INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 1, PP. 61–68
Manuscript received December 16, 2024; revised March, 2025. doi: 10.24425/ijet.2025.153545

Quantum-safe forward secure password authenticated key life-cycle management scheme with key update mechanism

Mariusz Jurkiewicz

***Abstract*—In this paper we construct and consider a new password authenticated key life-cycle management scheme (PAKMS) with key update mechanism, which uses random q-ary lattices as its domain. We justify that the scheme is existentially forward unforgeable under a chosen password attack (fu-cpwda). To this end, we show that breaking this scheme let us to construct a polynomial-time adversary that is able to solve small integer solution (SIS) problem. Since the security of the scheme is based on computational hardness of SIS problem, it tuns out to be resistant to both classical and quantum computations. The key-updating mechanism is based on some properties of binary trees, with a number of leaves being the same as a number of time periods in the scheme. The forward-security is gained under the assumption that one out of two hash functions is modeled as a random oracle.**

***Keywords*—Forward security; q-ary lattices password authentication; random-oracle model; SIS problem**

I. Introduction

NOWADAYS, many services are available in clouds, allowing users to leverage powerful computing resources without having to purchase or maintain hardware and software. However, this brings some new challenges concerning, in particular, secure access to resources and protection against a data breach. A key to this seams to be to provide trustworthy methods for storing and processing user's passwords. In early days of computers' era passwords were stored in non-encrypted form as tuples $(\mathsf{login}, \mathsf{password})$. However, back in the 60s, it was noticed that this method is not secure and must be avoided. Currently, the most common methods is to store passwords i encrypted form, usually taking advantage of cryptographic hash functions along with a random string called salt. In this paper, we design and analyze a new password-authenticated scheme that, in addition, provide a decent level of security against quantum computing. The proposed scheme allows to handle many services available in the domain, where after registration and setting a pair $(\mathsf{login}, \mathsf{password})$, a user has access to the chosen number of them. It must be highlighted, that unlike the other indicated practices, neither password nor login are stored anywhere. This is made possible by using the concepts of asymmetric cryptography. After verification of credentials and log-in to a specific service, a user is granted access to the resources. These resources are stored as encrypted data, and the authentication unlock a secret (symmetric) key, which is also associated with verification parameters. In addition, the important part of the scheme is a self-acting update mechanism, used to periodic change of secret key-material. The idea is similar to so called UE schemes, that recently are gaining increasing interest in the crypto-community [1], [2], [3]. However, the model of security is ideologically close to $\mathsf{eu\text{-}cma}$.

The domain for the presented scheme is the lattice theory which seams to be the most promising post-quantum substrate for the modern asymmetric-crypto primitives. This is all the more true given that the famous mishaps like breaking SIDH and Rainbow. In order to design the scheme, we use q-ary lattices that provide a lot of flexibility in obtaining required statistical properties. We focus on the important question of forward-security of the scheme [4], [5]. The construction is partially based on the so-called Fiat-Shamir with aborts approach [6]–[9], which was proposed by Lyubashevsky [6] and refers to the idea from statistics called rejection sampling.

M. Jurkiewicz is with Faculty of Cybernetics, Military University of Technology, Warsaw, Poland (e-mail: mariusz.jurkiewicz@wat.edu.pl).

II. Preliminaries

A. Notation

If $k \in \mathbb{Z}_{>0}$, then we use the following notation: $[k] = \{1, \dots k\}$ and $[k]_0 = \{0, 1, \dots k\}$. The norms ℓ_2 and ℓ_∞ are denoted by $\|\cdot\|$ and $\|\cdot\|_\infty$, respectively. Vectors are in column form and are denoted by bold lower case letters (e.g., $\mathbf{x}$). We view a matrix as the set of its column vectors and denote by bold capital letters (e.g., $\mathbf{A}$). The ith coefficient of a vector $\mathbf{x}$ is denoted x_i, whereas the jth column of $\mathbf{A}$ is denoted by $\mathbf{A}[j]$. The norm of a matrix $\mathbf{A}$ is defined as follows: $\|\mathbf{A}\| = \max_j \|\mathbf{A}[j]\|$. 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 $\mathbf{B}$.

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 $\Delta(X_n, Y_n) \leq \mathsf{negl}(n)$, where $\Delta(X_n, Y_n)$ is called the *statistical distance* between $\{X_n\}_{n\in I}$ and $\{Y_n\}_{n\in I}$ and is defined as the function $\Delta(X_n, Y_n) = \frac{1}{2}\sum_{r\in\mathcal{R}_n} |\Pr[X_n = r] - \Pr[Y_n = r]|$. We refer to [10] for more details.

B. Lattices

For a set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \subset \mathbb{R}^m$, a m-dimensional *lattice* is defined as a set of all integer linear combinations of the vectors from $\mathbf{B}$, e.i. $\mathcal{L} = \mathcal{L}(\mathbf{B}) = \left\{\sum_{i=1}^{k} \alpha_i \mathbf{b}_i \mid \alpha_i \in \mathbb{Z}\right\}$. The set $\mathbf{B}$ is called a *basis* of $\mathcal{L}$, and $k = \#\mathbf{B}$ is called the *rank* of $\mathcal{L}$. If $k = m$ then we say that $\mathcal{L}$ is a *full-rank* lattice. All this means in particular that every lattice is a $\mathbb{Z}$-module. Furthermore, it is easily seen that a lattice $\mathcal{L}$ is is an additive subgroup of $\mathbb{R}^m$ and therefore, it induces the quotient group $\mathbb{R}^m / \mathcal{L}$ of cosets $\{\mathbf{x} + \mathcal{L}\}_{\mathbf{x} \in \mathbb{R}^m}$, with respect to the addition operation $(\mathbf{x} + \mathcal{L}) + (\mathbf{y} + \mathcal{L}) = (\mathbf{x} + \mathbf{y}) + \mathcal{L}$. 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 $\mathbf{x} + \mathcal{L}$. It turns out that the measure of fundamental domain is an invariant of the lattice and, therefore, it is called the *determinant* of $\mathcal{L}$ and denoted by $\det \mathcal{L}$.

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 $q \in \mathbb{Z}_{\geq 1}$. By definition, a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ is an integer lattice iff $\mathbf{B} \in \mathbb{Z}^{m \times m}$ is an integer square matrix, which implies that the determinant $\det \mathcal{L}$ is a positive integer. In addition, $\mathbb{Z}^m / \mathcal{L}$ is a finite group and $|\mathbb{Z}^m / \mathcal{L}| = \det \mathcal{L}$.

Let $n, m, q \in \mathbb{Z}_{q \geq 1}$, $n < m$, and $\mathbf{A} \in \mathbb{Z}q^{n \times m}$ be a full-rank matrix. In this paper we make use of a special kind of q-ary lattices of the full rank m, that are defined as follows: $\mathcal{L}_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} \equiv 0 \pmod q\}$. A matrix $\mathbf{A}$, generating $\mathcal{L} = \mathcal{L}_q^{\perp}(\mathbf{A})$, is often called a *parity matrix*, however, it must be pointed out that $\mathbf{A}$ is not a base of $\mathcal{L}$. Particularly, if q is a prime, then $|\det \mathcal{L}_q^{\perp}(\mathbf{A})| = |\mathbb{Z}^m / \mathcal{L}_q^{\perp}(\mathbf{A})| = q^n$. The lattice $\mathcal{L}_q^{\perp}(\mathbf{A})$ is closely related with another structure that, for a fixed $\mathbf{u} \in \mathbb{Z}_q^n$, is defined by $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{x} \equiv \mathbf{u} \pmod q\}$. Although $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A})$ is usually called a q-ary lattice, this is not formally correct since it does not contain $\mathbf{0}$ for $\mathbf{v} \neq \mathbf{0}$. Since $\mathcal{L}_q^{\perp}(\mathbf{A})$ is a $\mathbb{Z}$-module, it is natural to call $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A})$ an *affine lattice over* $\mathcal{L}_q^{\perp}(\mathbf{A})$. Note that if $\mathbf{v} \in \mathcal{L}_q^{\mathbf{u}}(\mathbf{A})$, then $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A})$ and $\mathcal{L}_q^{\perp}(\mathbf{A})$ are connected by the relation $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}) = \mathbf{v} + \mathcal{L}_q^{\perp}(\mathbf{A})$.

Theorem 1: ([11]) For $n \in \mathbb{Z}_{\geq 1}$, an odd $q \in \mathbb{Z}_{\geq 3}$ and integer $m \geq 6n \lg q$, there is a probabilistic polynomial-time algorithm $\mathsf{TrapGen}$ that, on input q, n, m, outputs $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m \times m}$, where $\mathbf{A}$ is $(m \cdot q^{-n/6})$-uniform over $\mathbb{Z}_q^{n \times m}$, and $\mathbf{T}_{\mathbf{A}}$ is a short (good) basis of $\mathcal{L}_q^{\perp}(\mathbf{A})$ except with negligible probability in n. More precisely, $\|\mathbf{T}_{\mathbf{A}}\| \leq \mathcal{O}(n \log q)$ and $\|\mathbf{T}_{\mathbf{A}}^*\| \leq \mathcal{O}(\sqrt{n \log q})$.

By consideration conducted in [12], it is relatively easy to construct a PPT algorithm $\mathsf{SampleISIS}$ that takes as input $\mathbf{V} \in \mathbb{Z}_q^{n \times m}$ along with its trapdoor $\mathbf{T}_{\mathbf{V}} \in \mathbb{Z}^{m \times m}$, and $\mathbf{u} \in \mathbb{Z}_q^n$, and outputs a sample $\mathbf{e} \in \mathbb{Z}^m$ from the distribution $\mathcal{D}_{\mathcal{L}_q^{\mathbf{u}}(\mathbf{V}), s}$. Furthermore, we have $\mathbf{V}\mathbf{e} \equiv \mathbf{u} \pmod q$ with overwhelming probability. It turns out, that we can easily generalize this assertion, taking a matrix instead of $\mathbf{u}$. To this end, we just apply $\mathsf{SampleISIS}$ separately to the consecutive columns of that matrix. Consequently, we obtain the following useful theorem.

Theorem 2: Let $n, k, m, q \in \mathbb{Z}_{>0}$ be such that q is prime, and $m \geq 6n \lg q$. There is a PPT algorithm k-$\mathsf{SampleISIS}$ that, on input $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, its associated trapdoor $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m \times m}$, a Gaussian parameter $s \geq \|\mathbf{T}_{\mathbf{A}}^*\| \cdot \omega\left(\sqrt{\log m}\right)$, and $\mathbf{U} \in \mathbb{Z}_q^{n \times k}$, outputs a matrix $\mathbf{E} \in \mathbb{Z}^{m \times k}$ from the joint distribution $\left(\mathcal{D}_{\mathcal{L}_q^{\mathbf{u}_j}(\mathbf{A}), s}\right)_{j \in [k]}$. Furthermore, the matrices $\mathbf{A}, \mathbf{U}$ and $\mathbf{E}$ are related by the formula $\mathbf{A}\mathbf{E} \equiv \mathbf{U} \pmod q$ with overwhelming probability.

C. Discrete Gaussian

For any real $s > 0$ and $\mathbf{c} \in \mathbb{R}^m$, the Gauss function $\rho_{s,\mathbf{c}} : \mathbb{R}^m \to \mathbb{R}$ centered on $\mathbf{c}$ with parameter s is defined as $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp\left(-\pi \cdot s^{-2} \|\mathbf{x} - \mathbf{c}\|^2\right)$, and as a notational convenience, we write $\rho_s = \rho_{s,\mathbf{0}}$, $\rho = \rho_1$, i.e. $\rho(\mathbf{x}) = e^{-\pi \|\mathbf{x}\|^2}$. For a lattice $\mathcal{L} \subseteq \mathbb{Z}^m$ we put $\rho_{s,\mathbf{c}}(\mathcal{L}) = \sum_{\mathbf{x} \in \mathcal{L}} \rho_{s,\mathbf{c}}(\mathbf{x})$, and define the discrete Gaussian distribution over $\mathcal{L}$ with center $\mathbf{c}$ and parameter s as $\mathcal{D}_{\mathcal{L},s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x}) / \rho_{s,\mathbf{c}}(\mathcal{L})$, $\mathbf{x} \in \mathcal{L}$. For notational convenience, we let $\mathcal{D}_{\mathcal{L},s} = \mathcal{D}_{\mathcal{L},s,\mathbf{0}}$, $\mathcal{D}_{s,\mathbf{c}}^m = \mathcal{D}_{\mathbb{Z}^m,s,\mathbf{c}}$, and $\mathcal{D}_s^m = \mathcal{D}_{\mathbb{Z}^m,s}$.

Below, we provide some basic properties of discrete Gaussians over lattices, which are important for consideration conducted herein.

Lemma 1: 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 $s \geq \|\mathbf{T}_{\mathbf{A}}^*\| \cdot \omega\left(\sqrt{\log m}\right)$, we have

1) ([13], [14]): $\Pr_{\mathbf{x} \leftarrow \mathcal{D}_{\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}),s,\mathbf{c}}}\left[\|\mathbf{x} - \mathbf{c}\| > s \cdot \sqrt{m}\right] \leq \mathsf{negl}(n)$.
2) ([14], [15]): $\Pr_{\mathbf{x} \leftarrow \mathcal{D}_{\mathcal{L}_q^{\perp}(\mathbf{A}),s}}\left[\mathbf{x} = 0\right] \leq \mathsf{negl}(n)$.
3) ([14], [16]): 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.

D. Small integer solution problems

Definition 1: ([6], [13]) *The small integer solution problem* $\mathsf{SIS}_{q,n,m,\beta}$ (in the ℓ_2 norm) is: given $q \in \mathbb{Z}_{\geq 1}$, 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 $\mathbf{S}\mathbf{x} \equiv \mathbf{0} \pmod q$ and $\|\mathbf{x}\| \leq \beta$. Equivalently, the SIS problem asks to find a vector $\mathbf{x} \in \mathcal{L}_q^{\perp}(\mathbf{S}) / \{\mathbf{0}\}$ with $\|\mathbf{x}\| \leq \beta$.
An inhomogeneous variant of SIS problem, that is called ISIS, is presented below.

Definition 2: ([12]) The inhomogeneous small integer solution problem $\mathsf{ISIS}_{q,n,m,\beta}$ (in the ℓ_2 norm) is: given $q \in \mathbb{Z}_{\geq 1}$, 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 $\mathbf{S}\mathbf{x} \equiv \mathbf{u} \pmod q$ and $\|\mathbf{x}\| \leq \beta$.
Both problems are as hard as worst-case of the lattice-based SIVP problem.

Theorem 3: ([12], [13]) For any positive integers n, m, real $\beta = \mathsf{poly}(n)$ and prime $q \geq \beta \cdot \omega\left(\sqrt{n \log n}\right)$, 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 $\gamma = \beta \cdot \widetilde{\mathcal{O}}(\sqrt{n})$.
The next, important, lemma is inspired by [6].

Lemma 2: ([9]) Assume that $d \in \mathbb{Z}_{\geq 9}$ and let $m > 24 + n \lg q / \lg(2d+1)$. 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 $\Pr[\exists\, \mathbf{e}' \in (-[d] \cup [d]_0)^m : \mathbf{e}' \neq \mathbf{e},\ \mathbf{A}\mathbf{e} = \mathbf{A}\mathbf{e}' \pmod q] > 1 - 2^{-101}$.

III. Forward Security of Password Authenticated Key Life-cycle Management Schemes with Key Update Mechanism

Assume there is given an alphabet Σ without a blank symbol, positive integers τ_l, τ_p (values of τ_l and τ_p do not depend on n), and finite set of available services $\mathscr{S}$. We define a *password authenticated key life-cycle management scheme (*PAKMS*) with key update mechanism* as a tuple of algorithms $\Pi = (\mathscr{G}, \mathsf{GVParam}, \mathsf{IntMKGen}, \mathsf{UsrKGen}, \mathsf{KUpd}, \mathsf{PwdVrfy})$ such that:

- $\mathscr{G}$ (*system parameters generation*) is a PT algorithm, which on input the value 1^n of a security parameter, maximum number of epochs E, outputs the system parameters params.
- $\mathsf{GVParam}$ (*global verification parameters generation*) is a PPT algorithm, which on input 1^n, params, and E, outputs a set of global verification parameters gvp.
- $\mathsf{IntMKGen}$ (*initial user's master key generation*) is a PPT algorithm, which on input 1^n, gvp, $\mathsf{login} \in \Sigma^{\leq \tau_l}$, $\mathsf{password} \in \Sigma^{\leq \tau_p}$, and E, outputs a user's master key $\mathsf{mk}_0 = (\mathsf{msk}_0, \mathsf{mvp})$ for the initial epoch $\mathsf{e} = 0$; msk's are called *master secret keys*, whereas mvp is called *master verification parameter* and is unchanged.
- $\mathsf{UsrKGen}$ (*generation of users' key-material for a service*) is a PPT algorithm, which takes as input login, $\mathsf{password}$, gvp, mvp, a service identifier srvId, mk_e, and a current epoch e, and outputs the service (symmetric) encryption secret key esk along with its verification parameters evp.
- KUpd (*user's master secret key update*) is a PPT algorithm, which takes as input a mater secret key msk_e for the epoch $\mathsf{e} < \mathsf{E} - 1$, and outputs a master secret key $\mathsf{msk}_{\mathsf{e}+1}$ for the subsequent epoch $\mathsf{e} + 1$.
- $\mathsf{PwdVrfy}$ (*password verification*) is DPT algorithm, which on input login, $\mathsf{password}$, srvId, gvp, mvp, evp, and e, outputs one out of two values, namely $\mathsf{accepted}$ or $\mathsf{rejected}$

Below, we described the proposed security model for PAKMS. To this end, let $\mathcal{A}$ be an adversary and assume that the system parameters params have been generated and they have been revealed to the adversary. In addition, $\mathcal{A}$ has been granted access to four oracles: $\mathsf{IntMKGen}$,KUpd, $\mathsf{UsrKGen}$ and break in oracle Break. In order to improve the consistency let $q_0 = q_0(n)$ denote the maximum number of queries to $\mathsf{IntMKGen}$ oracle, and assume without loss of generality that $\mathcal{A}$ always makes exactly this many queries; note that q_0 must be polynomial. Let us consider the following experiment $\mathbf{Exp}_{\mathcal{A},\Pi}^{\mathsf{fu\text{-}cpwda}}$ (forward unforgeability under a chosen password attack):

1) The adversary $\mathcal{A}$ is given mvp, and can request initial master keys (mk_0) for q_0 different pairs (credentials) (login, $\mathsf{password}$). Denote by C the (finite) set of this credentials, and let C_l be an associated set of all login's used.
2) $\mathsf{e} \leftarrow 0$;
3) **while** $\mathsf{e} < \mathsf{E}$
   2.1. $\mathsf{UsrKGen}$: For a current epoch e the adversary $\mathcal{A}$ requests the key-material for chosen pairs form C and chosen services, as many times as it likes (obviously, as $\mathcal{A}$ is PPT, this number is a polynomial of n).
   2.2. KUpd : If $t < T - 1$ is a current epoch, then $\mathcal{A}$ requests update: $\mathsf{e} \leftarrow \mathsf{e}+1$, $\mathsf{msk}_{\mathsf{e}+1} \leftarrow \mathsf{KUpd}(\mathsf{msk}_\mathsf{e})$, for every element of S.
   2.3 If Break then break the loop **while**;
   Break : If $\mathcal{A}$ is intended to move to the breach phase then it points out a specific $\mathsf{login}^\star \in C_l$ and launches the oracle Break. Then the experiment records the break-in epoch $\bar{\mathsf{e}} = \mathsf{e}$ and sends the current master secret key $\mathsf{msk}_{\bar{\mathsf{e}}}$ 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 other oracles.
4) Eventually $(\mathsf{e}^\star, \mathsf{password}^\star, \mathsf{srvId}^\star, \mathsf{evp}^\star) \leftarrow \mathcal{A}(1^n, \mathsf{state})$. If $\mathsf{e}^\star < \bar{\mathsf{e}}$ and $\mathsf{PwdVrfy}(\mathsf{login}^\star, \mathsf{password}^\star, \mathsf{srvId}^\star, \mathsf{gvp}^\star, \mathsf{mvp}^\star, \mathsf{evp}^\star, \mathsf{e}^\star) = \mathsf{accepted}$ and $\mathsf{UsrKGen}$ oracle has been never queried about a triple $(\mathsf{login}^\star, \mathsf{password}^\star, \mathsf{srvId}^\star)$ for the epoch $\mathsf{e}^\star$, then output 1, otherwise output 0.

A scheme PAKMS with key update mechanism $\Pi = (\mathscr{G}, \mathsf{GVParam}, \mathsf{IntMKGen}, \mathsf{KUpd}, \mathsf{UsrKGen}, \mathsf{PwdVrfy})$ is called to be *existentially forward unforgeable under a chosen-password attack* if for each efficient PPT adversary $\mathcal{A}$, its advantage $\mathsf{Adv}_{\Pi,n}^{\mathsf{fu\text{-}cpwda}}(\mathcal{A}) = \Pr[\mathbf{Exp}_{\mathcal{A},\Pi}^{\mathsf{fu\text{-}cpwda}}(1^n, \mathsf{E}) = 1]$ is a negligible function of n.

IV. Construction of the Scheme

In the following subsections we provide a detailed description of the algorithms that make up the presented PAKMS scheme.

Let Σ be a fixed alphabet without a blank symbol, and let $\tau_l, \tau_p \in \mathbb{Z}_{>0}$ be such that they do not depend on n (they are associated with an outer security policy). Without loss of generality we may assume that letters of Σ can be encoded as base-l numerals.

A. System parameters generation

Let $\ell \in \mathbb{Z}_{>0}$. We denote by n a value of the security parameter and by $\mathsf{E} = 2^\ell$ the number of epoch. The algorithm $\mathscr{G}$, on input 1^n, and the number of epoch E, outputs $\mathsf{params} = (q, m, \eta_{\min}, k, r, \mathsf{h}, \mathsf{H})$, where

- $q = \mathsf{poly}(n)$, $q \geq 3$ is a prime;
- $m \geq \lceil 6n \lg q \rceil$;
- $\mathsf{h} : \mathbb{Z}_q^n \times \{0,1\}^n \to \mathbb{B}_\mathsf{h}^k = \{\mathbf{w} \in \mathbb{R}^k \mid w_i \in \{-1, 0, 1\},\ \|\mathbf{w}\| \leq \sqrt{r}\}$ is a collision-resistant hash function;
- $\eta_{\min}$ is required minimal entropy of h;
- $k \in \mathbb{Z}_{>0}$ and $r \in [k]_0$ are such that $2^{\eta_{\min}} \leq \sum_{i=0}^{r} \binom{k}{i}$; then $H(\mathsf{h}) \geq \eta_{\min}$;
- $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ is a cryptographic hash function.

B. Global Verification Parameters Generation

The algorithm GVParam takes as input 1^n, params, and $\mathsf{E} = 2^\ell$. Fist, it chooses uniformly and independently 2ℓ matrices $\mathbf{V}_i^{(b)} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $i \in [\ell-1]_0, b \in \{0,1\}$, and constructs the ordered set $\mathsf{gvp} \leftarrow \left\{\mathbf{V}_{\ell-1}^{(0)}, \mathbf{V}_{\ell-1}^{(1)}, \mathbf{V}_{\ell-2}^{(0)}, \mathbf{V}_{\ell-2}^{(1)}, \dots \mathbf{V}_0^{(0)}, \mathbf{V}_0^{(1)}\right\}$. Next, it outputs gvp.

C. User Initial Master Key Generation

We start by defining the algorithm which is used as a subroutine in the attempt to derive an initial and updated key-material. Since this algorithm contains the mechanisms providing forward-security, and therefore „memory loss" of the past epochs, we call it Dory.

Algorithm 1 Dory

Input: ℓ, gvp, mvp, $\mathsf{msk}[2]$, $\mathsf{e} = b_{\ell-1}\cdots b_0$
Output: $\mathsf{msk} = (\mathsf{msk}[1], \mathsf{msk}[2])$

$(\mathbf{T}, h) \leftarrow \mathsf{msk}[2].\mathsf{pop}()$
$h \leftarrow h - 1$
while $h \geq 0$ **do**
  $\mathbf{N}_{h+1} \leftarrow \left[\mathbf{V}_\ell \mid \mathbf{V}_{\ell\text{-}1}^{(b_{\ell\text{-}1})} \mid \dots \mid \mathbf{V}_{h+1}^{(b_{h+1})}\right]$
  $\mathsf{tmp} \leftarrow \mathsf{ExtBasis}\left(\mathbf{T}_{\mathbf{N}_{h+1}}, \left[\mathbf{N}_{h+1} \mid \mathbf{V}_h^{(1)}\right]\right)$
  $\mathsf{msk}[2].\mathsf{push}(\mathsf{tmp}, h)$
  $\mathbf{T} \leftarrow \mathsf{ExtBasis}\left(\mathbf{T}_{\mathbf{N}_{h+1}}, \left[\mathbf{N}_{h+1} \mid \mathbf{V}_h^0\right]\right)$
  $h \leftarrow h - 1$
end while
$\mathsf{msk}[1] \leftarrow \mathbf{T}$

The algorithm IntMKGen should be viewed as registration to a domain, providing a number of separated services. Formally, it takes as input 1^n, gvp, E, $\mathsf{login} \in \Sigma^{\leq \tau_l}$, $\mathsf{password} \in \Sigma^{\leq \tau_p}$, and runs as follows:

1) The algorithm $\mathsf{TrapGen}(q,n)$ (Theorem 1) is run, and it outputs $(\mathbf{V}_\ell, \mathbf{T}_{\mathbf{V}_\ell})$, where $\mathbf{V}_\ell \in \mathbb{Z}_q^{n\times m}$ is a matrix and $\mathbf{T}_{\mathbf{V}_\ell} \in \mathbb{Z}_q^{m\times m}$ its trapdoor. Furthermore, $\mathsf{mvp} \leftarrow \mathbf{V}_\ell$. The matrix $\mathbf{V}_\ell$ gets bound with login in such a way that it constitutes a system mirror image of login.
2) A positive $\varepsilon = o(1)$ is chosen to derive $s_0 \geq \|\mathbf{T}^*_{\mathbf{V}_\ell}\| \cdot (\lg(\ell+1)m)^{\frac{1}{2}+\varepsilon}$.
3) It is created an empty stack $\mathsf{msk}[2]$, to be next initialized with $(\mathbf{T}_{\mathbf{V}_\ell}, \ell)$ by invoking $\mathsf{msk}[2].\mathsf{push}(\mathbf{T}_{\mathbf{V}_\ell}, \ell)$.
4) The algorithm $\mathsf{Dory}(\ell, \mathsf{gvp}, \mathsf{mvp}, \mathsf{msk}[2], \mathsf{e}=0)$ is launches to get $\mathsf{msk}_0 = (\mathsf{msk}_0[1], \mathsf{msk}[2])$.
5) Eventually, IntMKGen outputs $\mathsf{mk}_0 = (\mathsf{msk}_0, \mathsf{mvp})$.

D. Service Key-material Generation

Let srvId be an identifier of a specific service. The algorithm UsrKGen takes as input login, password, gvp, mvp, srvId, mk_e, a current epoch e, and runs as follows:

1) A matrix $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n\times k}$ is chosen uniformly at random. The matrix $\mathbf{U}$ is an identifier of the service srvId (this specific one) for the whole epoch e.
2) It is constructed the matrix $\mathbf{V}_\mathsf{e} = \left[\mathbf{V}_\ell \mid \mathbf{V}_{\ell-1}^{(b_{\ell-1})} \mid \mathbf{V}_{\ell-2}^{(b_{\ell-2})} \mid \cdots \mid \mathbf{V}_0^{(b_0)}\right] \in \mathbb{Z}_q^{n\times(\ell+1)m}$, where $\mathsf{e} = (b_{\ell-1}\cdots b_0)_2$.
3) The algorithm k-SampleISIS is run, given as input a tuple $(\mathbf{V}_\mathsf{e}, \mathsf{msk}_\mathsf{e}[1], s_0, \mathbf{U})$, it outputs $\mathbf{E}_\mathsf{e} \in \mathbb{Z}^{(\ell+1)m\times k}$, such that $\mathbf{V}_\mathsf{e}\cdot\mathbf{E}_\mathsf{e} = \mathbf{U} \pmod q$.
4) The service encryption secret key is derived $\mathsf{esk} \leftarrow \mathsf{H}(\mathsf{password}\|\mathbf{E}_\mathsf{e})$.
5) The numbers $\alpha_1, \alpha_2 \geq 12$ are set, and next the algorithm computes $s_1 = \max\{\alpha_1\sqrt{r}, (\lg k)^{\frac{1}{2}+\varepsilon}\}$, and $s_2 = \max\left\{\alpha_2 s_0(1+\alpha_1\sqrt{k})\sqrt{(\ell+1)mr}, (\lg((\ell+1)m))^{\frac{1}{2}+\varepsilon}\right\}$, where $\varepsilon = o(1)$.
6) The algorithm samples $\mathbf{b} \leftarrow \mathcal{D}_{s_1}^k$, $\mathbf{a} \leftarrow \mathcal{D}_{s_2}^{(\ell+1)m}$, $\mathbf{r} \xleftarrow{\$} \{0,1\}^n$, and computes $\mathbf{x}_1 \leftarrow \mathbf{V}_\mathsf{e}\mathbf{a} + \mathbf{U}\mathbf{b} \pmod q$, $\mathbf{x}_2 \leftarrow \mathsf{H}(\mathsf{login}\|\mathsf{password}\|\mathbf{r})$, and gets $\boldsymbol{\vartheta}_1 \leftarrow \mathsf{h}(\mathbf{x}_1, \mathbf{x}_2)$.
7) Set $\boldsymbol{\vartheta}_1' \leftarrow \boldsymbol{\vartheta}_1 + \mathbf{b}$ and go to the next step with probability $\min\left(\frac{\mathcal{D}_{s_1}^k(\boldsymbol{\vartheta}_1')}{M_1\mathcal{D}_{\boldsymbol{\vartheta}_1, s_1}^k(\boldsymbol{\vartheta}_1')}, 1\right)$; otherwise the algorithm is restarted.
8) Set $\boldsymbol{\vartheta}_2 \leftarrow \mathbf{E}_\mathsf{e}\boldsymbol{\vartheta}_1' + \mathbf{a}$ and output $\boldsymbol{\vartheta}_2$ with probability $\min\left(\frac{\mathcal{D}_{s_2}^{(\ell+1)m}(\boldsymbol{\vartheta}_2)}{M_2\mathcal{D}_{\mathbf{E}_\mathsf{e}\boldsymbol{\vartheta}_1', s_2}^{(\ell+1)m}(\boldsymbol{\vartheta}_2)}, 1\right)$; otherwise the algorithm is restarted.
9) If $\|\boldsymbol{\vartheta}_2\| \leq s_2\sqrt{(\ell+1)m}$ then the algorithm sets $\mathsf{evp} \leftarrow (\boldsymbol{\vartheta}_1, \boldsymbol{\vartheta}_2, \mathbf{r})$, otherwise it is restarted.
10) Eventually, the algorithm outputs esk along with evp.

E. Master Secret Key Update

In this section we define a user's master secret key update mechanism KUpd, which is based on some properties of binary trees. We refer to [5] for the comprehensive description of this idea. The algorithm takes as an input a current mater secret key msk_e, and carries out the following steps:

1) It updates $\mathsf{e} \leftarrow \mathsf{e} + 1$.
2) If $\mathsf{e} \equiv 1 \pmod 2$, then the following steps are conducted:
   3.1 $(\mathbf{T}, h) \leftarrow \mathsf{msk}[2].\mathsf{pop}()$ and $\mathsf{msk}_\mathsf{e}[1] \leftarrow \mathbf{T}$;
   3.2 The updated mater secret key for the new epoch is of the form $\mathsf{msk}_\mathsf{e} = (\mathsf{msk}_\mathsf{e}[1], \mathsf{msk}[2])$.
3) If $\mathsf{e} \equiv 0 \pmod 2$, then the following steps are done:
   4.1 $(\mathsf{msk}[1], \mathsf{msk}[2]) \leftarrow \mathsf{Dory}(\ell, \mathsf{gvp}, \mathsf{mvp}, \mathsf{msk}[2], \mathsf{e})$.
   4.2 The updated mater secret key for the new epoch is of the form $\mathsf{msk}_\mathsf{e} = (\mathsf{msk}_\mathsf{e}[1], \mathsf{msk}[2])$.

F. Credentials Verification

This algorithm PwdVrfy is launched just after a user's intention to log-in to a specific service srvId in the domain. The only assumption here is that the user has previously logged-in to this service, meaning formally that the algorithm UsrKGen has already generated a proper key-material. The algorithm takes as input login, password, srvId, gvp, mvp, evp, e and runs as follows:

1) The algorithm constructs the matrix $\mathbf{V}_\mathsf{e} = [\mathbf{V}_\ell | \mathbf{V}_{\ell-1}^{(b_{\ell-1})} | \mathbf{V}_{\ell-2}^{(b_{\ell-2})}| \cdots \| \mathbf{V}_0^{(b_0)}] \in \mathbb{Z}_q^{n\times(\ell+1)m}$ associated with the epoch e.
2) It sets $\widehat{\mathbf{x}}_1 \leftarrow \mathbf{V}_\mathsf{e}\boldsymbol{\vartheta}_2 - \mathbf{U}\boldsymbol{\vartheta}_1 \pmod q$.

3) If $\vartheta_1 = \mathsf{h}(\widehat{\mathbf{x}}_1, \mathsf{H}(\mathsf{login}\|\mathsf{password}\|\mathbf{r}))$ and $\|\vartheta_2\| \leq s_2\sqrt{(\ell+1)m}$, then output accepted, otherwise return rejected.

From the practical perspective, it should be also defined an algorithm which updates the key-material for the supported services. It seems to be important since although the update of the master keys is able to be done without user's intervention, the process of updating a service's key requires to provide the associated credentials. The natural design of such an algorithm is that after logging-in to a chosen service, it starts by verifying the current credentials, next requires to change a password and, having it given, runs UsrKGen so as to carry out update for all services supported by the user. It is, therefore, seen that such an algorithm is based entirely on PwdVrfy and UsrKGen, and thus constitutes at most a negligible added value in terms of the security of the scheme and consequently can be skipped.

G. Correctness of the verification process

Let $\mathsf{evp} = (\vartheta_1, \vartheta_2, \mathbf{r})$ be the union of verification parameters for the credentials $(\mathsf{login}, \mathsf{password})$, and an epoch e. We have $\vartheta_1 = \mathsf{h}(\mathbf{x}_1, \mathbf{x}_2)$, where $\mathbf{x}_1 = \mathbf{V}_\mathsf{e}\mathbf{a} + \mathbf{U}\mathbf{b} \pmod q$, $\mathbf{x}_2 = \mathsf{H}(\mathsf{login}\|\mathsf{password}\|\mathbf{r})$, and $\vartheta_2 = \mathbf{E}_\mathsf{e}\vartheta_1' + \mathbf{a} \pmod q$, where $\vartheta_1' = \vartheta_1 + \mathbf{b} \pmod q$. Then $\mathbf{V}_\mathsf{e}\vartheta_2 - \mathbf{U}\vartheta_1 = \mathbf{V}_\mathsf{e}(\mathbf{E}_\mathsf{e}\vartheta_1' + \mathbf{a}) = \mathbf{U}\mathbf{b} + \mathbf{V}_\mathsf{e}\mathbf{a}$. This implies $\mathsf{h}\left(\mathbf{V}_\mathsf{e}\vartheta_2 - \mathbf{U}\vartheta_1 \pmod q, \mathsf{H}(\mathsf{login}\|\mathsf{password}\|\mathbf{r})\right) = \mathsf{h}\left(\mathbf{V}_\mathsf{e}\mathbf{a} + \mathbf{U}\mathbf{b} \pmod q, \mathsf{H}(\mathsf{login}\|\mathsf{password}\|\mathbf{r})\right) = \mathsf{h}(\mathbf{x}_1, \mathbf{x}_2) = \vartheta_1$.
Both vectors ϑ_1' and ϑ_2 were sampled from distributions $\mathcal{D}^k_{\vartheta_1, s_1}$ and $\mathcal{D}^{(\ell+1)m}_{\mathbf{E}_t\vartheta_1', s_2}$. However, our task is to associate them with the other distributions, namely $\mathcal{D}^k_{s_1}$ and $\mathcal{D}^{(\ell+1)m}_{s_2}$. To this end, we use the rejection sampling method (Theorem 4.6, [6]), and show that M's and s's can be chosen in such a way that the outputs distributions of the steps 7, 8 of UsrKGen, are statistically close to the distributions in which ϑ_1' and ϑ_2 are sampled form $\mathcal{D}^k_{s_1}$ and $\mathcal{D}^{(\ell+1)m}_{s_2}$, respectively. Then, Lemma 1.1, implies that the estimation $\|\vartheta_2\| \leq s_2\sqrt{(\ell+1)m}$ holds with overwhelming probability. Further, by Lemma 4.5 of [6] we conclude that $M_1 \geq e^{\frac{24\alpha_1+1}{2\alpha_1^2}}$ and $M_2 \geq e^{\frac{24\alpha_2+1}{2\alpha_2^2}}$, with probabilities at least $1 - 2^{-100}$. This, therefore, means that the optimal choices are $M_1 \approx \exp\left(\frac{24\alpha_1+1}{2\alpha_1^2}\right)$ and $M_2 \approx \exp\left(\frac{24\alpha_2+1}{2\alpha_2^2}\right)$.

V. Justification of the Scheme Security

In this section we prove that any security breach of our PAKMS scheme will enable us to find a non-trivial solution to SIS problem.

Theorem 4: Assume that n is a value of a security parameter, $\ell \in \mathbb{Z}_{>0}$, and that $\Pi = (\mathscr{G}, \mathsf{GVParam}, \mathsf{IntMKGen}, \mathsf{UsrKGen}, \mathsf{KUpd}, \mathsf{PwdVrfy})$ is a PAKMS scheme presented in Section IV, with the associated message space $\mathcal{M} = \{0,1\}^*$. If $\mathcal{A}$ is a fu-cpwda -adversary attacking Π in the random oracle model, with a number of at most q_h queries to this oracle, then for $\beta = (2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}$ there is a PPT adversary $\mathcal{B}$ attacking $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ problem with advantage

$$\mathbf{Adv}^{\mathsf{SIS}}_n(\mathcal{B}) \approx \frac{1}{\#\mathscr{S}2^{\tau_l+1}q_\mathsf{h}\mathsf{E}^2} \cdot \left(\mathbf{Adv}^{\mathsf{fu\text{-}cpwda}}_{\Pi,n}(\mathcal{A})\right)^2, \quad (1)$$

and a running time $(\mathrm{time}(\mathcal{A}))$.

Proof. Let $\mathcal{A}$ be an adversary against Π. Our goal here is to construct an algorithm $\mathcal{B}$, which solves $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ problem, for $\beta = (2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}$ and, in addition, it exploits $\mathcal{A}$ as its subroutine. 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 sent to $\mathcal{B}$ by its $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$-game challenger. Therefore, the challenge is to find $\mathbf{x} \in \mathbb{Z}^{(1+2\ell)m}$, such that

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

First, $\mathcal{B}$ chooses $\mathsf{e}^* \xleftarrow{\$} \mathsf{E} - 1$, $\mathsf{login}^* \xleftarrow{\$} \Sigma^{\leq\tau_l}$, and $\mathsf{srvId}^* \xleftarrow{\$} \mathscr{S}$. According to the assumptions, the function h is modeled as a random oracle, and $\mathcal{A}$ is entitled to ask at most $q := q_\mathsf{h}$ queries to this oracle. Therefore, $\mathcal{B}$ chooses vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{q_\mathsf{h}}$ uniformly at random from $\{\mathbf{w} \in \mathbb{R}^k \mid w_i \in \{-1, 0, 1\},\ \|\mathbf{w}\| \leq \sqrt{r}\}$, and constructs an ordered set $\mathcal{W}_\mathsf{h} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{q_\mathsf{h}}\}$, where the ordering relation "$\preccurlyeq$" is defined as follow $\mathbf{w}_i \preccurlyeq \mathbf{w}_j$ iff $i < j$. Let $(b^*_{\ell-1}, b^*_{\ell-2}, \ldots, b^*_0)_2$ be the binary representation of e^*, then $\mathcal{B}$ creates gvp as follows

$$\mathsf{gvp} \leftarrow \left(\mathbf{V}^{(b=0)}_{\ell-1}, \mathbf{V}^{(b=1)}_{\ell-1}, \ldots \mathbf{V}^{(b=0)}_0, \mathbf{V}^{(b=1)}_0, \mathbf{U}\right),$$

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

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

In addition, $\mathcal{B}$ sets the master ver. parameter of login^* as $\mathsf{mvp}_* \leftarrow \mathbf{V}_\ell = \mathbf{S}_\ell$.

Next, $\mathbf{B}$ puts $T_{\max} := \max_{i\in[\ell-1]_0}\{\|\mathbf{T}^*_{\mathbf{V}^{(b)}_i}\| \mid b \neq t^*_i\}$. and chooses a positive $\varepsilon = o(1)$ so as to compute $d = s_0\sqrt{(1+\ell)m}$, where a Gauss distribution parameter s_0 is chosen in such a way to meet the following conditions:

- $s_0 \geq T_{\max}\cdot(\lg((\ell+1)m))^{\frac{1}{2}+\varepsilon}$;
- $(\ell+1)m > 24 + (n\lg q)/\lg(2d+1)$ (see Lemma 2).

h-Query. $\mathcal{B}$ prepares a list L to record all queries and responses as follows

(1) At the beginning, the 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 the list L consists of a pair $((\mathbf{x}_1, \mathbf{x}_2), \mathsf{h}(\mathbf{x}_1, \mathbf{x}_2))$ and, in this case, $\mathcal{B}$ outputs $\mathsf{h}(\mathbf{x}_1, \mathbf{x}_2)$.
   b) Otherwise, there is taken the first element $\mathbf{w} \in \mathcal{W}_\mathsf{h}$ which has not yet been used (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 on output.

Queries. *Master secret key update* KUpd. Let $\mathsf{e} \neq \mathsf{e}^*$, then there exists at least one $i \in [\ell]$ such that $b_i \neq b_i^*$. Set $i_0 := \max\{i \in [\ell-1]_0 \mid b_i \neq b_i^*\}$. If only $i_0 \neq \ell$, then $b_i = b_i^*$ for $i > i_0$. As $\mathcal{B}$ knows $\mathbf{V}_{i_0}^{(b_{i_0})} \in \mathbb{Z}_q^{n\times m}$ along with its trapdoor $\mathbf{T}_{\mathbf{V}_{i_0}^{(b_{i_0})}} \in \mathbb{Z}^{m\times m}$, it uses ExtBasis (see Theorem 5 of [17]) in order to get

$$\mathbf{T}_{\mathbf{V}_{\mathsf{e}}} \leftarrow \mathsf{ExtBasis}\left(\mathbf{T}_{\mathbf{V}_{i_0}^{(b_{i_0})}}, \left[\mathsf{mvp} \mid \mathbf{V}_{\ell-1}^{(t^*_{\ell-1})}|\mathbf{V}_{\ell-2}^{(b^*_{\ell-2})}|\cdots |\mathbf{V}_{i_0-1}^{(b^*_{i_0-1})}|\mathbf{V}_{i_0}^{(b_{i_0})}|\cdots|\mathbf{V}_0^{(b_0)}\right]\right),$$

if $\mathsf{login} \neq \mathsf{login}^*$ then $\mathsf{mvp} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, otherwise $\mathsf{mvp} \leftarrow \mathsf{mvp}_*$.

User Initial Master Key Generation IntMKGen. If $\mathsf{login} \neq \mathsf{login}_*$ it just run IntMKGen, otherwise the simulation is the same as in the simulation of KUpd.

Service Key-material Generation UsrKGen.

- If $\mathsf{login} \neq \mathsf{login}^*$, $\mathcal{B}$ just run UsrKGen.
- If $\mathsf{login} = \mathsf{login}^*$,
  - If $\mathsf{srvId} = \mathsf{srvId}^*$, $\mathcal{B}$ runs k-times the algorithm $\mathbf{E}_* \leftarrow \mathsf{SampleD}\left(\{|i\rangle\}_{i=0}^{m-1},\ s_0, \mathbf{c}=0\right)$ (see [18]), where the distribution $\mathbf{E}_*$ is $\mathcal{D}_{s_0}^{(1+\ell)m\times k}$. Note that $\|\mathbf{E}_*\| = \max\{\|\mathbf{e}_{*,1}\|, \ldots, \|\mathbf{e}_{*,k}\|\} \leq d$, with overwhelming probability, by Lemma 1.1. Next, $\mathcal{B}$ sets $\mathbf{U}_* = \mathbf{V}_{\mathsf{e}^*}\cdot\mathbf{E}_*$, $\mathbf{V}_{\mathsf{e}^*} = [\mathsf{mvp}_* \mid \mathbf{V}_{\ell-1}^{(b^*_{\ell-1})}|\mathbf{V}_{\ell-2}^{(b^*_{\ell-2})}|\cdots|\mathbf{V}_0^{(b^*_0)}] \in \mathbb{Z}_q^{n\times(\ell+1)m}$.
    - If $\mathsf{e} \neq \mathsf{e}^*$ then $\mathcal{B}$ makes $\mathbf{V}_{\mathsf{e}} = [\mathsf{mvp}_* \mid \mathbf{V}_{\ell-1}^{(b_{\ell-1})}|\mathbf{V}_{\ell-2}^{(b_{\ell-2})}|\cdots|\mathbf{V}_0^{(b_0)}]$ runs k-SampleISIS that takes as input $(\mathbf{T}_{\mathbf{V}_{\mathsf{e}}}, \mathbf{V}_{\mathsf{e}}, s_0, \mathbf{U}_*)$, and outputs $\mathbf{E}_{\mathsf{e}}$ from the distribution $\mathcal{D}_{q,\mathbf{A},s}^{k,\mathbf{U}}$. Having done this, evp for password are generated as in Section IV-D.
    - If $\mathsf{e} = \mathsf{e}^*$ then $\mathcal{B}$ assigns $\mathbf{E}_{\mathsf{e}^*} \leftarrow \mathbf{E}_*$.
  - If $\mathsf{srvId} \neq \mathsf{srvId}^*$ then $\mathcal{B}$ runs k-SampleISIS that takes as input $(\mathbf{T}_{\mathbf{V}_{\mathsf{e}}}, \mathbf{V}_{\mathsf{e}}, s_0, \mathbf{U})$, and outputs $\mathbf{E}_{\mathsf{e}}$ from the distribution $\mathcal{D}_{q,\mathbf{A},s}^{k,\mathbf{U}}$. Having done this, evp for password are generated as in Section IV-D.

Break in Break. If the adversary $\mathcal{A}$ runs the break in oracle, the current epoch $\bar{\mathsf{e}}$ is saved and the adversary is given the proper master secret key $\mathsf{msk}_{\bar{\mathsf{e}}}$. This key consists of two components, namely $\mathsf{msk}_{\mathsf{e}}[1]$ and $\mathsf{msk}_{\mathsf{e}}[2]$. In order to generate $\mathsf{msk}_{\mathsf{e}}[1]$, $\mathcal{B}$ proceeds in the same way as in KUpd. When it comes to the component $\mathsf{msk}_{\mathsf{e}}[2]$, $\mathcal{B}$ uses Dory (with obvious changes, as it only focuses on the node's indexes) so as to get the identifiers of nodes from $\mathsf{msk}_{\mathsf{e}}[2]$. Having done this, $\mathcal{B}$ is able to derive nodes corresponding to these identifiers. To this end, it does the following. At first, it takes an identifier $(b_\ell - 1 \ldots b_h)_2$ and indicate $i_0 := \max\{i \in [\ell-1]_0 \mid \mathsf{e}_i \neq \mathsf{e}_i^*\}$. Such i_0 exists, since there is not node in $\mathsf{msk}_{\mathsf{e}}[2]$ that lies on the branch linking the root with a leaf associated with e^*. $\mathcal{B}$ knows a pair $\left(\mathbf{V}_{i_0}^{(b_{i_o})}, \mathbf{T}_{\mathbf{V}_{i_0}^{(b_{i_0})}}\right)$, therefore it runs ExtBasis in order to obtain $\mathbf{T}_{\mathbf{N}_h} \leftarrow \mathsf{ExtBasis}\left(\mathbf{T}_{\mathbf{V}_{i_0}^{(b_{i_0})}}, \mathbf{N}_h\right)$, where $\mathbf{N}_h = \left[\mathbf{V}_\ell \mid \mathbf{V}_{\ell\text{-}1}^{(b_{\ell\text{-}1})} \mid \ldots \mid \mathbf{V}_h^{(b_h)}\right]$, $\mathbf{V}_\ell = \mathsf{mvp}$.

Forgery. $\mathcal{B}$ is statistically indistinguishable from a real challenger in the experiment $\mathbf{Exp}_{\mathcal{A},\Pi}^{\mathsf{fu\text{-}cpwda}}$. Therefore, the adversary $\mathcal{A}$ eventually outputs a forgery for an epoche$\widehat{\mathsf{e}^*}$. If $\widehat{\mathsf{e}^*} \neq \mathsf{e}^*$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ accepts the forgery $(\mathsf{e}^*, \mathsf{login}^*, \mathsf{evp}^*)$, where $\mathsf{evp} = (\boldsymbol{\vartheta}_1^*, \boldsymbol{\vartheta}_2^*, \mathbf{r}^*)$ and

- $\|\boldsymbol{\vartheta}_2^*\| \leq s_2\sqrt{(\ell+1)m}$;
- $\boldsymbol{\vartheta}_1^* = \mathsf{h}\left(\mathbf{V}_{\mathsf{e}^*}\boldsymbol{\vartheta}_2^* - \mathbf{U}\boldsymbol{\vartheta}_1^* \pmod q, \mathsf{H}(\mathsf{login}^*\|\mathsf{password}^*\|\mathbf{r}^*)\right)$;
- $\mathbf{A}_{\mathsf{e}^*} = [\mathsf{mvp}_*|\mathbf{V}_{\ell-1}^{(b^*_{\ell-1})}|\mathbf{V}_{\ell-2}^{(b^*_{\ell-2})}|\cdots|\mathbf{V}_0^{(b^*_0)}] \in \mathbb{Z}_q^{n\times(\ell+1)m}$.

Let $\mathbf{w}_i \in \mathcal{W}_\mathsf{h}$ be such that $\boldsymbol{\vartheta}_1^* = \mathbf{w}_i$. The answers to h-queries are taken successively form $\mathcal{W}_\mathsf{h}$, following the relation "$\preccurlyeq$". $\mathcal{B}$ picks vectors $\widehat{\mathbf{w}}_i, \widehat{\mathbf{w}}_{i+1}, \ldots, \widehat{\mathbf{w}}_{q_\mathsf{h}}$ independently and uniformly at random from $\mathbb{B}_\mathsf{h}^k$, 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 has the form $\widehat{\mathcal{W}_\mathsf{h}} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}, \widehat{\mathbf{w}}_i, \widehat{\mathbf{w}}_{i+1}, \ldots, \widehat{\mathbf{w}}_{q_\mathsf{h}}\}$.
Having this done, $\mathcal{B}$ runs $\mathcal{A}$ again with the same parameters (params) and the same random tape ρ as in the first run, but this time it uses $\widehat{\mathcal{W}_\mathsf{h}}$ instead of $\mathcal{W}_\mathsf{h}$ to answer the h-queries. By Lemma General Forking Lemma (see [19]), $\mathcal{A}$ outputs a new forgery $\left(\widehat{\mathsf{e}^*}, \widehat{\mathsf{login}^*}, \widehat{\mathsf{evp}^*} = (\widehat{\boldsymbol{\vartheta}_1^*}, \widehat{\boldsymbol{\vartheta}_2^*}, \widehat{\mathbf{r}^*})\right)$ using the same h-queries. If $\widehat{\mathsf{e}^*} \neq \mathsf{e}^*$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ accepts the forgery and then this means, in particular, that $\|\widehat{\boldsymbol{\vartheta}_2^*}\| \leq s_2\sqrt{(\ell+1)m}$ and $\widehat{\mathbf{w}}_i = \widehat{\boldsymbol{\vartheta}_1^*} = \mathsf{h}\left(\mathbf{V}_{\mathsf{e}^*}\widehat{\boldsymbol{\vartheta}_2^*} - \mathbf{U}_*\widehat{\boldsymbol{\vartheta}_1^*} \pmod q, \mathsf{H}(\mathsf{login}^*\|\widehat{\mathsf{password}^*}\|\widehat{\mathbf{r}^*})\right)$. In addition, the probability P_1 that $\widehat{\mathbf{w}}_i \neq \mathbf{w}_i$ is $P_1 \approx \mathsf{Adv}_{\Pi,n}^{\mathsf{fu\text{-}cpwda}}(\mathcal{A})/q_\mathsf{h}$.

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 h-queries responses. This implies that the two executions of $\mathcal{A}$ are identical up to the i-th h-query, what means that the arguments of both i-th h-queries must be the same. Thus $\mathbf{A}_{t^*}\boldsymbol{\vartheta}_2^* - \mathbf{U}\boldsymbol{\vartheta}_1^* \equiv \mathbf{A}_{t^*}\widehat{\boldsymbol{\vartheta}_2^*} - \mathbf{U}\widehat{\boldsymbol{\vartheta}_1^*} \pmod q$ and $\mathsf{H}(\mathsf{login}^*\|\mathsf{password}^*\|\mathbf{r}^*) = \mathsf{H}(\mathsf{login}^*\|\widehat{\mathsf{password}^*}\|\widehat{\mathbf{r}^*})$. This implies

$$\begin{aligned} 0 &\equiv \mathbf{A}_{t^*}(\boldsymbol{\vartheta}_2^* - \widehat{\boldsymbol{\vartheta}_2^*}) - \mathbf{U}(\boldsymbol{\vartheta}_1^* - \widehat{\boldsymbol{\vartheta}_1^*}) \\ &\equiv \mathbf{A}_{t^*}\left(\boldsymbol{\vartheta}_2^* - \widehat{\boldsymbol{\vartheta}_2^*} - \mathbf{E}_*(\boldsymbol{\vartheta}_1^* - \widehat{\boldsymbol{\vartheta}_1^*})\right) \pmod q. \end{aligned} \tag{2}$$

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

$$\mathbf{V}_{\mathsf{e}^*}\mathbf{x}_0 \equiv \mathbf{0} \pmod q \quad \text{with } \|\mathbf{x}_0\| \leq \beta.$$

It remains to evaluate how likely it is that $\boldsymbol{\vartheta}_2^* - \widehat{\boldsymbol{\vartheta}_2^*} - \mathbf{E}_*(\boldsymbol{\vartheta}_1^* - \widehat{\boldsymbol{\vartheta}_1^*}) \neq \mathbf{0}$. Let $j_0 = \min\left\{j \in [k] \mid \vartheta_{1,j}^* \neq \widehat{\vartheta_{1,j}^*}\right\}$; j_0 exists as $\boldsymbol{\vartheta}_1^* = \mathbf{w}_i \neq \widehat{\mathbf{w}}_i = \widehat{\boldsymbol{\vartheta}_1^*}$. Set $\mathbf{e} = \mathbf{E}_*[j_0]$. Then, since $d \geq 9$ and $(\ell+1)m > 24 + (n\lg q)/\lg(2d+1)$, Lemma 2 says 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 $1 - 2^{-101}$. This let $\mathcal{B}$ to construct a matrix $\mathbf{E}'_*$ such that all its columns, except for the column j_0, are the same as $\mathbf{E}_*$. Therefore, by definition of $\mathbf{E}'_*$, if

$$\boldsymbol{\vartheta}_2^* - \widehat{\boldsymbol{\vartheta}_2^*} - \mathbf{E}_*(\boldsymbol{\vartheta}_1^* - \widehat{\boldsymbol{\vartheta}_1^*}) = \mathbf{0}, \tag{3}$$

then

$$\boldsymbol{\vartheta}_2^* - \widehat{\boldsymbol{\vartheta}_2^*} - \mathbf{E}'_*(\boldsymbol{\vartheta}_1^* - \widehat{\boldsymbol{\vartheta}_1^*}) \neq \mathbf{0}. \tag{4}$$

It means that for every matrix $\mathbf{E}_*$ satisfying (3), 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 (4) holds and that $\mathbf{A}_{t^*}\mathbf{E}_* = \mathbf{A}_{t^*}\mathbf{E}'_*$. Since these matrices are statistically indistinguishable to the adversary $\mathcal{A}$, the likelihood of choosing between them is at least $1/2$. Therefore, the probability P_2 that $\mathbf{x}_0 \neq \mathbf{0}$ is $P_2 \geq 1/2 - 1/2^{101} \approx 1/2$

Having given $\mathbf{x}_0$, $\mathcal{B}$ constructs a vector $\mathbf{x} = [x_{(1+2\ell)m-1}, \ldots, x_0] \in \mathbb{Z}_q^{(1+2\ell)m}$ in the following way

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

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

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

where $\|\mathbf{x}_0\| \leq \beta = (2s_2 + 2s_0\sqrt{r})\sqrt{(\ell+1)m}$. This implies that the probability of solving $\mathsf{SIS}_{q,n,(1+2\ell)m,\beta}$ problem is the same as the probability of an event that $\mathbf{x} \neq \mathbf{0}$. To summarize, the probability of $\mathcal{B}$ not aborting is equal to $2^{\tau_l} \cdot T^2 \cdot \#\mathscr{S}$, and $P_1 \cdot P_2 \approx \mathsf{Adv}^{\mathsf{fu\text{-}cpwda}}_{\Pi,n}(\mathcal{A})/2q_{\mathsf{h}}$. Therefore, we obtain

Taking into account the values of P_1 and P_1, and the fact that the probability of $\mathcal{B}$ not aborting is exactly equal to $1/T^2$, we have that

$$\Pr[\mathbf{x} \neq \mathbf{0}] \approx \mathsf{Adv}^{\mathsf{fu\text{-}cpwda}}_{\Pi,n}(\mathcal{A})/(q_{\mathsf{h}}2^{\tau_l+1} \cdot T^2 \cdot \#\mathscr{S}).$$

This finishes the proof. □

VI. Summary of Parameters

In this section we summarize the parameters that are of crucial importance for the security of the scheme (see Table I). In addition, we give an example set of parameters that provide 80 bits of security. The choice of this value is due to the fact that it is assumed 80 bits represents a minimal security requirement and draws a red line in gaining any sort of cybersecurity. Since the number of periods $T = 2^\ell$ does not influence directly on the security level, we can skip it in the Table I.

TABLE I
PAKMS PARAMETERS AND THE SET OF SAMPLES FOR $\ell = 8$ PROVIDING 80B SECURITY

Parameter	Definition	Samples
n	security parameter	512
q	a prime ≥ 3	$2^{27}+29$
$\eta_{\min}$	h's min. entropy	101
k, r	$k \in \mathbb{Z}_{>0}, r \in [k]_0, \sum_{i=0}^{r}\binom{k}{i} \geq 2^{\eta_{min}}$	101, 101
α_1, α_2	≥ 12	12, 12
m	$\max\{\lceil 6n\lg q\rceil, \left\lceil \frac{1}{\ell+1}\cdot\left(24+\frac{(n\lg q)}{\lg(2d+1)}\right)\right\rceil\}$	82945
M_1, M_2	$M_1 = M_2 = \exp\left(\frac{24\alpha_1+1}{2\alpha_1^2}\right)$	2.7277

The role of the parameters is clearly explained in the text. However, we feel compelled to add a few clarifying remarks. Namely, the parameters $\eta_{\min}$, k, r defines the size of the challenges so as to get correctness error of at most 2^{-101}. Furthermore, at first glance it appears that the value of m in Table I can be derived only if d is known. However, the condition $\left\lceil \frac{1}{\ell+1}\cdot\left(24+\frac{(n\lg q)}{\lg(2d+1)}\right)\right\rceil > \lceil 6n\lg q\rceil$ holds only for relatively small n, q, d and, consequently, in practical instantiates we always have $m = \lceil 6n\lg q\rceil$.

TABLE II
PAKMS PARAMETERS THAT NOT INFLUENCE DIRECTLY ON THE SECURITY LEVEL

Parameter	Definition
T	a number of periods $T = 2^\ell$, $\ell \in \mathbb{Z}_{>0}$
$T_{\max}$	$\max_{i\in[\ell-1]_0}\{\|\mathbf{T}^*_{\mathbf{A}_i^{(b)}}\| \mid b \neq t_i^*\}$
ε	$o(1)$
s_0	$\geq T_{\max}\cdot(\lg((\ell+1)m))^{\frac{1}{2}+\varepsilon}$
s_1	$\geq \max\left\{\alpha_1\sqrt{r}, (\lg k)^{\frac{1}{2}+\varepsilon}\right\}$
s_2	$\geq \max\left\{\alpha_2 s_0(1+\alpha_1\sqrt{k})\sqrt{(\ell+1)mr}, (\lg((\ell+1)m))^{\frac{1}{2}+\varepsilon}\right\}$
d	$s_0\sqrt{(1+\ell)m}$
β	$(2s_2+2s_0\sqrt{r})\sqrt{(\ell+1)m}$

The choice of n, k, q should guarantee the computational infeasibility of SIS problem. To this end, an associated random q-ary lattice $\mathcal{L}$ is defined in order to exploit some known lattice reduction algorithms so as to find short vectors in this lattice. Gama and Nguyen justify in [20] that the length of the vector obtained by running the best known algorithms on a random m-dimensional q-ary lattice $\mathcal{L}$ is close to $\min\{q, ((\det\mathcal{L})^{1/m}\cdot\delta^m)\} = \min\{q, q^{n/m}\cdot\delta^m\}$, with overwhelming probability. The parameter δ, called a Hermite factor, depends on the quality of the used lattice-reduction algorithm. Further, Micciancio and Regev [21] notice 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 $\mathcal{L}$ when $m \approx \sqrt{n\lg q/\lg\delta}$. For smaller m, the lattice is too sparse and does not contain vectors that are short enough. For larger m, the high dimension prevents lattice reduction algorithms from finding short vectors. Eventually, Micciancio and Regev conclude that 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 $\min\left\{q, 2^{2\cdot\sqrt{n\lg q\lg\delta}}\right\}$.

VII. Conclusions

We constructed a new password authenticated key life-cycle management scheme with key update mechanism. Since the construction is based on q-ary lattices, the scheme is secure against known quantum attacks and, to the best of our knowledge, is the first post-quantum scheme of this sort. In addition, we formally proved that the scheme is existentially forward unforgeable under a chosen password attack as long as SIS problem is computationally hard. This is significant since it meets the requirement that cryptographic constructions should be proven secure with respect to clearly stated definitions of security and relative to well-defined cryptographic assumptions. Such approach is the essence of modern cryptography.

It must be highlighted that the scheme's keys are represented by matrices of large dimensions, and their number grows as ℓ increases. Storing such a huge amount of data is not feasible for most practical applications. This drawback (inherently related with lattice-based solutions) can be eliminated by using a cryptographically secure deterministic pseudo random number generator (PRNG) and storing only the seeds of that PRNG.

We indicated the parameters and thoroughly explained their influence on the scheme's security. In particular, we derived the set od samples that provides a cut-off level of 80-bits security. This allows to conduct an introductory estimation regarding the order of magnitude for parameters in therms of reaching a desired security level in practical implementations. Further, taking into account the size of these samples vs the complexity of presented scheme, we get a good trade-off in comparison with other lattice-based solutions.

REFERENCES

[1] D. Slamanig and C. Striecks, "Revisiting Updatable Encryption: Controlled Forward Security, Constructions and a Puncturable Perspective," in *Theory of Cryptography Conference*. Springer, 2023, pp. 220–250.

[2] P. Miao, S. Patranabis, and G. Watson, "Unidirectional Updatable Encryption and Proxy Re-Encryption from DDH," in *IACR International Conference on Public-Key Cryptography*. Springer, 2023, pp. 368–398.

[3] Y. J. Galteland and J. Pan, "Backward-leak UNI-directional Updatable Encryption from (Homomorphic) Public Key Encryption," *Cryptology ePrint Archive*, 2022.

[4] M. Bellare and S. K. Miner, "A Forward-Secure Digital Signature Scheme," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 431–448. [Online]. Available: https://doi.org/10.1007/3-540-48405-1_28

[5] M. Jurkiewicz, "Binary Tree Based Forward Secure Signature Scheme in the Random Oracle Model," *International Journal of Electronics and Telecommunications*, pp. 717–726, 2021.

[6] V. Lyubashevsky, "Lattice Signatures without Trapdoors," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 738–755. [Online]. Available: https://doi.org/10.1007/978-3-642-29011-4_43

[7] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A Lattice-based Digital Signature Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 238–268, 2018. [Online]. Available: https://doi.org/10.13154/tches.v2018.i1.238-268

[8] P. Zhang, H. Jiang, Z. Zheng, P. Hu, and Q. Xu, "A New Post-quantum Blind Signature from Lattice Assumptions," *IEEE Access*, vol. 6, pp. 27 251–27 258, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2833103

[9] M. Jurkiewicz, "Quantum-resistant forward-secure digital signature scheme based on q-ary lattices," *Journal of Telecommunications and Information Technology*, pp. 90–103, 2024. [Online]. Available: https://doi.org/10.26636/jtit.2024.2.1581

[10] O. Goldreich, *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2003.

[11] J. Alwen and C. Peikert, "Generating Shorter Bases for Hard Random Lattices," *Theory of Computing Systems*, vol. 48, pp. 535–553, 2011. [Online]. Available: https://doi.org/10.1007/s00224-010-9278-3

[12] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for Hard Lattices and New Cryptographic Constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 197–206.

[13] D. Micciancio and O. Regev, "Worst-Case to Average-Case Reductions Based on Gaussian Measures," *SIAM Journal on Computing*, vol. 37, no. 1, pp. 267–302, 2007. [Online]. Available: https://doi.org/10.1109/FOCS.2004.72

[14] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai Trees, or How to Delegate a Lattice Basis," *Journal of Cryptology*, vol. 25, pp. 601–639, 2012.

[15] C. Peikert and A. Rosen, "Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices," in *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 2006, pp. 145–166.

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

[17] S. Agrawal, D. Boneh, and X. Boyen, "Efficient Lattice (H) IBE in the Standard Model," in *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*. Springer, 2010, pp. 553–572.

[18] C. Peikert, "An Efficient and Parallel Gaussian Sampler for Lattices," in *Annual Cryptology Conference*. Springer, 2010, pp. 80–97.

[19] M. Bellare and G. Neven, "Multi-signatures in the Plain Public-key Model and a General Forking Lemma," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 390–399. [Online]. Available: https://doi.org/10.1145/1180405.1180453

[20] N. Gama and P. Q. Nguyen, "Predicting Lattice Reduction," in *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*. Springer, 2008, pp. 31–51. [Online]. Available: https://doi.org/10.1007/978-3-540-78967-3_3

[21] D. Micciancio and O. Regev, "Lattice-based Cryptography," in *Post-quantum Cryptography*. Springer, 2009, pp. 147–191.