Fundamenta Informaticae, Volume 194, Issue 4, Article 4, 2025
DOI: 10.46298/fi.14425
https://fi.episciences.org/

Unidirectional Key Update in Updatable Encryption, Revisited

Mariusz Jurkiewicz, Kamila Prabucka[*†]
Military University of Technology, Warsaw, Poland
{*mariusz.jurkiewicz, kamila.prabucka*}*@wat.edu.pl*

Abstract. We introduce a new, efficient updatable encryption (UE) scheme based on FrodoPKE, a key encapsulation mechanism grounded in the Learning With Errors (LWE) problem and derived from Frodo KEM, a third-round alternate candidate in the NIST Post-Quantum Cryptography Standardization process. The proposed scheme supports uni-directional key updates with backward-leak protection and is proven secure in the rand-ind-eu-cpa model. Its security relies on the hardness of the LWE problem, providing strong guarantees against both classical and quantum adversaries under the assumption that LWE is computationally intractable.

Keywords: Updatable Encryption; Unidirectional Key Update; Lattices; LWE problem.

1. Introduction

Updatable encryption (UE) enables a third party to periodically rotate encryption keys across epochs without requiring the client to download, decrypt, re-encrypt, and re-upload the encrypted data. Instead, ciphertexts can be updated efficiently using *update tokens*, which are compact transformation artifacts generated during key rotation. These tokens can be sent publicly to the storage provider, who applies them to the ciphertexts to bring them in line with the current epoch key, all without learning anything about the underlying plaintexts.

UE has become an important cryptographic primitive for managing long-term encrypted data in outsourced storage systems, where frequent key rotations are essential for maintaining forward secrecy

Address for correspondence: Military University of Technology, Kaliskiego 2 00-908 Warsaw
[*]The authors are grateful to the anonymous reviewers for their thorough reading of the manuscript and for their insightful and constructive feedback.
[†]Supported by the Grant DOB/002/RON/ID1/2018 by The National Centre for Research and Development.

and minimizing exposure in case of key compromise. Efficient key updates that do not require client-side processing or data access are particularly attractive in cloud-based or distributed settings, where bandwidth, latency, and trust boundaries are significant concerns.

To convey the underlying intuition of UE while maintaining mathematical rigor, consider a connected and closed set $\mathsf{E} \subseteq \mathbb{R}_{\geq 0}$ containing 0, where $\mathbb{R}_{\geq 0}$ denotes the time axis and 0 serves as the origin. The set E is partitioned into a family $\mathcal{E} := \{\mathsf{e}_i\}_{i \in \mathbb{Z}_{\geq 0}}$ of closed and bounded subsets. If E is bounded, there exists $i_{\max} \in \mathbb{Z}_{\geq 1}$ such that $\mathsf{e}_i \neq \emptyset$ for all $i \leq i_{\max}$ and $\mathsf{e}_i = \emptyset$ for $i > i_{\max}$. If E is unbounded, then all sets e_i are nonempty. Furthermore, $\mathsf{E} = \bigcup_{i \in \mathbb{Z}_{\geq 0}} \mathsf{e}_i$, and for each $i \in \mathbb{Z}_{\geq 1}$ with $\mathsf{e}_i \neq \emptyset$, we have $\mathsf{e}_{i-1} \cap \mathsf{e}_i \neq \emptyset$ and $\mathrm{int}(\mathsf{e}_{i-1}) \cap \mathrm{int}(\mathsf{e}_i) = \emptyset$. Each nonempty set e_i is referred to, in the formal sense, as an *epoch*. It is worth noting that adjacent epochs may share a single point of intersection; this property is convenient for ensuring consistency in the construction and, since such intersections have Lebesgue measure zero on the real line, they do not affect the analysis. We introduce a natural ordering on epochs as follows: we write $\mathsf{e}_1 \leq \mathsf{e}_2$ if every element of e_1 is less than or equal to every element of e_2. To emphasize strict ordering, we write $\mathsf{e}_1 < \mathsf{e}_2$ whenever $\mathsf{e}_1 \leq \mathsf{e}_2$ and $\mathsf{e}_1 \neq \mathsf{e}_2$. Moreover, for any epoch e, we define its adjacent successor, denoted $\mathsf{e} + 1$, as the unique epoch with index one greater than that of e; similarly, $\mathsf{e} + 2$ denotes the epoch following $\mathsf{e} + 1$, and so forth. For convenience, and to avoid unnecessary proliferation of notation, we will also identify each epoch with its index; for example, $\mathsf{e} = 0$ means that we are actually working with e_0. An *updatable encryption scheme* over a nonempty message space $\mathcal{M}$ consists of a collection of probabilistic polynomial-time (PPT) algorithms $(\mathsf{UE.KG}, \mathsf{UE.TG}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd})$ operating over epochs, starting at $\mathsf{e} = 0$:

- $\mathsf{UE.KG}$ generates an epoch key $\mathsf{k_e}$,
- $\mathsf{UE.TG}$ takes two consecutive epoch keys $\mathsf{k_e}$ and $\mathsf{k_{e+1}}$ and outputs an update token $\Delta_{\mathsf{e}+1}$,
- $\mathsf{UE.Enc}$ encrypts a message $\mathbf{m} \in \mathcal{M}$ under $\mathsf{k_e}$ to produce a ciphertext $\mathsf{ct_e}$,
- $\mathsf{UE.Dec}$ decrypts a ciphertext $\mathsf{ct_e}$ using $\mathsf{k_e}$ to recover the message $\mathbf{m}$,
- $\mathsf{UE.Upd}$ takes a token $\Delta_{\mathsf{e}+1}$ and a ciphertext $\mathsf{ct_e}$, and outputs an updated ciphertext for epoch $\mathsf{e} + 1$.

We say that a UE scheme is *correct* if, for any message $\mathbf{m} \in \mathcal{M}$ and any two epochs $\mathsf{e}_1, \mathsf{e}_2 \in \mathcal{E}$ with $\mathsf{e}_1 < \mathsf{e}_2$, a ciphertext encrypted at epoch e_1 and sequentially updated through each intermediate epoch up to e_2 can be correctly decrypted under the final key, except with negligible probability. Formally, correctness requires that

$$\Pr\left[\mathsf{UE.Dec}(\mathsf{k_{e_2}}, \mathsf{ct_{e_2}}) \neq \mathbf{m}\right] \leq \mathsf{negl}(n).$$

In this process, the keys $\mathsf{k_{e_1}}, \ldots, \mathsf{k_{e_2}}$ are generated using $\mathsf{UE.KG}(1^n)$. The message $\mathbf{m}$ is encrypted under $\mathsf{k_{e_1}}$ to produce the initial ciphertext $\mathsf{ct_{e_1}} \leftarrow \mathsf{UE.Enc}(\mathsf{k_{e_1}}, \mathbf{m})$, and for each $j \in \{\mathsf{e}_1 + 1, \ldots, \mathsf{e}_2\}$, an update token $\Delta_j \leftarrow \mathsf{UE.TG}(\mathsf{k}_{j-1}, \mathsf{k}_j)$ is generated and applied to update the ciphertext via $\mathsf{ct}_j \leftarrow \mathsf{UE.Upd}(\Delta_j, \mathsf{ct}_{j-1})$.

In this work, we present a new UE scheme that is secure in the rand-ind-eu-cpa model and is based on the Learning With Errors (LWE) assumption, a well-established foundation for post-quantum cryptography. Our design builds on FrodoPKE, a lattice-based key encapsulation mechanism grounded in the LWE problem and known for its simplicity and conservative assumptions. The resulting scheme is secure against both classical and quantum adversaries and achieves strong directional guarantees: it supports *backward-leak uni-directional updates*, meaning the past key k_e can be derived from the future key $\mathsf{k}_{\mathsf{e}+1}$ and the update token $\Delta_{\mathsf{e}+1}$, but not vice versa. This directionality is significant: as shown in prior work [21], backward-leak uni-directional schemes offer stronger security than both forward-leak and bi-directional models, establishing a clear hierarchy among UE constructions.

The security of our scheme against both classical and quantum adversaries does not rely solely on the properties of FrodoPKE. Rather, it is a consequence of the fact that every element of the construction is carefully designed within the framework of lattice-based cryptography. The scheme employs only well-established operations, including Gaussian noise sampling, matrix-vector multiplication over integer lattices, and arithmetic on lattice-structured data, all of which are known to preserve security in both classical and quantum models. The design explicitly avoids relying on computational assumptions that are known to be vulnerable to quantum attacks, such as those based on factoring or discrete logarithms. Because the construction remains entirely within the domain of assumptions grounded in the hardness of the LWE problem, it maintains strong security guarantees throughout. This argument is formally supported by Theorem 6.9, which establishes a reduction from breaking the rand-ind-eu-cpa-security of our scheme in the backward-leak setting to solving the LWE problem, under standard lattice assumptions.

A key advantage of our construction is its efficiency. Unlike some earlier schemes where ciphertext or key sizes grow with the number of epochs, our scheme maintains constant-size ciphertexts and keys, regardless of how many times updates are performed. This makes it particularly well-suited for real-world applications requiring scalability and long-term security guarantees.

In summary, our construction advances the state of updatable encryption by combining three desirable properties: strong directional security in the backward-leak model, post-quantum resilience via the LWE assumption, and practical efficiency with constant-size ciphertexts and keys. This positions our scheme as a compelling option for secure, scalable, and future-proof encrypted storage.

2. Related Work and Discussion

The concept of updatable encryption was formally introduced by Boneh et al. [5], marking the beginning of a broader investigation into secure ciphertext updates across key epochs. Early efforts focused on ciphertext-dependent schemes, where each update token is generated relative to a specific ciphertext. This line of work was initiated by Everspaugh et al. [12], and subsequently refined by Boneh et al. [4] and Chen et al. [11], who expanded the model and proposed stronger security definitions to better capture realistic adversarial capabilities.

A notable advancement was made by Lehmann and Tackmann [18], who introduced ciphertext-independent UE, allowing the same update token to apply uniformly across all ciphertexts within a given epoch. They also explored bi-directional key updates, where ciphertexts can be updated both forward and backward in time. This framework was refined by follow-up work from Klooss et al. [17],

Boyd et al. [6], and Jiang [15], who examined the interplay between directionality, update efficiency, and security.

A major theoretical contribution came from Nishimaki [21], who emphasized the importance of update direction in determining security guarantees. He distinguished between forward-leak and backward-leak uni-directional schemes: forward-leak tokens can derive future keys, while backward-leak tokens derive only past keys. He demonstrated that backward-leak uni-directionality provides strictly stronger security than both forward-leak and bi-directional models, establishing a clear hierarchy among directional schemes. This result redefined the understanding of secure key updates and has since become a foundational insight in UE research.

Jiang and Pan [16] extended this line of research by showing that backward-leak uni-directional schemes can achieve security guarantees equivalent to those of no-directional schemes, which currently represent the strongest known model. Independently, Slamanig and Striecks [25] proposed a pairing-based, no-directional UE construction that incorporates an expiry mechanism. In their scheme, each ciphertext is associated with an explicit expiration epoch $e_\perp$, beyond which it becomes undecryptable. Specifically, if a token Δ_{e+1} is used to update a ciphertext past its expiry (i.e., when $e+1 > e_\perp$), the ciphertext becomes irrecoverable. This enhanced model invalidates the equivalence result previously established by Jiang [15] and enables uni-directional updates in a setting that does not impose directional constraints. Their construction is proven secure under the SXDH assumption, with ciphertext and key sizes growing as $\mathcal{O}\big((\log^2 T)\big)$, where T denotes the total number of epochs. In contrast, Nishimaki [21] achieves post-quantum security in the backward-leak model while keeping ciphertext and key sizes independent of T. His scheme builds on a variant of the Regev encryption scheme [24], leveraging the hardness of the LWE problem to ensure resistance against quantum attacks. Our construction follows this approach: it also achieves post-quantum security in the backward-leak uni-directional setting and maintains constant ciphertext and key sizes, regardless of the number of epochs.

The construction proposed in this paper is based on the computational hardness of the LWE problem, introduced by Regev [24], which is widely believed to be resistant even to quantum attacks. This assumption forms the foundation of our scheme's security. We specifically build on FrodoPKE [1, 2], a lattice-based public-key encryption scheme whose security is directly grounded in the LWE assumption [24]. As a result, FrodoPKE offers strong protection against both classical and quantum adversaries, making it highly suitable for post-quantum cryptographic applications. Although FrodoPKE and Regev's original encryption scheme differ in formal aspects such as message encoding, matrix dimensions, and noise generation, they share a common conceptual framework. Both follow the same high-level structure: a public key formed from noisy linear combinations, encryption using randomized inputs and controlled noise, and decryption through careful noise cancellation. FrodoPKE can therefore be viewed as a robust and scalable realization of Regev's foundational idea. Given this close conceptual alignment, it is reasonable to expect that FrodoPKE retains the same desirable characteristics as Regev's scheme when used as a foundation for building LWE-based updatable encryption. By combining the strong directional guarantees of backward-leak updatable encryption [21] with the post-quantum security of FrodoPKE, our construction achieves an effective balance between efficiency, simplicity, and long-term robustness. Compared to earlier bi-directional and pairing-based approaches [18, 6], it offers clear advantages in both practical deployment and future-proof security.

3. Preliminaries

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

Vectors are in column form and they 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 entry of a vector $\mathbf{x}$ is denoted x_i, and the jth column of a matrix $\mathbf{A}$ is denoted $\mathbf{a}_j$ or $\mathbf{A}[j]$. We identify a matrix $\mathbf{A}$ with the ordered set $\{\mathbf{a}_j\}$ of its column vector. For convenience, we define the norm of a matrix $\mathbf{A}$ as the maximum absolute value of its entries, i.e. $\|\mathbf{A}\|_{\max} = \max_j \|\mathbf{a}_j\|_\infty$. Note that usually $\|\mathbf{A}\|_{\max} \neq \|\mathbf{A}\|_\infty$ $(= \sup_{\|\mathbf{x}\|_\infty=1} \|\mathbf{A}\mathbf{x}\|_\infty)$. Roughly speaking, $\|\cdot\|_{\max}$ treats $\mathbf{A}$ more as a (multi) vector rather than an operator. If the columns of $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_k\}$ are linearly independent, then $\mathbf{A}^* = \{\mathbf{a}_1^*, \dots, \mathbf{a}_k^*\}$ denote the Gram-Schmidt orthogonalization of vectors $\mathbf{a}_1, \dots, \mathbf{a}_k$ 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 $\mathcal{X} = \{X_n\}_{n\in I}, \mathcal{Y} = \{Y_n\}_{n\in I}$ be two families of random variables such that X_n, Y_n take values in a finite set $\mathcal{R}_n$. We call $\mathcal{X}$ and $\mathcal{Y}$ *statistically/perfectly indistinguishable* 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 function $\Delta(X_n, Y_n) = \frac{1}{2}\sum_{r\in\mathcal{R}_n} |\Pr[X_n = r] - \Pr[Y_n = r]|$. (See [14] for more details).

Lemma 3.1. (smudging lemma ([3]))
Let $B_1 = B_1(n)$, and $B_2 = B_2(n)$ be positive integers and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \xleftarrow{\$} [-B_2, B_2]$ be chosen uniformly at random. If $B_1/B_2 = \mathsf{negl}(n)$, the distribution of e_2 is statistically indistinguishable from that of $e_2 + e_1$.

3.1. Lattices

In this subsection, we define lattices and review some of their fundamental properties. For a thorough and accessible introduction to lattice theory, we refer the reader to [22].

A subset $\mathcal{L} \subseteq \mathbb{R}^m$ is called an m-dimensional *lattice* if it is a discrete additive subgroup of $\mathbb{R}^m$. That is, $\mathcal{L}$ is a lattice if and only if the following conditions are satisfied:

1. $\mathbf{0} \in \mathcal{L}$, and for any $\mathbf{v}, \mathbf{w} \in \mathcal{L}$, we have $-\mathbf{v}$, $\mathbf{v} + \mathbf{w} \in \mathcal{L}$;
2. there exists $\varepsilon > 0$ such that for every $\mathbf{v} \in \mathcal{L}$, the intersection $\mathcal{L} \cap \{\mathbf{w} \in \mathbb{R}^m \mid \|\mathbf{v} - \mathbf{w}\| < \varepsilon\}$ contains only $\mathbf{v}$.

Because a lattice $\mathcal{L}$ is is an additive subgroup of $\mathbb{R}^m$, it induces the 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 $(\mathbf{x} + \mathcal{L}) + (\mathbf{y} + \mathcal{L}) = (\mathbf{x} + \mathbf{y}) + \mathcal{L}$.

Lattices admit a particularly clear representation: for any lattice $\mathcal{L}$, there exists a set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ such that every point in $\mathcal{L}$ is an integer linear combination of the vectors in $\mathbf{B}$. In other words,

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

The set $\mathbf{B}$ is called a *basis* of the lattice $\mathcal{L}$, and its cardinality $k = \#\mathbf{B}$ is referred to as the *rank* of $\mathcal{L}$. If $k = m$, we say that $\mathcal{L}$ is a *full-rank* lattice. A lattice basis $\mathbf{B}$ is not unique; in fact, for any unimodular matrix $\mathbf{U} \in \mathbb{Z}^{m \times m}$ (i.e., a matrix with $|\det(\mathbf{U})| = 1$), the set $\mathbf{B} \cdot \mathbf{U}$ 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 representative $\mathbf{x} - \mathbf{B} \cdot \lfloor \mathbf{B}^{-1} \cdot \mathbf{x} \rceil$. The measure of fundamental parallelepiped can be easily computed as $\mathrm{vol}\mathcal{P}(\mathbf{B}) = \sqrt{\det \mathbf{B} \cdot \mathbf{B}^T}$, in addition 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 $\mathrm{vol}\mathcal{P}(\mathbf{B}_1) = \mathrm{vol}\mathcal{P}(\mathbf{B}_2)$. Therefore, the measure of fundamental parallelepipeds is invariant of the lattice and is called the *determinant* of the lattice $\mathcal{L}$ and denoted by $\det \mathcal{L}$.

A full-rank lattice $\mathcal{L}$ is called an *integer lattice* if $\mathcal{L} \subseteq \mathbb{Z}^m$. An integer lattice is called a *q-ary lattice* if it satisfies $q\mathbb{Z}^m \subseteq \mathcal{L} \subseteq \mathbb{Z}^m$ for some $q \in \mathbb{Z}_{\geq 1}$.

By definition, a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ is an integer lattice if and only if $\mathbf{B} \in \mathbb{Z}^{m \times m}$ is an integer square matrix. In this case, the determinant $\det \mathcal{L} = |\det \mathbf{B}|$ is a positive integer. It is easy to show that every integer lattice is also a $(\det \mathcal{L})$-ary lattice. For a q-ary lattice $\mathcal{L}$, the following elementary algebraic facts hold:

- $\mathbb{Z}^m / q\mathbb{Z}^m \cong \mathbb{Z}_q^m$ (as additive groups);
- $\mathbb{Z}^m / \mathcal{L} \cong \mathbb{Z}_q^m / (\mathcal{L} \bmod q)$ (as additive groups).

Moreover, $\mathbb{Z}^m / \mathcal{L}$ is a finite group, and its order is given by $|\mathbb{Z}^m / \mathcal{L}| = \det \mathcal{L}$.

3.2. Gaussians

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

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

and

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

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

For any real $s > 0$, *the rounded Gaussian distribution* with parameter (or width) s, denoted Ψ_s, is the distribution over $\mathbb{Z}$ obtained by rounding a sample from $\mathcal{D}_s$ to the nearest integer:

$$\Psi_s(x) = \int_{\{z \mid \lfloor z \rceil = x\}} \mathcal{D}_s(z)dz.$$

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

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

For notational convenience, we let

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

We collect several known results from the literature concerning discrete Gaussians over lattices, specialized to the family relevant to our setting.

Lemma 3.2. Let $n < m$, and let $\mathbf{T}$ be any basis of the lattice $\mathcal{L}$ corresponding to some matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ 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 assume that $s \geq \|\mathbf{T}^*\| \cdot \omega\left(\sqrt{\log m}\right)$. Then:

1. [20, 10]: $\Pr_{\mathbf{x}\leftarrow\mathcal{D}_{\mathcal{L},s,\mathbf{c}}}\left[\|\mathbf{x}-\mathbf{c}\| > s\cdot\sqrt{m}\right] \leq \mathsf{negl}(n)$.
2. [23, 10]: $\Pr_{\mathbf{x}\leftarrow\mathcal{D}_{\mathcal{L},s}}\left[\mathbf{x} = 0\right] \leq \mathsf{negl}(n)$.

4. Security Model for Updatable Encryption

In this section, we present the formal security model for updatable encryption. We begin by defining the leakage sets and then describe key leakage and token leakage, followed by a detailed description of the adversary's available oracles within the security game. Finally, we outline the indistinguishability experiment under chosen-plaintext attacks, which forms the foundation of our security definition.

Leakage sets. The security notions for UE are based on the concept of *leakage sets*, originally introduced by Klooss et al. [17] and Jiang [15]. These sets are used to track the epochs in which the adversary has compromised a key, an update token, or a version of a challenge-equal ciphertext.

- $\mathcal{K}$: set of epochs in which the adversary corrupted the epoch key through $\mathcal{O}.\mathsf{Corr}$;
- $\mathcal{T}$: set of epochs in which the adversary corrupted the update token through $\mathcal{O}.\mathsf{Corr}$;
- $\mathcal{C}$: set of epochs in which the adversary obtained a challenge-equal ciphertext through $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$.

In addition to these sets, it is also necessary to track ciphertexts and their updates that are accessible to the adversary:

- $\mathcal{L}$: the set of non-challenge ciphertexts of the form $(\mathsf{qid}, \mathsf{ct}, \mathsf{e}; \mathbf{m})$, where qid is a query identifier incremented with each call to $\mathcal{O}.\mathsf{Enc}$, ct is the ciphertext, e is the epoch in which the encryption occurred, and $\mathbf{m}$ is the plaintext. These ciphertexts are learned through $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$;
- $\widetilde{\mathcal{L}}$: the set of challenge-equal ciphertexts of the form $(\widetilde{\mathsf{ct}}, \mathsf{e})$, where $\widetilde{\mathsf{ct}}$ is the challenge-equal ciphertext and e is the associated epoch. These ciphertexts are revealed through $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$.

The sets $\mathcal{L}$ and $\widetilde{\mathcal{L}}$, along with their respective extensions, are used exclusively in the deterministic variants of UE, as detailed by Jiang [15]. In the randomized setting, however, two plaintext-based sets, $\mathcal{Q}^*$ and $\widetilde{\mathcal{Q}}^*$, are introduced instead. Specifically, $\mathcal{Q}^*$ consists of pairs $(\mathbf{m}, \mathsf{e})$ such that the adversary has learned or is able to produce a ciphertext in epoch e under the message $\mathbf{m}$. The set $\widetilde{\mathcal{Q}}^*$ consists of pairs $(\bar{\mathbf{m}}, \mathsf{e})$ or $(\bar{\mathbf{m}}_1, \mathsf{e})$, where $(\bar{\mathbf{m}}, \bar{\mathsf{ct}})$ is the input to the challenge query $\mathcal{O}.\mathsf{Chall}$ and $\bar{\mathbf{m}}_1$ is the underlying message of the challenge ciphertext $\bar{\mathsf{ct}}$. These represent challenge-equal ciphertexts that the adversary can generate in epoch e under either message. This distinction enables a one-to-one correspondence between ciphertexts and their associated messages in both the deterministic and randomized settings. In contrast, the scheme presented in this paper is purely randomized, and we therefore do not consider the deterministic case.

Note that, since the adversary may learn or infer additional information through known tokens, the extended sets $\mathcal{K}^*$, $\mathcal{T}^*$, $\mathcal{C}^*$, $\mathcal{L}^*$, and $\widetilde{\mathcal{L}}^*$ are defined as supersets of the original sets $\mathcal{K}$, $\mathcal{T}$, $\mathcal{C}$, $\mathcal{L}$, and $\widetilde{\mathcal{L}}$, respectively.

Key Leakage. Key updates can be classified into three distinct cases; see Jiang [15] for a more detailed treatment:

- the pair $(\mathsf{k}_\mathsf{e}, \Delta_{\mathsf{e}+1})$ does not allow the derivation of $\mathsf{k}_{\mathsf{e}+1}$ (no-directional key update);
- the pair $(\mathsf{k}_\mathsf{e}, \Delta_{\mathsf{e}+1})$ allows the derivation of $\mathsf{k}_{\mathsf{e}+1}$ (uni–directional key update);
- either $(\mathsf{k}_\mathsf{e}, \Delta_{\mathsf{e}+1})$ or $(\mathsf{k}_{\mathsf{e}+1}, \Delta_{\mathsf{e}+1})$ enables the derivation of $\mathsf{k}_{\mathsf{e}+1}$ or k_e, respectively (bi-directional key update).

At first glance, it may seem that a bi-directional key update implies a uni-directional key update. However, this is not the case, as unidirectionality is defined precisely by the restriction that updates are possible in only one direction. In particular, this property does not hold in the bi-directional setting. As noted by Nishimaki [21], the conventional classification into no-, uni-, and bi-directional key updates does not encompass all possible cases. He highlighted a particularly important additional scenario: a reverse form of unidirectionality, in which knowledge of $(\mathsf{k}_{\mathsf{e}+1}, \Delta_{\mathsf{e}+1})$ enables the derivation of k_e. This case merits separate treatment as it represents a distinct class of key update directionality. As a result, the notion of uni-directional key update was refined into two distinct subtypes: *forward-leak uni-directional key update* (f-uni), which corresponds to the original uni model (i.e., $\mathcal{K}^*_{\mathsf{f\text{-}uni}} = \mathcal{K}^*_{\mathsf{uni}}$), and *backward-leak uni-directional key update* (b-uni). If $l + 1 \in \mathbb{Z}_{>0}$ denotes the total number of epochs, the set $\mathcal{K}^*_{\mathsf{b\text{-}uni}}$ is defined as follows:

$$\mathcal{K}^*_{\mathsf{b\text{-}uni}} \leftarrow \{\mathsf{e} \in [0,l] \mid \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\},$$

$$\text{where } \mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \Leftrightarrow (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}+1) \wedge (\mathsf{e}+1) \in \mathcal{T}).$$

Interestingly, b-uni-directional key updates provide stronger security guarantees than their bi-directional counterparts. Furthermore, Jiang, Galteland, and Pan [16] established a striking result: security in the backward-leak uni-directional setting is equivalent to that of the no-directional setting.

Token Leakage. A token is considered known to the adversary if it has either been explicitly corrupted or can be inferred from two consecutive epoch keys. Accordingly, the extended set $\mathcal{T}^*_{\mathsf{b\text{-}uni}}$ is defined as a function of $\mathcal{T}$ and $\mathcal{K}^*_{\mathsf{b\text{-}uni}}$:

$$\mathcal{T}^*_{\mathsf{b\text{-}uni}} \leftarrow \{\mathsf{e} \in [0,l] \,|\, (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^*_{\mathsf{b\text{-}uni}} \wedge (\mathsf{e}-1) \in \mathcal{K}^*_{\mathsf{b\text{-}uni}})\}\,.$$

Oracles in security games for updatable encryption. The security game corresponding to the variant of UE considered in this work involves the following oracles: $\mathbf{Setup}$, $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Dec}$, $\mathcal{O}.\mathsf{Next}$, $\mathcal{O}.\mathsf{Upd}$, $\mathcal{O}.\mathsf{Corr}$, $\mathcal{O}.\mathsf{Chall}$, and $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$.

$\mathbf{Setup}(1^n)$

$\mathsf{e} \leftarrow 0$
$\mathsf{k_e} \xleftarrow{\$} \mathsf{UE.KG}(1^n), \Delta_\mathsf{e} \leftarrow \perp$
$\mathsf{qid} \leftarrow 0, \mathsf{twf} \leftarrow 0, \mathsf{phase} \leftarrow 0$
$\mathcal{L}, \widetilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$

$\mathcal{O}.\mathsf{Enc}(\mathbf{m})$

$\mathsf{qid} \leftarrow \mathsf{qid} + 1$
$\mathsf{ct} \xleftarrow{\$} \mathsf{UE.Enc}(\mathsf{k_e}, \mathbf{m})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathsf{qid}, \mathsf{ct}, \mathsf{e}; \mathbf{m}\}$
return ct

$\mathcal{O}.\mathsf{Dec}(\mathsf{ct})$

$\mathbf{m}'$ or $\perp \leftarrow \mathsf{UE.Dec}(\mathsf{k_e}, \mathsf{ct})$
if $(\mathbf{m}', \mathsf{e}) \in \widetilde{\mathcal{Q}}^*$ **then**
  $\mathsf{twf} \leftarrow 1$
return $\mathbf{m}'$ **or** $\perp$

$\mathcal{O}.\mathsf{Upd}(\mathsf{ct}_{\mathsf{e}-1})$

if $(j, \mathsf{ct}_{\mathsf{e}-1}, \mathsf{e}-1, \mathbf{m}) \notin \mathcal{L}$ **then**
  return $\perp$
$\mathsf{ct_e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \mathsf{ct}_{\mathsf{e}-1})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, \mathsf{ct_e}, \mathsf{e}; \mathbf{m})\}$
return $\mathsf{ct_e}$

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

if $\hat{\mathsf{e}} > \mathsf{e}$ **then**
  return $\perp$
if $\mathsf{inp} = \mathsf{key}$ **then**
  $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
  return $\mathsf{k}_{\hat{\mathsf{e}}}$
if $\mathsf{inp} = \mathsf{token}$ **then**
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
  return $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Next}()$

$\mathsf{e} \leftarrow \mathsf{e} + 1$
$\mathsf{k}_\mathsf{e} \xleftarrow{\$} \mathsf{UE.KG}(1^n)$
$\Delta_\mathsf{e} \leftarrow \mathsf{UE.TG}(\mathsf{k}_{\mathsf{e}-1}, \mathsf{k}_\mathsf{e})$
if $\mathsf{phase} = 1$ **then**
  $\widetilde{\mathsf{ct}}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \widetilde{\mathsf{ct}}_{\mathsf{e}\text{-}1})$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
  $\widetilde{\mathcal{L}} \leftarrow \widetilde{\mathcal{L}} \cup \{(\widetilde{\mathsf{ct}}_\mathsf{e}, \mathsf{e})\}$

$\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}()$

if $\mathsf{phase} \neq 1$ **then**
  return $\perp$
return $\widetilde{\mathsf{ct}}_\mathsf{e}$

$\mathcal{O}.\mathsf{Chall}(\bar{\mathbf{m}}, \bar{\mathsf{ct}}, \mathsf{b})$

if $\mathsf{phase} = 1$ **then**
  return $\perp$
$\mathsf{phase} \leftarrow 1; \widetilde{\mathsf{e}} \leftarrow \mathsf{e}$
if $(\cdot, \bar{\mathsf{ct}}, \widetilde{\mathsf{e}} - 1, \bar{\mathbf{m}}_1) \notin \mathcal{L}$ **then**
  return $\perp$
if $\mathsf{b} = 0$ **then**
  $\widetilde{\mathsf{ct}}_{\widetilde{\mathsf{e}}} \leftarrow \mathsf{UE.Enc}(\mathsf{k}_{\widetilde{\mathsf{e}}}, \bar{\mathbf{m}})$
else
  $\widetilde{\mathsf{ct}}_{\widetilde{\mathsf{e}}} \leftarrow \mathsf{UE.Upd}(\Delta_{\widetilde{\mathsf{e}}}, \bar{\mathsf{ct}})$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{\widetilde{\mathsf{e}}\}$
$\widetilde{\mathcal{L}} \leftarrow \widetilde{\mathcal{L}} \cup \{(\widetilde{\mathsf{ct}}_{\widetilde{\mathsf{e}}}, \widetilde{\mathsf{e}})\}$
return $\widetilde{\mathsf{ct}}_{\widetilde{\mathsf{e}}}$

Each of the algorithms and global variables defined above plays a specific role within the security experiment. Several global technical variables are maintained to accurately track the evolution of the system state as the experiment progresses. The unsigned integer qid serves as a universal query counter, increasing with each call to the encryption oracle ($\mathcal{O}.\mathsf{Enc}$) and uniquely indexing every query instance. Two Boolean variables further regulate the experiment. The variable twf is set to 1 whenever a trivial or forbidden event arises; for example, this occurs if the adversary obtains the secret key for a particular epoch, which would enable direct decryption of the challenge ciphertext or any of its updated variants within that epoch. The variable phase indicates the current stage of the experiment, with $\mathsf{phase} = 1$ indicating entry into the challenge phase. The main algorithms $\mathbf{Setup}$, $\mathcal{O}.\mathsf{Enc}$, and $\mathcal{O}.\mathsf{Upd}$ adhere to the definition of UE schemes and function as expected in this context. The decryption oracle ($\mathcal{O}.\mathsf{Dec}$) does not participate in our security proofs, since our analysis is limited to the chosen plaintext attack (CPA) setting. Its presence serves only for completeness and for consistency with established literature, where broader models such as chosen ciphertext attacks (CCA) are considered. The $\mathcal{O}.\mathsf{Next}$ oracle advances the experiment to the next epoch by generating a new epoch key and update token. If the experiment is currently in the challenge phase, $\mathcal{O}.\mathsf{Next}$ also updates the challenge ciphertext and records each new version so that all such versions remain available to the adversary for later queries. The $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$ oracle allows the adversary to obtain the current, updated version of the challenge ciphertext for the present epoch, referencing the state maintained by $\mathcal{O}.\mathsf{Next}$. The $\mathcal{O}.\mathsf{Corr}$ oracle provides a means for the adversary to corrupt keys and tokens, thereby obtaining secret keys or update tokens for epochs of its choice. The $\mathcal{O}.\mathsf{Chall}$ oracle initiates the challenge phase by returning either a freshly encrypted message or an updated ciphertext as the challenge, which commences the central distinguishing experiment. Together, these variables and oracles, combined with the definition of the relevant leakage sets, establish a rigorous and comprehensive framework for formalizing the adversary's capabilities and for modeling the progression of the security experiment in the context of updatable encryption.

Challenge-Equal Ciphertext Leakage. The adversary learns all challenge-equal ciphertexts in epochs from the set $\mathcal{C}$. Additionally, such ciphertexts can be inferred via known update tokens, which may reveal further information through indirect derivations. It is easy to see that the structure of the extended set $\mathcal{C}^*$ depends on whether ciphertexts are updated in a uni- or bi-directional manner. Importantly, it remains independent of whether key and token updates follow the b-uni-directional model. Let $\mathsf{cc} \in \{\mathsf{uni}, \mathsf{bi}\}$; the procedure for computing the sets $\mathcal{C}^*_{\mathsf{b\text{-}uni},\mathsf{cc}}$ is described by Jiang [15].

Security of UE scheme in CPA model.

$\mathbf{Exp}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa\text{-}0}}_{\mathsf{UE},\mathcal{A}}$

1 : **do** $\mathbf{Setup}(1^n)$
2 : $\mathsf{b}' \leftarrow \mathcal{A}^{\mathsf{oracles(b=0)}}(1^n)$
3 : **if** $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ **then**
4 : $\quad \mathsf{twf} \leftarrow 1$
5 : **if** $\mathsf{twf} = 1$ **then**
6 : $\quad \mathsf{b}' \xleftarrow{\$} \{0,1\}$
7 : **return** b'

$\mathbf{Exp}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa\text{-}1}}_{\mathsf{UE},\mathcal{A}}$

1 : **do** $\mathbf{Setup}(1^n)$
2 : $\mathsf{b}' \leftarrow \mathcal{A}^{\mathsf{oracles(b=1)}}(1^n)$
3 : **if** $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ **then**
4 : $\quad \mathsf{twf} \leftarrow 1$
5 : **if** $\mathsf{twf} = 1$ **then**
6 : $\quad \mathsf{b}' \xleftarrow{\$} \{0,1\}$
7 : **return** b'

Although our UE scheme is a general randomized construction, it can be readily reformulated as a deterministic one. Nevertheless, in this work, we focus exclusively on the randomized setting and accordingly tailor the security model to this case. The security definition follows the standard semantic security approach, in which the adversary must distinguish between two experiments, namely $\mathbf{Exp}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa\text{-}0}}_{\mathsf{UE},\mathcal{A}}$ and $\mathbf{Exp}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa\text{-}1}}_{\mathsf{UE},\mathcal{A}}$. We emphasize that the proposed scheme achieves indistinguishability under chosen-plaintext attacks (CPA).

A UE scheme is said to be rand-ind-eu-cpa-secure if, for every PPT adversary $\mathcal{A}$, the following advantage is negligible as a function of the security parameter n:

$$\mathsf{Adv}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa}}_{\mathcal{A}}(n) := \left| \Pr\left[\mathbf{Exp}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa\text{-}0}}_{\mathsf{UE},\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa\text{-}1}}_{\mathsf{UE},\mathcal{A}} = 1\right] \right|.$$

5. Frodo based UE scheme

5.1. Frodo KEM

Since the construction of our UE scheme is based on FrodoPKE, we begin by presenting the framework for the Frodo learning with errors key encapsulation mechanism [1, 2]. To that end, let $D \in \mathbb{Z}_{>0}$ and define $q = 2^D$, with $B \in \mathbb{Z}_{[0,D]}$, and $n, \bar{m}, \bar{n} \in \mathbb{Z}_{>0}$, where $n \equiv 0 \pmod 8$. Additionally, let $\mathcal{M} = \{0,1\}^\ell$ denote the message space (see below).

The error distribution χ used in FrodoPKE is a discrete, symmetric distribution over $\mathbb{Z}$, centered at zero with small support. It approximates a rounded continuous Gaussian distribution. The support of χ is defined as $S_\chi = \{-s, -s+1, \ldots, -1, 0, 1, \ldots, s-1, s\}$ for some $s \in \mathbb{Z}_{>0}$. The probabilities

satisfy $\chi(z) = \chi(-z)$ for all $z \in S_\chi$ and are determined by a discrete probability density function represented as a table $T_\chi = (T_\chi(0), T_\chi(1), \ldots, T_\chi(s))$ of $s+1$ positive integers, which correspond to cumulative distribution function. Sampling from χ is performed using inversion sampling techniques; see [1, 2] for full details. The distribution χ plays a central role in the algorithm Frodo.SampleMatrix (see [1, 2]), which is used to sample error matrices. For notational simplicity, the output of this algorithm will be denoted as $\mathbf{E} \xleftarrow{\$} \chi(\mathbb{Z}_q^{\dim})$ throughout the paper.

The public-key encryption scheme FrodoPKE serves as a foundational component of our construction and, as mentioned above, is based on the computational hardness of the LWE problem. This assumption provides strong security guarantees against both classical and quantum adversaries. FrodoPKE follows the standard architecture of lattice-based encryption schemes and consists of four PPT algorithms. Formally, it is defined as the following tuple: FrodoPKE is defined as a tuple of algorithms: (FrodoPKE.Setup, FrodoPKE.KeyGen, FrodoPKE.Enc, FrodoPKE.Dec), each of which is described in detail below.

FrodoPKE.Setup(1^n)

1 : $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times n}$
2 : parameters $\leftarrow (\mathbf{A}, 1^n, D, B, n, \bar{m}, \bar{n})$
3 : **return** parameters

FrodoPKE.KeyGen(parameters)

1 : $\mathbf{S}, \mathbf{E} \xleftarrow{\$} \chi(\mathbb{Z}_q^{n \times \bar{n}})$
2 : $\mathbf{B} \leftarrow \mathbf{A} \cdot \mathbf{S} + \mathbf{E} \in \mathbb{Z}_q^{n \times \bar{n}}$
3 : **return** pk $\leftarrow \mathbf{B}$, sk $\leftarrow \mathbf{S}$

FrodoPKE.Enc(pk, $\mathbf{m} \in \mathcal{M}$)

1 : $\mathbf{S}', \mathbf{E}' \xleftarrow{\$} \chi(\mathbb{Z}_q^{\bar{m} \times n})$
2 : $\mathbf{E}'' \xleftarrow{\$} \chi(\mathbb{Z}_q^{\bar{m} \times \bar{n}})$
3 : $\mathbf{B}' \leftarrow \mathbf{S}' \cdot \mathbf{A} + \mathbf{E}' \in \mathbb{Z}_q^{\bar{m} \times n}$
4 : $\mathbf{V} \leftarrow \mathbf{S}' \cdot \mathbf{B} + \mathbf{E}'' \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$
5 : $\mathbf{C}_1 \leftarrow \mathbf{B}'$, $\mathbf{C}_2 \leftarrow \mathbf{V} +$ encode$(\mathbf{m})$
6 : **return** ct $\leftarrow (\mathbf{C}_1, \mathbf{C}_2)$

FrodoPKE.Dec(ct, sk)

1 : $\mathbf{M} \leftarrow \mathbf{C}_2 - \mathbf{C}_1 \cdot \mathbf{S} \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$
2 : $\mathbf{m}' \leftarrow$ decode$(\mathbf{M})$
3 : **return** $\mathbf{m}'$

Although the functions encode and decode are formally defined in the Frodo specifications [1, 2], we briefly outline the intuition behind their construction. The encode function maps bit strings of length $\ell = B \cdot \bar{m} \cdot \bar{n}$ to $\bar{m} \times \bar{n}$ matrices over $\mathbb{Z}_q$. To achieve this, each consecutive B-bit substring is interpreted as an integer $k \in [0, 2^B)$ and then mapped to an element of $\mathbb{Z}_q$ via the transformation $\mathsf{ec}(k) = k \cdot q/2^B$. The corresponding decode function inverts this process using the operation $\mathsf{dc}(c) = \lfloor c \cdot 2^B/q \rceil \pmod{2^B}$.

The scheme is correct as long as the error remains sufficiently small.

Lemma 5.1. ([1]) Let $q = 2^D$ and $B \leq D$. Then, for any integer $0 \leq k < 2^B$ and any error term e satisfying $-q/2^{B+1} \leq e < q/2^{B+1}$, it holds that $\mathsf{dc}(\mathsf{ec}(k) + e) = k$.

This result ensures that the decoding procedure remains robust to small additive errors introduced during encryption, as long as the noise stays within a carefully bounded interval. Intuitively, the

rounding operation at the heart of decode acts as a form of noise tolerance, enabling perfect recovery of the original message bit string despite the presence of moderate LWE noise.

6. Construction of EU scheme based on Frodo

To update a ciphertext ct_e, we homomorphically decrypt it using the update token Δ_{e+1} and re-encrypt the resulting message under the next epoch's public key, yielding $\mathsf{ct}_{e+1} \leftarrow \mathsf{FrodoPKE.Enc}(\mathsf{pk}_{e+1}, \mathbf{m})$. The update token Δ_{e+1} can be naturally viewed as a homomorphic encryption of the secret key sk_e under the public key pk_{e+1}. This transformation relies on two supporting functions: *bit ordering* (Ord) and *Tensor-D* ($\bigotimes_D$), both of which are formally defined below. These functions are designed to enable the structured manipulation of ciphertexts necessary for homomorphic key updates within the lattice-based framework. The overall approach is motivated by the key-switching technique originally introduced by Brakerski et al. [8, 7, 9], which enables ciphertexts to be transformed from one encryption key to another. It also draws on the noise smudging lemma [3, 21], which is essential for ensuring that the update process remains secure and error-resilient.

Bit-ordering (Ord). This function takes as input a matrix $\mathbf{X} \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$ and produces an output matrix $\mathbf{Y} \in \{0,1\}^{\bar{m} \times \bar{n} D}$. Let $\mathbf{X}[i, \cdot] = [x_{i1}, x_{i2}, \ldots, x_{i\bar{n}}]$ denote the i-th row of $\mathbf{X}$. Each entry $x_{i,j} \in \mathbb{Z}_q$ is represented using D bits, meaning it can be written as $x_{i,j} = \sum_{k=1}^{D} 2^{k-1} y_{i,j,k}$. The operator Ord transforms the row $\mathbf{X}[i, \cdot]$ into the i-th row of $\mathbf{Y}$, given by $\mathbf{Y}[i, \cdot] = [\mathbf{y}_{i1}, \mathbf{y}_{i2}, \ldots, \mathbf{y}_{iD}]$, where each $\mathbf{y}_{i,k} = [y_{i,j,k}]_j \in \{0,1\}^{\bar{n}}$.

Tensor-D ($\bigotimes_D$). This function transforms a matrix from $\mathbb{Z}_q^{n \times \bar{n}}$ to $\mathbb{Z}_q^{nD \times \bar{n}}$. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\bar{n}}] \in \mathbb{Z}_q^{n \times \bar{n}}$, where $\mathbf{x}_j$ represents the j-th column of $\mathbf{X}$. The transformation $\bigotimes_D(\mathbf{X}) = \mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{\bar{n}}] \in \mathbb{Z}_q^{nD \times \bar{n}}$, where each column $\mathbf{y}_j$ is defined as $\mathbf{y}_j = [1, 2, \ldots, 2^{D-1}]^T \otimes \mathbf{x}_j$.

Formal product Consider matrices $\mathbf{A} \in \mathbb{Z}_q^{l \times r}$, $\mathbf{B} \in \mathbb{Z}_q^{r \times s_1}$, and $\mathbf{C} \in \mathbb{Z}_q^{r \times s_2}$. The formal product $\mathbf{A} \cdot (\mathbf{B}, \mathbf{C})$ represents the multiplication of $\mathbf{A}$ with the concatenation of $\mathbf{B}$ and $\mathbf{C}$, denoted as $\mathbf{A} \cdot [\mathbf{B} \mid \mathbf{C}]$. This operation yields $[\mathbf{A} \cdot \mathbf{B} \mid \mathbf{A} \cdot \mathbf{C}]$. By partitioning the resulting matrix along the concatenation, we establish the formal equivalence $(\mathbf{A} \cdot \mathbf{B}, \mathbf{A} \cdot \mathbf{C}) = \mathbf{A} \cdot (\mathbf{B}, \mathbf{C})$.

Remark 6.1. Note that if $\mathbf{C}$ and $\mathbf{S}$ belong to the domains of Bit-ordering and Tensor-D, respectively, then the following equality holds $\mathsf{Ord}(\mathbf{C}) \cdot \bigotimes_D(\mathbf{S}) = \mathbf{C} \cdot \mathbf{S}$.

With a comprehensive understanding of the bit-ordering, tensor-D and formal product functions established, we can now delve into the specifics of our UE scheme. These foundational components are integral to the scheme's architecture, enabling efficient key updates and ensuring the continuous security of encrypted data. The following exposition will detail the design principles, operational procedures, and the interplay of these core functions that facilitate seamless and secure encryption transitions.

UE.Setup(1^n)

1 : parameters $\xleftarrow{\$}$ FrodoPKE.Setup(1^n)
2 : parameters $= (\mathbf{A}, 1^n, D, B, n, \bar{m}, \bar{n})$
3 : **return** parameters

UE.KG(parameters)

1 : $(\mathbf{B}_e, \mathbf{S}_e) \xleftarrow{\$}$ FrodoPKE.KeyGen(parameters)
2 : $\mathsf{sk}_e \leftarrow \mathbf{S}_e,\ \mathsf{pk}_e \leftarrow \mathbf{B}_e$
3 : **return** $\mathsf{k}_e \leftarrow (\mathsf{sk}_e, \mathsf{pk}_e)$

UE.Enc($\mathsf{k}_e, \mathbf{m} \in \mathcal{M}$)

1 : $\mathsf{k}_e = (\mathbf{S}_e, \mathbf{B}_e)$
2 : $(\mathbf{C}_1, \mathbf{C}_2) \xleftarrow{\$}$ FrodoPKE.Enc($\mathbf{B}_e, \mathbf{m}$)
3 : $\mathsf{ct} \leftarrow (\mathbf{C}_1, \mathbf{C}_2) \in \mathbb{Z}_q^{\bar{m}\times n} \times \mathbb{Z}_q^{\bar{m}\times\bar{n}}$
4 : **return** ct

UE.Dec($\mathsf{k}_e, \mathsf{ct}$)

1 : $\mathsf{k}_e = (\mathbf{S}_e, \mathbf{B}_e),\ \mathsf{ct} = (\mathbf{C}_1, \mathbf{C}_2)$
2 : $\mathbf{m}' \leftarrow$ FrodoPKE.Dec(ct, $\mathbf{S}_e$)
3 : **return** $\mathbf{m}'$

UE.TG($\mathsf{k}_e, \mathsf{k}_{e+1}$)

1 : $\mathsf{k}_e = (\mathbf{S}_e, \mathbf{B}_e),\ \mathsf{k}_{e+1} = (\mathbf{S}_{e+1}, \mathbf{B}_{e+1})$
2 : $\mathbf{S}'_{(1)}, \mathbf{E}'_{(1)} \xleftarrow{\$} \chi(\mathbb{Z}_q^{nD\times n}),\ \mathbf{E}''_{(1)} \xleftarrow{\$} \chi(\mathbb{Z}_q^{nD\times\bar{n}})$
3 : $\mathbf{S}'_{(2)}, \mathbf{E}'_{(2)} \xleftarrow{\$} \chi(\mathbb{Z}_q^{n\times n}),\ \mathbf{E}''_{(2)} \xleftarrow{\$} \chi(\mathbb{Z}_q^{n\times\bar{n}})$
4 : $\Delta_{e+1}^{(1)} \leftarrow \left(\mathbf{S}'_{(1)} \cdot \mathbf{A} + \mathbf{E}'_{(1)}, \mathbf{S}'_{(1)} \cdot \mathbf{B}_{e+1} + \mathbf{E}''_{(1)} - \bigotimes\nolimits_D(\mathbf{S}_e)\right)$
5 : $\Delta_{e+1}^{(2)} \leftarrow \left(\mathbf{S}'_{(2)} \cdot \mathbf{A} + \mathbf{E}'_{(2)}, \mathbf{S}'_{(2)} \cdot \mathbf{B}_{e+1} + \mathbf{E}''_{(2)}\right)$
6 : **return** $\Delta_{e+1} \leftarrow (\Delta_{e+1}^{(1)}, \Delta_{e+1}^{(2)})$

UE.Upd($\Delta_{e+1}, \mathsf{ct}_e$)

1 : $\Delta_{e+1} = (\Delta_{e+1}^{(1)}, \Delta_{e+1}^{(2)}),\ \mathsf{ct}_e = (\mathbf{C}_{e,1}, \mathbf{C}_{e,2})$
2 : $(\mathbf{C}_1^{(1)}, \mathbf{C}_2^{(1)}) \leftarrow \mathsf{Ord}(\mathbf{C}_{e,1}) \cdot \Delta_{e+1}^{(1)}$
3 : $\mathbf{R} \xleftarrow{\$} \chi(\mathbb{Z}_q^{\bar{m}\times n})$
4 : $(\mathbf{C}_1^{(2)}, \mathbf{C}_2^{(2)}) \leftarrow \mathbf{R} \cdot \Delta_{e+1}^{(2)}$
5 : $\mathsf{ct}_{e+1} \leftarrow \left(\mathbf{C}_1^{(1)} + \mathbf{C}_1^{(2)}, \mathbf{C}_{e,2} + \mathbf{C}_2^{(1)} + \mathbf{C}_2^{(2)}\right)$
6 : **return** ct_{e+1}

Remark 6.2. Note that to generate the token Δ_{e+1}, UE.TG only needs sk_e and pk_{e+1}, rather than the complete keys k_e and k_{e+1}. This aligns with the idea introduced earlier.

The following theorem outlines the conditions under which the scheme operates correctly.

Theorem 6.3. Suppose χ is the error distribution defined in Sec. 5.1, with support $S_\chi = \{-s, \ldots, 0, \ldots, s\}$. Assume T is the maximum number of epochs. If

$$2(n^2Ds^3 + n^2s^3) + nDs + ns^2 < \frac{q}{T \cdot 2^{B+1}},$$

then the scheme is correct.

Proof:
Let $\mathsf{ct}_\mathsf{e} = (\mathbf{C}_{\mathsf{e},1}, \mathbf{C}_{\mathsf{e},2})$ and $\mathsf{ct}_{\mathsf{e}+1} = (\mathbf{C}_{\mathsf{e}+1,1}, \mathbf{C}_{\mathsf{e}+1,2})$, where the components are defined as in $\mathsf{UE.Upd}$. It is straightforward to verify that

$$\mathbf{C}_{\mathsf{e}+1,2} - \mathbf{C}_{\mathsf{e}+1,1} \cdot \mathbf{S}_{\mathsf{e}+1} = \mathbf{C}_{\mathsf{e},2} - \mathbf{C}_{\mathsf{e},1} \cdot \mathbf{S}_\mathsf{e} + \mathsf{Error}$$

where the error term is given by

$$\begin{aligned}\mathsf{Error} = \left(\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{S}'_{(1)} + \mathbf{R} \cdot \mathbf{S}'_{(2)}\right) \mathbf{E} - \left(\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{E}'_{(1)} + \mathbf{R} \cdot \mathbf{E}'_{(2)}\right) \mathbf{S}_{\mathsf{e}+1} \\ + \mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{E}''_{(1)} + \mathbf{R} \cdot \mathbf{E}''_{(2)}\end{aligned} \tag{1}$$

Here, the primed variables involving $\mathbf{E}$ (i.e., $\mathbf{E}'_{(1)}, \mathbf{E}'_{(2)}, \mathbf{E}''_{(1)}, \mathbf{E}''_{(2)}$) denote independent noise samples generated within $\mathsf{UE.TG}$, following the structure of LWE-based encryption. The matrix $\mathbf{E}$ denotes the noise introduced when generating (or, in this case, updating) $\mathsf{pk}_{\mathsf{e}+1} = \mathbf{B}_{\mathsf{e}+1}$ from $\mathsf{sk}_{\mathsf{e}+1} = \mathbf{A} \cdot \mathbf{S}_{\mathsf{e}+1}$, in the same form as described in $\mathsf{FrodoPKE.KeyGen}$.
Given that $\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \in \{0,1\}^{\bar{m} \times nD}$, $\mathbf{S}'_{(1)} \in S_\chi^{nD \times n}$, and $\mathbf{E} \in S_\chi^{n \times n}$, we can derive the following bound

$$\left\| \left(\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{S}'_{(1)} + \mathbf{R} \cdot \mathbf{S}'_{(2)}\right) \mathbf{E} \right\|_{\max} \leq n^2Ds^2 + n^2s^3. \tag{2}$$

Similarly, we have

$$\left\| \left(\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{E}'_{(1)} + \mathbf{R} \cdot \mathbf{E}'_{(2)}\right) \mathbf{S}_{\mathsf{e}+1} \right\|_{\max} \leq n^2Ds^2 + n^2s^3, \tag{3}$$

$$\left\| \mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{E}''_{(1)} \right\|_{\max} \leq nDs, \quad \left\| \mathbf{R} \cdot \mathbf{E}''_{(2)} \right\|_{\max} \leq ns^2. \tag{4}$$

By applying the triangle inequality to (1) together with the bounds in (2)–(4), we obtain:

$$\|\mathsf{Error}\|_\infty \leq 2(n^2Ds^3 + n^2s^3) + nDs + ns^2 < q/(T \cdot 2^{B+1}).$$

Therefore, each ciphertext update introduces noise of magnitude at most $q/(T \cdot 2^{B+1})$. It follows that after T updates, the total accumulated noise remains below $q/2^{B+1}$. Hence, by Lemma 5.1, the proof is complete. □

6.1. Security proof

In this section, we prove that the scheme satisfies rand-ind-eu-cpa-security in the backward-leak, unidirectional setting. This security notion ensures that even if secret keys from earlier epochs are leaked, an adversary cannot compromise ciphertexts encrypted under newer public keys. Our proof adopts the *firewall technique*, a methodology introduced by Jiang and Nishimaki [15, 21], which provides a systematic framework for analyzing security in key-evolving encryption schemes.

As a first step, we show that the scheme inherently supports uni-directional ciphertext updates, an essential property that follows directly from its security guarantees.

Theorem 6.4. For any PPT adversary $\mathcal{A}$, the advantage of converting a ciphertext under public key $\mathsf{pk}_{\mathsf{e}+1}$ into a valid ciphertext under pk_{e}, given access to $(\mathsf{sk}_{\mathsf{e}}, \mathsf{pk}_{\mathsf{e}})$, $\mathsf{pk}_{\mathsf{e}+1}$, and $\Delta_{\mathsf{e}+1}$, is negligible.

Proof:
Suppose, for contradiction, that the adversary $\mathcal{A}$ is capable of transforming a ciphertext under $\mathsf{pk}_{\mathsf{e}+1}$ into a valid ciphertext under pk_{e}. Given that FrodoPKE is ind-cpa-secure, we construct a PPT adversary $\mathcal{B}$ that breaks the security of ciphertexts under $\mathsf{pk}_{\mathsf{e}+1}$.

The adversary $\mathcal{B}$ is given the public parameters parameters and the public key $\mathsf{pk}_{\mathsf{e}+1}$. It then generates a key pair $(\mathsf{sk}_{\mathsf{e}}, \mathsf{pk}_{\mathsf{e}}) \leftarrow \mathsf{UE.KG}(\mathsf{parameters})$ and computes the update token $\Delta_{\mathsf{e}+1} \leftarrow \mathsf{UE.TG}(\mathsf{sk}_{\mathsf{e}}, \mathsf{pk}_{\mathsf{e}+1})$ (see Remark 6.2). Next, $\mathcal{B}$ selects two equal-length messages $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}$ and submits them to its ind-cpa challenger. The challenger randomly selects a bit $b \xleftarrow{\$} \{0,1\}$, encrypts $\mathbf{m}_b$ as $\mathsf{ct}^* \leftarrow \mathsf{FrodoPKE.Enc}(\mathsf{pk}_{\mathsf{e}+1}, \mathbf{m}_b)$, and returns the challenge ciphertext ct^* to $\mathcal{B}$. At this point, $\mathcal{B}$ runs $\mathcal{A}$ on input $((\mathsf{sk}_{\mathsf{e}}, \mathsf{pk}_{\mathsf{e}}), \Delta_{\mathsf{e}+1}, \mathsf{ct}^*)$. The adversary $\mathcal{A}$ outputs a ciphertext ct' under pk_{e}. Then, $\mathcal{B}$ decrypts this ciphertext by computing $\mathbf{m}' \leftarrow \mathsf{FrodoPKE.Dec}(\mathsf{ct}', \mathsf{sk}_{\mathsf{e}})$ and outputs a guess b' such that $\mathbf{m}' = \mathbf{m}_{b'}$. Since $\mathcal{A}$ successfully converts the ciphertext with overwhelming probability, it follows that $b' = b$ with non-negligible advantage, contradicting the assumed ind-cpa-security of FrodoPKE. This completes the proof. □

In the next step, we reformulate the security game and propose new hybrid algorithms that simulate the scheme with negligible error, while remaining easier to analyze. To proceed, we revisit the update procedure. By the scheme's definition, we have:

$$
\begin{aligned}
\mathbf{C}_1^{(1)} &= \mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{S}'_{(1)} \cdot \mathbf{A} + \mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{E}'_{(1)}, \\
\mathbf{C}_2^{(1)} &= \mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{S}'_{(1)} \cdot \mathbf{B}_{\mathsf{e}+1} + \mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{E}''_{(1)} - \underbrace{\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \bigotimes_{D}(\mathbf{S}_{\mathsf{e}})}_{\mathbf{C}_{\mathsf{e},1} \cdot \mathbf{S}_{\mathsf{e}}}, \\
\mathbf{C}_1^{(2)} &= \mathbf{R} \cdot \mathbf{S}'_{(2)} \cdot \mathbf{A} + \mathbf{R} \cdot \mathbf{E}'_{(2)}, \\
\mathbf{C}_2^{(2)} &= \mathbf{R} \cdot \mathbf{S}'_{(2)} \cdot \mathbf{B}_{\mathsf{e}+1} + \mathbf{R} \cdot \mathbf{E}''_{(2)}.
\end{aligned}
$$

A straightforward calculation yields

$$\mathbf{C}_1^{(1)} + \mathbf{C}_1^{(2)} = \Bigg(\underbrace{\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{S}'_{(1)} + \mathbf{R} \cdot \mathbf{S}'_{(2)}}_{\mathbf{S}^\dagger}\Bigg) \cdot \mathbf{A} + \underbrace{\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{E}'_{(1)} + \mathbf{R} \cdot \mathbf{E}'_{(2)}}_{\mathbf{E}^\dagger},$$

and

$$\begin{aligned}\mathbf{C}_2^{(1)} + \mathbf{C}_2^{(2)} + \mathbf{C}_{\mathsf{e},2} = &\Bigg(\underbrace{\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{S}'_{(1)} + \mathbf{R} \cdot \mathbf{S}'_{(2)}}_{\mathbf{S}^\dagger}\Bigg) \cdot \mathbf{B}_{\mathsf{e}+1} \\ &+ \underbrace{\mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{E}''_{(1)} + \mathbf{R} \cdot \mathbf{E}''_{(2)} + \mathbf{E}''_{\mathsf{e}}}_{\mathbf{E}^{\dagger\dagger}} \\ &- \underbrace{\mathbf{E}' \cdot \mathbf{S}_{\mathsf{e}} + \mathbf{S}' \cdot \mathbf{E}}_{\text{"garbage"}} + \mathsf{encode}(\mathbf{m})\end{aligned}$$

The primed variants of $\mathbf{E}$ represent independent noise samples generated within UE.TG, while $\mathbf{E}$ refers to the noise used in the public key update. See the explanation in the proof of Theorem 6.3 for further context.

This leads us to the following conclusion.

Conclusion 6.5. By Lemma 3.1, a ciphertext $\mathsf{ct}_{\mathsf{e}+1}$ is statistically indistinguishable from $(\mathbf{S}^\dagger \cdot \mathbf{A} + \mathbf{E}^\dagger, \mathbf{S}^\dagger \cdot \mathbf{B}_{\mathsf{e}+1} + \mathbf{E}^{\dagger\dagger} + \mathsf{encode}(\mathbf{m}))$. This implies that we can simulate an updated ciphertext using the original ciphertext, its associated plaintext and randomness, the new epoch's public key, and fresh randomness for generating the token $\Delta_{\mathsf{e}+1}$.

$\mathsf{Hyb.UE.Upd}(\mathsf{ct}_{\mathsf{e}}, \mathbf{B}_{\mathsf{e}+1}, \mathbf{m}, \mathbf{E}''_{\mathsf{e}}, (\mathbf{S}'_{(1)}, \mathbf{S}'_{(2)}, \mathbf{E}''_{(1)}, \mathbf{E}''_{(2)}))$

1 : Parse $\mathsf{ct}_{\mathsf{e}} = (\mathbf{C}_{\mathsf{e},1}, \mathbf{C}_{\mathsf{e},2})$

2 : Choose $\mathbf{R} \xleftarrow{\$} \chi(\mathbb{Z}_q^{\bar{m} \times n})$

3 : Set $\mathbf{S}^\dagger = \mathsf{Ord}(\mathbf{C}_{\mathsf{e},1}) \cdot \mathbf{S}'_{(1)} + \mathbf{R} \cdot \mathbf{S}'_{(2)}$

4 : Set $\mathsf{ct}_{\mathsf{e}+1} = \big(\mathbf{S}^\dagger \cdot \mathbf{A} + \mathbf{E}^\dagger, \mathbf{S}^\dagger \cdot \mathbf{B}_{\mathsf{e}+1} + \mathbf{E}^{\dagger\dagger} + \mathsf{encode}(\mathbf{m})\big)$

5 : Output $(\mathsf{ct}_{\mathsf{e}+1}, \mathbf{E}''_{\mathsf{e}})$.

$\mathsf{Sim.UE.KG}(\mathsf{parameters})$

1 : Choose $\mathbf{B}_e^+ \xleftarrow{\$} \mathbb{Z}_q^{n \times \bar{n}}$

2 : Output $\mathsf{pk}_{\mathsf{e}} = \mathbf{B}_e^+$.

$\mathsf{Sim.UE.TG}(\mathsf{parameters})$

1 : Choose $\Delta_{\mathsf{e}+1}^{(1)+} \xleftarrow{\$} \mathbb{Z}_q^{nD \times n} \times \mathbb{Z}_q^{nD \times \bar{n}}$, and $\Delta_{\mathsf{e}+1}^{(2)+} \xleftarrow{\$} \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^{D \times \bar{n}}$

2 : Output $\Delta_{\mathsf{e}+1}^{+} = \left(\Delta_{\mathsf{e}+1}^{(1)+}, \Delta_{\mathsf{e}+1}^{(2)+}\right)$.

Sim.UE.Upd(parameters)

1 : Choose $(\mathbf{C}_1, \mathbf{C}_2) \xleftarrow{\$} \mathbb{Z}_q^{\bar{m}\times n} \times \mathbb{Z}_q^{\bar{m}\times \bar{n}}$

2 : Output $\mathsf{ct}_{\mathsf{e}+1} = (\mathbf{C}_1, \mathbf{C}_2)$.

Sim.UE.Enc(parameters)

1 : Choose $(\mathbf{C}_1, \mathbf{C}_2) \xleftarrow{\$} \mathbb{Z}_q^{\bar{m}\times n} \times \mathbb{Z}_q^{\bar{m}\times \bar{n}}$

2 : Output $\mathsf{ct}_{\mathsf{e}} = (\mathbf{C}_1, \mathbf{C}_2)$.

It follows directly from Conclusion 6.5 that the following lemma holds.

Lemma 6.6. The ciphertext update $\mathsf{UE.Upd}(\Delta_{\mathsf{e}+1}, \mathsf{ct}_{\mathsf{e}})$ is statistically indistinguishable form $\mathsf{Hyb.UE.Upd}(\mathsf{ct}_{\mathsf{e}}, \mathbf{B}_{\mathsf{e}+1}, \mathbf{m}, \mathbf{E}''_{\mathsf{e}}, (\mathbf{S}'_{(1)}, \mathbf{S}'_{(2)}, \mathbf{E}''_{(1)}, \mathbf{E}''_{(2)}))$.

From the lemma above, we derive the following conclusion.

Conclusion 6.7. The oracle $\mathcal{O}.\mathsf{Upd}(\mathsf{ct}_{\mathsf{e}})$ can be simulated by $\mathsf{Hyb.UE.Upd}(\mathsf{ct}_{\mathsf{e}}, \mathbf{B}_{\mathsf{e}+1}, \mathbf{m}, \mathbf{E}''_{\mathsf{e}}, (\mathbf{S}'_{(1)}, \mathbf{S}'_{(2)}, \mathbf{E}''_{(1)}, \mathbf{E}''_{(2)}))$.

Let $T+1$ be a number of epochs (they are counted from zero). Define the following hybrid games:

$\mathsf{Hyb}_i(b)$: This is the same as $\mathbf{Exp}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa}\text{-}b}_{\mathsf{UE},\mathcal{A}}(n)$ except the following difference: When the adversary sends a query $(\bar{\mathbf{m}}, \bar{\mathsf{ct}})$ to $\mathcal{O}.\mathsf{Chall}$ or an empty query to $\mathcal{O}.\mathsf{Upd}\widetilde{\mathsf{C}}$ at epoch j:

- for $j < i$ return an honestly generated challenge-equal ciphertext, i.e.

  if $b = 0$ **then** $\mathsf{UE.Enc}(\mathsf{k}_{\tilde{\mathsf{e}}}, \bar{\mathbf{m}})$,
  else $\mathsf{UE.Upd}(\Delta_{\tilde{\mathsf{e}}}, \bar{\mathsf{ct}})$.

- $j \geq i$, return a random ciphertext.

It is easy to see that $\mathsf{Hyb}_{T+1}(b)$ is the same as the original rand-ind-eu-cpa game in the backward-leak uni-directional setting $\mathbf{Exp}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa}\text{-}b}_{\mathsf{UE},\mathcal{A}}(n)$. Let $\mathcal{U}(n)$ be a random variable distributed uniformly in $[0, T]$, by the standard hybrid argument, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}{}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa}}(n) &= \left|\Pr\left[\mathsf{Hyb}_{T+1}(1) = 1\right] - \Pr\left[\mathsf{Hyb}_{T+1}(0) = 1\right]\right| \\
&= \left|\sum_{i=0}^{T} \left\{\left(\Pr\left[\mathsf{Hyb}_{i+1}(1) = 1\right] - \Pr\left[\mathsf{Hyb}_i(1) = 1\right]\right)\right.\right. \\
&\quad \left.\left. - \left(\Pr\left[\mathsf{Hyb}_{i+1}(0) = 1\right] - \Pr\left[\mathsf{Hyb}_i(0) = 1\right]\right)\right\}\right|
\end{aligned}$$

$$
\begin{aligned}
&= \Bigg| \sum_{i=0}^{T} \Big(\Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)+1}(1) = 1 \mid \mathcal{U}(n) = i\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)}(1) = 1 \mid \mathcal{U}(n) = i\right] \Big) \\
&\quad - \sum_{i=0}^{T} \Big(\Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)+1}(0) = 1 \mid \mathcal{U}(n) = i\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)}(0) = 1 \mid \mathcal{U}(n) = i\right] \Big) \Bigg| \qquad (5) \\
&= (T+1) \Bigg| \sum_{i=0}^{T} \Big(\Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)+1}(1) = 1 \wedge \mathcal{U}(n) = i\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)}(1) = 1 \wedge \mathcal{U}(n) = i\right] \Big) \\
&\quad - \sum_{i=0}^{T} \Big(\Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)+1}(0) = 1 \wedge \mathcal{U}(n) = i\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)}(0) = 1 \wedge \mathcal{U}(n) = i\right] \Big) \Bigg| \\
&\quad \leq (T+1) \left| \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)+1}(1) = 1\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)}(1) = 1\right] \right| \\
&\quad + (T+1) \left| \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)+1}(0) = 1\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)}(0) = 1\right] \right|
\end{aligned}
$$

The condition $\mathsf{Hyb}_0(0) = \mathsf{Hyb}_0(1)$ trivially holds since all challenge-equal ciphertexts are random ciphertexts. We use this fact in (5).

By the above computation, we have to prove that

$$\left| \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)+1}(b) = 1\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)}(b) = 1\right] \right| \leq \mathsf{negl}(n), \quad \text{where } b \in \{0,1\}.$$

In order to use the firewall technique [15, 21], we define the next hybrid game:

$\mathsf{Hyb}'_i(b)$: This is the same as $\mathsf{Hyb}_i(b)$ except that the game chooses $\mathsf{fwl}, \mathsf{fwr} \leftarrow [0, T]$. If the adversary corrupts k_j such that $j \in [\mathsf{fwl}, \mathsf{fwr}]$ or $\Delta_{\mathsf{fwr}+1}$, the game aborts.

The guess is correct with probability $(T+1)^{-2}$. Therefore, we have

$$
\begin{aligned}
&\left| \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)+1}(b) = 1\right] - \Pr\left[\mathsf{Hyb}_{\mathcal{U}(n)}(b) = 1\right] \right| \\
&\qquad \leq (T+1)^2 \left| \Pr\left[\mathsf{Hyb}'_{\mathcal{U}(n)+1}(b) = 1\right] - \Pr\left[\mathsf{Hyb}'_{\mathcal{U}(n)}(b) = 1\right] \right|.
\end{aligned}
$$

Note that to prove the security of our UE scheme it remains to show that the following condition holds.

$$\left| \Pr\left[\mathsf{Hyb}'_{\mathcal{U}(n)+1}(b) = 1\right] - \Pr\left[\mathsf{Hyb}'_{\mathcal{U}(n)}(b) = 1\right] \right| \leq \mathsf{negl}(n). \qquad (6)$$

Intuitively, this condition asserts that the adversary cannot distinguish between the final two hybrids in our sequence with more than negligible advantage. The lemma below formalizes this indistinguishability under the LWE assumption.

Lemma 6.8. If the LWE assumption holds, then Equation (6) also holds.

We omit the proof of this lemma, as it closely parallels the argument in Lemma 5.7 of Nishimaki's work [21], with only minor adaptations to suit our setting.

Our analysis leads to the following security guarantee for the proposed scheme.

Theorem 6.9. Under the LWE assumption, the presented FrodoPKE-based UE scheme is rand-ind-eu-cpa secure in the backward-leak uni-directional setting. That is, $\mathsf{Adv}^{\mathsf{rand\text{-}ind\text{-}eu\text{-}cpa}}_{\mathcal{A}}(n) \leq \mathsf{negl}(n)$.

As stated, this result formally establishes the security of our scheme against both classical and quantum adversaries. While the underlying FrodoPKE construction is already known to be quantum-resistant due to its reliance on the LWE assumption, it was not a priori evident that the full updatable encryption scheme would inherit this level of security. The theorem above certifies that, despite the additional structure and operations introduced by the UE framework, the overall design remains robust in the post-quantum setting. This provides strong evidence that the scheme is suitable for deployment in environments requiring long-term confidentiality under realistic quantum threat models.

7. Implementation and Performance Analysis

This section presents a detailed performance analysis of the updatable encryption scheme, implemented using various configurations of FrodoPKE. Specifically, we evaluate both AES-based and SHAKE-based instantiations of FrodoKEM at three different security levels defined by the NIST Post-Quantum Cryptography (PQC) standardization project, namely security levels 1, 3, and 5. These levels correspond to FrodoKEM-640, FrodoKEM-976, and FrodoKEM-1344, respectively.

The numerical suffixes in these names (640, 976, and 1344) refer to the matrix dimensions used in the scheme's underlying lattice-based cryptographic operations. Each configuration is designed to meet a specific NIST PQC security level:

- Level 1 (FrodoKEM-640) provides security approximately equivalent to AES-128,
- Level 3 (FrodoKEM-976) corresponds roughly to AES-192 security,
- Level 5 (FrodoKEM-1344) targets security comparable to AES-256.

Each security level supports two main variants, distinguished by the choice of pseudorandom number generator (PRNG): one using AES and the other using SHAKE (SHAKE128 or SHAKE256). While both variants offer the same level of security, they differ in performance characteristics. AES-based versions may benefit from hardware acceleration available on modern CPUs, whereas SHAKE-based versions are typically more portable and may be preferable in environments without AES-specific hardware support.

For each FrodoKEM variant, we measure and report the average execution time and standard deviation for key generation, encryption, decryption, token generation, and ciphertext update. These results provide a comprehensive view of the performance trade-offs associated with each parameter set and PRNG choice, helping to assess the practicality of deploying updatable encryption at different post-quantum security levels.

We implemented the updatable encryption scheme based on FrodoPKE in the Python programming language using the NumPy library. Our implementation is based on the official FrodoKEM reference implementation. While Python is not typically considered a high-performance language, the

Table 1. Execution times for the updatable encryption scheme based on FrodoPKE using AES.

Algorithm variants	FrodoPKE-640		FrodoPKE-976		FrodoPKE-1344	
Algorithm part	Avg. time [s]	Std.	Avg. time [s]	Std.	Avg. time [s]	Std.
UE.KG(params)	0.0198	0.0006	0.0465	0.0030	0.0878	0.0035
UE.Enc($\mathrm{k}_e, \mathbf{m} \in \mathcal{M}$)	0.0218	0.0011	0.0475	0.0010	0.0939	0.0010
UE.Dec($\mathrm{k}_e, \mathrm{ct}_e$)	0.0003	0.00001	0.0004	0.00001	0.0004	0.00001
UE.TG($\mathrm{k}_e, \mathrm{k}_{e+1}$)	6.3251	0.2018	18.0592	0.9919	64.1637	0.8593
UE.Upd($\Delta_{e+1}, \mathrm{ct}_e$)	0.1224	0.0057	0.5130	0.0057	1.2693	0.0379

Table 2. Execution times for the updatable encryption scheme based on FrodoPKE using SHAKE.

Algorithm variants	FrodoPKE-640		FrodoPKE-976		FrodoPKE-1344	
Algorithm part	Avg. time [s]	Std.	Avg. time [s]	Std.	Avg. time [s]	Std.
UE.KG(params)	0.0198	0.0004	0.0471	0.0030	0.0894	0.0063
UE.Enc($\mathrm{k}_e, \mathbf{m} \in \mathcal{M}$)	0.0218	0.0002	0.0484	0.0014	0.0945	0.0006
UE.Dec($\mathrm{k}_e, \mathrm{ct}_e$)	0.0002	0.00001	0.0004	0.00001	0.0005	0.00003
UE.TG($\mathrm{k}_e, \mathrm{k}_{e+1}$)	6.3118	0.0649	18.0746	0.4328	63.3840	0.8250
UE.Upd($\Delta_{e+1}, \mathrm{ct}_e$)	0.1214	0.0012	0.5166	0.0153	1.2572	0.0267

main objective here is to provide a comparative analysis of the execution time for individual components of the scheme. Python remains a popular choice for such tasks due to its simplicity and the availability of powerful scientific libraries. In particular, NumPy offers efficient tools for handling numerical computations, including large matrix operations.

We conducted 100 independent test runs to measure execution times, analyzing how they vary across algorithm components, parameter sets, and input randomness. Among all components, decryption consistently showed the lowest computation time with minimal variance. In contrast, token generation proved to be the most computationally demanding step, primarily due to the multiplication of large matrices whose dimensions grow with the security level of the chosen FrodoPKE variant. This results in a significant increase in runtime as the security level rises. However, this performance cost is less critical in practice, as token generation is performed by a trusted party and occurs outside the main ciphertext update process. Moreover, tokens are typically generated infrequently - only once per key rotation - and the lifespan of a key in an updatable encryption scheme is generally measured in days or longer. As a result, the high computational cost of token generation does not hinder the practi-

cal deployment of the scheme. These findings illustrate the expected trade-offs between post-quantum security levels and computational performance, particularly in the context of token generation and update operations.

References

[1] Alkim E, Avanzi R, Bos J, Ducas L, de la Piedra A, Pöppelmann T, Schwabe P, Stebila D. FrodoKEM Learning With Errors Key Encapsulation, Algorithm Specifications and Supporting Documentation. 2021. pp. 1–59.

[2] Alkim E, Avanzi R, Bos J, Ducas L, de la Piedra A, Pöppelmann T, Schwabe P, Stebila D. FrodoKEM: Learning With Errors Key Encapsulation. Preliminary Standardization Proposal (submitted to ISO). 2023. pp. 1–59.

[3] Asharov G, Jain A, López-Alt A, Tromer E, Vaikuntanathan V, Wichs D. Multiparty computation with low communication, computation and interaction via threshold FHE. In: Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31. Springer, 2012 pp. 483–501.

[4] Boneh D, Eskandarian S, Kim S, Shih M. Improving Speed and Security in Updatable Encryption Schemes. In: Advances in Cryptology – ASIACRYPT 2020, volume 12491 of *Lecture Notes in Computer Science*. Springer, 2020 pp. 559–589. doi:10.1007/978-3-030-64837-4_19.

[5] Boneh D, Lewi K, Montgomery H, Raghunathan A. Key Homomorphic PRFs and Their Applications. In: Annual Cryptology Conference. Springer, 2013 pp. 410–428.

[6] Boyd C, Davies GT, Gjøsteen K, Jiang Y. Fast and Secure Updatable Encryption. In: Advances in Cryptology – CRYPTO 2020, volume 12171 of *Lecture Notes in Computer Science*. Springer, 2020 pp. 464–493. doi:10.1007/978-3-030-56880-1_16.

[7] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 2014. **6**(3):1–36.

[8] Brakerski Z, Vaikuntanathan V. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Annual cryptology conference. Springer, 2011 pp. 505–524.

[9] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on computing*, 2014. **43**(2):831–871.

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

[11] Chen L, Li Y, Tang Q. CCA Updatable Encryption Against Malicious Re-Encryption Attacks. In: Advances in Cryptology – ASIACRYPT 2020, volume 12491 of *Lecture Notes in Computer Science*. Springer, 2020 pp. 590–620. doi:10.1007/978-3-030-64837-4_20.

[12] Everspaugh A, Paterson KG, Ristenpart T, Scott S. Key Rotation for Authenticated Encryption. In: Advances in Cryptology – CRYPTO 2017, volume 10401 of *Lecture Notes in Computer Science*. Springer, 2017 pp. 98–129. doi:10.1007/978-3-319-63688-7_4.

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

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

[15] Jiang Y. The Direction of Updatable Encryption does not Matter Much. In: Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26. Springer, 2020 pp. 529–558.

[16] Jiang-Galteland Y, Pan J. Backward-Leak Uni-Directional Updatable Encryption from (Homomorphic) Public Key Encryption. In: Canteaut A, Ishai Y (eds.), Public-Key Cryptography – PKC 2023, volume 13940 of *Lecture Notes in Computer Science*. Springer, 2023 pp. 399–428. doi:10.1007/978-3-031-31368-4_14.

[17] Klooß M, Lehmann A, Rupp A. (R) CCA Secure Updatable Encryption with Integrity Protection. In: Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38. Springer, 2019 pp. 68–99.

[18] Lehmann A, Tackmann B. Updatable Encryption with Post-Compromise Security. In: Advances in Cryptology – EUROCRYPT 2018, volume 10821 of *Lecture Notes in Computer Science*. Springer, 2018 pp. 685–716. doi:10.1007/978-3-319-78375-8_22.

[19] Micciancio D, Goldwasser S. Complexity of Lattice Problems: a Cryptographic Perspective, volume 671. Springer Science & Business Media, 2002.

[20] Micciancio D, Regev O. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM Journal on Computing*, 2007. **37**(1):267–302.

[21] Nishimaki R. The direction of updatable encryption does matter. In: IACR International Conference on Public-Key Cryptography. Springer, 2022 pp. 194–224.

[22] Peikert C. A Decade of Lattice Cryptography. *Cryptology ePrint Archive*, 2015.

[23] Peikert C, Rosen A. 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.

[24] Regev O. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM (JACM)*, 2009. **56**(6):1–40.

[25] Slamanig D, Striecks C. Revisiting Updatable Encryption: Controlled Forward Security, Constructions and a Puncturable Perspective. Cryptology ePrint Archive, Paper 2021/268, 2023. `https://eprint.iacr.org/2021/268/20231003:110230`.