# Towards More Efficient Registration-Based Encryption from LWE

Toi Tomita[1]

[1] Yokohama National University, Japan

tomita-toi-sk@ynu.ac.jp

**Abstract**

Registration-based encryption (RBE) effectively addresses the key escrow problem in identity-based encryption. However, existing post-quantum RBE schemes suffer from prohibitive ciphertext sizes in the gigabyte range for systems with $2^{10}$ registered users. This poor scalability is a major obstacle to the large-scale implementation of RBE in society. In this work, we propose a framework for constructing efficient RBE schemes that can be instantiated from the learning with errors (LWE) assumption. Specifically, the ciphertext size remains around 221 MB even as the number of registered users increases. The core techniques involve introducing decomposable laconic encryption and integrating it with a refined snapshotting trick. Our work represents an important milestone towards achieving practical post-quantum RBEs.

## Contents

**1 Introduction** **2**
- 1.1 Background . . . 2
- 1.2 Our Contribution . . . 2
- 1.3 Technical Overview . . . 3
- 1.4 Related Works . . . 4
- 1.5 Concurrent Works . . . 4

**2 Preliminaries** **4**
- 2.1 Registration-Based Encryption . . . 5

**3 Decomposable Laconic Encryption** **6**

**4 Efficient RBE from dLE** **8**
- 4.1 Construction . . . 9
- 4.2 Correctness, Compactness, and Efficiency . . . 9
- 4.3 Security . . . 12

**5 Instantiation from Lattice** **13**
- 5.1 Preliminaries on Lattices . . . 13
- 5.2 Lattice-Based dLE Scheme . . . 14
- 5.3 Efficiency and Comparison . . . 17

# 1 Introduction

## 1.1 Background

Registration-based encryption (RBE), introduced by Garg et al. [GHMR18], is a novel primitive that bridges the gap between traditional public-key encryption (PKE) and identity-based encryption (IBE). The core appeal of RBE is its ability to eliminate the key escrow problem inherent in IBE. In an IBE system, a trusted key generation center (KGC) possesses a master secret key that can decrypt any message. This vulnerability has long hindered the adoption of IBE in privacy-critical environments [BF01, Rog15]. RBE modifies the conventional trust model in which users generate their own key pairs locally and then register their public keys and their identities with a semi-trusted key curator (KC). The KC maintains a succinct digest (called a public parameter) of all registered users, enabling any sender to encrypt messages using only the recipient's identity and the current system parameters. The public parameter is updated each time a user registers. Therefore, registered users must obtain the helper decryption key used in the decryption process from the KC. Importantly, the RBE definition requires that the number of updates increase logarithmically with the number of users. It is not feasible to update all users' keys each time a new public key is registered, which is why this requirement is necessary. Since the initial works [GHMR18, GHM+19], rapid interest in RBEs has emerged, and they have advanced on many fronts, including efficiency [CES21, DKL+23, GKMR23, FKdP23], security [GV20, CHH+24], and theoretical lower bounds [MQR22, MQ23, HMQS23, DMQ25].

Despite its theoretical elegance, constructing efficient RBE schemes, especially those secure against quantum adversaries, has proven exceptionally challenging. Early RBE constructions [GHMR18, GHM+19] relied on highly theoretical and prohibitively inefficient cryptographic primitives, with estimated ciphertext sizes in the terabyte range [CES21], rendering them completely un-deployable. While pairing-based RBE schemes [GKMR23, FKdP23] have reached practical efficiency (with ciphertexts under 1 KB), they lack post-quantum resistance. On the other hand, existing post-quantum RBE schemes based on the learning with errors (LWE) assumption [DKL+23, FKdP23] have ciphertext sizes exceeding gigabytes for systems with $2^{10}$ registered users. This pervasive issue presents a fundamental obstacle to their adoption. Thus, we pose the following pressing question, which forms the central motivation of our work:

*Can we construct more efficient post-quantum RBE schemes?*

## 1.2 Our Contribution

In this work, we answer this question in the affirmative. We propose a framework for constructing efficient RBE schemes that can be instantiated from the LWE assumption. Specifically, the ciphertext size remains around 221 MB even as the number of registered users increases. Compared to existing works [DKL+23, FKdP23], our results significantly improve the concrete efficiency of the post-quantum RBE scheme. The core techniques are summarized as follows:

- We introduce *decomposable laconic encryption (dLE)*, a novel primitive inspired by the decomposability found in multi-recipient encryption. The key insight of dLE is that a ciphertext can be decomposed into two constituent parts. Through this decomposition, our framework achieves remarkable compactness and scalability.
- Moving beyond the traditional approach to construct RBEs, we implement a *refined snapshotting trick*. As a result of this, our ciphertext size scales only by $O(\mathsf{hw}(N))$, a significant improvement over $O(\ell \cdot \log N)$ for [DKL+23] and $O(\log N_{\max} \cdot \log N)$ for [FKdP23]. Here, $\mathsf{hw}(N)$ denotes the Hamming weight of the binary representation of $N$, $\ell$ is the length of the identity, and $N_{\max}$ is the maximum number of registered users.

We compare our scheme with the state-of-the-art post-quantum RBE schemes in Table 1. This table shows that our scheme achieves a significantly smaller ciphertext size and superior scalability compared with prior schemes. Furthermore, the limitations inherent in [FKdP23]'s scheme are not present in our scheme. Although our ciphertext size is still large at 221 MB, our work represents an important milestone towards achieving practical post-quantum RBEs.

| | $N_{\max}$ | Ciphertext size [MB] | | | Unbounded users | Adaptive compactness |
|---|---|---|---|---|---|---|
| | | $N = 2^{10}$ | $N = 2^{20}$ | $N = 2^{30}$ | | |
| [DKL+23] | $\infty$ | 4428.38 | 8856.76 | 13285.14 | ✓ | ✓ |
| [FKdP23] | $2^{20}$ | 1523.03 | 3046.07 | - | - | - |
| | $2^{30}$ | 2211.98 | 4423.96 | 6635.94 | | |
| **Ours** | $\infty$ | 221.43 | 221.45 | 221.47 | ✓ | ✓ |

Table 1: A comparison of the concrete ciphertext sizes of LWE-based RBE schemes. $N_{\max}$ is the maximum number of registered users. $N$ is the number of current registered users. "Unbounded users" means the maximum number of users is not bounded in advance. "Adaptive compactness" implies that it satisfies the standard definition of compactness.

### 1.3 Technical Overview

Here, we provide an overview of our techniques for obtaining efficient RBEs.

**The Bottleneck of the Previous Approach.** Most existing RBE constructions [GHMR18, GHM+19, GV20, DKL+23, GKMR23, FKdP23] rely on the *powers-of-two* approach [HLWW23] or its variant, the *snapshotting trick* [GV20]. These allow an RBE scheme without update frequency requirements to be converted into an RBE scheme with a low update frequency. Especially in the snapshotting trick, the KC maintains a logarithmic number of snapshots of the registration state (represented as a digest of the base scheme). These snapshots divide the registered user space into disjoint groups of sizes $2^j$ (e.g., $1, 2, 4, \ldots, 2^\lambda$). A snapshot is only replaced or merged when a new registration completes a group of a specific size. This ensures that the size of public parameters and the total number of updates per user all scale logarithmically with $N$. However, these approaches increase the ciphertext size by a factor of $O(\log N)$ because the encryption process must generate a ciphertext for each of the $\log N$ active digests of the base scheme. In the post-quantum setting (e.g., LWE-based schemes [DKL+23, FKdP23]), the ciphertext size of the base scheme already reaches several hundred megabytes. Consequently, even when $\log N = 10$, the resulting overhead leads to impractical ciphertext sizes of several gigabytes.

**Our Key Insight: Leveraging Decomposability.** To overcome this issue, our framework transitioned from a paradigm of simple repetition to one of effective repetition via *decomposability*. Inspired by recent advancements of post-quantum multi-recipient encryption (mPKE) [KKPP20, HKP+21, AHK+23], we introduce a novel primitive: Decomposable Laconic Encryption (dLE) is a decomposable variant of laconic encryption that was introduced in [DKL+23]. It can be considered a form of RBE without requiring an update frequency. While (d)LE can be converted to RBE using the previous approaches, it incurs multiplicative overhead. The key insight of dLE is that a ciphertext can be decomposed into two constituent parts: a relatively large, *digest-independent* part ($\mathsf{ct}^{\mathsf{ind}}$) and a very compact, *digest-dependent* part ($\mathsf{ct}^{\mathsf{dep}}$). This allows the encryptor to generate only a single digest-independent part, regardless of the number of active digests. The only additional cost associated with an increase in active digests is the compact digest-dependent part. To utilize dLE securely as described above, dLE must have a slightly stronger security requirement. That is to say, the message is hidden, even given one digest-independent part and multiple digest-dependent parts generated by different digests and *shared randomness*. We call such security *multi-digest security*. For more details, we give the formal definition of dLE in Section 3.

Fortunately, the LWE-based laconic encryption scheme proposed in [DKL+23] is already decomposable since the ciphertext can be decomposed into $\mathsf{ct}^{\mathsf{ind}}$ and $\mathsf{ct}^{\mathsf{dep}}$, where $\mathbf{A}_{\mathsf{id}}$ is a matrix determined by the identity $\mathsf{id}$, $\mathbf{y}$ is the digest, $\mathbf{r}_0, \ldots, \mathbf{r}_\ell$ are the LWE secrets, and $\mathbf{e}$ and $e$ are the LWE noises. That is, $\mathsf{ct}^{\mathsf{dep}}$ is only one LWE ciphertext and is significantly compact. However, simply applying the security proof in [DKL+23] repeatedly to each digest is insufficient to prove the security of a multi-digest. The original proof assumes that each ciphertext is generated with independent randomness. In contrast, multi-digest security involves multiple $\mathsf{ct}^{\mathsf{dep}}$ that are strongly correlated with a single $\mathsf{ct}^{\mathsf{ind}}$ through shared randomness $\mathbf{r}_0$. Therefore, when randomizing $\mathsf{ct}^{\mathsf{dep}}$ for a specific digest, the simulation must be performed without compromising the consistency of $\mathsf{ct}^{\mathsf{dep}}$ for the other digests or $\mathsf{ct}^{\mathsf{ind}}$. For more details, we refer the reader to Section 5.

**Our Generic Transformation.** Rather than a naive application of previous approaches, our RBE construction employs a refined snapshotting trick tailored to dLE. In our transformation, the public parameter $\mathsf{pp}$ consists of a set of digests $(\tilde{\mathsf{dig}}_1, \ldots, \tilde{\mathsf{dig}}_{\mathsf{hw}(N)})$, where each digest represents a snapshot of the registration state. The number of these snapshots is strictly limited to $\mathsf{hw}(N)$. Upon the registration of a new user, snapshots are merged in a manner analogous to a binary counter, ensuring that older snapshots are only replaced when a new snapshot can cover an equivalent power-of-two number of users. To encrypt a message, the encryptor does not generate multiple full ciphertexts. Instead, the encryptor produces a single digest-independent part ($\tilde{\mathsf{ct}}^{\mathsf{ind}}$) and then attaches multiple digest-dependent parts ($\tilde{\mathsf{ct}}_i^{\mathsf{dep}}$) corresponding to each active snapshot. The resulting ciphertext is $\mathsf{ct} = (\tilde{\mathsf{ct}}^{\mathsf{ind}}, \tilde{\mathsf{ct}}_1^{\mathsf{dep}}, \ldots, \tilde{\mathsf{ct}}_{\mathsf{hw}(N)}^{\mathsf{dep}})$. The size of $\mathsf{ct}$ is computed as $|\mathsf{ct}| = |\tilde{\mathsf{ct}}^{\mathsf{ind}}| + \mathsf{hw}(N) \cdot |\tilde{\mathsf{ct}}_i^{\mathsf{dep}}|$ instead of $O(\log N) \cdot (|\tilde{\mathsf{ct}}^{\mathsf{ind}}| + |\tilde{\mathsf{ct}}_i^{\mathsf{dep}}|)$.

In particular, when instantiated with the above dLE scheme, the ciphertext size only scales with $\mathsf{hw}(N)$ LWE ciphertexts. The concrete breakdown of the ciphertext size is $|\tilde{\mathsf{ct}}^{\mathsf{ind}}| = 221.41$ MB and $|\tilde{\mathsf{ct}}_i^{\mathsf{dep}}| = 1.86$ KB. Consequently, while the ciphertext size in previous LWE-based RBE schemes doubled from $2^{10}$ to $2^{20}$ users, our ciphertext size grows only by a few tens of kilobytes. This remarkable efficiency can be attributed to the fact that a substantial portion of the ciphertext is shared across all snapshots. At the same time, the growth part is confined to the compact digest-dependent components. The synergy between dLE and snapshotting enables the scheme to achieve a more efficient post-quantum RBE, yielding an order-of-magnitude reduction in ciphertext size. A more detailed description of our transformation is provided in Section 4. We believe that our techniques could be useful for more advanced applications.

### 1.4 Related Works

In recent years, there has been growing interest in generalizing RBEs to registered fine-grained encryption, such as registered attribute-based encryption (RABE) [HLWW23] and registered function-based encryption (RFE) [FFM+23]. Most proposals are based on indistinguishability obfuscation [HLWW23, FFM+23, DPY24, LWW25] or pairing-based assumptions [HLWW23, FFM+23, ZZGQ23, ZLZ+24, GLWW24, AT24, DPY24, BLM+24, ZCHZ24, LWW25, AAH+25]. Very recently, we have had some lattice-based RABE constructions from private/public-coin evasive LWE [FWW23, ZCHZ24], succinct LWE [CHW25, WW25], or decomposed LWE [AMR25] assumptions, which are significantly stronger variants of the LWE assumption.

### 1.5 Concurrent Works

## 2 Preliminaries

**Notations.** The basic notations used in this paper are summarized here. For $n, m \in \mathbb{N}$ with $n \leq m$, we write $[n, m] \coloneqq \{n, \ldots, m\}$ and $[m] \coloneqq \{1, \ldots, m\}$. For $b \in \{0,1\}$, we write $\bar{b} \coloneqq 1 - b$. For $x \in \{0,1\}^n$, let $\mathsf{hw}(x)$ denote its Hamming weight. When $x \in \mathbb{N}$, $\mathsf{hw}(x)$ denotes the Hamming weight of the binary representation of $x$. For $x \in \{0,1\}^n$ and $1 \leq \ell \leq n$, let $x_{1:\ell}$ denote the length-$\ell$ prefix of $x$. We write $x \leftarrow\!\!\$\, \mathcal{S}$ to denote that $x$ is sampled uniformly from a finite set $\mathcal{S}$. We write $x \leftarrow \mathcal{D}$ to denote that $x$ is sampled from the distribution $\mathcal{D}$.

Throughout this work, we write $\lambda$ to denote the security parameter. We write $\mathsf{poly}(\lambda)$ to denote a function that is $O(\lambda^c)$ for some constant $c \in \mathbb{N}$ and $\mathsf{negl}(\lambda)$ to denote a function that is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$. We say that two families of distributions $\mathcal{D}_0 \coloneqq \{\mathcal{D}_{0,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable, denoted as $\mathcal{D}_0 \approx_c \mathcal{D}_1$ if no probabilistic polynomial time (PPT) algorithm can distinguish them with non-negligible probability. We say that they are statistically indistinguishable, denote as $\mathcal{D}_0 \approx_s \mathcal{D}_1$, if the statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is bounded by $\mathsf{negl}(\lambda)$. We use bold uppercase letters (e.g., $\mathbf{M}$) to denote matrices and bold lowercase letters (e.g., $\mathbf{v}$) to denote vectors. For an algorithm $\mathsf{A}$ and data $x$, $\mathsf{A}^x$ (resp., $\mathsf{A}^{[x]}$) denotes a read (resp., read/write) random access operation performed by $\mathsf{A}$ on $x$.

## 2.1 Registration-Based Encryption

Here, we define registration-based encryption (RBE) [GHMR18].

**Definition 2.1** (Registration-Based Encryption)**.** Let $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda\in\mathbb{N}}$ be the identity space and $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be the message space. A registration-based encryption (RBE) scheme with identity space $\mathcal{ID}$ and message space $\mathcal{M}$ is a tuple of PPT algorithms with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: On input the security parameter $\lambda$, the setup algorithm outputs a common reference string $\mathsf{crs}$.
- $\mathsf{KGen}(\mathsf{crs}) \to (\mathsf{sk}, \mathsf{pk})$: On input $\mathsf{crs}$, the key-generation algorithm outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$: On input $\mathsf{crs}$, the current public parameter $\mathsf{pp}$, an identity $\mathsf{id} \in \mathcal{ID}$ to be registered, and a corresponding public key $\mathsf{pk}$, the registration algorithm *deterministically* outputs the updated public parameter $\mathsf{pp}'$ via (read/write) random access to the auxiliary information $\mathsf{aux}$. (The system is initialized with public parameters $\mathsf{pp}$ and auxiliary information $\mathsf{aux}$ set to $\perp$.)
- $\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct}$: On input $\mathsf{crs}$, $\mathsf{pp}$, a recipient identity $\mathsf{id} \in \mathcal{ID}$, and a message $\mathsf{m} \in \mathcal{M}$, the encryption algorithm outputs a ciphertext $\mathsf{ct}$.
- $\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to \mathsf{hsk}$: On input $\mathsf{pp}$ and an identity $\mathsf{id} \in \mathcal{ID}$, the update algorithm *deterministically* outputs a helper decryption key $\mathsf{hsk}$ via (read) random access to $\mathsf{aux}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct}) \to \mathsf{m} \in \mathcal{M} \cup \{\mathsf{GetUpd}, \perp\}$: On input a secret key $\mathsf{sk}$, a helper decryption key $\mathsf{hsk}$, and $\mathsf{ct}$, the decryption algorithm outputs either a message $\mathsf{m} \in \mathcal{M}$, or a special symbol in $\{\mathsf{GetUpd}, \perp\}$. (Here, $\mathsf{GetUpd}$ indicates that a key update might be needed for decryption.)

**Definition 2.2** (Correctness, Compactness, and Efficiency)**.** Let $\Pi$ be an RBE scheme. For a security parameter $\lambda \in \mathbb{N}$ and an adversary $\mathcal{A}$, we define the following game $\mathsf{Cor}_{\Pi,\mathcal{A}}(1^\lambda)$ between $\mathcal{A}$ and a challenger $\mathsf{Chal}$:

- **Initialization:** $\mathsf{Chal}$ sets $(\mathsf{pp}, \mathsf{aux}, \mathsf{hsk}, \mathcal{S}, \mathsf{id}^*, t) := (\perp, \perp, \perp, \emptyset, \perp, 0)$ and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, and sends $\mathsf{crs}$ to $\mathcal{A}$.
- Every round $\mathcal{A}$ does one of these actions:
  (a) Registering a new (non-target) identity: $\mathcal{A}$ sends a new identity $\mathsf{id} \notin \mathcal{S}$ and a public key $\mathsf{pk}$ to $\mathsf{Chal}$, which registers and and updates $\mathsf{pp} := \mathsf{Reg}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$. $\mathsf{Chal}$ also updates $\mathcal{S} := \mathcal{S} \cup \{\mathsf{id}\}$.
  (b) Registering the target identity: If the target identity $\mathsf{id}^*$ is already chosen, then do nothing. Otherwise, $\mathcal{A}$ sends a target identity $\mathsf{id}^*$ to $\mathsf{Chal}$. If $\mathsf{id}^* \notin \mathcal{S}$ then $\mathsf{Chal}$ samples an honest key pair $(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KGen}(\mathsf{crs})$ and registers and updates $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}^*)$, $\mathcal{S} := \mathcal{S} \cup \{\mathsf{id}^*\}$, and sends $\mathsf{pk}^*$ to $\mathcal{A}$.
  (c) Encrypting to the target identity: If the target identity $\mathsf{id}^*$ is not yet chosen, then do nothing. Otherwise, $\mathsf{Chal}$ sets $t := t + 1$and receives a message $\mathsf{m}_t$ from $\mathcal{A}$. $\mathsf{Chal}$ sends a ciphertext $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{m}_t)$ to $\mathcal{A}$.
  (d) Decrypting for target identity: $\mathcal{A}$ sends $i \in [t]$ to $\mathsf{Chal}$, which computes $\mathsf{m}'_i := \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}, \mathsf{ct}_i)$. If $\mathsf{m}'_i = \mathsf{GetUpd}$, then $\mathsf{Chal}$ computes the helper decryption key $\mathsf{hsk} := \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ and computes $\mathsf{m}'_i := \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}, \mathsf{ct}_i)$. If $\mathsf{m}'_j \neq \mathsf{m}_j$, the game halts with outputs $b = 1$.

  If $\mathcal{A}$ finished making queries and the game has not halted (as a result of a decryption query), then the game outputs $b = 0$.

Let $N = |\mathcal{S}|$. We say that $\Pi$ is correct and efficient if for all security parameters $\lambda \in \mathbb{N}$ and all (possibly unbounded) adversaries $\mathcal{A}$ making at most a polynomial number of queries, the following properties hold:

- **Correctness:** It holds that $\Pr\left[\mathsf{Cor}_{\Pi,\mathcal{A}}(1^\lambda) = 1\right] = \mathsf{negl}(\lambda)$.
- **(Adaptive) compactness:** It holds that $|\mathsf{pp}| = \mathsf{poly}(\lambda, \log N)$ and $|\mathsf{hsk}^*| = \mathsf{poly}(\lambda, \log N)$ (at all points in the game).
- **(Weakly) efficiency:** Then, in the course of the above game, $\mathsf{Chal}$ invokes the update algorithm $\mathsf{Upd}$ at most $O(\log N)$ times, where each invocation runs in $\mathsf{poly}(\lambda, \log N)$ time in the RAM model of computation.

**Definition 2.3** (Security)**.** Let $\Pi$ be an RBE scheme. For a security parameter $\lambda \in \mathbb{N}$ and an adversary $\mathcal{A}$, we define the following game $\mathsf{Sec}_{\Pi,\mathcal{A}}(1^\lambda)$ between $\mathcal{A}$ and a challenger $\mathsf{Chal}$.

- **Initialization:** $\mathsf{Chal}$ sets $(\mathsf{pp}, \mathsf{aux}, \mathsf{hsk}, \mathcal{S}, \mathsf{id}^*) := (\bot, \bot, \bot, \emptyset, \bot)$ and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, and sends $\mathsf{crs}$ to $\mathcal{A}$.
- Every round $\mathcal{A}$ does one of these actions:
  (a) Registering a new (non-target) identity: $\mathcal{A}$ sends a new identity $\mathsf{id} \notin \mathcal{S}$ and a public key $\mathsf{pk}$ to $\mathsf{Chal}$, which registers and and updates $\mathsf{pp} := \mathsf{Reg}^{\mathsf{aux}}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$. $\mathsf{Chal}$ also updates $\mathcal{S} := \mathcal{S} \cup \{\mathsf{id}\}$.
  (b) Registering the target identity: If the target identity $\mathsf{id}^*$ is already chosen, then do nothing. Otherwise, $\mathcal{A}$ sends a target identity $\mathsf{id}^*$ to $\mathsf{Chal}$. If $\mathsf{id}^* \notin \mathcal{S}$ then $\mathsf{Chal}$ samples an honest key pair $(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KGen}(\mathsf{crs})$ and registers and updates $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}^*)$, $\mathcal{S} := \mathcal{S} \cup \{\mathsf{id}^*\}$, and sends $\mathsf{pk}^*$ to $\mathcal{A}$.
- **Encrypting for target identity:** $\mathcal{A}$ sends $\mathsf{m}_0, \mathsf{m}_1 \in \mathcal{M}$. If no target identity $\mathsf{id}^*$ is chosen, $\mathcal{A}$ sends $\mathsf{id}^*$ to $\mathsf{Chal}$. Then, $\mathsf{Chal}$ samples a random bit $b \leftarrow\!\!\$\, \{0,1\}$, generates an encryption to the target identity $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{m}_b)$, and sends $\mathsf{ct}^*$ to $\mathcal{A}$.
- $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. The game outputs 1 if $b = b'$ and 0 otherwise.

We say that $\Pi$ is secure if for all security parameter $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, it holds that $\Pr[\mathsf{Sec}_{\Pi,\mathcal{A}}(1^\lambda) = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.

# 3 Decomposable Laconic Encryption

In this section, we formally introduce *decomposable* laconic encryption (dLE), which is a generalization of the standard laconic encryption in [DKL+23] with modifications described in Theorem 3.2.

**Definition 3.1** (Decomposable Laconic Encryption)**.** Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be the message space. A decomposable laconic encryption (dLE) scheme with message space $\mathcal{M}$ is a tuple of PPT algorithms with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{crs}, \mathsf{dig}, \mathsf{st})$: On input a security parameter $\lambda$ and an index-length parameter $\ell$, the setup algorithm outputs a common reference string $\mathsf{crs}$, an initial digest $\mathsf{dig}$, and an initial state $\mathsf{st}$. We assume that $\mathsf{crs}$ implicitly defines a randomness space $\mathcal{R} = (\mathcal{R}^{\mathsf{ind}}, \mathcal{R}^{\mathsf{dep}})$, where $\mathcal{R}^{\mathsf{ind}}$ is the randomness space for digest-independent encryption and $\mathcal{R}^{\mathsf{dep}}$ is the randomness space for digest-dependent encryption.
- $\mathsf{KGen}(\mathsf{crs}) \to (\mathsf{sk}, \mathsf{pk})$: On input $\mathsf{crs}$, the key-generation algorithm outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{IsValid}(\mathsf{crs}, \mathsf{pk}) \to b$: On input $\mathsf{crs}$ and a public key $\mathsf{pk}$, the key-validation algorithm *deterministically* outputs a bit $b \in \{0,1\}$ indicating whether $\mathsf{pk}$ is valid or not.
- $\mathsf{MemUpd}^{[\mathsf{st}]}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{pk}) \to \mathsf{dig}'$: On input $\mathsf{crs}$, the current digest $\mathsf{dig}$, an index $\mathsf{id} \in \{0,1\}^\ell$, and $\mathsf{pk}$, the membership update algorithm *deterministically* outputs the updated digest $\mathsf{dig}'$ via (read/write) random access to the state $\mathsf{st}$.

- $\mathsf{Enc}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{m}; \mathsf{r}^{\mathsf{ind}}, \mathsf{r}^{\mathsf{dep}}) \to \mathsf{ct} = (\mathsf{ct}^{\mathsf{ind}}, \mathsf{ct}^{\mathsf{dep}})$: The *decomposable* encryption algorithm, runs with randomness $(\mathsf{r}^{\mathsf{ind}}, \mathsf{r}^{\mathsf{dep}}) \in \mathcal{R}^{\mathsf{ind}} \times \mathcal{R}^{\mathsf{dep}}$, splits into the following two algorithms:
  - $\mathsf{Enc}^{\mathsf{ind}}(\mathsf{crs}, \mathsf{id}; \mathsf{r}^{\mathsf{ind}}) \to \mathsf{ct}^{\mathsf{ind}}$: On input $\mathsf{crs}$, $\mathsf{id} \in \{0,1\}^\ell$, and $\mathsf{r}^{\mathsf{ind}} \in \mathcal{R}^{\mathsf{ind}}$, the digest-independent encryption algorithm outputs a digest-independent ciphertext $\mathsf{ct}^{\mathsf{ind}}$.
  - $\mathsf{Enc}^{\mathsf{dep}}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{m}; \mathsf{r}^{\mathsf{ind}}, \mathsf{r}^{\mathsf{dep}}) \to \mathsf{ct}^{\mathsf{dep}}$: On input $\mathsf{crs}$, $\mathsf{dig}$, $\mathsf{id} \in \{0,1\}^\ell$, a message $\mathsf{m} \in \mathcal{M}$, and $(\mathsf{r}^{\mathsf{ind}}, \mathsf{r}^{\mathsf{dep}}) \in \mathcal{R}^{\mathsf{ind}} \times \mathcal{R}^{\mathsf{dep}}$, the digest-dependent encryption algorithm outputs a digest-dependent ciphertext $\mathsf{ct}^{\mathsf{dep}}$.
- $\mathsf{WGen}^{\mathsf{st}}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{pk}) \to \mathsf{wit}$: On input $\mathsf{crs}$, $\mathsf{dig}$, $\mathsf{id} \in \{0,1\}^\ell$, and $\mathsf{pk}$, the witness generation algorithm *deterministically* outputs a membership witness $\mathsf{wit}$ via (read) random access to $\mathsf{st}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ct}) \to \mathsf{m} \in \mathcal{M} \cup \{\bot\}$: On input $\mathsf{sk}$, $\mathsf{wit}$, and $\mathsf{ct}$, the decryption algorithm outputs either a message $\mathsf{m} \in \mathcal{M}$ or a special symbol $\bot$.

Furthermore, all the above algorithms run in time at most $\mathsf{poly}(\lambda, \ell)$.

*Remark* 3.2 (Differences from standard laconic encryption). Below, we summarize the difference between decomposable and standard laconic encryption [DKL+23].

- The most important modification is *decomposability*, which allows the encryption algorithm to be split into two parts: a digest-independent and a digest-dependent part. We also introduce *ciphertext compactness* (Theorem 3.4) and *multi-digest security* (Theorem 3.5). Ciphertext compactness requires that a digest-dependent ciphertext be sufficiently compact. Multi-digest security requires that the message is hidden from the challenge ciphertext, which consists of one digest-independent part and *multiple digest-dependent parts* generated by different digests and shared randomness.
- Inspired by [HLWW23], we make a minor change by adding a key-validation algorithm. The algorithm verifies the validity of the public key. We define completeness (Theorem 3.3) to require that public keys generated by a key-generation algorithm pass verification by the key-validation algorithm.
- In the correctness game (Theorem 3.4), we modify the challenge ciphertext to be generated from *the time-specific digest chosen by the adversary instead of the most recent one.*

We define the completeness requirement of a dLE scheme. Completeness requires that the public key generated by $\mathsf{KGen}$ be valid.

**Definition 3.3** (Completeness)**.** We say that a dLE scheme $\Pi$ is complete if for all $\lambda, \ell \in \mathbb{N}$, all $(\mathsf{crs}, \mathsf{dig}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, and all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{crs})$, it holds that $\mathsf{IsValid}(\mathsf{crs}, \mathsf{pk}) = 1$.

We define the correctness and compactness requirements of a dLE scheme. The correctness requirement is essentially the same as in [DKL+23], except that the challenge ciphertext is generated from the time-specific digest chosen by the adversary instead of the most recent one. The compactness requires that a digest, a witness, and a digest-dependent ciphertext are sufficiently compact.

**Definition 3.4** (Correctness and Compactness)**.** Let $\Pi$ be a dLE scheme. For parameters $\lambda, \ell \in \mathbb{N}$ and an adversary $\mathcal{A}$, we define the following game $\mathsf{Cor}_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$ between $\mathcal{A}$ and a challenger $\mathsf{Chal}$:

- **Setup phase:** $\mathsf{Chal}$ samples $(\mathsf{crs}, \mathsf{dig}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and gives them to $\mathcal{A}$. $\mathsf{Chal}$ also initializes a counter $\mathsf{ctr} := 0$ and an (initially-empty) dictionary mapping indices to key-pair $\mathsf{Dict}_{\mathsf{H}} = \emptyset$. Finally, it gives $(\mathsf{crs}, \mathsf{dig}, \mathsf{st})$ to $\mathcal{A}$.
- **Query phase:** $\mathcal{A}$ can now issue the following queries:
  - Honest query: In this query, $\mathcal{A}$ specifies an index $\mathsf{id} \in \{0,1\}^\ell$. In response, $\mathsf{Chal}$ increments $\mathsf{ctr} := \mathsf{ctr} + 1$, generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{crs})$, runs $\mathsf{dig} := \mathsf{MemUpd}^{[\mathsf{st}]}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{pk})$, and replies to $\mathcal{A}$ with $(\mathsf{dig}, \mathsf{st}, \mathsf{sk}, \mathsf{pk})$. In addition, $\mathsf{Chal}$ adds the mapping $\mathsf{id} \mapsto (\mathsf{sk}, \mathsf{pk})$ to the dictionary $\mathsf{Dict}_{\mathsf{H}}$ and stores $(\mathsf{dig}_{\mathsf{ctr}}, \mathsf{st}_{\mathsf{ctr}}) := (\mathsf{dig}, \mathsf{st})$.

– Malicious query: In this query, $\mathcal{A}$ specifies an index $\mathsf{id} \in \{0,1\}^\ell$ and a public key $\mathsf{pk}$. In response, $\mathsf{Chal}$ first checks that $\mathsf{IsValid}(\mathsf{crs}, \mathsf{pk}) = 1$. If the check fails, it outputs $\perp$. Otherwise, $\mathsf{Chal}$ increments $\mathsf{ctr} := \mathsf{ctr} + 1$ and runs $\mathsf{dig} := \mathsf{MemUpd}^{[\mathsf{st}]}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{pk})$. Then, $\mathsf{Chal}$ replies to $\mathcal{A}$ with $(\mathsf{dig}, \mathsf{st})$. In addition, $\mathsf{Chal}$ stores $(\mathsf{dig}_{\mathsf{ctr}}, \mathsf{st}_{\mathsf{ctr}}) := (\mathsf{dig}, \mathsf{st})$.

- **Challenge phase:** At some point, $\mathcal{A}$ specifies an index $\mathsf{id}^* \in \{0,1\}^\ell$, a message $\mathsf{m}^* \in \mathcal{M}$, and a time index $t \in [\mathsf{ctr}]$. $\mathsf{Chal}$ outputs $b = 1$ if $\mathsf{id}^* \notin \mathsf{Dict}_{\mathsf{H}}$. Otherwise, $\mathsf{Chal}$ computes $\mathsf{ct}^* = (\mathsf{ct}^{\mathsf{ind}}, \mathsf{ct}^{\mathsf{dep}}) \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{dig}_t, \mathsf{id}^*, \mathsf{m}^*)$. Then, it retrieves $(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{Dict}_{\mathsf{H}}[\mathsf{id}^*]$, generates $\mathsf{wit}^* := \mathsf{WGen}^{\mathsf{st}_t}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}^*, \mathsf{pk}^*)$, and computes $\mathsf{m} := \mathsf{Dec}(\mathsf{sk}^*, \mathsf{wit}^*, \mathsf{ct}^*)$. Finally, $\mathsf{Chal}$ outputs $b = 1$ if $\mathsf{m} \neq \mathsf{m}^*$, $b = 0$ otherwise.

We say that a dLE scheme $\Pi$ is correct and compact if for all $\lambda, \ell \in \mathbb{N}$ and all (possibly unbounded) adversaries $\mathcal{A}$, the following properties holds:

- **Correctness:** It holds that $\Pr\left[\mathsf{Cor}_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell) = 1\right] = \mathsf{negl}(\lambda)$.
- **Compactness:** It holds that $|\mathsf{dig}| = \mathsf{poly}(\lambda, \ell)$ (at all points in the game), $|\mathsf{wit}^*| = \mathsf{poly}(\lambda, \ell)$, $|\mathsf{ct}^{\mathsf{ind}}| = \mathsf{poly}(\lambda, \ell)$, and $|\mathsf{ct}^{\mathsf{dep}}| = \mathsf{poly}(\lambda, \log \ell)$.

We define the *multi-digest* security requirements of a dLE scheme. The security requirement is essentially the same as in [DKL+23], except that the challenge ciphertext is generated from the time-specific digest chosen by the adversary instead of the most recent one.

**Definition 3.5** ($L$-Digest Security)**.** Let $\Pi$ be a dLE scheme. For parameters $\lambda, \ell, L \in \mathbb{N}$ and an adversary $\mathcal{A}$, define the following game $\mathsf{Sec}_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$ between $\mathcal{A}$ and a challenger $\mathsf{Chal}$:

- **Setup phase:** $\mathsf{Chal}$ samples $(\mathsf{crs}, \mathsf{dig}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$. $\mathsf{Chal}$ also initializes a counter $\mathsf{ctr} := 0$ and a set $\mathcal{C}_{\mathsf{M}} := \emptyset$. Finally, it gives $(\mathsf{crs}, \mathsf{dig}, \mathsf{st})$ to $\mathcal{A}$.
- **Query phase:** $\mathcal{A}$ can now issue the following queries:
  - Honest query: In this query, $\mathcal{A}$ specifies an index $\mathsf{id} \in \{0,1\}^\ell$. In response, $\mathsf{Chal}$ increments $\mathsf{ctr} := \mathsf{ctr} + 1$, generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{crs})$, and runs $\mathsf{dig} := \mathsf{MemUpd}^{[\mathsf{st}]}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{pk})$. Then, $\mathsf{Chal}$ replies to $\mathcal{A}$ with $(\mathsf{dig}, \mathsf{st}, \mathsf{pk})$. In addition, $\mathsf{Chal}$ stores $(\mathsf{dig}_{\mathsf{ctr}}, \mathsf{st}_{\mathsf{ctr}}) := (\mathsf{dig}, \mathsf{st})$.
  - Malicious query: In this query, $\mathcal{A}$ specifies an index $\mathsf{id} \in \{0,1\}^\ell$ and a public key $\mathsf{pk}$. In response, $\mathsf{Chal}$ first checks that $\mathsf{IsValid}(\mathsf{crs}, \mathsf{pk}) = 1$. If the check fails, it outputs $\perp$. Otherwise, $\mathsf{Chal}$ increments $\mathsf{ctr} := \mathsf{ctr} + 1$ and runs $\mathsf{dig} \leftarrow \mathsf{MemUpd}^{[\mathsf{st}]}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{pk})$ if $\mathsf{IsValid}(\mathsf{crs}, \mathsf{pk}) = 1$. Then, $\mathsf{Chal}$ replies to $\mathcal{A}$ with $(\mathsf{dig}, \mathsf{st})$. In addition, $\mathsf{Chal}$ adds $\mathsf{id}$ to the set $\mathcal{C}_{\mathsf{M}}$ and stores $(\mathsf{dig}_{\mathsf{ctr}}, \mathsf{st}_{\mathsf{ctr}}) := (\mathsf{dig}, \mathsf{st})$.
- **Challenge phase:** At some point, $\mathcal{A}$ specifies an index $\mathsf{id}^* \in \{0,1\}^\ell$, two messages $\mathsf{m}_0, \mathsf{m}_1 \in \mathcal{M}$, and $L$ distinct time indices $t_1, \ldots, t_L \in [\mathsf{ctr}]$. $\mathsf{Chal}$ outputs 0 if $\mathsf{id}^* \in \mathcal{C}_{\mathsf{M}}$. Otherwise, $\mathsf{Chal}$ samples $\mathsf{r}^{\mathsf{ind}} \leftarrow\!\!\$\, \mathcal{R}^{\mathsf{ind}}$ and $\mathsf{r}_1^{\mathsf{dep}}, \ldots, \mathsf{r}_L^{\mathsf{dep}} \leftarrow\!\!\$\, \mathcal{R}^{\mathsf{dep}}$, computes
$$\mathsf{ct}^{\mathsf{ind}} := \mathsf{Enc}^{\mathsf{ind}}(\mathsf{crs}, \mathsf{id}^*; \mathsf{r}^{\mathsf{ind}}), \qquad \mathsf{ct}_i^{\mathsf{dep}} := \mathsf{Enc}^{\mathsf{dep}}(\mathsf{crs}, \mathsf{dig}_{t_i}, \mathsf{id}^*, \mathsf{m}_b; \mathsf{r}^{\mathsf{ind}}, \mathsf{r}_i^{\mathsf{dep}}), \quad \forall i \in [L],$$
  and replies to $\mathcal{A}$ with $(\mathsf{ct}^{\mathsf{ind}}, \mathsf{ct}_1^{\mathsf{dep}}, \ldots, \mathsf{ct}_L^{\mathsf{dep}})$.
- **Output phase:** $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. The game outputs 1 if $b = b'$ and 0 otherwise.

We say that a dLE scheme $\Pi$ is $L$-digest secure if for all security parameters $\lambda \in \mathbb{N}$, all polynomials $\ell = \ell(\lambda)$ and $L = L(\lambda)$, and all PPT adversaries $\mathcal{A}$, it holds that $\Pr[\mathsf{Sec}_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell) = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.

# 4 Efficient RBE from dLE

In this section, we present our generic transformation from a dLE scheme to an efficient RBE scheme.

## 4.1 Construction

Let $\Pi_{\mathsf{dLE}} = (\widetilde{\mathsf{Setup}}, \widetilde{\mathsf{KGen}}, \widetilde{\mathsf{IsValid}}, \widetilde{\mathsf{MemUpd}}, \widetilde{\mathsf{Enc}} = (\widetilde{\mathsf{Enc}}^{\mathsf{ind}}, \widetilde{\mathsf{Enc}}^{\mathsf{dep}}), \widetilde{\mathsf{WGen}}, \widetilde{\mathsf{Dec}})$ be a dLE scheme with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$. In Fig. 1, we describe our construction of RBE scheme $\Pi_{\mathsf{RBE}} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ with identity space $\mathcal{ID} = \{0,1\}^\ell$ and the same message space $\mathcal{M}$, where $\ell = O(\lambda)$ is the index-length parameter for $\Pi_{\mathsf{dLE}}$. If a collision-resistant hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell$ is available, one can use an arbitrary string as an identity. In the description, we adopt the following conventions:

- The auxiliary information $\mathsf{aux} = (\mathsf{ctr}, \mathsf{Dict}_1, \mathsf{Dict}_2, \mathsf{pp}, \tilde{\mathsf{st}})$ consists of the following components:
  - A counter $\mathsf{ctr} \geq 0$ is the current number of registered users in the system.
  - A dictionary $\mathsf{Dict}_1$ mapping a counter $\mathsf{ctr}$ to the $\mathsf{ctr}$-th registered identity/key pair $(\mathsf{id}, \mathsf{pk})$.
  - A dictionary $\mathsf{Dict}_2$ mapping an identity $\mathsf{id}$ to a collection of the helper decryption keys $\mathsf{hsk} = \{(\mathsf{ctr}, \tilde{\mathsf{wit}})\}$ associated with $\mathsf{id}$, where $\tilde{\mathsf{wit}}$ is a membership witness generated upon the $\mathsf{ctr}$-th registration.
  - The current public parameter $\mathsf{pp} = (\mathsf{ctr}, \tilde{\mathsf{dig}}_1, \ldots, \tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr})})$, where $\mathsf{ctr}$ is the current counter and $\tilde{\mathsf{dig}}_1, \ldots, \tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr})}$ are the digests of $\Pi_{\mathsf{dLE}}$. Of the digests, $\tilde{\mathsf{dig}}_1$ is the oldest version and $\tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr})}$ is the newest version.
  - The state $\tilde{\mathsf{st}}$ corresponding to the newest digest $\tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr})}$.
- We use a function $\delta : \mathbb{N} \to \mathbb{N}$ that, for an input $x \in \mathbb{N}$, outputs the position where the least significant bit is 1 in the binary representation of $x$.

## 4.2 Correctness, Compactness, and Efficiency

Here, we will show that the correctness, compactness, and efficiency of $\Pi_{\mathsf{RBE}}$ are implied by the correctness and compactness of $\Pi_{\mathsf{dLE}}$.

**Theorem 4.1** (Correctness)**.** If $\Pi_{\mathsf{dLE}}$ is complete and correct, $\Pi_{\mathsf{RBE}}$ is correct.

*Proof.* Consider the correctness game for $\Pi_{\mathsf{RBE}}$ (Theorem 2.2) played by an adversary $\mathcal{A}_{\mathsf{RBE}}$ and a challenger $\mathsf{Chal}$. Let $\mathsf{crs} = \tilde{\mathsf{crs}} \leftarrow \widetilde{\mathsf{Setup}}(1^\lambda, 1^\ell)$ be the common reference string, let $(\mathsf{sk}^*, \mathsf{pk}^*) := (\tilde{\mathsf{sk}}^*, \tilde{\mathsf{pk}}^*) \leftarrow \widetilde{\mathsf{KGen}}(\tilde{\mathsf{crs}})$ be the target-key sampled by $\mathsf{Chal}$ in response to a target-identity registration query on the challenge identity $\mathsf{id}^*$, and let $\mathsf{ctr}^*$ be the counter when $(\mathsf{id}^*, \mathsf{pk}^*)$ is registered. Further, let $\mathsf{aux} = (\mathsf{ctr}, \mathsf{Dict}_1, \mathsf{Dict}_2, \mathsf{pp}, \tilde{\mathsf{st}})$ be the auxiliary information (maintained by $\mathsf{Chal}$) at any point in the correctness game *after* $\mathcal{A}_{\mathsf{RBE}}$ has made a target-identity registration query. We can write $\mathsf{pp} = (\mathsf{ctr}, \tilde{\mathsf{dig}}_1, \ldots, \tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr})})$ with $\mathsf{ctr} \geq \mathsf{ctr}^*$.

We need to show that in the correctness game, for any (possibly unbounded) adversary $\mathcal{A}_{\mathsf{RBE}}$, the probability that $b = 1$ (indicating decryption failure) is negligible. Let $\mathsf{m}_t \in \mathcal{M}$ be the $t$-th encryption query made by $\mathcal{A}_{\mathsf{RBE}}$, and let $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{m}_t)$ be the resulting ciphertext. Consider a decryption query on any index $j \in [t]$. Here, $\mathsf{Chal}$ computes $\mathsf{m}'_j \leftarrow \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$.

- By construction, we can write $\mathsf{ct}_j = (\mathsf{ctr}_{\mathsf{enc}}, \tilde{\mathsf{ct}}_j^{\mathsf{ind}}, \tilde{\mathsf{ct}}_{j,1}^{\mathsf{dep}}, \ldots, \tilde{\mathsf{ct}}_{j,\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})}^{\mathsf{dep}})$ and $\mathsf{hsk}^* = \{(\mathsf{ctr}_i, \tilde{\mathsf{wit}}_i)\}_{i\in[d]}$.
- If $\mathsf{ctr}_{\mathsf{enc}} < \mathsf{ctr}_1$, $\mathsf{Chal}$ replies with $\perp$.
- If there is no counter such that $(\mathsf{ctr}_{\mathsf{enc}} \oplus \mathsf{ctr}_i) < 2^{\delta(\mathsf{ctr}_i)}$, $\mathsf{Chal}$ updates $\mathsf{hsk}^* := \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ (As will be shown later, the updated helper decryption key $\mathsf{hsk}^*$ always includes the counter $\mathsf{ctr}_i$ that satisfies the above).
- $\mathsf{Chal}$ replies with $\mathsf{m}'_j \leftarrow \widetilde{\mathsf{Dec}}(\tilde{\mathsf{sk}}^*, \tilde{\mathsf{wit}}_i, (\tilde{\mathsf{ct}}_j^{\mathsf{ind}}, \tilde{\mathsf{ct}}_{j,\mathsf{hw}(\mathsf{ctr}_i)}^{\mathsf{dep}}))$.

We now examine three failure cases.

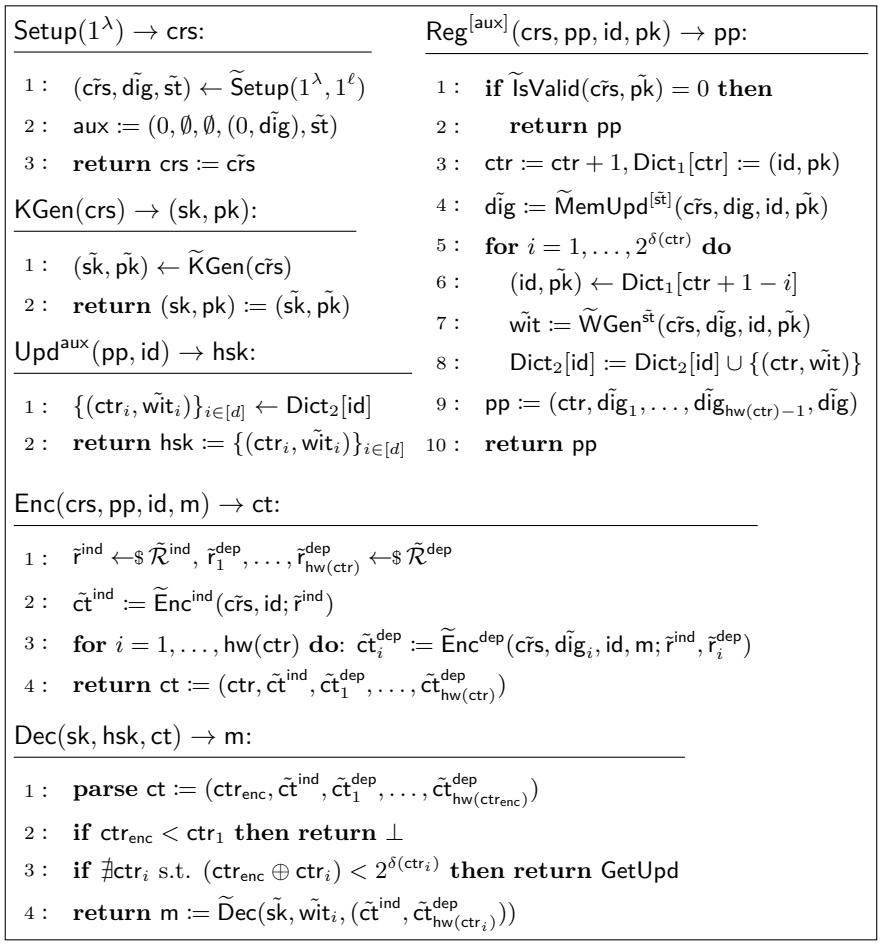



Figure 1: An RBE scheme $\Pi_{\mathsf{RBE}}$ from a dLE scheme $\Pi_{\mathsf{dLE}}$.

- Case 1: $\mathsf{ctr}_{\mathsf{enc}} < \mathsf{ctr}_1$. By construction, we have $\mathsf{ctr}_1 = \mathsf{ctr}^*$. In the correctness game, $\mathsf{Chal}$ only encrypts messages after the target identity $\mathsf{id}^*$ is registered. Thus, $\mathsf{ctr}_{\mathsf{enc}} \geq \mathsf{ctr}^* = \mathsf{ctr}_1$ always holds. Therefore, this case never occurs.

- Case 2: No suitable counter $\mathsf{ctr}_i$ exists. For any $\mathsf{ctr}_{\mathsf{enc}} \geq \mathsf{ctr}^*$, let the binary representation of $\mathsf{ctr}_{\mathsf{enc}}$ be $\sum_{j=1}^{\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})} 2^{d_j}$ with $d_1 > d_2 > \cdots > d_{\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})}$. This defines a canonical partition of $[1, \mathsf{ctr}_{\mathsf{enc}}]$ into $\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})$ intervals $\mathcal{B}_i := [T_i - 2^{d_i} + 1, T_i]$, where $T_i = \sum_{j=1}^{i} 2^{d_j}$. Since $\mathsf{ctr}^* \in [1, \mathsf{ctr}_{\mathsf{enc}}]$, there exists $i \in [\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})]$ such that $\mathsf{ctr}^* \in \mathcal{B}_i$. By the $\mathsf{Reg}$ algorithm, when the global counter $\mathsf{ctr}$ reached $T_i$, the loop in step 5 added the tuple $(T_i, \tilde{\mathsf{wit}})$ to $\mathsf{Dict}_2[\mathsf{id}^*]$ because $\mathsf{id}^*$ was within the $2^{\delta(T_i)}$ most recent registrations. Furthermore, the condition $(\mathsf{ctr}_{\mathsf{enc}} \oplus T_i) \leq 2^{\delta(T_i)}$ is satisfied because $\mathsf{ctr}_{\mathsf{enc}}$ and $T_i$ share the same bits of all positions higher than $d_i = \delta(T_i)$, and $\mathsf{ctr}_{\mathsf{enc}} \geq T_i$. This means that there exists a suitable counter $\mathsf{ctr}_i = T_i$. Therefore, this failure case is resolved by obtaining the latest $\mathsf{hsk}^*$.

- Case 3: $\Pi_{\mathsf{dLE}}$ decryption fails. Finally, we consider the case that $\mathsf{m}_j \neq \mathsf{m}'_j$. By the correctness of $\Pi_{\mathsf{dLE}}$, this would occur with only negligible probability if the witness $\tilde{\mathsf{wit}}_i$ and the digest $\tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr}_i)}$ behind $\tilde{\mathsf{ct}}_{j,\mathsf{hw}(\mathsf{ctr}_i)}^{\mathsf{dep}}$ were *synchronized*. Here, $\tilde{\mathsf{wit}}_i$ is the witness for the $\mathsf{id}^*$ generated through the execution of $\widetilde{\mathsf{WGen}}^{\tilde{\mathsf{st}}}(\tilde{\mathsf{crs}}, \tilde{\mathsf{dig}}, \mathsf{id}^*, \tilde{\mathsf{pk}}^*)$ during the $\mathsf{ctr}_i$-th registration. Hence, it is sufficient to show that $\tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr}_i)}$ is generated by the execution of $\widetilde{\mathsf{MemUpd}}$ in the $\mathsf{ctr}_i$-th registration. The condition $(\mathsf{ctr}_{\mathsf{enc}} \oplus \mathsf{ctr}_i) < 2^{\delta(\mathsf{ctr}_i)}$

ensures $\mathsf{ctr}_{\mathsf{enc}}$ and $\mathsf{ctr}_i$ share the same prefix beyond the $\delta(\mathsf{ctr}_i)$-th bit. By the description of the $\mathsf{Reg}$ algorithm, this guarantees that $\tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr}_i)}$ is generated by the execution of $\widetilde{\mathsf{MemUpd}}$ in the $\mathsf{ctr}_i$-th registration.

Therefore, the overall probability of decryption failure is negligible. This completes the proof. □

**Theorem 4.2** (Compactness)**.** If $\Pi_{\mathsf{dLE}}$ is compact, $\Pi_{\mathsf{RBE}}$ is compact.

*Proof.* We consider each requirement separately:

**Compactness of the public parameter:** By construction, the public parameter $\mathsf{pp}$ simply consists of a counter $\mathsf{ctr}$ indicating the current number of registered users, along with $\mathsf{hw}(\mathsf{ctr})$ digests $\tilde{\mathsf{dig}}_1, \ldots, \tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr})}$ for the underlying dLE scheme $\Pi_{\mathsf{dLE}}$. By compactness of $\Pi_{\mathsf{dLE}}$ and $\ell = O(\lambda)$, we have $|\tilde{\mathsf{dig}}_i| = \mathsf{poly}(\lambda, \ell) = \mathsf{poly}(\lambda)$ for all $i \in [\mathsf{hw}(\mathsf{ctr})]$. Since $\mathsf{hw}(\mathsf{ctr}) \leq \log(\mathsf{ctr})$, we have $|\mathsf{pp}| \leq |\mathsf{ctr}| + \log(\mathsf{ctr}) \cdot |\tilde{\mathsf{dig}}_i| = \mathsf{poly}(\lambda, \log(\mathsf{ctr}))$.

**Compactness of the helper decryption key:** Let $\mathsf{hsk} = \{(\mathsf{ctr}_i, \tilde{\mathsf{wit}}_i)\}_{i\in[d]}$ be a helper decryption key for some $\mathsf{id}$. By construction, we have $\mathsf{ctr}_i < \mathsf{ctr}_{i+1}$ and $\mathsf{ctr}_{i+1} - 2^{\delta(\mathsf{ctr}_{i+1})} + 1 \leq \mathsf{ctr}_i$ for all $i \in [d-1]$. The latter is because $(\mathsf{ctr}_{i+1}, \tilde{\mathsf{wit}}_{i+1})$ is added to $\mathsf{Dict}_2[\mathsf{id}]$ when $\mathsf{ctr}_{i+1} - 2^{\delta(\mathsf{ctr}_{i+1})} + 1 \leq \mathsf{ctr}_i$ occurs.

Then, we show that

$$\mathsf{ctr}_{i+1} - 2^{\delta(\mathsf{ctr}_{i+1})} + 1 \leq \mathsf{ctr}_i < \mathsf{ctr}_{i+1} \tag{1}$$

implies $\delta(\mathsf{ctr}_i) < \delta(\mathsf{ctr}_{i+1})$. By definition of $\delta$, we can write $\mathsf{ctr}_{i+1} = k_{i+1} \cdot 2^{\delta(\mathsf{ctr}_{i+1})}$ for some odd integer $k_i$. Thus, the inequality (1) can be written

$$(k_{i+1} - 1) \cdot 2^{\delta(\mathsf{ctr}_{i+1})} + 1 \leq \mathsf{ctr}_i \leq k_{i+1} \cdot 2^{\delta(\mathsf{ctr}_{i+1})} - 1.$$

This means that $\mathsf{ctr}_i \equiv 1 \mod 2^{\delta(\mathsf{ctr}_{i+1})}$. If $\delta(\mathsf{ctr}_i) \geq \delta(\mathsf{ctr}_{i+1})$, then $\mathsf{ctr}_i$ must be divisible by $2^{\delta(\mathsf{ctr}_i)}$, which contradicts the above. Therefore, we have $\delta(\mathsf{ctr}_i) < \delta(\mathsf{ctr}_{i+1})$. From the above argument, we have $\delta(\mathsf{ctr}_1) < \delta(\mathsf{ctr}_2) < \cdots < \delta(\mathsf{ctr}_d) \leq \log N$. and then $d \leq \log N$. Hence, $\mathsf{hsk}$ consists of at most $\log N$ pairs of counters and witnesses. By compactness of $\Pi_{\mathsf{dLE}}$, we have $|\tilde{\mathsf{wit}}_i| = \mathsf{poly}(\lambda, \ell) = \mathsf{poly}(\lambda)$ since $\ell = O(\lambda)$. Therefore, $|\mathsf{hsk}| = \mathsf{poly}(\lambda, \log N)$. □

Furthermore, by the (ciphertext) compactness of $\Pi_{\mathsf{dLE}}$, $\Pi_{\mathsf{RBE}}$ satisfies:

$$|\mathsf{ct}| = |\mathsf{ctr}| + |\tilde{\mathsf{ct}}^{\mathsf{ind}}| + \sum_{i=1}^{\mathsf{hw}(\mathsf{ctr})} |\tilde{\mathsf{ct}}_i^{\mathsf{dep}}| \leq |\tilde{\mathsf{ct}}^{\mathsf{ind}}| + O(\log N) \cdot \mathsf{poly}(\lambda, \log \ell). \tag{2}$$

**Theorem 4.3** (Efficiency)**.** If $\Pi_{\mathsf{dLE}}$ is compact, $\Pi_{\mathsf{RBE}}$ is (weakly) efficient.

*Proof.* We consider each requirement separately:

**Running time of update:** By construction, the $\mathsf{GetUpd}$ operation simply looks up the helper decryption key $\mathsf{hsk}$. By Theorem 4.2, we have $|\mathsf{hsk}| = \mathsf{poly}(\lambda, \log N)$. Since the auxiliary information maintains a dictionary $\mathsf{Dict}_2$ mapping each identity to its helper decryption key, the update operation can be implemented in $\mathsf{poly}(\lambda, \log N)$ time in the RAM model of computation.

**Number of updates:** Let $\mathsf{hsk} = \{(\mathsf{ctr}_i, \tilde{\mathsf{wit}}_i)\}_{i\in[d]}$ be a helper decryption key for some $\mathsf{id}$. By construction and Theorem 4.2, a new element is added to $\mathsf{Dict}_2[\mathsf{id}]$ whenever $\delta(\mathsf{ctr}_i) < \delta(\mathsf{ctr}_{i+1})$, so a registered identity $\mathsf{id}$ must invoke the $\mathsf{Upd}$ algorithm. Therefore, for any identity, its $i$-th update occurs when $2^i$ new identities are registered after its $(i-1)$-th update. Since there are at most $N$ registrations, the number of updates received by any identity is at most $\log N$. □

## 4.3 Security

We will show that the security of $\Pi_{\mathsf{RBE}}$ is implied by the multi-digest security of $\Pi_{\mathsf{dLE}}$. Recall the security game from Theorem 2.3.

**Theorem 4.4** (Security)**.** If $\Pi_{\mathsf{dLE}}$ is $\lceil \log N \rceil$-digest secure, $\Pi_{\mathsf{RBE}}$ is secure.

*Proof.* Let $\mathcal{A}_{\mathsf{RBE}}$ be an adversary for $\Pi_{\mathsf{RBE}}$, Let $\mathsf{ctr}_{\mathsf{enc}}$ be the counter used for encrypting the target identity, and let $\mathsf{ct}^*$ be the challenge ciphertext. By construction, we can write $\mathsf{ct}^* = (\mathsf{ctr}_{\mathsf{enc}}, \tilde{\mathsf{ct}}^{\mathsf{ind}}, \tilde{\mathsf{ct}}_1^{\mathsf{dep}}, \ldots, \tilde{\mathsf{ct}}_{\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})}^{\mathsf{dep}})$. Here, $\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}}) \leq \log N$. We use $\mathcal{A}_{\mathsf{RBE}}$ to construct a PPT algorithm $\mathcal{B}_{\mathsf{dLE}}$ for the security of $\Pi_{\mathsf{dLE}}$:

- **Setup phase:** Given $(\tilde{\mathsf{crs}}, \tilde{\mathsf{dig}}, \tilde{\mathsf{st}})$, $\mathcal{B}_{\mathsf{dLE}}$ proceeds as follows:
  1. $\mathcal{B}_{\mathsf{dLE}}$ initializes a counter $\mathsf{ctr} := 0$ and dictionaries $\mathsf{Dict}_1, \mathsf{Dict}_2 := \emptyset$.
  2. $\mathcal{B}_{\mathsf{dLE}}$ also initializes $\mathsf{pp} := (0, \tilde{\mathsf{dig}})$, $\mathsf{aux} = (\mathsf{ctr}, \mathsf{Dict}_1, \mathsf{Dict}_2, \mathsf{pp}, \tilde{\mathsf{st}})$, $\mathcal{S} := \emptyset$, $\mathsf{id}^* := \bot$.
  3. In addition, $\mathcal{B}_{\mathsf{dLE}}$ internally maintains an ordered list $\mathsf{List} := \bot$ which will track the digests and states generated during $\mathsf{Reg}$ algorithm execution.
  4. Finally, $\mathcal{B}_{\mathsf{dLE}}$ gives $\mathsf{crs} := \tilde{\mathsf{crs}}$ to $\mathcal{A}_{\mathsf{RBE}}$.
- **Query phase:** $\mathcal{B}_{\mathsf{dLE}}$ simulates responses to actions made by $\mathcal{A}_{\mathsf{RBE}}$ as follows:
  - Registering a new (non-target) identity: When $\mathcal{A}_{\mathsf{RBE}}$ specifies an identity $\mathsf{id}$ and a public key $\mathsf{pk} = \tilde{\mathsf{pk}}$, $\mathcal{B}_{\mathsf{dLE}}$ outputs the current public parameter $\mathsf{pp}$ if $\widetilde{\mathsf{IsValid}}(\tilde{\mathsf{crs}}, \tilde{\mathsf{pk}}) = 0$. Otherwise, it proceeds as follows:
    1. $\mathcal{B}_{\mathsf{dLE}}$ obtains $(\tilde{\mathsf{dig}}, \tilde{\mathsf{st}})$ by making a malicious query on $(\mathsf{id}, \tilde{\mathsf{pk}})$.
    2. $\mathcal{B}_{\mathsf{dLE}}$ updates $\mathsf{ctr} := \mathsf{ctr} + 1$, $\mathsf{pp} := (\mathsf{ctr}, \tilde{\mathsf{dig}}_1, \ldots, \tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr})-1}, \tilde{\mathsf{dig}})$, $\mathsf{Dict}_1[\mathsf{ctr}] := (\mathsf{id}, \tilde{\mathsf{pk}})$, $\mathsf{aux} = (\mathsf{ctr}, \mathsf{Dict}_1, \mathsf{Dict}_2, \mathsf{pp}, \tilde{\mathsf{st}})$, $\mathcal{S} := \mathcal{S} \cup \{\mathsf{id}\}$, and $\mathsf{List}[\mathsf{ctr}] := (\tilde{\mathsf{dig}}, \tilde{\mathsf{st}})$.
  - Registering the target identity: When $\mathcal{A}_{\mathsf{RBE}}$ specifies an identity $\mathsf{id}^*$, $\mathcal{B}_{\mathsf{dLE}}$ proceeds as follows:
    1. $\mathcal{B}_{\mathsf{dLE}}$ obtains $(\tilde{\mathsf{dig}}, \tilde{\mathsf{st}}, \tilde{\mathsf{pk}})$ by making an honest query on $\mathsf{id}$.
    2. $\mathcal{B}_{\mathsf{dLE}}$ updates $\mathsf{ctr} := \mathsf{ctr} + 1$, $\mathsf{pp} := (\mathsf{ctr}, \tilde{\mathsf{dig}}_1, \ldots, \tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr})-1}, \tilde{\mathsf{dig}})$, $\mathsf{Dict}_1[\mathsf{ctr}] := (\mathsf{id}^*, \tilde{\mathsf{pk}})$, $\mathsf{aux} = (\mathsf{ctr}, \mathsf{Dict}_1, \mathsf{Dict}_2, \mathsf{pp}, \tilde{\mathsf{st}})$, $\mathcal{S} := \mathcal{S} \cup \{\mathsf{id}^*\}$, and $\mathsf{List}[\mathsf{ctr}] := (\tilde{\mathsf{dig}}, \tilde{\mathsf{st}})$.
    3. $\mathcal{B}_{\mathsf{dLE}}$ replies to $\mathcal{A}_{\mathsf{RBE}}$ with $\mathsf{pk}^* := \tilde{\mathsf{pk}}$.
- **Challenge phase:** After $\mathcal{A}_{\mathsf{RBE}}$ outputs a pair of messages $\mathsf{m}_0, \mathsf{m}_1$ and $\mathsf{id}^*$ (if no target identity $\mathsf{id}^*$ is chosen), $\mathcal{B}_{\mathsf{dLE}}$ proceeds as follows: $\mathcal{B}_{\mathsf{dLE}}$ constructs $\mathsf{ct}^* = (\mathsf{ctr}_{\mathsf{enc}}, \tilde{\mathsf{ct}}^{\mathsf{ind}}, \tilde{\mathsf{ct}}_1^{\mathsf{dep}}, \ldots, \tilde{\mathsf{ct}}_{\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})}^{\mathsf{dep}})$ as follows:
  1. Let $\mathsf{pp} = (\mathsf{ctr}_{\mathsf{enc}}, \tilde{\mathsf{dig}}_1, \ldots, \tilde{\mathsf{dig}}_{\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})})$ be the current public parameter.
  2. For $j \in [\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})]$, let $t_j \in [\mathsf{ctr}_{\mathsf{enc}}]$ be the time index at which $\tilde{\mathsf{dig}}_j$ were generated, respectively.
  3. $\mathcal{B}_{\mathsf{dLE}}$ obtains $\tilde{\mathsf{ct}}^* = (\tilde{\mathsf{ct}}^{\mathsf{ind}}, \tilde{\mathsf{ct}}_1^{\mathsf{dep}}, \ldots, \tilde{\mathsf{ct}}_{\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})}^{\mathsf{dep}})$ by querying $(t_1, \ldots, t_{\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})}, \mathsf{id}^*, \mathsf{m}_0^*, \mathsf{m}_1^*)$ during the challenge phase.
  4. $\mathcal{B}_{\mathsf{dLE}}$ finally gives $\mathsf{ct}^* = (\mathsf{ctr}_{\mathsf{enc}}, \tilde{\mathsf{ct}}^{\mathsf{ind}}, \tilde{\mathsf{ct}}_1^{\mathsf{dep}}, \ldots, \tilde{\mathsf{ct}}_{\mathsf{hw}(\mathsf{ctr}_{\mathsf{enc}})}^{\mathsf{dep}})$ to $\mathcal{A}_{\mathsf{RBE}}$.
- **Output phase:** $\mathcal{A}_{\mathsf{RBE}}$ outputs a bit $b' \in \{0, 1\}$, which $\mathcal{B}_{\mathsf{dLE}}$ also outputs.

According to the above simulation, if $\mathsf{id}^*$ is registered as a non-target identity, then $\mathcal{B}_{\mathsf{dLE}}$ also does not make a malicious query on $\mathsf{id}^*$. Therefore, the game will not end due to the challenge query made by $\mathcal{B}_{\mathsf{dLE}}$. Then, we show that $\mathcal{B}_{\mathsf{dLE}}$ perfectly simulates an execution of the RBE security game for $\mathcal{A}_{\mathsf{RBE}}$. In the case where $\tilde{\mathsf{ct}}_j^{\mathsf{dep}} \leftarrow \widetilde{\mathsf{Enc}}^{\mathsf{dep}}(\tilde{\mathsf{crs}}, \tilde{\mathsf{dig}}_j, \mathsf{id}^*, \mathsf{m}_0)$, $\mathcal{B}_{\mathsf{dLE}}$ simulates the challenger in the security game of $\mathsf{RBE}$ when $b = 0$ whereas in the another case, $\mathcal{B}_{\mathsf{dLE}}$ simulates the challenger in the security game of $\mathsf{RBE}$ when $b = 1$. Correspondingly, $\mathcal{B}_{\mathsf{dLE}}$ wins the dLE security game with the same probability. □

# 5 Instantiation from Lattice

In this section, we present a lattice-based RBE construction obtained by instantiating the transformation from the previous section.

## 5.1 Preliminaries on Lattices

Here, we give preliminaries on lattices necessary to describe our dLE scheme. **Cyclotomic Rings.** Let $\mathcal{K} = \mathbb{Q}(\zeta)$ be a cyclotomic field and $\mathcal{R} = \mathbb{Z}[\zeta]$ its ring of integers, where $\zeta \in \mathbb{C}$ is a root of unity. Write $\deg_{\mathcal{R}}$ for the degree of $\mathcal{R}$. The (infinity) norm $\|a\|$ of an element $a = \sum_{i\in[0,\deg_{\mathcal{R}}-1]} a_i\zeta^i \in \mathcal{R}$ is defined as the infinity norm of its coefficient vector $(a_0, \ldots, a_{\deg_{\mathcal{R}}-1})^\top$, i.e. $\|a\| \coloneqq \max_i \|a_i\|_\infty$. For a vector $\mathbf{x} = (x_1, \ldots, x_m)^\top \in \mathcal{R}^m$, its norm is defined as $\|\mathbf{x}\| \coloneqq \max_{i\in[m]} \|x_i\|$. For $q \in \mathbb{N}$, we write $\mathcal{R}_q \coloneqq \mathcal{R}/q\mathcal{R}$.

We then recall the ring expansion factor and its property.

**Definition 5.1** (Expansion Factor)**.** The expansion factor of $\mathcal{R}$, denoted by $\gamma_{\mathcal{R}}$, is $\gamma_{\mathcal{R}} \coloneqq \max_{a,b\in\mathcal{R}\setminus\{0\}} \frac{\|a\cdot b\|}{\|a\|\cdot\|b\|}$.

**Lemma 5.2** ([AL21])**.** If $\mathcal{R}$ is a prime-power cyclotomic ring, then $\gamma_{\mathcal{R}} \le 2\deg_{\mathcal{R}}$. If $\mathcal{R}$ is a power-of-2 cyclotomic ring, then $\gamma_{\mathcal{R}} \le \deg_{\mathcal{R}}$.

We also recall a version of the leftover hash lemma over cyclotomic rings.

**Lemma 5.3** (Adapted from [BJRW20, Lemma 7])**.** Let $n = \mathsf{poly}(\lambda)$, $q \in \mathbb{N}$, and $m \ge n \cdot \lceil \log q \rceil + \omega(\log \lambda)$. Then, it holds that

$$\left\{ (\mathbf{B}, \mathbf{y}) : \begin{array}{l} \mathbf{B} \leftarrow\$ \mathcal{R}_q^{n\times m}, \mathbf{x} \leftarrow\$ \mathcal{R}_2^m \\ \mathbf{y} \coloneqq \mathbf{B}\mathbf{x} \bmod q \end{array} \right\} \approx_s \left\{ (\mathbf{B}, \mathbf{y}) : \begin{array}{l} \mathbf{B} \leftarrow\$ \mathcal{R}_q^{n\times m} \\ \mathbf{y} \leftarrow\$ \mathcal{R}_q^n \end{array} \right\}.$$

**Gadget Matrix.** Let $q, n, m \in \mathbb{N}$ with $m = n \cdot \lceil \log q \rceil$. The gadget matrix [MP12] is defined as $\mathbf{G} \coloneqq \mathbf{I}_n \otimes \begin{pmatrix} 1 & 2 & \cdots & 2^{\lfloor \log q \rfloor} \end{pmatrix} \in \mathcal{R}_q^{n\times m}$. Further, we write $\mathbf{G}^{-1}(\cdot)$ to denote the binary decomposition.

**Discrete Gaussian Distribution.** For a positive real $\sigma$, the Gaussian function over $\mathbb{R}$ with parameter $\sigma$ is defined as $\rho_\sigma(x) \coloneqq \exp(-\pi|x|^2/\sigma^2)$ with support $\mathbb{R}$. Then, we define discrete Gaussian distributions over $\mathbb{Z}$ and $\mathcal{R}$.

**Definition 5.4** (Discrete Gaussian Distribution)**.** The discrete Gaussian distribution over $\mathbb{Z}$ with parameter $\sigma$ is defined as $\mathcal{D}_{\mathbb{Z},\sigma}(x) \coloneqq \rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$ with support $\mathbb{Z}$, where $\rho_\sigma(\mathbb{Z}) \coloneqq \sum_{y\in\mathbb{Z}} \rho_\sigma(y)$ The discrete Gaussian distribution over $\mathcal{R}$ with parameter $\sigma$, denoted by $\mathcal{D}_{\mathcal{R},\sigma}$, is defined by independently sampling $\deg_{\mathcal{R}}$ coefficients $x_i$ from $\mathcal{D}_{\mathbb{Z},\sigma}$, and setting $x \coloneqq \sum_{i\in[0,\deg_{\mathcal{R}}-1]} x_i \cdot \zeta^i \in \mathcal{R}$.

We will use the following properties of discrete Gaussian distributions.

**Lemma 5.5** (Derived from [MP12, Lemma 7])**.** For any $k > 0$, it holds that $\Pr_{x\leftarrow\mathcal{D}_{\mathcal{R},\sigma}}[\|x\| > k \cdot \sigma] < 2 \cdot \deg_{\mathcal{R}} \cdot \exp(-\pi \cdot k^2)$.

**Learning with Errors and Variants.** We define the (module) learning with errors (LWE) problem and its leaky variant as follows.

**Definition 5.6** (LWE)**.** Let $\mathsf{params} = (\mathcal{R}, n, m, q, \mathcal{E})$ be parametrised by $\lambda$, where $\mathcal{R}$ is a ring, $n, m, q \in \mathbb{N}$, $\mathcal{E}$ is distributions over $\mathcal{R}^m$. For a bit $b \in \{0, 1\}$ and an adversary $\mathcal{A}$, we define the experiment $\mathsf{Exp\text{-}LWE}^b_{\mathsf{params},\mathcal{A}}(1^\lambda)$ as in Fig. 2. The $\mathsf{LWE}_{\mathsf{params}}$ assumption states that for any PPT $\mathcal{A}$, it holds that

$$\left|\Pr[\mathsf{Exp\text{-}LWE}^0_{\mathsf{params},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp\text{-}LWE}^1_{\mathsf{params},\mathcal{A}}(1^\lambda) = 1]\right| \le \mathsf{negl}(\lambda).$$

$\underline{\mathsf{Exp\text{-}LWE}^b_{\mathsf{params},\mathcal{A}}(1^\lambda)}$

$\mathbf{A} \leftarrow\!\$\, \mathcal{R}_q^{n\times m}$
$\mathbf{s} \leftarrow\!\$\, \mathcal{R}_q^n, \mathbf{e} \leftarrow \mathcal{E}$
**if** $b = 0$ **then** $\mathbf{b}^\top := \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top \bmod q$
**if** $b = 1$ **then** $\mathbf{b} \leftarrow\!\$\, \mathcal{R}_q^m$
**return** $b' \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})$

$\underline{\mathsf{Exp\text{-}LLWE}^b_{\mathsf{params},\mathcal{A}}(1^\lambda)}$

$\mathbf{A} \leftarrow\!\$\, \mathcal{R}_q^{n\times m}$
$(\mathbf{L}, \mathsf{state}) \leftarrow \mathcal{A}_0(\mathbf{A})$ // assert $\mathbf{L} \in \mathcal{L}$
$\mathbf{s} \leftarrow\!\$\, \mathcal{R}_q^n, \mathbf{e} \leftarrow \mathcal{E}, \mathbf{f} \leftarrow \mathcal{F}$
**if** $b = 0$ **then** $\mathbf{b}^\top := \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top \bmod q$
**if** $b = 1$ **then** $\mathbf{b} \leftarrow\!\$\, \mathcal{R}_q^m$
$\mathbf{l}^\top := \mathbf{e}^\top\mathbf{L} + \mathbf{f}^\top$
**return** $b' \leftarrow \mathcal{A}_1(\mathbf{b}, \mathbf{l}, \mathsf{state})$

Figure 2: The security experiments $\mathsf{Exp\text{-}LWE}^b_{\mathsf{params},\mathcal{A}}(1^\lambda)$ and $\mathsf{Exp\text{-}LLWE}^b_{\mathsf{params},\mathcal{A}}(1^\lambda)$.

**Definition 5.7** (Leaky LWE [LSW25, Definition 3])**.** Let $\mathsf{params} = (\mathcal{R}, n, m, q, Q, \mathcal{E}, \mathcal{F}, \mathcal{L})$ be parametrized by $\lambda$, where $\mathcal{R}$ is a ring, $n, m, q, Q \in \mathbb{N}$, $\mathcal{E}$ and $\mathcal{F}$ are distributions over $\mathcal{R}^m$ and $\mathbb{Z}^Q$, respectively, and $\mathcal{L} \subseteq \mathcal{R}^{m\times Q}$ is a polynomial-time decidable family of tuple of matrices. For $b \in \{0,1\}$ and an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we define the experiment $\mathsf{Exp\text{-}LLWE}^b_{\mathsf{params},\mathcal{A}}(1^\lambda)$ as in Fig. 2. The $\mathsf{LLWE}_{\mathsf{params}}$ assumption states that for any PPT $\mathcal{A}$, it holds that

$$\left|\Pr[\mathsf{Exp\text{-}LLWE}^0_{\mathsf{params},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp\text{-}LLWE}^1_{\mathsf{params},\mathcal{A}}(1^\lambda) = 1]\right| \le \mathsf{negl}(\lambda).$$

**Lemma 5.8** (Hardness of LLWE [LSW25, Theorem 3])**.** Let $\mathcal{R}$ be a ring and $n$, $m$, $q$, $Q$, $\mathcal{E}^*$, $\mathcal{E}$, $\mathcal{F}$, $\mathcal{L}$, and $\epsilon$ be parameters such that all constraints in the following condition are satisfied:

- $\mathcal{E}^* = \mathcal{D}^m_{\mathcal{R},\sigma^*}$, $\mathcal{E} = \mathcal{D}^m_{\mathcal{R},\sigma}$, and $\mathcal{F} = \mathcal{D}_{\mathcal{R},\sigma'}$, where $\sigma^*, \sigma, \sigma' \in \mathbb{R}$,
- $\sigma^* \ge \sqrt{2}\eta_\epsilon(\mathcal{R}^m)$,
- $\mathcal{L} := \left\{\mathbf{L} \in \mathcal{R}^{m\times Q} : \|\mathbf{L}^\top\mathbf{L}\| \le \sigma'^2 \cdot \left((2\eta_\epsilon(\mathcal{R}^{n+m})^2 + (\sigma^*)^2)^{-1} - \sigma^{-2}\right)\right\}$.

Then, for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{\mathsf{LWE}}_{\mathsf{params}^*,\mathcal{B}}(\lambda) \ge \mathsf{Adv}^{\mathsf{LLWE}}_{\mathsf{params},\mathcal{A}}(\lambda) - 4\epsilon/(1-\epsilon),$$

where $\mathsf{params}^* = (\mathcal{R}, n, m, q, \mathcal{E}^*)$ and $\mathsf{params} = (\mathcal{R}, n, m, q, Q, \mathcal{E}, \mathcal{F}, \mathcal{L})$.

## 5.2 Lattice-Based dLE Scheme

Our dLE scheme is syntactically built upon the laconic encryption scheme of Döttling et al. [DKL+23]. Let $n, m, \bar{m}, q \in \mathbb{N}$ and $\sigma, \sigma', \bar{\sigma} \in \mathbb{R}$ be parametrized by $\lambda$.

**Correctness, Completeness, and Compactness.** Our dLE scheme inherits correctness directly from the LE scheme in [DKL+23].

**Theorem 5.9** (Correctness)**.** If $q > 2\sqrt{\lambda}\,(\sigma' + 2\ell m\gamma_{\mathcal{R}}\sigma + \bar{m}\gamma_{\mathcal{R}}\bar{\sigma}) + 1$, our dLE scheme is correct.

*Proof.* Observe that the decryption is correct whenever

$$\mathsf{noise} := \left| e' - \sum_{j=0}^{\ell-1} \mathbf{e}_j^\top \begin{pmatrix} \mathbf{u}_{\mathsf{id}_{1:j}\|0} \\ \mathbf{u}_{\mathsf{id}_{1:j}\|1} \end{pmatrix} - \mathbf{e}_\ell^\top\mathbf{x} \right| < \frac{q-1}{4}.$$

By Theorem 5.5, with probability $1 - \mathsf{negl}(\lambda)$, we have $|e'| \le \frac{\sqrt{\lambda}}{2}\sigma'$, $\|\mathbf{e}_j\| \le \frac{\sqrt{\lambda}}{2}\sigma$ for all $j \in [0, \ell-1]$, and $\|\mathbf{e}_\ell\| \le \frac{\sqrt{\lambda}}{2}\bar{\sigma}$. Since $\|\mathbf{x}\| \le 1$ and $\left\|\mathbf{u}_{\mathsf{id}_{1:j}\|b}\right\| \le 1$ for all $j \in [\ell-1]$ and $b \in \{0,1\}$, we have $\mathsf{noise} \le \frac{\sqrt{\lambda}}{2}\,(\sigma' + 2\ell m\gamma_{\mathcal{R}}\sigma + \bar{m}\gamma_{\mathcal{R}}\bar{\sigma}) < \frac{q-1}{4}$ with probability $1 - \mathsf{negl}(\lambda)$. □

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{crs}, \mathsf{dig}, \mathsf{st})$:

$\mathbf{A}_0, \mathbf{A}_1 \leftarrow\$ \mathcal{R}_q^{n\times m}$, $\mathbf{B} \leftarrow\$ \mathcal{R}_q^{n\times \bar{m}}$
$\mathbf{y}_\epsilon := \mathbf{y}^* \leftarrow\$ \mathcal{R}_q^n$
$\mathsf{crs} := (\ell, \mathbf{A}_0, \mathbf{A}_1, \mathbf{B}, \mathbf{y}^*)$, $\mathsf{dig} := \mathbf{y}_\epsilon$
$\mathsf{st} := (\mathcal{T} := \{\epsilon\}, \{\mathbf{y}_v\}_{v\in\mathcal{T}})$
**return** $(\mathsf{crs}, \mathsf{dig}, \mathsf{st})$

$\mathsf{KGen}(\mathsf{crs}) \to (\mathsf{sk}, \mathsf{pk})$:

$\mathbf{x} \leftarrow\$ \mathcal{R}_2^{\bar{m}}$, $\mathbf{y} := \mathbf{B}\mathbf{x} \bmod q$
**return** $(\mathsf{sk}, \mathsf{pk}) := (\mathbf{x}, \mathbf{y})$

$\mathsf{IsValid}(\mathsf{crs}, \mathsf{pk}) \to b$:

**if** $\mathbf{y} \in \mathcal{R}_q^n$ **then return** 1
**else return** 0

$\mathsf{MemUpd}^{[\mathsf{st}]}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{pk}) \to \mathsf{dig}$:

$\mathcal{T} := \mathcal{T} \cup \{\mathsf{id}\}$
**for** $j = \ell - 1, \ldots, 0$ **do**
  **if** $(\mathsf{id}_{1:j}\|0) \notin \mathcal{T} \wedge (\mathsf{id}_{1:j}\|1) \notin \mathcal{T}$ **then** $\mathcal{T} := \mathcal{T} \setminus \{\mathsf{id}_{1:j}\}$
  **else**
    **if** $(\mathsf{id}_{1:j}\|\bar{\mathsf{id}}_{j+1}) \notin \mathcal{T}$ **then** $\mathbf{y}_{\mathsf{id}_{1:j}\|\bar{\mathsf{id}}_{j+1}} := \mathbf{y}^*$
    $\mathbf{u}_{\mathsf{id}_{1:j}\|0} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}_{1:j}\|0}), \mathbf{u}_{\mathsf{id}_{1:j}\|1} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}_{1:j}\|1})$
    $\mathcal{T} := \mathcal{T} \cup \{\mathsf{id}_{1:j}\}$
    $\mathbf{y}_{\mathsf{id}_{1:j}} := \mathbf{A}_0\mathbf{u}_{\mathsf{id}_{1:j}\|0} + \mathbf{A}_1\mathbf{u}_{\mathsf{id}_{1:j}\|1} \bmod q$
**return** $\mathsf{dig} := \mathbf{y}_\epsilon$

$\mathsf{Enc}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct} = (\mathsf{ct}^{\mathsf{ind}}, \mathsf{ct}^{\mathsf{dep}})$:

$\mathbf{r}_0, \ldots, \mathbf{r}_\ell \leftarrow\$ \mathcal{R}_q^n$
$\mathbf{e}_0, \ldots, \mathbf{e}_{\ell-1} \leftarrow \mathcal{D}_{\mathcal{R},\sigma}^{2m}$
$\mathbf{e}_\ell \leftarrow \mathcal{D}_{\bar{\sigma}}^{\bar{m}}$
$\mathsf{r}^{\mathsf{ind}} := (\mathbf{r}_0, \ldots, \mathbf{r}_\ell, \mathbf{e}_0, \ldots, \mathbf{e}_\ell)$
$\mathsf{ct}^{\mathsf{ind}} := \mathsf{Enc}^{\mathsf{ind}}(\mathsf{crs}, \mathsf{id}; \mathsf{r}^{\mathsf{ind}})$
$\mathsf{r}^{\mathsf{dep}} := e' \leftarrow \mathcal{D}_{\mathcal{R},\sigma'}$
$\mathsf{ct}^{\mathsf{dep}} := \mathsf{Enc}^{\mathsf{dep}}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{m}; \mathsf{r}^{\mathsf{ind}}, \mathsf{r}^{\mathsf{dep}})$
**return** $\mathsf{ct} := (\mathsf{ct}^{\mathsf{ind}}, \mathsf{ct}^{\mathsf{dep}})$

$\mathsf{WGen}^{\mathsf{st}}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{pk}) \to \mathsf{wit}$:

**for** $j = \ell - 1, \ldots, 0$ **do**
  $\mathbf{u}_{\mathsf{id}_{1:j}\|0} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}_{1:j}\|0})$
  $\mathbf{u}_{\mathsf{id}_{1:j}\|1} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}_{1:j}\|1})$
**return** $\mathsf{wit} := (\mathbf{u}_{\mathsf{id}_{1:j}\|0}, \mathbf{u}_{\mathsf{id}_{1:j}\|1})_{j=0}^{\ell-1}$

$\mathsf{Enc}^{\mathsf{ind}}(\mathsf{crs}, \mathsf{id}; \mathsf{r}^{\mathsf{ind}})$

**for** $j = 0, \ldots, \ell - 1$ **do**
  $\mathbf{B}_j := \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ \bar{\mathsf{id}}_{j+1} \cdot \mathbf{G} & \mathsf{id}_{j+1} \cdot \mathbf{G} \end{pmatrix}$
  $\mathbf{c}_j^\top := (\mathbf{r}_j^\top, \mathbf{r}_{j+1}^\top)\mathbf{B}_j + \mathbf{e}_j^\top \bmod q$
$\mathbf{c}_\ell^\top := \mathbf{r}_\ell^\top\mathbf{B} + \mathbf{e}_\ell^\top \bmod q$
**return** $\mathsf{ct}^{\mathsf{ind}} := (\mathbf{c}_0, \ldots, \mathbf{c}_\ell)$

$\mathsf{Enc}^{\mathsf{dep}}(\mathsf{crs}, \mathsf{dig}, \mathsf{id}, \mathsf{m}; \mathsf{r}^{\mathsf{ind}}, \mathsf{r}^{\mathsf{dep}})$

$d := \mathbf{r}_0^\top\mathbf{y}_\epsilon + e' + \lfloor q/2 \rfloor \cdot \mathsf{m} \bmod q$
**return** $\mathsf{ct}^{\mathsf{ind}} := d$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ct}) \to \mathsf{m}'$:

$\mathbf{c}_{\mathsf{id}}^\top := \sum_{j=0}^{\ell-1} \mathbf{c}_j^\top \begin{pmatrix} \mathbf{u}_{\mathsf{id}_{1:j}\|0} \\ \mathbf{u}_{\mathsf{id}_{1:j}\|1} \end{pmatrix}$
$\mu := d - \mathbf{c}_{\mathsf{id}}^\top - \mathbf{c}_\ell^\top\mathbf{x} \bmod q$
**return** $\mathsf{m}' := \mathsf{Decode}(\mu)$

Figure 3: Construction of lattice-based dLE.

We then show the completeness and compactness of our dLE scheme.

**Theorem 5.10** (Completeness)**.** Our dLE scheme is complete.

*Proof.* Completeness holds since $\mathsf{IsValid}$ always outputs 1 on all public keys $\mathsf{pk} \in \mathcal{R}_q^n$. By construction, the public keys output by $\mathsf{KGen}$ are in $\mathcal{R}_q^n$. □

**Theorem 5.11** (Compactness)**.** If $q = O(\ell)$, our dLE scheme is compact.

*Proof.* By construction, we have $\mathsf{dig} \in \mathcal{R}_q^n$, $\mathsf{wit} \in (\mathcal{R}_2^m)^{2\ell}$, $\mathsf{ct}^{\mathsf{ind}} \in \mathcal{R}_q^{2m\ell+\bar{m}}$, and $\mathsf{ct}^{\mathsf{dep}} \in \mathcal{R}_q$. If $q = O(\ell)$, then we have $|\mathsf{dig}| = \mathsf{poly}(\lambda, \log \ell)$, $|\mathsf{wit}| = \mathsf{poly}(\lambda, \ell)$, $|\mathsf{ct}^{\mathsf{ind}}| = \mathsf{poly}(\lambda, \ell)$, and $|\mathsf{ct}^{\mathsf{dep}}| = \mathsf{poly}(\lambda, \log \ell)$. Thus, our dLE scheme is compact. □

**Multi-Digest Security.** We show the multi-digest security of our dLE scheme.

**Theorem 5.12** (Multi-Digest Security)**.** Let $\bar{m} \geq n \cdot \lceil \log q \rceil + \omega(\log \lambda)$ and $L = \mathsf{poly}(\lambda)$. Further let $\mathsf{params} = (\mathcal{R}, n, 2m, q, 1, \mathcal{D}_{\mathcal{R},\sigma}^{2m}, \mathcal{D}_{\mathcal{R},\sigma'}, \mathcal{L})$ and $\mathsf{params}' = (\mathcal{R}, n, \bar{m} + L, q, \mathcal{D}_{\mathcal{R},\bar{\sigma}}^{\bar{m}})$ be parametrized by $\lambda$, where $\mathcal{L} = \{\mathbf{l} \in \mathcal{R}^{2mm} : \|\mathbf{l}\|_2 \leq \sqrt{(2\ell m + \bar{m}) \deg_{\mathcal{R}}}\}$. If the $\mathsf{LLWE}_{\mathsf{params}}$ and $\mathsf{LWE}_{\mathsf{params}'}$ assumptions hold, our dLE scheme is $L$-digest secure.

*Proof.* First, we will briefly explain the difference in the proof in [DKL+23]. Unlike the proof for a single digest, we introduce sets of time indices, $\{\mathcal{H}_i\}$, and adopt a method that sequentially isolates multiple $\mathsf{ct}^{\mathsf{dep}} = d$ from the outside in according to the path's branching depth. This allows us to sequentially randomize each digest-dependent part while maintaining correlation with $\mathbf{r}_0$.

Let $\mathsf{ctr} \in \mathbb{N}$ be the counter when the adversary $\mathcal{A}$ makes the challenge query. We assume that $\mathcal{A}$ specifies an index $\mathsf{id}^* = (\mathsf{id}_1^*, \ldots, \mathsf{id}_\ell^*) \in \{0,1\}^\ell$, a pair of messages $\mathsf{m}_0^*, \mathsf{m}_1^* \in \mathcal{M}$, and $L$ time indices $t_1, \ldots, t_L \in [\mathsf{ctr}]$. For $l \in [L]$, let $\mathsf{dig}_{t_l} = \mathbf{y}_{\epsilon,l}$ be the $t_l$-th root node, let $\mathsf{st}_{t_l} = (\mathcal{T}_l, \{\mathbf{y}_{v,l}\}_{v \in \mathcal{T}_l})$ be the $t_l$-th tree, and let $k_l \in [0, \ell]$ be such that $\mathsf{id}_{1:k_l}^*$ is a leaf node in the $t_l$-th tree for which $\mathcal{A}$ does not have a corresponding preimage. Furthermore, we denote $\mathbf{y}_{i,l}^* = \mathbf{y}_{\mathsf{id}_{1:i}^*,l}$ at the nodes $\mathsf{id}_{1:1}^*, \ldots, \mathsf{id}_{1:i}^*$ in $t_l$-th tree. Finally, we consider $L' (\leq L)$ sets $\mathcal{H}_1, \ldots, \mathcal{H}_{L'} \subseteq [L]$ satisfying the following conditions:

1. For any $i \in [L']$ and any $h, h' \in \mathcal{H}_i$, we have $k_h = k_{h'}$.
2. For any $i, i' \in [L']$ with $i < i'$, any $h \in \mathcal{H}_i$, and any $h' \in \mathcal{H}_{i'}$, we have $k_h < k_{h'}$.

In the following, let $\mathsf{ct}^* = (\mathbf{c}_0, \ldots, \mathbf{c}_{\ell-1}, \mathbf{c}_\ell, d_1, \ldots, d_L)$ be the challenge ciphertext. Consider the following hybrids:

- $\mathsf{Hyb}_0 (= \mathsf{Sec}_{\Pi,\mathcal{A}}^0(1^\lambda, 1^\ell))$: In this hybrid, $\mathsf{ct}^*$ is computed as follows:

$$\begin{aligned} \mathbf{c}_j^\top &= \mathbf{r}_j^\top \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \end{pmatrix} + \mathbf{r}_{j+1}^\top \begin{pmatrix} \bar{\mathsf{id}}_{j+1}^* \cdot \mathbf{G} & \mathsf{id}_{j+1}^* \cdot \mathbf{G} \end{pmatrix} + \mathbf{e}_j^\top, \quad \forall j \in [0, \ell-1], \\ \mathbf{c}_\ell^\top &= \mathbf{r}_\ell^\top \mathbf{B} + \mathbf{e}_\ell^\top, \\ d_l &= \mathbf{r}_0^\top \mathbf{y}_{\epsilon,l} + e_l' + \lfloor q/2 \rfloor \cdot \mathsf{m}_0^*, \ \forall l \in [L]. \end{aligned} \tag{3}$$

- $\mathsf{Hyb}_{1,i}$ for $i = 1, \ldots, L'$: Let $k = k_h$ for some $h \in \mathcal{H}_i$. In this hybrid, we sample $\mathbf{c}_0, \ldots, \mathbf{c}_{k-1} \leftarrow\!\!\$\, \mathcal{R}_q^{2m}$ and $\mathbf{e}_0, \ldots, \mathbf{e}_{k-1} \leftarrow \mathcal{D}_{\mathcal{R},\sigma}^{2m}$ and set

$$d_l = \begin{cases} d \leftarrow\!\!\$\, \mathcal{R}_q, & \text{if } l \in \bigcup_{j=1}^{i-1} \mathcal{H}_j \\ \sum_{j=0}^{k-1} (\mathbf{c}_j - \mathbf{e}_j)^\top \mathbf{z}_{j,l} + \mathbf{r}_k^\top \mathbf{y}_{k,l}^* + e_l' + \lfloor q/2 \rfloor \cdot \mathsf{m}_0^*, & \text{if } l \in \mathcal{H}_i, \\ \sum_{j=0}^{k} (\mathbf{c}_j - \mathbf{e}_j)^\top \mathbf{z}_{j,l} + \mathbf{r}_{k+1}^\top \mathbf{y}_{k+1,l}^* + e_l' + \lfloor q/2 \rfloor \cdot \mathsf{m}_0^*, & \text{if } l \in \bigcup_{j=i+1}^{L'} \mathcal{H}_j, \end{cases}$$

where $\mathbf{z}_{j,l} = \begin{pmatrix} \mathbf{u}_{\mathsf{id}_{1:j}^* \| 0, l} \\ \mathbf{u}_{\mathsf{id}_{1:j}^* \| 1, l} \end{pmatrix} \in \mathcal{R}_2^{2m}$. Then, we compute $\mathbf{c}_k, \ldots, \mathbf{c}_\ell$ as in Eq. (3).

In this hybrid, $d_l$ for $l \in \bigcup_{j=1}^{i-1} \mathcal{H}_j$ has already randomized, whereas $d_l$ for $l \in \bigcup_{j=i+1}^{L'} \mathcal{H}_j$ are still calculated to maintains consistency with the common $\mathbf{c}_0, \ldots, \mathbf{c}_k$. This structure enables the elimination of the influence of the shared randomness $\mathbf{r}_0$ one by one.

- $\mathsf{Hyb}_{2,i}$ for $i = 1, \ldots, L'$: Let $k = k_h$ for some $h \in \mathcal{H}_i$. In this hybrid, we uniformly randomly sample $\mathbf{c}_k$ from $\mathcal{R}_q^{2m}$ if $k \neq \ell$, and from $\mathcal{R}_q^{\bar{m}}$ if $k = \ell$, and $d_l \leftarrow\!\!\$\, \mathcal{R}_q$ for all $l \in \mathcal{H}_i$.
- $\mathsf{Hyb}_3 (= \mathsf{Sec}^1_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell))$: This hybrid is the same as $\mathsf{Hyb}_0$ except that $d_l$ for all $l \in [L]$ is computed as $d_l = \mathbf{r}_0^\top \mathbf{y}_{\epsilon,l} + e'_l + \lfloor q/2 \rfloor \cdot \mathsf{m}^*_1$.

To complete the proof, we prove the following lemmas. □

**Lemma 5.13.** If the $\mathsf{LLWE}_{\mathsf{params}}$ assumption holds, it holds that $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_{1,1}$.

*Proof.* This can be proved in the same way as the proof of [DKL+23, Lemma 3]. □

**Lemma 5.14.** Let $\bar{m} \geq n \cdot \lceil \log q \rceil + \omega(\log \lambda)$. If the $\mathsf{LWE}_{\mathsf{params}'}$ assumption holds, it holds that $\mathsf{Hyb}_{1,i} \approx_c \mathsf{Hyb}_{2,i}$ for all $i \in [L']$.

*Proof.* This can be proved in the same way as the proof of [DKL+23, Lemma 4]. □

**Lemma 5.15.** If the $\mathsf{LLWE}_{\mathsf{params}}$ assumption holds, it holds that $\mathsf{Hyb}_{2,i} \approx_c \mathsf{Hyb}_{1,i+1}$ for all $i \in [L']$.

*Proof.* This can be proved in the same way as the proof of [DKL+23, Lemma 3]. □

**Lemma 5.16.** Let $\bar{m} \geq n \cdot \lceil \log q \rceil + \omega(\log \lambda)$. If the $\mathsf{LLWE}_{\mathsf{params}}$ and $\mathsf{LWE}_{\mathsf{params}'}$ assumptions hold, it holds that $\mathsf{Hyb}_{2,L'} \approx_c \mathsf{Hyb}_3$.

*Proof.* This can be proved by applying Theorems 5.13 to 5.15 in reverse order. □

## 5.3 Efficiency and Comparison

Our RBE construction of Section 4.1 can be instantiated with our dLE scheme of Section 5.2. Here, we discuss the efficiency of this concrete instantiation and compare it to existing schemes.

The ciphertext size of our dLE scheme is $|\mathsf{ct}^{\mathsf{ind}}| = (2m\ell + \bar{m}) \cdot |\mathcal{R}_q|$ and $|\mathsf{ct}^{\mathsf{dep}}| = |\mathcal{R}_q|$. Combined with Eq. (2), we achieve the following ciphertext size:

$$|\mathsf{ct}| = \lceil \log N \rceil + |\mathsf{ct}^{\mathsf{ind}}| + \mathsf{hw}(N) \cdot |\mathsf{ct}^{\mathsf{dep}}| = \lceil \log N \rceil + (2m\ell + \bar{m} + \mathsf{hw}(N)) \cdot |\mathcal{R}_q|$$

where $N$ is the number of currently registered users. The last inequality is an equality if and only if $N = 2^\kappa - 1$ for some $\kappa \in \mathbb{N}$. In most cases, $\mathsf{hw}(N)$ is strictly less than $\lceil \log N \rceil$. The case $N = 2^\kappa$ is of particular note. In this case, $\mathsf{hw}(N) = 1$ and the ciphertext size achieves $|\mathsf{ct}| = \kappa + (2m\ell + \bar{m} + 1) \cdot |\mathcal{R}_q|$ This is the same size as the dLE ciphertext except for the overhead of $\kappa$ bits.

We provide a comparison of ciphertext sizes for (M)LWE-based RBE schemes in Table 2. For the concrete comparison in Table 1, we chose the following parameters with reference to [DKL+23]: $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^{256}+1)$ and $(\log q, n, m, \bar{m}, \ell) = (58, 4, 232, 512, 256)$.

| | Ciphertext size |
|---|---|
| [DKL+23] | $(n + 4m\ell + 2\bar{m} + 2) \cdot \lceil \log N \rceil \cdot \lvert\mathcal{R}_q\rvert$ |
| [FKdP23] | $(n + 8(2m(\log N_{\max} + 1)) + \bar{m} + 1) \cdot \lceil \log N \rceil \cdot \lvert\mathcal{R}_q\rvert$ |
| **Ours** | $\lceil \log N \rceil + (2m\ell + \bar{m} + \mathsf{hw}(N)) \cdot \lvert\mathcal{R}_q\rvert$ |

Table 2: A comparison of ciphertext sizes for (M)LWE-based RBE schemes.

## Acknowledgement

This work was supported by JSPS KAKENHI Grant Number JP25K21197, Japan.

## References

[AAH+25] Kyoichi Asano, Nuttapong Attrapadung, Keisuke Hara, Keitaro Hashimoto, and Yohei Watanabe. Key revocation in registered attribute-based encryption. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 102–133. Springer, Cham, May 2025. 4

[AHK+23] Joël Alwen, Dominik Hartmann, Eike Kiltz, Marta Mularczyk, and Peter Schwabe. Post-quantum multi-recipient public key encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1108–1122. ACM Press, November 2023. 3

[AL21] Martin R. Albrecht and Russell W. F. Lai. Subtractive sets over cyclotomic rings - limits of Schnorr-like arguments over lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 519–548, Virtual Event, August 2021. Springer, Cham. 13

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Key-homomorphic computations for RAM: Fully succinct randomised encodings and more. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 236–268. Springer, Cham, August 2025. 4

[AT24] Nuttapong Attrapadung and Junichi Tomida. A modular approach to registered ABE for unbounded predicates. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 280–316. Springer, Cham, August 2024. 4

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Berlin, Heidelberg, August 2001. 2

[BJRW20] Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen. Towards classical hardness of module-LWE: The linear rank case. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 289–317. Springer, Cham, December 2020. 13

[BLM+24] Pedro Branco, Russell W. F. Lai, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Ivy K. Y. Woo. Traitor tracing without trusted authority from registered functional encryption. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 33–66. Springer, Singapore, December 2024. 4

[CES21] Kelong Cong, Karim Eldefrawy, and Nigel P. Smart. Optimizing registration based encryption. In Maura B. Paterson, editor, *18th IMA International Conference on Cryptography and Coding*, volume 13129 of *LNCS*, pages 129–157. Springer, Cham, December 2021. 2

[CHH+24] Sohto Chiku, Keisuke Hara, Keitaro Hashimoto, Toi Tomita, and Junji Shikata. How to apply Fujisaki-Okamoto transformation to registration-based encryption. In Markulf Kohlweiss, Roberto Di Pietro, and Alastair R. Beresford, editors, *CANS 2024, Part II*, volume 14906 of *LNCS*, pages 145–165. Springer, Singapore, September 2024. 2

[CHW25] Jeffrey Champion, Yao-Ching Hsieh, and David J. Wu. Registered ABE and adaptively-secure broadcast encryption from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 3–34. Springer, Cham, August 2025. 4

[DKL+23] Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 417–446. Springer, Cham, April 2023. 2, 3, 6, 7, 8, 14, 16, 17

[DMQ25] Jesko Dujmovic, Giulio Malavolta, and Wei Qi. Registration-based encryption in the plain model. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part I*, volume 15674 of *LNCS*, pages 194–226. Springer, Cham, May 2025. 2

[DPY24] Pratish Datta, Tapas Pal, and Shota Yamada. Registered FE beyond predicates: (attribute-based) linear functions and more. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 65–104. Springer, Singapore, December 2024. 4

[FFM+23] Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (inner-product) functional encryption. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 98–133. Springer, Singapore, December 2023. 4

[FKdP23] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 166–200. Springer, Singapore, December 2023. 2, 3, 17

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 498–531. Springer, Cham, August 2023. 4

[GHM+19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 63–93. Springer, Cham, April 2019. 2, 3

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 689–718. Springer, Cham, November 2018. 2, 3, 5

[GKMR23] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1065–1079. ACM Press, November 2023. 2, 3

[GLWW24] Rachit Garg, George Lu, Brent Waters, and David J. Wu. Reducing the CRS size in registered ABE systems. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 143–177. Springer, Cham, August 2024. 4

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 621–651. Springer, Cham, August 2020. 2, 3

[HKP+21] Keitaro Hashimoto, Shuichi Katsumata, Eamonn Postlethwaite, Thomas Prest, and Bas Westerbaan. A concrete treatment of efficient continuous group key agreement via multi-recipient PKEs. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1441–1462. ACM Press, November 2021. 3

[HLWW23] Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 511–542. Springer, Cham, April 2023. 3, 4, 7

[HMQS23] Mohammad Hajiabadi, Mohammad Mahmoody, Wei Qi, and Sara Sarfaraz. Lower bounds on assumptions behind registration-based encryption. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part II*, volume 14370 of *LNCS*, pages 306–334. Springer, Cham, November / December 2023. 2

[KKPP20] Shuichi Katsumata, Kris Kwiatkowski, Federico Pintore, and Thomas Prest. Scalable ciphertext compression techniques for post-quantum KEMs and their applications. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 289–320. Springer, Cham, December 2020. 3

[LSW25] Russell W. F. Lai, Monisha Swarnakar, and Ivy K. Y. Woo. Leaky LWE: Learning with errors with semi-adaptive secret- and error-leakage. *IACR Communications in Cryptology*, 2(3), 2025. 14

[LWW25] George Lu, Brent Waters, and David J. Wu. Multi-authority registered attribute-based encryption. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part III*, volume 15603 of *LNCS*, pages 3–33. Springer, Cham, May 2025. 4

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012. 13

[MQ23] Mohammad Mahmoody and Wei Qi. Online mergers and applications to registration-based encryption and accumulators. In Kai-Min Chung, editor, *ITC 2023*, volume 267 of *LIPIcs*, pages 15:1–15:23. Schloss Dagstuhl, June 2023. 2

[MQR22] Mohammad Mahmoody, Wei Qi, and Ahmadreza Rahimi. Lower bounds for the number of decryption updates in registration-based encryption. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 559–587. Springer, Cham, November 2022. 2

[Rog15] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. 2

[WW25] Hoeteck Wee and David J. Wu. Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 204–235. Springer, Cham, August 2025. 4

[ZCHZ24] Yijian Zhang, Jie Chen, Debiao He, and Yuqing Zhang. Bounded collusion-resistant registered functional encryption for circuits. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 32–64. Springer, Singapore, December 2024. 4

[ZLZ+24] Ziqi Zhu, Jiangtao Li, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered functional encryptions from pairings. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 373–402. Springer, Cham, May 2024. 4

[ZZGQ23] Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered ABE via predicate encodings. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 66–97. Springer, Singapore, December 2023. 4