcal{D}_{\mathbb{Z}^m,s}.$$

We summarize several facts from the literature about discrete Gaussians over lattices, again if focus to our interest area.

Lemma 3. Let $n < m$ and $\mathbf{T}_\mathbf{A}$ be any basis of $\mathcal{L}_q^{\perp}(\mathbf{A})$ for some $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ whose columns generate $\mathbb{Z}_q^n$, let $\mathbf{u} \in \mathbb{Z}_q^n$ and $\mathbf{c} \in \mathbb{Z}^m$ be arbitrary, and let $s \geqslant \|\mathbf{T}_\mathbf{A}^*\| \cdot \omega\left(\sqrt{\log m}\right)$, we have:

- $\Pr_{\mathbf{x} \leftarrow \mathcal{D}_{\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}),s,\mathbf{c}}}\left[\|\mathbf{x} - \mathbf{c}\| > s \cdot \sqrt{m}\right] \leqslant \mathsf{negl}(n)$ [6], [13],
- $\Pr_{\mathbf{x} \leftarrow \mathcal{D}_{\mathcal{L}_q^{\perp}(\mathbf{A}),s}}[\mathbf{x} = 0] \leqslant \mathsf{negl}(n)$ [6], [14].
- A set of $\mathcal{O}(m^2)$ independent samples from $\mathcal{D}_{\mathcal{L}_q^{\perp}(\mathbf{A}),s}$ contains a set of m linearly independent vectors, except with negligible probability in n (see [6], [15]).

Theorem 4. There is a probabilistic polynomial-time algorithm $\mathsf{SampleD}$ that, given a basis $\mathbf{B}$ of m-dimensional lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, a parameter $s \geqslant \sigma_1(\mathbf{B}) \cdot \omega\left(\sqrt{\log m}\right)$, and a center $\mathbf{c} \in \mathbb{R}^m$, outputs a sample from a distribution that is statistically close to $\mathcal{D}_{\mathcal{L},s,\mathbf{c}}$.

The above scheme is an alternative to the $\mathsf{SampleD}$ algorithm given in [17], and it is more efficient and fully parallelizable. Value $\sigma_1(\mathbf{B})$ is the largest singular value of $\mathbf{B}$, which is never smaller than $\|\mathbf{B}^*\|$ but is also not much larger in most important cases – see [16] for details.

Lemma 5. Let $n, m, q \in \mathbb{Z}_{>0}$ with q prime and let $m \geqslant 2n \lg q$. Then for all but the q^{-n} fraction of all $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the subset-sums of the columns of $\mathbf{A}$ generate $\mathbb{Z}_q^n$; i.e. for every syndrome $\mathbf{u} \in \mathbb{Z}_q^n$ there is $\mathbf{x} \in \{0,1\}^m$, such that $\mathbf{u} \equiv \mathbf{A}\mathbf{x} \pmod q$ [14], [17], [18].

Lemma 6. Let $n, m, q \in \mathbb{Z}_{>0}$ with q prime, $m \geqslant 2n \lg q$, and $s \geqslant \omega\left(\sqrt{\log m}\right)$. If the columns of $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ generate $\mathbb{Z}_q^n$, then for fixed $\mathbf{u} \in \mathbb{Z}_q^n$ and an arbitrary solution $\mathbf{t} \in \mathbb{Z}^m$ to $\mathbf{A}\mathbf{t} \equiv \mathbf{u} \pmod q$, the conditional distribution of $\mathbf{e} \leftarrow \mathcal{D}_s^m$, given $\mathbf{A}\mathbf{e} \equiv \mathbf{u} \pmod q$ is exactly $\mathbf{t} + \mathcal{D}_{\mathcal{L}_q^{\perp}(\mathbf{A}),s,-\mathbf{t}}$ [17].

Lemma 7. Let $n, m, q \in \mathbb{Z}_{>0}$ with q prime and let $m \geqslant 2n \lg q$. Then, for all but the $2q^{-n}$ fraction of all $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and for any $s \geqslant \omega\left(\sqrt{\log m}\right)$, the distribution of the syndrome $\mathbf{u} \equiv \mathbf{A}\mathbf{x} \pmod q$ is statistically close to uniform over $\mathbb{Z}_q^n$, where $\mathbf{x} \leftarrow \mathcal{D}_s^m$. Besides, the conditional distribution of $\mathbf{x} \leftarrow \mathcal{D}_s^m$ given $\mathbf{u} \equiv \mathbf{A}\mathbf{x} \pmod q$ is $\mathcal{D}_{\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}),s}$ [7], [11].

Now a variant of the rejection sampling algorithm can be formulated, in which one generates samples from a desired probability distribution by using samples from another distribution. The use of this method will enable to achieve transcript security of presenting the signature scheme. We start with the lemma that allows to bound the success probability of the algorithm.

Lemma 8. For any $\mathbf{c} \in \mathbb{Z}^m$, if $s = \alpha\|\mathbf{c}\|$, for $\alpha \in \mathbb{R}_{>0}$, then [5]:

$$\Pr_{\mathbf{x}\leftarrow\mathcal{D}_s^m}\left[\frac{\mathcal{D}_s^m(\mathbf{x})}{\mathcal{D}_{\mathbf{c},s}^m(\mathbf{x})} < \exp\left(\frac{12}{\alpha} + \frac{1}{2\alpha^2}\right)\right] > 1 - 2^{-100}.$$

Theorem 9 (Rejection sampling). Let V be a subset of $\mathbb{Z}^m$ in which all elements have norms less than T, s be some element in $\mathbb{R}$ such that $s = \omega(T\sqrt{\log m})$, and $\chi : V \in \mathbb{R}$ be a probability distribution. Then, there exists a constant $M = \mathcal{O}(1)$, such that the distribution of the following algorithm $\mathcal{A}$ [5]:

1) $\mathbf{c} \leftarrow \chi$
2) $\mathbf{x} \leftarrow \mathcal{D}_{\mathbf{c},s}^m$
3) output $(\mathbf{x}, \mathbf{c})$ with probability $\min\left(\frac{\mathcal{D}_s^m(\mathbf{x})}{M\mathcal{D}_{\mathbf{c},s}^m(\mathbf{x})}, 1\right)$

is within statistical distance $\frac{1}{M} \cdot 2^{-\omega(\log m)}$ of the distribution of the following algorithm $\mathcal{F}$:

1) $\mathbf{c} \leftarrow \chi$
2) $\mathbf{x} \leftarrow \mathcal{D}_s^m$
3) output $(\mathbf{x}, \mathbf{c})$ with probability $1/M$

Moreover, the probability that $\mathcal{A}$ outputs something is at least $\frac{1-2^{-\omega(\log m)}}{M}$.

More specifically, if $s = \alpha T$ for any $\alpha \in \mathbb{R}_{>0}$, then $M = \exp\left(\frac{12}{\alpha} + \frac{1}{2\alpha^2}\right)$, the output of algorithm $\mathcal{A}$ is within statistical distance $\frac{2^{-100}}{M}$ of the output of $\mathcal{F}$, and the probability that $\mathcal{A}$ outputs something is at least $\frac{1-2^{-100}}{M}$.

2.3. Trapdoors for Lattices

Lemma 10. Let $\delta \in \mathbb{R}_{>0}$ be any fixed constant and $q \in \mathbb{Z}_{\geqslant 3}$ be odd. There is a universal constant $C \in \mathbb{R}_{>0}$ and a probabilistic polynomial-time algorithm TrapGen that, on an input of a uniformly random $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times m_1}$, for any $m_1 \geqslant d = (1+\delta)n\lg q$, and any integer $m_2 \geqslant (4+2\delta)n\lg q$, outputs matrices $\mathbf{U} \in \mathbb{Z}^{m_2\times m_2}$, $\mathbf{G}, \mathbf{R} \in \mathbb{Z}^{m_1\times m_2}$, $\mathbf{P} \in \mathbb{Z}^{m_2\times m_1}$, such that $\mathbf{U}$ is nonsingular, and $(\mathbf{GP} + \mathbb{1}_{m_1}) \subset \mathcal{L}_q^{\perp}(\mathbf{A}_1)$ [19]. Moreover:

- $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$ is $(m_2 \cdot q^{-\delta n/2})$-uniform over $\mathbb{Z}_q^{n\times m}$, where $m = m_1 + m_2$ and $\mathbf{A}_2 = -\mathbf{A}_1\cdot(\mathbf{R}+\mathbf{G}) \in \mathbb{Z}_q^{n\times m_2}$,
- $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, given by the formula:

$$\mathbf{T_A} = \begin{bmatrix} (\mathbf{G}+\mathbf{R})\mathbf{U} & \mathbf{RP} - \mathbb{1}_{m_1} \\ \mathbf{U} & \mathbf{P} \end{bmatrix},$$

  is a short basis of $\mathcal{L}_q^{\perp}(\mathbf{A})$ and meets $\|\mathbf{T_A}\| \leqslant Cn\log q = \mathcal{O}(n\log q)$ with a probability of $1 - 2^{-\Omega(n)}$ over the choice of $\mathbf{R}$,
- $\|\mathbf{T}_{\mathbf{A}}^*\| \leqslant 1 + C\sqrt{d} = \mathcal{O}(\sqrt{n\log q})$ with probability of $1 - 2^{-\Omega(n)}$ over the choice of $\mathbf{R}$.

Guided by a practical perspective, the above lemma could be reformulated in the following manner.

Theorem 11. For $n \in \mathbb{Z}_{\geqslant 1}$, an odd $q \in \mathbb{Z}_{\geqslant 3}$ and integer $m \geqslant 6n\lg q$, there is a probabilistic polynomial-time algorithm TrapGen that, on an input of q, n, m, outputs $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, where $\mathbf{A}$ is $(m \cdot q^{-n/6})$-uniform over $\mathbb{Z}_q^{n\times m}$, and $\mathbf{T_A}$ is a short (good) basis of $\mathcal{L}_q^{\perp}(\mathbf{A})$ except with negligible probability in n.

2.4. Small Integer Solution Problems

The small integer solution (SIS) problem [13], [18] aims to find a short non-zero, is to find a short nonzero integer solution $\mathbf{x} \in \mathbb{Z}^m$ to the homogeneous linear system $\mathbf{Sx} = \mathbf{0} \pmod q$ for uniformly random $\mathbf{S} \in \mathbb{Z}_q^{n\times m}$.

Definition 12. The small integer solution problem $\mathsf{SIS}_{q,n,m,\beta}$ (in the ℓ_2 norm) is: given $q \in \mathbb{Z}_{\geqslant 1}$, a uniformly random matrix $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, and $\beta \in \mathbb{R}_{>0}$, find a nonzero integer vector $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Sx} \equiv \mathbf{0} \pmod q$ and $\|\mathbf{x}\| \leqslant \beta$ [5], [13].

Equivalently, the SIS problem asks to find a vector $\mathbf{x} \in \mathcal{L}_q^{\perp}(\mathbf{S})/\{\mathbf{0}\}$ with $\|\mathbf{x}\| \leqslant \beta$.

It turns out that the distribution given by SIS has decent properties, so it is convenient to introduce its formal definition.

Definition 13. $\mathsf{SIS}_{q,n,m,d}$ *distribution*: choose a uniformly random matrix:

$$\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$$

and a vector

$$\mathbf{e} \xleftarrow{\$} \{-d, \dots, 0, \dots, d\}^m,$$

and output $(\mathbf{S}, \mathbf{Se})$ [5].

Remark 14. If $d \gg q^{n/m}$, then the $\mathsf{SIS}_{q,n,m,d}$ distribution is actually statistically close to uniform over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$ (by the leftover hash lemma) [5].

An homogeneous variant of SIS problem, called ISIS, is presented below.

Definition 15. The inhomogeneous small integer solution problem $\mathsf{ISIS}_{q,n,m,\beta}$ (in the ℓ_2 norm) is: given $q \in \mathbb{Z}_{\geqslant 1}$, a uniformly random matrix $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, a syndrome $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$, and $\beta \in \mathbb{R}_{>0}$, find an integer vector $\mathbf{x} \in \mathbb{Z}^m$, such that $\mathbf{Sx} \equiv \mathbf{u} \pmod q$ and $\|\mathbf{x}\| \leqslant \beta$ [17].

Both problems turn out to be as hard as worst-case SIVP problem (see subsection 2.1).

Theorem 16. For any positive integers n, m, real $\beta = \mathsf{poly}(n)$ and prime $q \geqslant \beta \cdot \omega\left(\sqrt{n\log n}\right)$, the average-case problem $\mathsf{SIS}_{q,n,m,\beta}$ and $\mathsf{ISIS}_{q,n,m,\beta}$ are as hard as the worst-case problem SIVP_γ with $\gamma = \beta \cdot \widetilde{\mathcal{O}}(\sqrt{n})$ [13], [17].

The next lemma plays an important role in the proof of the main result. It is inspired by [15].

Lemma 17. Assume that $d \in \mathbb{Z}_{\geqslant 9}$ and

$$m > 24 + \frac{n\lg q}{\lg(2d+1)}.$$

Then, for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and for uniformly random $\mathbf{e} \xleftarrow{\$} (-[d] \cup [d]_0)^m$, we have:

$$\Pr\big[\exists\, \mathbf{e}' \in (-[d] \cup [d]_0)^m : \mathbf{e}' \neq \mathbf{e},\ \mathbf{Ae} = \mathbf{Ae}' \pmod q\big] > 1 - 2^{-101}.$$

Proof. The matrix $\mathbf{A}$ can be viewed as an operator transforming $\mathbb{Z}^m$ into $\mathbb{Z}_q^n$, therefore $\#\mathbf{A}(\mathbb{Z}^m) = q^n$. This means that in the set $(-[d] \cup [d]_0)^m \subset \mathbb{Z}^m$, there are at most q^n different elements that do not collide with each other. Since $\#(-[d] \cup [d]_0)^m = (2d+1)^m$, probability P of selecting a non-colliding element from the set $(-[d] \cup [d]_0)^m$ can be estimated as:

$$\frac{q^n}{(2d+1)^m} < \frac{(2d+1)^{(n\lg q)/\lg(2d+1)}}{(2d+1)^{24+(n\lg q)/\lg(2d+1)}} = \frac{1}{(2d+1)^{24}} < 2^{-101}.$$

Consequently, the probability of an event that there are at least two colliding elements in:

$$\big(-[d] \cup [d]_0\big)^m \text{ is } 1 - P > 1 - 2^{-101}.$$

□

2.5. One-way Preimage Sampleable Functions

This subsection presents some important mechanisms that ensure the high level of security of the proposed work. We start by reiterating the definition of a one-way preimage sampleable function.

Definition 18. For a given value n of a security parameter, a *family of preimage sampleable functions* (PSFs) is a tuple of PPT algorithms $(\mathsf{TrapGen}, \mathsf{SampleDom}, \mathsf{SamplePre})$ satisfying the following conditions [17]:

- *Generating a representative with a trapdoor:* $\mathsf{TrapGen}(q, n, m)$ outputs (A, T) with an overwhelming probability, where A is the description of an efficiently-computable (representative) function $f_A : D_n \to R_n$ (for some efficiently-recognizable domain D_n and range R_n depending on n), and T is some trapdoor information for f_A.

For the remaining properties, fix some $(A, T) \leftarrow \mathsf{TrapGen}(q, n, m)$.

- *Domain sampling with almost uniform output*: $\mathsf{SampleDom}(1^n)$ samples an $x \in D_n$ from some (possibly non-uniform) distribution over D_n, for which the distribution of $f_A(x)$ is statistically close to uniform over R_n.
- *Preimage sampling with trapdoor:* for every $y \in R_n$, $\mathsf{SamplePre}(T, y)$ samples from a distribution that is statistically close to the conditional distribution of $x \leftarrow \mathsf{SampleDom}(1^n)$, given $f_A(x) = y$.

Definition 19. For a given value n of a security parameter, a PSFs is called a *one-way PSFs*, if the additional condition holds [17]:

- *One-way nature, without a trapdoor*: for any PPT algorithm $\mathcal{A}$, the probability that $\mathcal{A}(1^n, A, y) \in f_A^{-1}(y) \subseteq D_n$ is negligible, where the probability is taken over the choice of A, the target value $y \leftarrow R_n$ chosen uniformly at random, and $\mathcal{A}$'s random coins.

We now show a one-way PSFs design based on the average-case hardness of ISIS. This is adapted directly from [17] and plays a key role in the main part of this paper. To this end, let $q \in \mathbb{Z}_{\geqslant 3}$ be odd and $m \geqslant 6n \log q$ be an integer:

1) In order to generate a representative function with trapdoor the $\mathsf{TrapGen}$ algorithm from Theorem 11 is used. It outputs $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, where $\mathbf{A}$ is is statistically close to a uniform matrix in $\mathbb{Z}_q^{n\times m}$, and $\mathbf{T_A}$ is a short basis of $\mathcal{L}_q^{\perp}(\mathbf{A})$ except with negligible probability in n. Let $s \geqslant \|\mathbf{T}_\mathbf{A}^*\| \cdot \omega\left(\sqrt{\log m}\right)$, the function $f_\mathbf{A}$ is defined as $f_\mathbf{A}(\mathbf{e}) \equiv \mathbf{Ae} \pmod q$, with domain $\mathsf{D}_m = \{\mathbf{e} \in \mathbb{Z}^m \mid \|\mathbf{e}\| \leqslant s\sqrt{m}\}$ and range $\mathsf{R}_n = \mathbb{Z}_q^n$.
2) The input distribution is $\mathcal{D}_s^m$ and is sampled using the algorithm $\mathsf{SampleD}$ from Theorem 4 with the standard basis for $\mathbb{Z}^m$. Correctness holds because a sample $\mathbf{e} \leftarrow \mathcal{D}_s^m$ lands in the D_m domain (except with negligible probability), by Lemma 3.1, and for all but an exponentially small fraction of all $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $f_\mathbf{A}(\mathbf{e})$ is statistically close to uniform over R_n, by Lemma 7.
3) Preimage sampling with trapdoor is conducted by the trapdoor inversion algorithm called, in this specific case, $\mathsf{SampleISIS}$, taking $\mathbf{A}$, $\mathbf{T_A}$, s, and $\mathbf{u} \in \mathbb{Z}_q^n$ as input and sampling from $f_\mathbf{A}^{-1}(\mathbf{u})$ as follows:
   - choose an arbitrary $\mathbf{t} \in \mathbb{Z}^m$ such that $\mathbf{At} \equiv \mathbf{u} \pmod q$ (by Lemma 5, such a $\mathbf{t}$ exists for all but at most q^{-1} fractions of $\mathbf{A}$; such a $\mathbf{t}$, with relatively large norm, can be efficiently find via elementary linear algebra)
   - using $\mathsf{SampleD}(\mathbf{T}_A, s, -\mathbf{t})$ sample $\mathbf{v} \leftarrow \mathcal{D}_{\mathcal{L}_q^{\perp}(\mathbf{A}),s,-\mathbf{t}}$
   - output $\mathbf{e} = \mathbf{t} + \mathbf{v}$.

   Because $s \geqslant \|\mathbf{T}_\mathbf{A}^*\| \cdot \omega\left(\sqrt{\log m}\right)$, Theorem 4 implies that $\mathsf{SampleD}$ samples from a distribution that is statistically close to $\mathcal{D}_{\mathcal{L}_q^{\perp}(\mathbf{A}),s,-\mathbf{t}}$. Then by Lemma 6 $\mathsf{SampleISIS}$ samples from the appropriate conditional distribution $\mathcal{D}_{\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}),s}$.

It is important to sample the input from the discrete Gaussian $\mathcal{D}_s^m$, rather than sampling from a continuous Gaussian over $\mathbb{R}^m$ with parameter s and rounding off each coordinate to the nearest integer (see [17] for details).

Theorem 20. The algorithms described above gives a family of one-way PSFs if $\mathsf{ISIS}_{q,n,m,\sqrt{m}}$ is hard [17].

We summarize the most important ideas of the preceding discussion in the following theorem.

Theorem 21. Let $n, m, q \in \mathbb{Z}_{>0}$ be such that q is a prime, and $m \geqslant 6n \lg q$. There is a PPT algorithm – $\mathsf{SampleISIS}$ – that, on input $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, its associated trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, a Gaussian parameter $s \geqslant \|\mathbf{T}_\mathbf{A}^*\| \cdot \omega\left(\sqrt{\log m}\right)$, and $\mathbf{u} \in \mathbb{Z}_q^n$, outputs $\mathbf{e} \in \mathsf{D}_m \subset \mathbb{Z}^m$ from the distribution $\mathcal{D}_{\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}),s}$. Furthermore, $\mathbf{Ae} \equiv \mathbf{u} \pmod q$ with overwhelming probability.

We conclude this subsection with a useful generalization of $\mathsf{SampleISIS}$ that is quite important both theoretically and practically. In the notation of Theorem 21, let $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ be an ordered set of vectors, which are the columns of a matrix

$\mathbf{U} \in \mathbb{Z}_q^{n\times k}$. Let k-SampleISIS be an algorithm that takes on input $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, its associated trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, a Gaussian parameter $s \geqslant \|\mathbf{T}^*_\mathbf{A}\| \cdot \omega\left(\sqrt{\log m}\right)$, and $\mathbf{U}$.
It works as follows:

- For subsequent j's from 1 to k, $\mathsf{SampleISIS}(\mathbf{A}, \mathbf{T_A}, s, \mathbf{u}_j)$ is run to output $\mathbf{e}_j \in \mathbb{Z}^m$ from distribution $\mathcal{D}_{\mathcal{L}_q^{\mathbf{u}_j}(\mathbf{A}),s}$.
- Having given the ordered set $\{\mathbf{e}_j\}_{j\in[k]}$, matrix $\mathbf{E} = \{\mathbf{e}_1, \dots \mathbf{e}_k\}$ is the output.

As the matrix $\mathbf{U}$ induces q-ary lattices $\mathcal{L}_q^{\mathbf{u}_j}(\mathbf{A})$ and they, in turn, induce a set of discrete Gaussian distributions $\mathcal{D}_{\mathcal{L}_q^{\mathbf{u}_j}(\mathbf{A}),s}$, the k-SampleISIS algorithm outputs $\mathbf{E}$ from the joint distribution of $\left(\mathcal{D}_{\mathcal{L}_q^{\mathbf{u_j}}(\mathbf{A}),s}\right)_{j\in[k]}$. In order to ease notation, this distribution will be denoted by $\mathcal{D}^{k,\mathbf{U}}_{q,\mathbf{A},s}$.

In addition, note that if $\mathbf{E} \leftarrow (\mathbf{A}, \mathbf{T_A}, s, \mathbf{U})$ then $\mathbf{A}, \mathbf{U}$ and $\mathbf{E}$ are related by the formula $\mathbf{AE} \equiv \mathbf{U} \pmod q$. Thereby, we have proven the following theorem.

Theorem 22. Let $n, k, m, q \in \mathbb{Z}_{>0}$ be such that q is prime, and $m \geqslant 6n \lg q$. There is a PPT k-SampleISIS algorithm that, on input $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, its associated trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$, a Gaussian parameter $s \geqslant \|\mathbf{T}^*_\mathbf{A}\| \cdot \omega\left(\sqrt{\log m}\right)$, and $\mathbf{U} \in \mathbb{Z}_q^{n\times k}$, outputs a matrix $\mathbf{E} \in \mathsf{D}^k_m \subset \mathbb{Z}^{m\times k}$ from the distribution $\mathcal{D}^{k,\mathbf{U}}_{q,\mathbf{A},s}$. Furthermore, matrices $\mathbf{A}, \mathbf{U}$ and $\mathbf{E}$ are related by the $\mathbf{AE} \equiv \mathbf{U} \pmod q$ formula with overwhelming probability.

2.6. Base Extension Mechanism

In subsection 2.3 we proved that a random lattice from our family of interest can be generated together with a relatively good (short) basis for the lattice. In this case, we say that the lattice is under *control*. The following theorem shows that we may extend this control of a lattice to an arbitrary higher-dimensional extension, without any loss of quality in the resulting basis.

Theorem 23. There is a deterministic polynomial-time algorithm ExtBasis with the following properties: given an arbitrary matrix $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times m_1}$ whose columns generate the entire group $\mathbb{Z}_q^n$, an arbitrary basis $\mathbf{T}_{\mathbf{A}_1} \in \mathbb{Z}^{m_1\times m_1}$ of $\mathcal{L}_q^\perp(\mathbf{A}_1)$, and an arbitrary $\mathbf{A}_2 \in \mathbb{Z}_q^{n\times m_2}$, $\mathsf{ExtBasis}(\mathbf{T}_{\mathbf{A}_1}, \mathbf{A} = [\mathbf{A}_1|\mathbf{A}_2])$, outputs a basis $\mathbf{T_A}$ of $\mathcal{L}_q^\perp(\mathbf{A}) \subseteq \mathbb{Z}^{m_1+m_2}$, such that $\|\mathbf{T}^*_\mathbf{A}\| = \|\mathbf{T}^*_{\mathbf{A}_1}\|$. Moreover, the same holds even for any given permutation of the columns of $\mathbf{A}$, e.g. if columns of $\mathbf{A}_2$ are both appended and prepended to $\mathbf{A}$ [6].

As an immediate conclusion of the last theorem we obtain the following assertion.

Theorem 24. There is a deterministic polynomial-time algorithm ExtBasis with the following properties: given an arbitrary $\mathbf{A}_2 \in \mathbb{Z}_q^{n\times m_2}$ whose columns generate the entire group $\mathbb{Z}_q^n$, an arbitrary basis $\mathbf{T}_{\mathbf{A}_2} \in \mathbb{Z}^{m_2\times m_2}$ of $\mathcal{L}_q^\perp(\mathbf{A}_2)$, and an arbitrary $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times m_1}, \mathbf{A}_3 \in \mathbb{Z}_q^{n\times m_3}$, $\mathsf{ExtBasis}(\mathbf{T}_{\mathbf{A}_2}, \mathbf{A} = [\mathbf{A}_1|\mathbf{A}_2|\mathbf{A}_3])$, outputs a basis $\mathbf{T_A}$ of $\mathcal{L}_q^\perp(\mathbf{A}) \subseteq \mathbb{Z}^{m_1+m_2+m_3}$, such that $\|\mathbf{T}^*_\mathbf{A}\| = \|\mathbf{T}^*_{\mathbf{A}_2}\|$ [20].

3. Forward Security of Schemes with Evolving Private Key

3.1. Schemes with Evolving Private Key

Forward-secure signature schemes are based on schemes with an evolving private key, which are defined as a tuple of PPT algorithms $\Pi = (\mathscr{G}, \mathsf{KGen}, \mathsf{KUpd}, \mathsf{Sign}, \mathsf{Vrfy})$ along with a message space $\mathcal{M}$, such that they fulfill the following properties:

- **Generation of system parameters** $\mathscr{G}$ is a PT algorithm which, on input of a security parameter value of 1^n and with a maximum number of time periods T, outputs the system parameters params.
- **Key generation** KGen is a PPT algorithm which, on input the system parameters params and with a maximum number of time periods T, outputs a public verification key pk with an initial secret signing key sk_0 for the initial period $t = 0$.
- **Key update** KUpd is a PPT algorithm. It takes on input the secret key sk_t for the time period $t < T - 1$, and outputs the secret key sk_{t+1} for the subsequent $t + 1$.
- **Signing** Sign is a is a PPT algorithm which takes on input the current secret key sk_t and a message $m \in \mathcal{M}$ and outputs a signature σ.
- **Verification algorithm** Vrfy is a DPT algorithm that, on input a public key pk, a message $m \in \mathcal{M}$, the proper time period t and a (purported) signature σ, outputs a single bit b, where $b = 1$ means *accepted* and $b = 0$ means *rejected*.

In addition, we assume that *statistical correctness* holds, i.e. that for all messages $m \in \mathcal{M}$ and periods $t \in [T-1]_0$, if $(\mathsf{sk}_0, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{params}, T)$ and $\mathsf{sk}_{i+1} \leftarrow \mathsf{KUpd}(\mathsf{sk}_i)$ for $i \in [t-1]_0$, then $\mathsf{Vrfy}_{\mathsf{pk}}(t, m, \mathsf{Sign}_{\mathsf{sk}_t}(m)) = 1$ with overwhelming probability.

3.2. Security Models of Schemes with Evolving Private Key

The presented security model is taken from [2]. Let $\mathcal{A}$ be an adversary and assume that the system parameters have been generated and they have been revealed to the adversary. Let us consider the following experiment $\mathbf{Exp}^{\mathsf{fu\text{-}cma}}_{\mathcal{A},\Pi}$:

1) Generate $\mathsf{params} \leftarrow \mathscr{G}(1^n, T)$ and $(\mathsf{sk}_0, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{params}, T)$.
2) The adversary $\mathcal{A}$ is given pk and granted access to three oracles: signing oracle Sign, key update oracle KUpd and break in oracle Break.
3) $t \leftarrow 0$.
4) **while** $t < T$
    4.1. Sign : For current secret key sk_t the adversary $\mathcal{A}$ requests signatures on as many messages as it likes (analogously to euf-cma it is denoted by $\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}_t}(\cdot)}(\mathsf{pk})$).
    4.2. KUpd : If $t < T - 1$ is the current time period, then $\mathcal{A}$ requests update: $t \leftarrow t + 1$, $\mathsf{sk}_{t+1} \leftarrow \mathsf{KUpd}(\mathsf{sk}_t)$.
    4.3 If Break then break the loop **while**.

Break : If $\mathcal{A}$ is intended to move to the forge phase, then it launches Break. Then the experiment records the break-in time $\bar{t} = t$ and sends the current signing key $\mathsf{sk}_{\bar{t}}$ to $\mathcal{A}$. This oracle can only be queried once, and after it has been queried, the adversary can make no further queries to the key update or signing oracles.

5) Eventually $(t^\star, m^\star, \sigma^\star) \leftarrow \mathcal{A}(1^n, \mathsf{state})$.
6) If $t^\star < \bar{t}$ and $\mathsf{Vrfy}_{\mathsf{pk}}(t^\star, m^\star, \sigma^\star) = 1$ and the signing oracle $\mathsf{Sign}_{\mathsf{sk}_{t^\star}}$ has been never queried about $m^\star$ within the time period $t^\star$, then output 1. Otherwise output is 0.

We refer to such an adversary as an fu-cma-adversary. The advantage of $\mathcal{A}$ in attacking the scheme Π is defined as:

$$\mathbf{Adv}^{\text{fu-cma}}_{\Pi,n}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{fu-cma}}_{\mathcal{A},\Pi}(1^n, T) = 1] \,.$$

A signature scheme is called to be forward-secure if no efficient adversary can succeed in the above game with non-negligible probability.

Definition 25. A signature scheme with an evolving private key $\Pi = (\mathscr{G}, \mathsf{KGen}, \ \mathsf{KUpd}, \mathsf{Sign}, \mathsf{Vrfy})$, is called to be *existentially forward unforgeable under a chosen-message attack* or just *forward-secure* if for all efficient PPT adversaries $\mathcal{A}$, their advantage $\mathbf{Adv}^{\text{fu-cma}}_{\Pi,n}(\mathcal{A})$ is a negligible function of n.

4. Construction of Forward-Secure Scheme

We start with a high-level description of the key updating mechanism that is based on some properties of binary-trees. Let $\ell \in \mathbb{Z}_{\geqslant 1}$ be a fixed height of a tree. Then, the tree consists of 2^ℓ leaves and, in our interpretation, each leaf is associated with a singular period (and a secret key).

One may see that for every node there is a unique path joining the root with this node (in particular, a leaf). Let us adopt a rule that for a given node, 0 is assigned to its left branch and 1 to its right branch (see Fig. 1).

This implies that each of these paths is uniquely represented by a bit-string $(t_{\ell-1}t_{\ell-2}\cdots t_h)_2$, where $h \in [\ell]_0$ is the node height. Furthermore, this bit-string also uniquely indicates the node itself and, thus, we associate it with this node as its identifier. In particular, for a time period $t \in [2^\ell\text{-}1]_0$, the bit-string $(t_{\ell-1}t_{\ell-2}\cdots t_0)_2$ expresses the unique path from the root to the leaf t, and simultaneously identifies t as its binary representation.

Let $\mathbf{A}_j^{(t_j)}$, $t_j \in [1]_0$ and $j \in [\ell\text{-}1]_0$ be chosen uniformly and independently form $\mathbb{Z}_q^{n\times m}$, and let $(\mathbf{A}_\ell, \mathbf{T}_{\mathbf{A}_\ell}) \leftarrow \mathsf{TrapGen}(q, n)$. We define a node corresponding to the identifier $(t_{\ell\text{-}1}\cdots t_h)_2$ as $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2 = [\mathbf{A}_\ell|\mathbf{A}_{\ell-1}^{(t_{\ell-1})}|\cdots|\mathbf{A}_h^{(t_h)}]$. If, additionally, $t \in [2^\ell\text{-}1]_0$ and $(t_{\ell\text{-}1}\cdots t_0)_2$ is its binary representation, then we set $\mathsf{Leaf}(t) = \mathsf{Node}(t_{\ell\text{-}1}\cdots t_0)_2$.

Note that if P is the path connecting the root with a $\mathsf{Leaf}(t = (t_{\ell-1}\cdots t_{h_1}\cdots t_{h_2}\cdots t_0)_2)$, and $\mathsf{Node}(t_{\ell-1}\cdots\ t_{h_1})_2$, $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_2})_2 \in P$, then the following relation holds $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_2})_2 = \left[\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_1})_2|\mathbf{A}_{h_1-1}^{(t_{h_1-1})}|\cdots|\mathbf{A}_{h_2}^{(t_{h_2})}\right]$.

As the matrices $\mathbf{A}_j^{(t_j)}$, $\mathbf{A}_\ell$ are assumed to be publicly known, anyone can create an arbitrary node. Further, note that any node $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2$ is from $\mathbb{Z}_q^{n\times m\cdot(\ell-h+1)}$ and, thus, it can be viewed as a parity matrix which generates a q-ary lattice $\mathcal{L}_q^\perp(\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2)$ with full-rank $m\cdot(\ell-h+1)\times m\cdot(\ell-h+1)$. The secret related with this lattice is its short basis, i.e. a trapdoor $\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2}$, which is also a secret associated with the node $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2$. Obviously, it is computationally hard to derive a trapdoor from a given node.

However, it turns out that knowledge of the root's secret, along with inherent properties of the binary-tree structure, increases computational efficiency. The role of the tree's root is played by the matrix A_ℓ, whereas the associated secret is its trapdoor $\mathbf{T}_{\mathbf{A}_\ell}$ being also a secret master-key for the scheme.

In order to get a secret related to a node $\mathsf{Node}(t_{\ell-1})$, we use $\mathbf{T}_{\mathsf{Node}(t_{\ell-1})} \leftarrow \mathsf{ExtBasis}\left(\mathbf{T}_{\mathbf{A}_\ell}, [\mathbf{A}_\ell|\mathbf{A}_{\ell-1}^{(t_{\ell-1})}]\right)$ and further, if $\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2}$ is a secret associated with $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2$ then to derive a secret for $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h+1})_2$, we do the following: $\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h+1})_2} \leftarrow \mathsf{ExtBasis}\left(\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2}, \left[\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2|\mathbf{A}_{h-1}^{(t_{h-1})}\right]\right)$.

The same idea can be iteratively applied to get this trapdoor directly from the root $\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2} \leftarrow \mathsf{ExtBasis}\left(\mathbf{T}_{\mathbf{A}_\ell}, \left[\mathbf{A}_\ell|\mathbf{A}_{\ell-1}^{(t_{\ell-1})}|\cdots|\mathbf{A}_h^{(t_h)}\right]\right)$. The above implies that if there is a path connecting the root with a leaf, which contains $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_1})_2$ and $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_2})_2$, where $h_1 > h_2$, and if there is given $\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_1})_2}$, then we can efficiently compute $\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_2})_2}$, by using $\mathsf{ExtBasis}\left(\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_1})_2}, \left[\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h_1})_2|\mathbf{A}_{h_1-1}^{(t_{h_1-1})}|\cdots|\mathbf{A}_{h_2}^{(t_{h_2})}\right]\right)$.

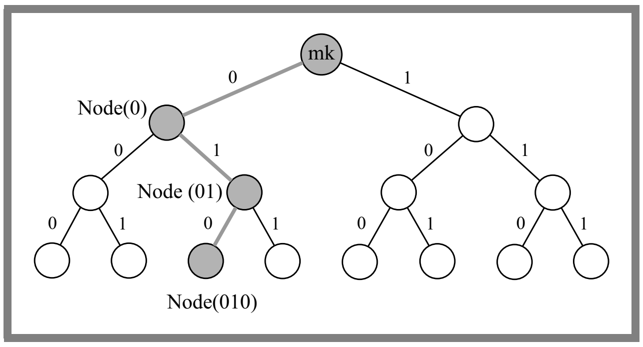

Fig. 1. Visualization of some basic ideas standing behind the scheme.

In addition, each secret key must consist of two components, one to make signatures, and one consisting of the minimal set of roots that allows to compute leaves associated with all future periods (see Fig. 2). In this proposal, these nodes are stored in a stack whose height equals ℓ at most, and the stack itself is filled in by using the $\mathsf{StackFilling}$ function defined by Algorithm 2 (see subsection 4.2).

4.1. Generation of System Parameters

Let n be a value of the security parameter. An efficient and polynomial-time system parameters generator algorithm $\mathscr{G}$, on input n and a binary tree height ℓ, outputs params $= (q, m, \eta_{min}, k, r, T, \mathsf{h}, \mathsf{H})$, where:

- $q = \mathsf{poly}(n)$, $q \geqslant 3$ is a prime,
- $m \geqslant \lceil 6n \lg q \rceil$,
- η_{min} is a required minimal entropy of the hash function h,
- $k \in \mathbb{Z}_{>0}$ and $r \in [k]_0$ are such that $2^{\eta_{min}} \leqslant \sum_{i=0}^{r} \binom{k}{i}$, as it guarantees that $H(\mathsf{h}) \geqslant \eta_{min}$,
- $\mathsf{h} : \mathbb{Z}_q^n \times \{0,1\}^n \to \mathbb{B}_{\mathsf{h}}^k = \{\mathbf{w} \in \mathbb{R}^k \mid w_i \in \{-1,0,1\}, \|\mathbf{w}\| \leqslant \sqrt{r}\}$ is a collision-resistant hash function,
- $\mathsf{H} : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^n$ is a hash function (not necessary satisfying any crystallographic requirements),
- $T = 2^\ell$ is the number of periods.

4.2. Key Generation

Before we describe the initial key's generation process, a very useful algorithm – StackFilling – is presented, allowing to obtain a new secret key along with a minimal set of trapdoors required to get keys for the consecutive periods. This set is denoted by Stack and, technically speaking, it is a stack that stores pairs $(\mathbf{T}, h)$, where h is the height of a trapdoor $\mathbf{T}$ on the binary tree. Note that h varies in the range of 0 to ℓ and, in particular, ℓ is the height of the root.

Algorithm 2. StackFilling

```
Input: $T$, Stack, $t = t_{\ell-1} \cdots t_0$
Output: Stack, scp
1. $(\mathbf{T}, h) \leftarrow$ Stack.pop()      ▷ $\mathbf{T} = \mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2}$; $h$ is the height of $\mathbf{T}$
2. $h \leftarrow h - 1$      ▷ After reindexing, $\mathbf{T} = \mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h+1})_2}$
3. While $h \geqslant 0$ then
4.     tmp $\leftarrow$ ExtBasis$\left(\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h+1})_2}, \left[\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h+1})_2 | \mathbf{A}_h^{(1)}\right]\right)$
           ▷ i.e. tmp $= \mathbf{T}_{\mathsf{Node}(t_{\ell-1}\cdots t_{h+1}1)_2}$
5.     Stack.push (tmp, $h$)
6.     $\mathbf{T} \leftarrow$ ExtBasis$\left(\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h+1})_2}, \left[\mathsf{Node}(t_{\ell\text{-}1}\cdots t_{h+1})_2 | \mathbf{A}_h^{(0)}\right]\right)$
7.     $h \leftarrow h - 1$
8. End While
9. scp $\leftarrow \mathbf{T}$
```

With this ingredient in place, the KGen algorithm could be described, with security parameter n acting as its input. With a maximum number of time periods T, it outputs an initial secret key sk_0 along with a long-term public key pk. Once a set of parameters params has been generated, then the formal definition of KGen is (Fig. 3):

1) Choose a matrix $\mathbf{U}$ uniformly at random from $\mathbb{Z}_q^{n\times k}$,
2) Choose matrices $\mathbf{A}_{\ell-1}^{(0)}, \mathbf{A}_{\ell-1}^{(1)}, \mathbf{A}_{\ell-2}^{(0)}, \mathbf{A}_{\ell-2}^{(1)}, \ldots \mathbf{A}_0^{(0)}, \mathbf{A}_0^{(1)}$ uniformly at random from $\mathbb{Z}_q^{n\times m}$,

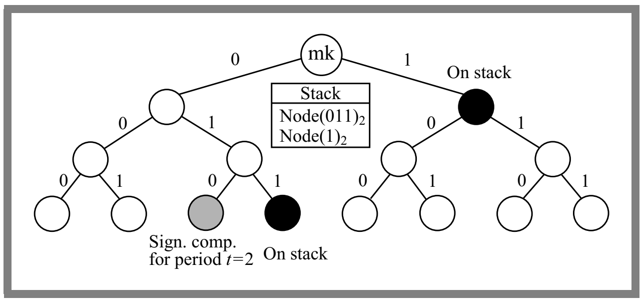

Fig. 2. Content of a secure key associated with a period t.

3) Launch the TrapGen(q, n) algorithm to get $(\mathbf{A}_\ell, \mathbf{T}_{\mathbf{A}_\ell})$, where $\mathbf{A}_\ell \in \mathbb{Z}_q^{n\times m}$ is a matrix and $\mathbf{T}_{\mathbf{A}_\ell} \in \mathbb{Z}_q^{m\times m}$ its trapdoor,
4) Fix a tiny $\varepsilon \in \mathbb{R}_{>0}$. Next, choose $s_0 \in \mathbb{R}_{>0}$ such that $s_0 \geqslant \|\mathbf{T}_{\mathbf{A}}^*\| \cdot (\lg(\ell+1)m)^{\frac{1}{2}+\varepsilon}$, and update params $\leftarrow$ (params, s_0); s_0 is a parameters for discrete Gaussian distribution,
5) Create an empty stack Stack, and next invoke Stack.push $(\mathbf{T}_{\mathbf{A}_\ell}, \ell)$ in order to initialize the stack with $(\mathbf{T}_{\mathbf{A}_\ell}, \ell)$,
6) Launch the StackFilling (params, Stack, $t = 0$) algorithm to be given (scp, Stack) (after making this step, Stack consists of ℓ elements),
7) KGen outputs the initial secret key $\mathsf{sk}_0 =$ (scp, Stack, $t = 0$) and the public key pk $\leftarrow$ $(\mathbf{A}_\ell, \mathbf{A}_{\ell-1}^{(0)}, \mathbf{A}_{\ell-1}^{(1)}, \mathbf{A}_{\ell-2}^{(0)}, \mathbf{A}_{\ell-2}^{(1)}, \ldots \mathbf{A}_0^{(0)}, \mathbf{A}_0^{(1)}, \mathbf{U})$.

4.3. Key Update

KUpd takes as an input the secret key sk_t associated with $t < T - 1$, and generates a secret key for the next period $t + 1$. This algorithm works as follows:

1) Parse $\mathsf{sk}_t =$ (scp, Stack, t),
2) Update $t \leftarrow t + 1$, after this step the variable t stores a value of the new time period,
3) If $t \equiv 1 \pmod 2$, then the following steps are conducted:
   3.1 $(\mathbf{T}, h) \leftarrow$ Stack.pop() and scp $\leftarrow \mathbf{T}$,
   3.2 The secret key for the new period is of the form $\mathsf{sk}_t =$ (scp, Stack, t),
4) If $t \equiv 0 \pmod 2$, then the following steps are performed:
   4.1 Run (scp, Stack) $\leftarrow$ StackFilling (params, Stack, t),

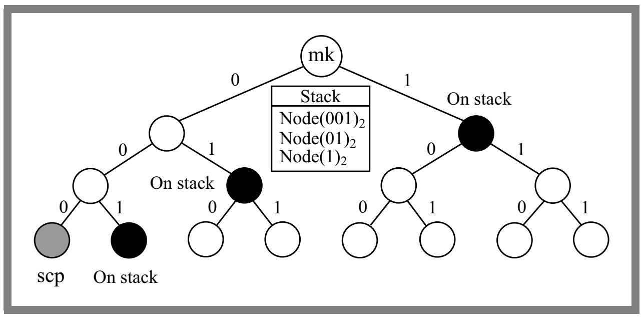

Fig. 3. The initial secret key $\mathsf{sk}_0 =$ (scp, Stack, $t = 0$) and output from StackFilling.

4.2 The secret key for the new period is of the form $\mathsf{sk}_t = (\mathsf{scp}, \textsc{Stack}, t)$.

4.4. Signing

The signing algorithm Sign uses, as input, the secret key sk_t associated with a period $t < T$ and message $\mathsf{msg} \in \mathcal{M}$, and outputs a value of the signature. As explained in the previous section, secret keys consist of three components, scp, Stack, and t. All of them plays a different role in the process of signing a message. Namely, Stack stores minimal amount of data required to update the secret key to the next period, while the signing component is needed only for making signatures within a current period.

To sign a message $\mathsf{msg} \in \mathcal{M}$ using sk_t, the signing algorithm Sign performs the following steps:

1) Construct matrix $\mathbf{A}_t = \left[\mathbf{A}_\ell | \mathbf{A}_{\ell-1}^{(t_{\ell-1})} | \mathbf{A}_{\ell-2}^{(t_{\ell-2})} | \cdots | \mathbf{A}_0^{(t_0)}\right] \in \mathbb{Z}_q^{n\times(\ell+1)m}$ associated with period t. Note that $\mathbf{A}_t = \mathsf{Leaf}(t)$, and $\mathsf{scp} \in \mathsf{sk}_t$ has the form of $\mathsf{scp} = \mathbf{T}_{\mathbf{A}_t}$,
2) Run k-SampleISIS, with its input being a tuple $(\mathbf{A}_t, \mathsf{scp}, s_0, \mathbf{U})$, in order to get a secret ephemeral key $\mathbf{E}_t \in \mathsf{D}^k_{(\ell+1)m} \subset \mathbb{Z}^{(\ell+1)m\times k}$, such that $\mathbf{A}_t \cdot \mathbf{E}_t = \mathbf{U} \pmod q$),
3) Set $\alpha_1, \alpha_2 \geqslant 12$, next choose
$$s_1 \geqslant \max\left\{\alpha_1\sqrt{r}, (\lg k)^{\frac{1}{2}+\varepsilon}\right\},$$
$$s_2 \geqslant \max\left\{\alpha_2 s_0(1+\alpha_1\sqrt{k})\sqrt{(\ell+1)mr},\ \left(\lg\left((\ell+1)m\right)\right)^{\frac{1}{2}+\varepsilon}\right\}.$$
4) Choose $\mathbf{b} \leftarrow \mathcal{D}^k_{s_1}$, $\mathbf{a} \leftarrow \mathcal{D}^{(\ell+1)m}_{s_2}$, and $\mathbf{r} \xleftarrow{\$} \{0,1\}^n$.
5) Compute $\mathbf{x}_1 \leftarrow \mathbf{A}_t\mathbf{a} + \mathbf{U}\mathbf{b} \pmod q$, $\mathbf{x}_2 \leftarrow \mathsf{H}(\mathsf{msg}, \mathbf{r})$, and derive $\boldsymbol{\sigma}_1 \leftarrow \mathsf{h}(\mathbf{x}_1, \mathbf{x}_2)$.
6) Set $\boldsymbol{\sigma}_1' \leftarrow \boldsymbol{\sigma}_1 + \mathbf{b}$ and go to the next step (i.e. output $\boldsymbol{\sigma}_1'$) with probability $\min\left(\frac{\mathcal{D}^k_{s_1}(\boldsymbol{\sigma}_1')}{M_1\mathcal{D}^k_{\boldsymbol{\sigma}_1,s_1}(\boldsymbol{\sigma}_1')}, 1\right)$; otherwise Sign is restarted.
7) Set $\boldsymbol{\sigma}_2 \leftarrow \mathbf{E}_t\boldsymbol{\sigma}_1' + \mathbf{a}$ and output $\boldsymbol{\sigma}_2$ with probability $\min\left(\frac{\mathcal{D}^{(\ell+1)m}_{s_2}(\boldsymbol{\sigma}_2)}{M_2\mathcal{D}^{(\ell+1)m}_{\mathbf{E}_t\boldsymbol{\sigma}_1',s_2}(\boldsymbol{\sigma}_2)}, 1\right)$; otherwise Sign is restarted.
8) If $\|\boldsymbol{\sigma}_2\| \leqslant s_2\sqrt{(\ell+1)m}$ then output $\sigma = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \mathbf{r})$.

Remark 26. The explanation why the estimates for s_1 and s_2 are as presented in point 3) is given in subsection 4.6.

4.5. Verification

The verification algorithm Vrfy takes as input t, pk, msg, and σ, and outputs one out of two values: accepted or rejected.

1) Construct the matrix $\mathbf{A}_t = \left[\mathbf{A}_\ell | \mathbf{A}_{\ell-1}^{(t_{\ell-1})} | \mathbf{A}_{\ell-2}^{(t_{\ell-2})} | \cdots \| \mathbf{A}_0^{(t_0)}\right] \in \mathbb{Z}_q^{n\times(\ell+1)m}$ associated with period t.
2) Set $\hat{\mathbf{x}}_1 \leftarrow \mathbf{A}_t\boldsymbol{\sigma}_2 - \mathbf{U}\boldsymbol{\sigma}_1 \pmod q$.
3) If $\boldsymbol{\sigma}_1 = \mathsf{h}\big(\hat{\mathbf{x}}_1, \mathsf{H}(\mathsf{msg}, \mathbf{d})\big)$ and $\|\boldsymbol{\sigma}_2\| \leqslant s_2\sqrt{(\ell+1)m}$, then output accepted , otherwise return rejected.

4.6. Correctness

In order to evaluate the correctness of the scheme, let us suppose that $\sigma = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \mathbf{r})$ is a signature of msg for a period t. Recall that $\boldsymbol{\sigma_1} = \mathsf{h}(\mathbf{x}_1, \mathbf{x}_2)$, where $\mathbf{x}_1 = \mathbf{A}_t\mathbf{a} + \mathbf{U}\mathbf{b} \pmod q$, $\mathbf{x}_2 = \mathsf{H}(\mathsf{msg}, \mathbf{d})$, and that $\boldsymbol{\sigma}_2 = \mathbf{E}_t\boldsymbol{\sigma}_1' + \mathbf{a} \pmod q$, where $\boldsymbol{\sigma}_1' = \boldsymbol{\sigma}_1 + \mathbf{b} \pmod q$. Then we have:

$$\mathbf{A}_t\boldsymbol{\sigma}_2 - \mathbf{U}\boldsymbol{\sigma}_1 = \mathbf{A}_t(\mathbf{E}_t\boldsymbol{\sigma}_1' + \mathbf{a}) = \mathbf{U}\mathbf{b} + \mathbf{A}_t\mathbf{a}.$$

This means that:

$$\begin{aligned}&\mathsf{h}\big(\mathbf{A}_t\boldsymbol{\sigma}_2 - \mathbf{U}\boldsymbol{\sigma}_1 \pmod q, \mathsf{H}(\mathsf{msg}, \mathbf{r})\big)\\ &= \mathsf{h}\big(\mathbf{A}_t\mathbf{a} + \mathbf{U}\mathbf{b} \pmod q, \mathsf{H}(\mathsf{msg}, \mathbf{r})\big)\\ &= \mathsf{h}(\mathbf{x}_1, \mathbf{x}_2) = \boldsymbol{\sigma}_1.\end{aligned}$$

Although vectors $\boldsymbol{\sigma}_1'$ and $\boldsymbol{\sigma}_2$ come from the distributions $\mathcal{D}^k_{\boldsymbol{\sigma}_1,s_1}$ and $\mathcal{D}^{(\ell+1)m}_{\mathbf{E}_t\boldsymbol{\sigma}_1',s_2}$, respectively, the target distributions for them that we will be aiming for are $\mathcal{D}^k_{s_1}$ and $\mathcal{D}^{(\ell+1)m}_{s_2}$. We will apply the rejection sampling, Theorem 9 shows that for an appropriately-chosen values of M and s (steps 6–7), of Sign will output values whose probabilities equal approximately $1/M$ and the statistical distance between the outputs is statistically close to the distribution in which $\boldsymbol{\sigma}_1'$ and $\boldsymbol{\sigma}_2$ are chosen form $\mathcal{D}^k_{s_1}$ and $\mathcal{D}^{(\ell+1)m}_{s_2}$, respectively.

Due to this fact and following Lemma 3, the flowing estimation $\|\boldsymbol{\sigma}_2\| \leqslant s_2\sqrt{(\ell+1)m}$ is obtained, with overwhelming probability.

Moreover, according to the assumption, $s_1, s_2 \geqslant \|\mathbf{T}^*_{\mathbf{A}}\| \cdot \omega\left(\sqrt{\log m}\right)$. Therefore, there are $\alpha_1, \alpha_2 \in \mathbb{R}_{>0}$, such that $s_1 = \alpha_1 \cdot \|\boldsymbol{\sigma}_1\|$ and $s_2 = \alpha_2 \cdot \|\mathbf{E}_t\boldsymbol{\sigma}_1'\|$. By Lemma 8 the following conditions hold:

$$\frac{\mathcal{D}^k_{s_1}(\mathbf{x})}{M_1\mathcal{D}^k_{\boldsymbol{\sigma}_1,s_1}(\mathbf{x})} < \frac{1}{M_1}\cdot e^{\frac{24\alpha_1+1}{2\alpha_1^2}}$$
$$\frac{\mathcal{D}^{(\ell+1)m}_{s_2}(\mathbf{x})}{M_2\mathcal{D}^{(\ell+1)m}_{\mathbf{E}_t\boldsymbol{\sigma}_1',s_2}(\mathbf{x})} < \frac{1}{M_2}\cdot e^{\frac{24\alpha_2+1}{2\alpha_2^2}},$$

with probabilities of at least $1 - 2^{-100}$. Due to the fact that Theorem 9 requires:

$$\frac{\mathcal{D}^k_{s_1}(\mathbf{x})}{M_1\mathcal{D}^k_{\boldsymbol{\sigma}_1,s_1}(\mathbf{x})} \leqslant 1 \quad\text{and}\quad \frac{\mathcal{D}^{(\ell+1)m}_{s_2}(\mathbf{x})}{M_2\mathcal{D}^{(\ell+1)m}_{\mathbf{E}_t\boldsymbol{\sigma}_1',s_2}(\mathbf{x})} \leqslant 1,$$

we conclude that:

$M_1 \geqslant e^{\frac{24\alpha_1+1}{2\alpha_1^2}}$ and $M_2 \geqslant e^{\frac{24\alpha_2+1}{2\alpha_2^2}}$.

It is easily seen that the optimal choice is:

$M_1 \approx \exp\left(\frac{24\alpha_1+1}{2\alpha_1^2}\right)$ and $M_2 \approx \exp\left(\frac{24\alpha_2+1}{2\alpha_2^2}\right)$.

It is instructive to justify the origin and accuracy of the estimates of Gaussian parameters s_1, s_2. Since the parameters s_1, s_2 are associated with probability distributions $\mathcal{D}^k_{s_1}$ and $\mathcal{D}^{(\ell+1)m}_{s_2}$, respectively, from Theorem 4, in order to use SampleD, the following conditions ought to be satisfied:

$$s_1 \geqslant \|\mathbb{1}_k\|(\lg k)^{\frac{1}{2}+\varepsilon} = (\lg k)^{\frac{1}{2}+\varepsilon}; \tag{1}$$

$$s_2 \geqslant \|\mathbb{1}_{(\ell+1)m}\|\big(\lg((\ell+1)m)\big)^{\frac{1}{2}+\varepsilon} = \big(\lg((\ell+1)m)\big)^{\frac{1}{2}+\varepsilon}. \tag{2}$$

The use of identity matrices $\mathbb{1}$ results from the fact that we consider lattices $\mathbb{Z}^k$ and $\mathbb{Z}^{(\ell+1)m}$ with their respective (short) canonical bases.

On the other hand, $\mathbf{b}$ and $\mathbf{E}_t$ are taken from distributions $\mathcal{D}^k_{s_1}$ and $\mathcal{D}^{k,\mathbf{U}}_{q,\mathbf{A}_t,s}$, respectively. Therefore, from Theorem 4, we get with overwhelming probability, that:

$$\|\mathbf{b}\| \leqslant s_1\sqrt{k} \quad \text{and} \quad \|\mathbf{E}_t\| \leqslant s_0\sqrt{(\ell+1)m}\|. \tag{3}$$

Furthermore:

$$\|\boldsymbol{\sigma}_1\| = \|\mathsf{h}(\mathbf{x}_1,\mathbf{x}_2)\| \leqslant \sqrt{r}\,. \tag{4}$$

The estimates (3) and (4) imply that:

$$\begin{aligned}\|\mathbf{E}_t\boldsymbol{\sigma}_1'\| &\leqslant \|\mathbf{E}_t\,\| \cdot (\|\boldsymbol{\sigma}_1\| + \|\mathbf{b}\|)\\ &\leqslant s_0\left(\sqrt{r} + s_1\sqrt{k}\right)\sqrt{(\ell+1)m}\,. \end{aligned}\tag{5}$$

According to (4) and Lemma 8, the following condition must hold:

$$\begin{aligned}s_1 \geqslant &\alpha_1 \cdot (\text{the best known upper bound of } \{\|\boldsymbol{\sigma}_1\|\})\\ =&\alpha_1\sqrt{r}\,.\end{aligned}\tag{6}$$

Similarly, conditions (5) – (6) and Lemma 8 imply:

$$\begin{aligned}s_2 &\geqslant \alpha_2 \cdot (\text{the best known upper bound of } \{\|\mathbf{E}_t\boldsymbol{\sigma}_1'\|\})\\ &= \alpha_2 s_0\left(\sqrt{r} + s_1\sqrt{k}\right)\sqrt{(\ell+1)m} \\ &\geqslant \alpha_2 s_0(1+\alpha_1\sqrt{k})\sqrt{(\ell+1)mr}\,.\end{aligned}\tag{7}$$

In consequence, by combining (1) with (6) and (2) with (7) the postulated estimates are obtained:

5. Security Proof

Here, we show that the considered key-evolving digital signature scheme is secure in terms of the forward-security model defined in Section 3. To this end, we prove that the ability to obtain a valid forgery leads us to the ability to construct a non-trivial solution to SIS problem. The proof itself exploits consequences of the general forking lemma and, due to this fact, it is conducted in the random oracle model.

Theorem 27. Let n be a value of a security parameter, $\ell \in \mathbb{Z}_{>0}$, and $\Pi = (\mathscr{G}, \mathsf{KGen}, \mathsf{KUpd}, \mathsf{Sign}, \mathsf{Vrfy})$ be a scheme considered in Section 4 with the associated message space $\mathcal{M} = \{0,1\}^*$. If $\mathcal{A}$ is a fu-cma-adversary attacking Π in the random oracle model, which makes at most q_{h} queries to the random oracle, then for $\beta = (2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}$, there exists a PPT algorithm $\mathcal{B}$ attacking $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ problem with advantage:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{SIS}}_n(\mathcal{B}) \geqslant &\frac{1}{q_{\mathsf{h}}\cdot T^2}\cdot\left(\mathbf{Adv}^{\text{fu-cma}}_{\Pi,n}(\mathcal{A}) - \frac{1}{\#(\mathbb{B}^k_{\mathsf{h}})}\right)\\ &\cdot\left(\mathbf{Adv}^{\text{fu-cma}}_{\Pi,n}(\mathcal{A}) - \frac{q_{\mathsf{h}}+1}{\#(\mathbb{B}^k_{\mathsf{h}})}\right)\cdot\left(\frac{1}{2} - \frac{1}{2^{101}}\right),\end{aligned}\tag{8}$$

and a running time of $\mathcal{O}(\text{time}(\mathcal{A}))$.

Proof. Assume that $\mathcal{A}$ is an adversary attacking Π, and suppose that parameters params of Π have been generated by $\mathscr{G}(1^n,\ell)$ as described in subsection 4.1. We will build an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ as a subroutine and which is aimed to attack $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ problem, where $\beta = (2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}$. To this end, let:

$$\mathbf{S} = \left[\mathbf{S}_\ell \mid \mathbf{S}^{(0)}_{\ell-1} \mid \mathbf{S}^{(1)}_{\ell-1} \mid \cdots \mid \mathbf{S}^{(0)}_0 \mid \mathbf{S}^{(1)}_0\right] \in \mathbb{Z}_q^{n\times(1+2\ell)m},$$

be a random matrix given to $\mathcal{B}$. According to the model and Definition 12, the challenge here is to find $\mathbf{x} \in \mathbb{Z}^{(1+2\ell)m}$ as:

$$\mathbf{S}\cdot\mathbf{x} \equiv \mathbf{0} \pmod q,\ \|\mathbf{x}\| \leqslant (2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}\,.$$

Setup. The simulator sets the previously generated params (see also subsection 4.1) as the parameters. After this step the adversary $\mathcal{B}$ chooses a time frame t^* according to the uniform distribution, i.e. $t^* \overset{\$}{\leftarrow} [T-1]$.

A collision-resistant hash function $\mathsf{h} \in \mathsf{params}$ is modeled as a random oracle and $\mathcal{A}$ is able to send at most $q := q_{\mathsf{h}}$ queries to this oracle. In order to appropriately simulate the oracle's random behavior, vectors $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{q_{\mathsf{h}}}$ are chosen uniformly at random from $\{\mathbf{w} \in \mathbb{R}^k \mid w_i \in \{-1,0,1\},\ \|\mathbf{w}\| \leqslant \sqrt{r}\}$. Next, there an ordered set $\mathcal{W}_{\mathsf{h}} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{q_{\mathsf{h}}}\}$, is designed, where the ordering relation "$\preccurlyeq$" is defined as follow $\mathbf{w}_i \preccurlyeq \mathbf{w}_j$ if $i < j$.

Let $t^* = (t^*_{\ell-1}, t^*_{\ell-2}, \dots, t^*_0)_2$ be a binary representation of t^*. $\mathcal{B}$ sets the public key, such that:

$$\mathsf{pk} \leftarrow \left(\mathbf{A}_\ell, \mathbf{A}^{(b=0)}_{\ell-1}, \mathbf{A}^{(b=1)}_{\ell-1}, \mathbf{A}^{(b=0)}_{\ell-2}, \mathbf{A}^{(b=1)}_{\ell-2}, \dots, \mathbf{A}^{(b=0)}_0, \mathbf{A}^{(b=1)}_0, \mathbf{U}\right),$$

in the manner shown below:

- $\mathbf{A}_\ell = \mathbf{S}_\ell$;

and for every $i \in [\ell-1]_0$:

- if $b = t^*_i$ then $\mathbf{A}^{(b)}_i = \mathbf{S}^{(t^*_i)}_i$,
- else, i.e. if $b \neq t^*_i$ then the algorithm $\mathsf{TrapGen}(q,n,m)$ is run to get $\mathbf{A}^{(b)}_i \in \mathbb{Z}_q^{n\times m}$ along with its trapdoor $\mathbf{T}_{\mathbf{A}^{(b)}_i} \in \mathbb{Z}^{m\times m}$.

Having done this, $\mathbf{B}$ puts

$T_{\max} := \max_{i\in[\ell-1]_0}\left\{\|\mathbf{T}^*_{\mathbf{A}^{(b)}_i}\| \mid b \neq t^*_i\right\}.$

Remark 28. Although matrices $\mathbf{A}_i$ is generated by $\mathsf{TrapGen}$ are not truly random, Theorem 11 shows that they are $(m\cdot q^{-n/6})$-uniform over $\mathbb{Z}_q^{n\times m}$. According to the assumed form of m (subsection 4.1), we get $m\cdot q^{-n/6} \to 0$ as $n \to 0$, and this convergence is very fast. Moreover, note that for any

$c \in \mathbb{Z}_{>0}$, we have $(n^c m) \cdot q^{-n/6} \to 0$ as $n \to 0$. Therefore, there exists a positive n_0, such that $(n^c m) \cdot q^{-n/6} < 1$ and equivalently $m \cdot q^{-n/6} < n^{-c}$, for $n > n_0$. This means that $m \cdot q^{-n/6} = \mathsf{negl}(n)$, i.e. the matrices $\mathbf{A}_i$ are chosen from a distribution, whose statistical distance to the being uniform is negligible.

Further, $\mathcal{B}$ chooses $\varepsilon \in \mathbb{R}_{>0}$. The best estimation is obtained when ε is a tiny number and puts $d = s_0\sqrt{(1+\ell)m}$, where a Gaussian distribution parameter s_0 is chosen in such a way, that the following two conditions hold:

– $s_0 \geqslant T_{\max} \cdot (\lg((\ell+1)m))^{\frac{1}{2}+\varepsilon}$ (see Lemma 7, Theorem 4 and, for further application, Theorem 22),

– $(\ell+1)m > 24 + (n \lg q)/\lg(2d+1)$ (see Lemma 17).

Further, it launches k-times the SampleD algorithm with the canonical basis $\{|i\rangle\}_{i=0}^{m-1}$, the Gaussian parameter s_0 and $\mathbf{c} = 0$ (see Theorem 4) being its input values in order to get $\mathbf{E}_* \leftarrow \mathcal{D}_{s_0}^{(1+\ell)m\times k}$. Note that:

$$\|\mathbf{E}_*\| = \max\{\|\mathbf{e}_{*,1}\|, \ldots, \|\mathbf{e}_{*,k}\|\} \leqslant d\,,$$

with overwhelming probability, by Lemma 3. After this step, $\mathcal{B}$ sets $\mathbf{U} = \mathbf{A}_{t^*} \cdot \mathbf{E}_*$, where $\mathbf{A}_{t^*} = [\mathbf{A}_\ell|\mathbf{A}_{\ell-1}^{(t^*_{\ell-1})}|\mathbf{A}_{\ell-2}^{(t^*_{\ell-2})}|\cdots|\mathbf{A}_0^{(t^*_0)}] \in \mathbb{Z}_q^{n\times(\ell+1)m}$.

Remark 29. Since $\mathbf{A}_{t^*}$ is a representative of a family of one-way PSFs (subsection 2.5) and $\mathbf{E}_* \leftarrow \mathcal{D}_s^{(1+\ell)m\times k}$, due to Lemma 7, the distribution of $\mathbf{U} = \mathbf{A}_{t^*} \cdot \mathbf{E}_*$ is statistically close to the uniform distribution over $\mathbb{Z}_q^{n\times k}$.

Finally, $\mathcal{B}$ updates $\mathsf{params} \leftarrow (\mathsf{params}, s_0)$ and sends params along with pk to the adversary $\mathcal{A}$.

h-Query. In this phase, adversary $\mathcal{A}$ makes hash queries. $\mathcal{B}$ prepares a hash list L to record all queries and responses as follows:

1) At the beginning, list L is empty,
2) Let $(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{Z}_q^n \times \{0,1\}^n$ be a k-th query to h:
   a) If $\mathcal{A}$ has already asked about $(\mathbf{x}_1, \mathbf{x}_2)$ then list L consists of a pair $\big((\mathbf{x}_1, \mathbf{x}_2), \mathsf{h}(\mathbf{x}_1, \mathbf{x}_2)\big)$ and, in this case, $\mathcal{B}$ outputs $\mathsf{h}(\mathbf{x}_1, \mathbf{x}_2)$,
   b) Otherwise, the first element $\mathbf{w} \in \mathcal{W}_\mathsf{h}$ which has not yet been used is taken (i.e. if $\mathbf{w}' \in \mathcal{W}_\mathsf{h}$ is such that $\mathbf{w}' \preccurlyeq \mathbf{w}$, then $\mathbf{w}'$ has been already used), a pair $((\mathbf{x}_1, \mathbf{x}_2), \mathbf{w})$ is appended to the list L and $\mathsf{h}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{w}$ is given as the output.

Queries. According to the security model given by $\mathbf{Exp}_{\mathcal{A},\Pi}^{\text{fu-cma}}$(subsection 3.2), adversary $\mathcal{A}$ has access to three oracles: signing oracle, key update oracle, and break in oracle.

Key update KUpd. Let $t \neq t^*$, then there exists at least one $i \in [\ell]$ such that $t_i \neq t_i^*$. Set $i_0 := \max\{i \in [\ell-1]_0 \mid t_i \neq t_i^*\}$. If only $i_0 \neq \ell$, then it is obvious that $t_i = t_i^*$ for every $i > i_0$. Since the adversary $\mathcal{B}$ knows $\mathbf{A}_{i_0}^{(t_{i_0})} \in \mathbb{Z}_q^{n\times m}$ and its associated trapdoor $\mathbf{T}_{\mathbf{A}_{i_0}^{(t_{i_0})}} \in \mathbb{Z}^{m\times m}$, it uses ExtBasis to get:

$$\mathbf{T}_{\mathbf{A}_t} \leftarrow \mathsf{ExtBasis}\Big(\mathbf{T}_{\mathbf{A}_{i_0}^{(t_{i_0})}}, \Big[\mathbf{A}_\ell|\mathbf{A}_{\ell-1}^{(t^*_{\ell-1})}|\mathbf{A}_{\ell-2}^{(t^*_{\ell-2})}|\cdots |\mathbf{A}_{i_0-1}^{(t^*_{i_0-1})}|\mathbf{A}_{i_0}^{(t_{i_0})}|\cdots|\mathbf{A}_0^{(t_0)}\Big]\Big)\,.$$

Signing Sign. In order to simulate a signature on a given message msg, $\mathcal{B}$ acts depending on the value of a time frame.

- If $t \neq t^*$, then $\mathcal{B}$ runs k-SampleISIS that takes as input a tuple $(\mathbf{T}_{\mathbf{A}_t}, \mathbf{A}_t, s_0, \mathbf{U})$ and outputs an ephemeral key $\mathbf{E}_t \in \mathsf{D}_{(\ell+1)m}^k$ from the distribution $\mathcal{D}_{q,\mathbf{A},s}^{k,\mathbf{U}}$. Having done this, a signature is affixed to msg as described in subsection 4.4,
- If $t = t^*$, then $\mathcal{B}$ assigns $\mathbf{E}_{t^*} \leftarrow \mathbf{E}_*$.

Remark 30. It must be emphasized that if $t = t^*$, then $\mathbf{E}_{t^*} \leftarrow \mathbf{E}_*$ is not generated by k-SampleISIS, which rises a query whether $\mathcal{B}$ properly simulates the signing algorithm of the scheme.

Fortunately, due to the use of rejection sampling, the distribution of $\boldsymbol{\sigma}_2$ is statistically close to $\mathcal{D}_{s_2}^{(\ell+1)m}$ and, in consequence, $\boldsymbol{\sigma}_2$ is independent of $\mathbf{E}_*$. Therefore, simulation conducted by $\mathcal{A}$, as for choosing ephemeral keys $\mathbf{E}$, is indistinguishable from any real instantiation of Sign.

Break in Break. If the adversary $\mathcal{A}$ runs the break-in oracle, the current time period $\bar{t}$ is saved and the adversary is given the proper secret key $\mathsf{sk}_{\bar{t}}$ (associated with $\bar{t}$). This key consists of two components, namely scp and Stack. In order to generate scp, $\mathcal{B}$ proceeds in the same way as in KUpd.

When it comes to the Stack component, $\mathcal{B}$ launches StackFillingID (Algorithm 3) which, based on an empty stackID stack and $\bar{t}$, returns (filled in stack) stackID consisting of the identifiers of nodes from Stack.

Algorithm 3. StackFillingID

```
Input: stackID, $t = (t_{\ell-1}\cdots t_0)_2$
Output: stackID
1: $(\mathsf{nodeID}, h) \leftarrow$ stackID.pop()   $\triangleright$ nodeID $= t_{\ell-1}\cdots t_h$;
2: $h \leftarrow h-1$   $\triangleright$ After reindexing, nodeID $= t_{\ell-1}\cdots t_{h+1}$
3: While $h \geqslant 0$ then
4:     tmp $\leftarrow t_{\ell-1}\cdots t_{h+1}1$   $\triangleright$ i.e. $t_h = 1$
5:     stackID.push((tmp, $h$))
6: End While
```

After that $\mathcal{B}$ is able to derive nodes corresponding to these identifiers. At first, it takes an identifier $(t_{\ell\text{-}1}\cdots t_h)_2 \in$ StackFillingID and indicates $i_0 := \max\{i \in [\ell-1]_0 \mid t_i \neq t_i^*\}$. Such i_0 exists, since there is not node in Stack that lies on the branch linking the root with $\mathsf{Leaf}(t^*)$. $\mathcal{B}$ knows a pair $\Big(\mathbf{A}_{i_0}^{(t_{i_0})}, \mathbf{T}_{\mathbf{A}_{i_0}^{(t_{i_0})}}\Big)$, hence it runs ExtBasis in order to obtain:

$$\mathbf{T}_{\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2} \leftarrow \mathsf{ExtBasis}\Big(\mathbf{T}_{\mathbf{A}_{i_0}^{(t_{i_0})}}, [\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2]\Big)\,,$$

where $\mathsf{Node}(t_{\ell\text{-}1}\cdots t_h)_2$ has the following form $\mathbf{A}_\ell|\mathbf{A}_{\ell-1}^{(t^*_{\ell-1})}|\cdots|\mathbf{A}_{i_0-1}^{(t^*_{i_0-1})}|\mathbf{A}_{i_0}^{(t_{i_0})}|\cdots|\mathbf{A}_h^{(t_h)}$.

Forgery. By Remarks 28, 29, and 30, $\mathcal{B}$ is statistically indistinguishable from a real challenger in the experiment $\mathbf{Exp}^{\mathsf{fu\text{-}cma}}_{\mathcal{A},\Pi}$, for the considered scheme denoted herein by Π. So for $\mathcal{A}$, the interaction with $\mathcal{B}$ is the same as while conducting the real experiment. Therefore, the adversary $\mathcal{A}$ eventually outputs a forgery for a time-frame $\widehat{t^*}$. In the case that $\widehat{t^*} \neq t^*$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ accepts the forgery, which is of the form $(t^*, \mathsf{msg}^*, \sigma^*)$, where $\sigma^* = (\boldsymbol{\sigma}_1^*, \boldsymbol{\sigma}_2^*, \mathbf{r}^*)$ and

- $\|\boldsymbol{\sigma}_2^*\| \leqslant s_2\sqrt{(\ell+1)m}$,
- $\boldsymbol{\sigma}_1^* = \mathsf{h}\left(\mathbf{A}_{t^*}\boldsymbol{\sigma}_2^* - \mathbf{U}\boldsymbol{\sigma}_1^* \pmod q), \mathsf{H}(\mathsf{msg}^*, \mathbf{r}^*)\right)$,
- $\mathbf{A}_{t^*} = [\mathbf{A}_\ell|\mathbf{A}_{\ell-1}^{(t^*_{\ell-1})}|\mathbf{A}_{\ell-2}^{(t^*_{\ell-2})}|\cdots|\mathbf{A}_0^{(t^*_0)}] \in \mathbb{Z}_q^{n\times(\ell+1)m}$.

Let $\mathbf{w}_i \in \mathcal{W}_\mathsf{h}$ be such that $\boldsymbol{\sigma}_1^* = \mathbf{w}_i$. Recall that answers to the successive h-queries are taken successively form $\mathcal{W}_\mathsf{h}$ and in accordance with the "$\preccurlyeq$" relation. This means that just before $\mathcal{A}$ sent the h-query for a value of $\boldsymbol{\sigma}_1^*$, all of the vectors $\mathbf{w}_j \preccurlyeq \mathbf{w}_i$ had already been used. In order to take advantage of the general forking lemma (Lemma 1), $\mathcal{B}$ picks vectors $\widehat{\mathbf{w}}_i, \widehat{\mathbf{w}}_{i+1}, \ldots, \widehat{\mathbf{w}}_{q_\mathsf{h}}$ independently and uniformly at random from $\mathbb{B}^k_\mathsf{h} = \{\mathbf{w} \in \mathbb{R}^k \mid w_i \in \{-1,0,1\},\ \|\mathbf{w}\| \leqslant \sqrt{r}\}$ and modifies $\mathcal{W}_\mathsf{h}$ in such a way that the vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{i-1}$ are kept, whereas $q-i+1$ of the remaining vectors are replaced by the newly-generated vectors $\widehat{\mathbf{w}}$'s.

After this update, the set of answers to h-queries is of the following form $\widehat{\mathcal{W}_\mathsf{h}} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}, \widehat{\mathbf{w}}_i, \widehat{\mathbf{w}}_{i+1}, \ldots, \widehat{\mathbf{w}}_{q_\mathsf{h}}\}$. Further, $\mathcal{B}$ runs the subroutine $\mathcal{A}$ again with the same parameters (params) and the same random tape ρ as in the first run, but it uses $\widehat{\mathcal{W}_\mathsf{h}}$ instead of $\mathcal{W}_\mathsf{h}$ to answer $\mathcal{A}$'s h-queries. By Lemma 1, $\mathcal{A}$ outputs a new forgery $\left(\widehat{t^*}, \widehat{\mathsf{msg}^*}, \widehat{\sigma^*} = (\widehat{\boldsymbol{\sigma}_1^*}, \widehat{\boldsymbol{\sigma}_2^*}, \widehat{\mathbf{r}^*})\right)$ using the same h-queries. If $\widehat{t^*} \neq t^*$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ accepts the forgery and then this means that $\|\widehat{\boldsymbol{\sigma}_2^*}\| \leqslant s_2\sqrt{(\ell+1)m}$ and $\widehat{\mathbf{w}}_i = \widehat{\boldsymbol{\sigma}_1^*} = \mathsf{h}\left(\mathbf{A}_{t^*}\widehat{\boldsymbol{\sigma}_2^*} - \mathbf{U}\widehat{\boldsymbol{\sigma}_1^*} \pmod q), \mathsf{H}(\widehat{\mathsf{msg}^*}, \widehat{\mathbf{r}^*})\right)$. In addition, the probability P_1 that satisfies $\widehat{\mathbf{w}}_i \neq \mathbf{w}_i$ is:

$$P_1 \geqslant \left(\tau - \frac{1}{\#(\mathbb{B}^k_\mathsf{h})}\right)\cdot\left(\frac{\tau - 1/\#(\mathbb{B}^k_\mathsf{h})}{q_\mathsf{h}} - \frac{1}{\#(\mathbb{B}^k_\mathsf{h})}\right), \quad (9)$$

where $\tau = \mathbf{Adv}^{\mathsf{fu\text{-}cma}}_{\Pi,n}(\mathcal{A})$.

Before asking the i-th h-query, $\mathcal{B}$ uses the same inputs, random tape ρ and $\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}$ to generate $\mathcal{A}$'s inputs, random tape and responses to h-queries. This implies that the two executions of $\mathcal{A}$ are identical up to the i-th h-query which, in turn means that the arguments of both i-th h-queries must be the same. Therefore:

$$\mathbf{A}_{t^*}\boldsymbol{\sigma}_2^* - \mathbf{U}\boldsymbol{\sigma}_1^* \equiv \mathbf{A}_{t^*}\widehat{\boldsymbol{\sigma}_2^*} - \mathbf{U}\widehat{\boldsymbol{\sigma}_1^*} \pmod q \quad \text{and}$$
$$\mathsf{H}(\mathsf{msg}^*, \mathbf{r}^*) = \mathsf{H}(\widehat{\mathsf{msg}^*}, \widehat{\mathbf{r}^*}) .$$

This implies that:

$$\begin{aligned} 0 &\equiv \mathbf{A}_{t^*}(\boldsymbol{\sigma}_2^* - \widehat{\boldsymbol{\sigma}_2^*}) - \mathbf{U}(\boldsymbol{\sigma}_1^* - \widehat{\boldsymbol{\sigma}_1^*}) \\ &\equiv \mathbf{A}_{t^*}\left(\boldsymbol{\sigma}_2^* - \widehat{\boldsymbol{\sigma}_2^*} - \mathbf{E}_*(\boldsymbol{\sigma}_1^* - \widehat{\boldsymbol{\sigma}_1^*})\right) \pmod q . \end{aligned} \quad (10)$$

Now, $\mathcal{B}$ sets $\mathbf{x}_0 = \boldsymbol{\sigma}_2^* - \widehat{\boldsymbol{\sigma}_2^*} - \mathbf{E}_*(\boldsymbol{\sigma}_1^* - \widehat{\boldsymbol{\sigma}_1^*})$, then $\|\mathbf{x}_0\| \leqslant (2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}$. Therefore by (10), if $\mathbf{x}_0 \neq \mathbf{0}$ then it is a solution of the following $\mathsf{SIS}_{q,n,(1+\ell)m,\beta}$ problem

$$\mathbf{A}_{t^*}\mathbf{x}_0 \equiv \mathbf{0} \pmod q \quad \text{with } \|\mathbf{x}_0\| \leqslant \beta .$$

This means that the probability that $\boldsymbol{\sigma}_2^* - \widehat{\boldsymbol{\sigma}_2^*} - \mathbf{E}_*(\boldsymbol{\sigma}_1^* - \widehat{\boldsymbol{\sigma}_1^*}) \neq \mathbf{0}$ still remains to be estimated. To this end, let $j_0 = \min\left\{j \in [k] \mid \sigma^*_{1,j} \neq \widehat{\sigma^*_{1,j}}\right\}$, hence the existence of this index is a consequence of the condition $\boldsymbol{\sigma}_1^* = \mathbf{w}_i \neq \widehat{\mathbf{w}}_i = \widehat{\boldsymbol{\sigma}_1^*}$. Let $\mathbf{e} = \mathbf{E}_*[j_0]$. Then, since $d \geqslant 9$ and $(\ell+1)m > 24 + (n\lg q)/\lg(2d+1)$, Lemma 17 concludes that the probability of existence of another $\mathbf{e}' \in (-[d] \cup [d]_0)^m$ such that $\mathbf{e}' \neq \mathbf{e}$ and $\mathbf{A}_{t^*}\mathbf{e}' = \mathbf{A}_{t^*}\mathbf{e}$ is at least $1 - 2^{-101}$. Having done this, matrix $\mathbf{E}'_*$ may be created, such that all its columns, except for the column j_0, are the same as $\mathbf{E}_*$, i.e. $\mathbf{E}'_* = \{\mathbf{e}'_j\}$, where $\mathbf{e}'_j = \mathbf{E}_*[j]$, for $j \neq j_0$ and $\mathbf{e}'_{j_0} = \mathbf{e}'$. Now, one may see that with this definition of $\mathbf{E}'_*$ if:

$$\boldsymbol{\sigma}_2^* - \widehat{\boldsymbol{\sigma}_2^*} - \mathbf{E}_*(\boldsymbol{\sigma}_1^* - \widehat{\boldsymbol{\sigma}_1^*}) = \mathbf{0} , \quad (11)$$

then

$$\boldsymbol{\sigma}_2^* - \widehat{\boldsymbol{\sigma}_2^*} - \mathbf{E}'_*(\boldsymbol{\sigma}_1^* - \widehat{\boldsymbol{\sigma}_1^*}) \neq \mathbf{0} . \quad (12)$$

In other words, we have showed that for every matrix $\mathbf{E}_*$ satisfying (11), with probability at least $1 - 2^{-101}$, it holds that there exists an another matrix $\mathbf{E}'_*$ which differs form $\mathbf{E}_*$ only in column j_0, and such that (12) is satisfied and that $\mathbf{A}_{t^*}\mathbf{E}_* = \mathbf{A}_{t^*}\mathbf{E}'_*$. These, in turn, mean that the likelihood of choosing between $\mathbf{E}_*$ and $\mathbf{E}'_*$ is at least $1/2$. Obviously, both these matrices are statistically indistinguishable to adversary $\mathcal{A}$.

To recapitulate, the probability P_2 that $\mathbf{x}_0 \neq \mathbf{0}$ satisfies the following condition:

$$P_2 \geqslant \frac{1}{2} - \frac{1}{2^{101}} . \quad (13)$$

Having obtained $\mathbf{x}_0$, $\mathcal{B}$ creates a vector:

$$\mathbf{x} = \left[x_{(1+2\ell)m-1}, \ldots, x_0\right] \in \mathbb{Z}_q^{(1+2\ell)m},$$

in the following way:

$$x_j = \begin{cases} x_{0,j} & \text{for } 2\ell m \leqslant j < (1+2\ell)m, \\ x_{0,j} & \text{for } j < 2\ell m \text{ and } \lfloor \frac{j}{m} \rfloor \pmod 2 = 1 - t^*_{\lfloor \frac{j}{2m} \rfloor}, \\ 0 & \text{elsewhere.} \end{cases}$$

Eventually, $\mathcal{B}$ outputs vector $\mathbf{x}$. Note that:

$$\mathbf{S}\cdot\mathbf{x} = \mathbf{A}_{t^*}\mathbf{x}_0 \equiv \mathbf{0} \pmod q ,$$

where $\|\mathbf{x}_0\| \leqslant \beta = (2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}$. This leads to the conclusion that the probability of getting a solution to $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ is the same as the probability of an event that $\mathbf{x} \neq \mathbf{0}$.

From inequalities (9), (13) and the fact that the probability of $\mathcal{B}$ not aborting is exactly equal to $1/T^2$, we have:

$$\Pr\left[\mathbf{x} \neq \mathbf{0}\right] \geqslant \frac{1}{T^2} \cdot \left(\frac{\tau - 1/\#(\mathbb{B}_{\mathsf{h}}^k)}{q_{\mathsf{h}}} - \frac{1}{\#(\mathbb{B}_{\mathsf{h}}^k)} \right) \cdot \left(\tau - \frac{1}{\#(\mathbb{B}_{\mathsf{h}}^k)} \right) \cdot \left(\frac{1}{2} - \frac{1}{2^{101}} \right),$$

where $\tau = \mathbf{Adv}_{\Pi,n}^{\mathsf{fu\text{-}cma}}(\mathcal{A})$. This finishes the proof. $\square$

Remark 31. It is easy to see that the left-hand side of the inequality (8) can be approximated by $\left(\mathbf{Adv}_{\Pi,n}^{\mathsf{fu\text{-}cma}}(\mathcal{A})\right)^2 / 2q_{\mathsf{h}}T^2$. This shows that the wording of Theorem 27 can be reformulated in such a way that if $\mathcal{A}$ is a fu-cma-adversary attacking Π in the random oracle model, which makes at most q_{h} queries to the random oracle, then there exists a PPT algorithm $\mathcal{B}$ attacking $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ problem with the following advantage:

$$\mathbf{Adv}_n^{\mathsf{SIS}}(\mathcal{B}) \approx \frac{1}{2q_{\mathsf{h}}T^2} \cdot \left(\mathbf{Adv}_{\Pi,n}^{\mathsf{fu\text{-}cma}}(\mathcal{A})\right)^2.$$

6. Parameters

In Table 1 summarizes the parameters of the proposed forward-secure digital signature scheme. The three independent values n, k, q need to be chosen in such a way to guarantee that the SIS problem is computationally infeasible. The basic idea of solving SIS problem is to define a random q-ary lattice $\mathcal{L}_q^{\perp}(\mathbf{A})$ and use lattice reduction algorithms to find short vectors in this lattice. Paper [21] shows that the length of the vector obtained by running the best known algorithms on a random m-dimensional q-ary lattice $\mathcal{L}_q^{\perp}(\mathbf{A})$ is close to $\min\left\{q, \left(\det\left(\mathcal{L}_q^{\perp}(\mathbf{A})\right)^{1/m} \cdot \delta^m\right)\right\} = \min\left\{q, q^{n/m} \cdot \delta^m\right\}$, where the equality holds with overwhelming probability. Parameter δ, called a Hermite factor, depends on the quality of the lattice-reduction algorithm being used. It is conjectured in [22] that $\delta = 1.007$ may be outside our reach for the foreseeable future.

It is worth noticing that although article [22] was published in 2011, the research still seems to be valid.

Based on the results from [22], Micciancio and Regev observed, in [23], that since $2^{2\cdot\sqrt{n \lg q \lg \delta}}$ is the minimum of a function $m \mapsto q^{n/m} \cdot \delta^m$ for $m = \sqrt{n \lg q / \lg \delta}$, lattice reduction algorithms can output the shortest vectors of $\mathcal{L}_q^{\perp}(\mathbf{A})$ when $m \approx \sqrt{n \lg q / \lg \delta}$. For lower m, the lattice is too sparse and does not contain short enough vectors. For larger m values, the high dimension prevents lattice reduction algorithms from finding short vectors.

To sum up, the authors concluded that the length of the shortest vector one can find in $\mathcal{L}_q^{\perp}(\mathbf{A})$ for a random $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ using lattice reduction algorithms is of length at least $\min\left\{q, 2^{2\cdot\sqrt{n \lg q \lg \delta}}\right\}$.

We wish to emphasize that the s_0 depends on $\mathbf{T}^*_{\mathbf{A}_\ell}$ and it is chosen after getting $(\mathbf{A}_\ell, \mathbf{T}_{\mathbf{A}_\ell}) \leftarrow \mathsf{TrapGen}(q,n)$. As far as parameters s_1 and s_2 are concerned, two options are available.

Tab. 1. Parameters of the proposed forward secure signature scheme.

Param.	Definition
n	Security parameter value
q	Prime number greater than or equal to 3
ℓ	Number of levels in a binary tree, $\ell \in \mathbb{Z}_{>0}$
T	Number of periods (leaves in a binary tree), $T = 2^\ell$
η_{min}	Min. entropy of the hash function (random oracle) h
k, r	$k \in \mathbb{Z}_{>0}, r \in [k]_0, \sum_{i=0}^{r} \binom{k}{i} \geqslant 2^{\eta_{min}}$
α_1, α_2	Real numbers greater than or equal to 12
$T_{\max}$	$\max_{i\in[\ell-1]_0}\{\lVert\mathbf{T}^*_{\mathbf{A}_i^{(b)}}\rVert \mid b \neq t_i^*\}$
ε	Tiny positive real number
s_0	At least $T_{\max} \cdot (\lg((\ell+1)m))^{\frac{1}{2}+\varepsilon}$
s_1	At least $\max\left\{\alpha_1\sqrt{r}, (\lg k)^{\frac{1}{2}+\varepsilon}\right\}$
s_2	At least $\max\left\{\alpha_2 s_0(1+\alpha_1\sqrt{k})\sqrt{(\ell+1)mr} \cdot (\lg((\ell+1)m))^{\frac{1}{2}+\varepsilon}\right\}$
d	$s_0\sqrt{(1+\ell)m}$
m	$\max\left\{\lceil 6n\lg q\rceil, \left\lceil \frac{1}{\ell+1} \cdot \left(24 + \frac{(n \lg q)}{\lg(2d+1)}\right)\right\rceil\right\}$
β	$(2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}$
M_1, M_2	$M_1 = M_2 = \exp\left(\frac{24\alpha_1+1}{2\alpha_1^2}\right)$

They may either be set together with s_0, or they may play the role of one-time parameters and be renewed while creating a new signature. The same remark applies to a parameter d which depends on s_0.

This can be a bit confusing, since due to the definition of m, a value of d ought to be known in advance, before setting the value of m. To be more precise, in addition to the primary assumption concerning m, according to which it must be at least $\lceil 6n \lg q\rceil$, Lemma 17 enforces $m > \left\lceil \frac{1}{\ell+1} \cdot \left(24 + \frac{(n \lg q)}{\lg(2d+1)}\right)\right\rceil$.

However, the left-hand side of the latter condition can exceed the value of $\lceil 6n \lg q\rceil$ only for very small n, q and d, meaning that, in real instances of the scheme, we always have that $\max\left\{\lceil 6n\lg q\rceil, \left\lceil \frac{1}{\ell+1} \cdot \left(24 + \frac{(n \lg q)}{\lg(2d+1)}\right)\right\rceil\right\} = \lceil 6n \lg q\rceil$. Therefore, the dependence of m on d is only of theoretical importance and it can be neglected while determining a set of parameters.

References

[1] M. Jurkiewicz, "Binary Tree Based Forward Secure Signature Scheme in the Random Oracle Model", *International Journal of Electronics and Telecommunications*, vol. 67, no. 4, pp. 717–726, 2021 (https://doi.org/10.24425/ijet.2021.137868).

[2] M. Bellare and S.K. Miner, "A Forward-Secure Digital Signature Scheme", *Advances in Cryptology – CRYPTO '99*, vol. 1666, pp. 431–448, 1999 (https://doi.org/10.1007/3-540-48405-1_28).

[3] J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS – A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions", *Post-Quantum Cryptography*, vol. 7071, pp. 117–129, 2010 (https://doi.org/10.1007/978-3-642-25405-5_8).

[4] L. Ducas *et al.*, "Crystals-dilithium: A Lattice-based Digital Signature Scheme", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, pp. 238–268, 2018 (https://doi.org/10.13154/tches.v2018.i1.238-268).

[5] V. Lyubashevsky, "Lattice Signatures without Trapdoors", *Advances in Cryptology – EUROCRYPT 2012*, pp. 738–755, 2012 (https://doi.org/10.1007/978-3-642-29011-4_43).

[6] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai Trees, or How to Delegate a Lattice Basis", *Advances in Cryptology – EUROCRYPT 2010,* vol. 6110, pp. 601–639, 2012 (https://doi.org/10.1007/978-3-642-13190-5_27).

[7] P. Zhang *et al.*, "A New Post-Quantum Blind Signature from Lattice Assumptions", *IEEE Access*, vol. 6, pp. 27251–27258, 2018 (https://doi.org/10.1109/ACCESS.2018.2833103).

[8] O. Goldreich, *Foundations of Cryptography: Volume 1, Basic Tools*, Cambridge University Press, 396 p., 2003 (ISBN: 9780521035361).

[9] D. Pointcheval and J. Stern, "Provably Secure Blind Signature Schemes", *Advances in Cryptology – ASIACRYPT '96,* pp. 252–265, 1996 (https://doi.org/10.1007/BFb0034852).

[10] M. Bellare and G. Neven, "Multi-signatures in the Plain Public-key Model and a General Forking Lemma", *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 390–399, 2006 (https://doi.org/10.1145/1180405.1180453).

[11] C. Peikert, *A Decade of Lattice Cryptography*, Now Foundations and Trends, 156 p., 2015 (https://doi.org/10.1561/0400000074).

[12] D. Micciancio and S. Goldwasser, *Complexity of Lattice Problems. A Cryptographic Perspective*, Springer Science & Business Media, 220 p., 2002 (https://doi.org/10.1007/978-1-4615-0897-7).

[13] D. Micciancio and O. Regev, "Worst-Case to Average-Case Reductions Based on Gaussian Measures", *45th Annual IEEE Symposium on Foundations of Computer Science*, Rome, Italy, 2004 (https://doi.org/10.1109/FOCS.2004.72).

[14] O. Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography", *Journal of the ACM*, vol. 56, no. 6, pp. 1–40, 2009 (https://doi.org/10.1145/1568318.1568324).

[15] C. Peikert and A. Rosen, "Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices", *Theory of Cryptography*, vol. 3876, pp. 145–166, 2006 (https://doi.org/10.1007/11681878_8).

[16] C. Peikert, "An Efficient and Parallel Gaussian Sampler for Lattices", *Advances in Cryptology – CRYPT 2010,* vol. 6223, pp. 80–97, 2010 (https://doi.org/10.1007/978-3-642-14623-7_5).

[17] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for Hard Lattices and New Cryptographic Constructions", *Proceedings of the fortieth Annual ACM Symposium on Theory of Computing*, pp. 197–206, 2008 (https://doi.org/10.1145/1374376.1374407).

[18] M. Ajtai, "Generating Hard Instances of Lattice Problems", *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 99–108, 1996 (https://doi.org/10.1145/237814.237838).

[19] J. Alwen and C. Peikert, "Generating Shorter Bases for Hard Random Lattices", *Theory of Computing Systems,* vol. 48, no. 3, pp. 535–553, 2011 (https://doi.org/10.1007/s00224-010-9278-3).

[20] S. Agrawal, D. Boneh, and X. Boyen, "Efficient Lattice (H)IBE in the Standard Model", *Advances in Cryptology – EUROCRYPT 2010*, vol. 6110, pp. 553–572, 2010 (https://doi.org/10.1007/978-3-642-13190-5_28).

[21] N. Gama and P.Q. Nguyen, "Predicting Lattice Reduction", *Advances in Cryptology – EUROCRYPT 2008*, vol. 4965, pp. 31–51, 2008 (https://doi.org/10.1007/978-3-540-78967-3_3).

[22] Y. Chen and P.Q. Nguyen, "BKZ 2.0: Better Lattice Security Estimates", *Advances in Cryptology – ASIACRYPT 2011*, pp. 1–20, 2011 (https://doi.org/10.1007/978-3-642-25385-0_1).

[23] D. Micciancio and O. Regev, "Lattice-based Cryptography", in: *Post-Quantum Cryptography*, ed. D.J. Bernstein, J. Buchmann, E. Dahmen, pp. 147–191, Springer, 2009 (https://doi.org/10.1007/978-3-540-88702-7_5).

Mariusz Jurkiewicz, Ph.D., Assistant Professor
Cybernetics Faculty
https://orcid.org/0000-0002-6314-4381
E-mail: mariusz.jurkiewicz@wat.edu.pl
Military University of Technology, Warsaw, Poland
https://www.wojsko-polskie.pl/wat