

# コンパクトな耐量子登録ベース暗号

富田 斗威<sup>1,a)</sup>

**概要：**登録ベース暗号 (RBE) は、ID ベース暗号 (IBE) における鍵預託問題に対する解決策の一つである。本論文では、暗号文のサイズが ID の長さとユーザー数の対数に線形に比例してスケールする、初の耐量子 RBE 方式を提案する。これにより、耐量子 RBE の効率性とスケーラビリティが大幅に向上する。私たちの結果は、RBE の構築で従来から用いられてきた powers-of-two アプローチのより効果的な活用法に関する発見に由来する。私たちのアプローチの核心は、多受信者暗号の文脈で用いられる技術から着想を得た「分解可能ラコニック暗号」という新しい暗号学的プリミティブである。

**キーワード：**登録ベース暗号、耐量子計算機安全性、格子

## Compact Post-Quantum Registration-Based Encryption

TOI TOMITA<sup>1,a)</sup>

**Abstract:** Registration-based encryption (RBE) is an elegant solution to the key escrow problem associated with identity-based encryption (IBE). In this paper, we present the first post-quantum RBE scheme where the size of the ciphertext scales linearly with the length of the ID and the logarithm of the number of users. This significantly improves the efficiency and scalability of post-quantum RBEs. Our results stem from our discovery of a more effective use of the powers-of-two approach, which is conventional for constructing RBEs. At the core of our approach is a new primitive called decomposable laconic encryption, inspired by techniques in the context of multi-recipient encryption.

**Keywords:** Registration-Based Encryption, Post-Quantum, Lattice

## 1. はじめに

### 1.1 背景

登録ベース暗号 (RBE) は、Garg ら [7] によって初めて提案され、従来の公開鍵暗号 (PKE) と ID ベース暗号 (IBE) の中間的なアプローチを提供する。RBE の主な動機は、IBE に内在する鍵供託問題を解決することである。IBE は、受信者の ID のみを使用してメッセージを暗号化するという魅力的な特徴を提供し、鍵交換や証明書検証の必要性を排除できるが、信頼できる第三者機関である鍵生成センター (KGC) がマスター秘密鍵を保持する。これにより、KGC がシステム内の任意の暗号文を復号化できるという

重大なセキュリティ上の脆弱性が生じる。さらに、ユーザーが復号化鍵を要求していないくとも、KGC はメッセージを復号できる。Rogaway [16] をはじめとする研究者たちが指摘するように、鍵供託問題は IBE の実利用における主要な障害となっている。

RBE は、信頼モデルを根本から変える洗練された解決策を提供する。RBE では、ユーザーが鍵ペアを自身で生成し、その公開鍵と自身の ID をシステムに登録する。ユーザーが公開鍵と ID を登録するたびに、鍵管理局 (KC) と呼ばれる第三者機関が公開パラメーターを更新し、ユーザーに補助復号鍵という復号に必要な鍵を返す。このとき、KC は秘密情報を保持せずに、内部的な処理も行わないため、透明性の高い第三者機関となっている。そのため、IBE における鍵預託問題を解決しており、その社会導入に大きく貢献できると考えられている。暗号化と復号化は IBE と

<sup>1</sup> 横浜国立大学 教育推進機構/先端科学高等研究院  
Organization for the Promotion of Education/Institute of Advanced Sciences, Yokohama National University  
a) tomita-toi-sk@ynu.ac.jp

同様に行われる。公開パラメーターと受信者の ID が与えられれば、送信者は受信者宛てのメッセージを暗号化できる。受信者はその後、自身の秘密鍵と補助復号鍵を用いてその暗号文を復号できる。正当性や安全性は、IBE と同様に定義される。RBE には、効率性に関する特有の要件がいくつかある。特に重要なのは、補助復号鍵の更新回数である。RBE ではユーザーが登録されるたびに公開パラメーターが更新されるため、それに対応して補助復号鍵も更新する必要がある。ただし、ユーザーが登録されるたびにすべてのユーザーの補助復号鍵を更新することは現実的でない。そのため、各ユーザーの補助復号鍵の更新回数は登録ユーザー数に対して劣線形であるという要件がある。さらに、公開パラメーターのサイズや各アルゴリズムの実行時間も、登録ユーザー数に対して劣線形である必要がある。

初期の研究 [7], [8] 以降、RBE に対する急速な関心が高まり、効率性 [2], [3], [5], [9]、セキュリティ [1], [10]、および理論的な下限 [4], [11], [14], [15] など、さまざまな方向に進展を遂げてきた。

**従来の RBE 方式の効率性について。**本稿では RBE の効率性に焦点を当てる。初期の RBE 方式 [7], [8] は、識別不可能性難読化やガーブル回路などの暗号プリミティブによる非ブラックボックス構成であった。しかし、これらの方法は理論的であり、実用上非効率的であった。[2] で議論されているように、これらの構成における暗号文の推定サイズはテラバイト規模であり、実利用は現実的ではありません。

Glaeser ら [9] の最近の構成、この状況を大幅に改善した。彼らはペアリングに基づく最初の具体的で高効率な RBE 方式を実現した。彼らの RBE 方式の暗号文サイズはわずか 914 バイトである。ただし、この方式は ID 空間が小さいという欠点があったが、Fiore ら [5] によってこの欠点は解決された。これは画期的な成果であり、RBE の実用的な展開が可能なことを初めて示した。

一方、Döttolting ら [3] は、LWE に基づく効率的な RBE 方式を提案した。これは耐量子 RBE の最初のブラックボックス構成である。彼らの方式は実現可能な効率性を達成しているが、その暗号文サイズは依然として実用的ではない。登録ユーザー数が  $N$  で ID の長さが  $\ell$  である場合、彼らの RBE 方式の暗号文は約  $O(\ell \log N)$  個の LWE 暗号文からなる。Fiore ら [5] は、 $2^{10}$  人の登録ユーザーを有するシステムにおいて、具体的な暗号文サイズが 2.4 GB であると報告している。その後、Fiore ら [5] は [3] よりも短い暗号文を持つ LWE ベースの RBE 方式を提案した。彼らの RBE 方式の暗号文は  $O(\log^2 N)$  個の LWE 暗号文からなる。ただし、これは「選択的コンパクトネスモデル」と呼ばれる制限されたモデル下でのみ実現可能である。

まとめると、従来の LWE ベースの RBE 方式の暗号文サイズは、制限モデルであっても少なくとも  $O(\log^2 N)$  個

の LWE 暗号文を必要とする。すなわち、登録ユーザーの数が増加するにつれ、従来の LWE ベースの方式はスケーラビリティに欠けます。したがって、本研究では LWE からコンパクトでスケーラブルな RBE 方式を構成可能かを考える。

## 1.2 貢献

本研究では、暗号文サイズがよりコンパクトな LWE ベースの RBE 方式を提案する。提案方式の暗号文は、約  $2\ell + \log N$  個の LWE 暗号文からなる。より正確には、 $2\ell + \text{hw}(N)$  個の LWE 暗号文からなる。ここで、 $\text{hw}(N)$  は  $N$  の二進表現のハミング重さである。 $N$  が 2 の幂乗の場合、 $\text{hw}(N) = 1$  となり、暗号文のサイズが最小化される。提案方式と従来の LWE ベースの RBE 方式の比較を表 1 に示す。表 1 から、我々の結果は LWE(またはより一般に耐量子)RBE 方式の効率とスケーラビリティを大幅に改善していることがわかる。

この結果を得るために、我々は新しい暗号学的プリミティブである分解可能なラコニック暗号を導入し、これを汎用的にコンパクトな RBE 方式に変換する新たなアプローチを開発した。このアイデアは、多受信者暗号の文脈で用いられている分解可能性 [13] という概念から着想を得ている。我々のアイデアは登録属性ベース/関数型暗号 [6], [12] などのより高度な暗号技術にも有用であると考えている。

## 2. 前準備

**記法.**自然数  $n \in \mathbb{N}$  に対して、 $[n] = \{1, \dots, n\}$  と表記する。二進数  $b \in \{0, 1\}^n$  に対して、そのハミング重みを  $\text{hw}(b)$  で表す。 $b \in \mathbb{N}$  のとき、 $\text{hw}(b)$  は  $b$  の二進表現のハミング重みを表す。分布  $\mathcal{D}$  に対して、 $x \leftarrow \mathcal{D}$  と表記して  $x$  が分布  $\mathcal{D}$  からサンプルされたことを表す。集合  $\mathcal{S}$  に対して、 $x \leftarrow \mathcal{S}$  と表記して  $x$  が集合  $\mathcal{S}$  から一様ランダムにサンプルされたことを表す。

本稿では、セキュリティパラメータを表すために  $\lambda$  と表記する。 $\text{poly}(\lambda)$  はある定数  $c \in \mathbb{N}$  に対して  $O(\lambda^c)$  となる関数を表し、 $\text{negl}(\lambda)$  はすべての  $c \in \mathbb{N}$  に対して  $o(\lambda^{-c})$  となる関数を表す。二つの分布の族  $\mathcal{D}_0 := \{\mathcal{D}_{0,\lambda}\}_{\lambda \in \mathbb{N}}$  と  $\mathcal{D}_1 := \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$  に対して、これらを無視できない確率で識別する確率的多項式時間 (PPT) アルゴリズムが存在しないならば、 $\mathcal{D}_0$  と  $\mathcal{D}_1$  は計算量的に識別不可能であるといい、 $\mathcal{D}_0 \approx_c \mathcal{D}_1$  と表記する。

### 2.1 登録ベース暗号

ここでは、登録ベース暗号を定義する。

**定義 2.1** (登録ベース暗号 [7]). ID 空間が  $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$  でメッセージ空間が  $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$  である登録ベース暗号 (RBE) は以下の PPT アルゴリズムからなる。

方式	暗号文サイズ	公開パラメーターと 補助復号鍵のサイズ	秘密鍵と 公開鍵のサイズ	更新回数
[3]	$(4m\ell + 2m + n + 2) \log N \cdot  \mathcal{R}_q $	$O(\log N)$	$O(1)$	$\log N$
[5] <sup>†</sup>	$2(4m \log N + 3m + 2) \log N \cdot  \mathcal{R}_q $	$O(\log N)$	$O(1)$	$\log N$
<b>提案方式</b>	$\log N + (2m\ell + m + \text{hw}(N)) \cdot  \mathcal{R}_q $	$O(\log N)$	$O(1)$	$\log N$

表 1 LWE ベースの RBE 方式の比較.  $N$  は登録ユーザー数を表す.  $n, q, m = O(n \log q)$  は格子のパラメータであり,  $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$  は多項式環である.

†: [5] の効率は制限されたモデルにおいてのみ達成される.

- $\text{Setup}(1^\lambda) \rightarrow \text{crs}$ : セットアップアルゴリズムはセキュリティパラメータ  $\lambda$  を入力として, 共通参照文字列  $\text{crs}$  を出力する.
- $\text{KGen}(\text{crs}) \rightarrow (\text{sk}, \text{pk})$ : 鍵生成アルゴリズムは共通参照文字列  $\text{crs}$  を入力として, 鍵ペア  $(\text{sk}, \text{pk})$  を出力する.
- $\text{Reg}^{[\text{aux}]}(\text{crs}, \text{id}, \text{pk}) \rightarrow \text{pp}$ : 登録アルゴリズムは共通参照文字列  $\text{crs}$  と  $\text{IDid} \in \{0, 1\}^\ell$  と公開鍵  $\text{pk}$  を入力として, 補助情報  $\text{aux}$  への読み書き可能なランダムアクセスを通じて公開パラメータ  $\text{pp}$  を決定的に出力する.
- $\text{Enc}(\text{crs}, \text{pp}, \text{id}, \text{m}) \rightarrow \text{ct}$ : 暗号化アルゴリズムは共通参照文字列  $\text{crs}$  と受信者の  $\text{IDid} \in \{0, 1\}^\ell$  とメッセージ  $\text{m} \in \mathcal{M}$  を入力として, 暗号文  $\text{ct}$  を出力する.
- $\text{Upd}^{[\text{aux}]}(\text{pp}, \text{id}) \rightarrow \text{hsk}$ : 更新アルゴリズムは共通参照文字列  $\text{crs}$  と  $\text{IDid} \in \{0, 1\}^\ell$  を入力として, 補助情報  $\text{aux}$  への読み取り可能なランダムアクセスを通じて補助復号鍵  $\text{hsk}$  を出力する.
- $\text{Dec}(\text{sk}, \text{hsk}, \text{ct}) \rightarrow \text{m} \in \mathcal{M} \cup \{\text{GetUpd}, \perp\}$ : 復号アルゴリズムは秘密鍵  $\text{sk}$  と補助復号鍵  $\text{hsk}$  と暗号文  $\text{ct}$  を入力として,  $\text{m} \in \mathcal{M} \cup \{\perp\}$  を出力する. ここで,  $\perp$  は復号エラーを表す特殊記号であり,  $\text{GetUpd}$  は補助復号鍵の更新が必要であることを表す.

**定義 2.2** (正当性, コンパクト性, 効率性). セキュリティパラメータ  $\lambda$  と攻撃者  $\mathcal{A}$  に対して,  $\mathcal{A}$  とチャレンジヤ  $\text{Chal}$  との間の正当性ゲームを以下のように定義する.

- **セットアップ:**  $\text{Chal}$  は  $\text{crs} \leftarrow \text{Setup}(1^\lambda)$  をサンプルし,  $\text{aux} := \perp$  と  $\text{pp}_0 := \perp$  を初期化する. また,  $\text{Chal}$  は登録された ID の集合  $\mathcal{S} := \emptyset$ , 暗号化クエリの回数を追跡するためのカウンター  $t := 0$ , チャレンジ  $\text{IDid}^* := \perp$  を初期化する. 最後に,  $\text{Chal}$  は  $\mathcal{A}$  に  $\text{crs}$  を渡す.
- **クエリフェーズ:**  $\mathcal{A}$  は以下のクエリを行う.
  - **非ターゲット ID 登録クエリ:** このクエリでは,  $\mathcal{A}$  は  $\text{IDid} \in \mathcal{ID}$  と公開鍵  $\text{pk}$  を指定する.  $\text{id} \in \mathcal{S}$  の場合,  $\text{Chal}$  は  $\perp$  を返す. そうでない場合,  $\text{Chal}$  は  $\text{id}$  を  $\mathcal{S}$  に追加し,  $\text{pp}_{|\mathcal{S}|} := \text{Reg}^{[\text{aux}]}(\text{crs}, \text{id}, \text{pk})$  を計算することによって  $(\text{id}, \text{pk})$  を登録し,  $(\text{pp}_{|\mathcal{S}|}, \text{aux})$  を  $\mathcal{A}$  に返す.
  - **ターゲット ID 登録クエリ:** このクエリでは,  $\mathcal{A}$  は  $\text{IDid}$  を指定する.  $\text{id}^* \neq \perp$  または  $\text{id} \in \mathcal{S}$  の場合,  $\text{Chal}$  は  $\perp$  を返す. そうでない場合,  $\text{Chal}$  は  $\text{id}$  を  $\mathcal{S}$  に追加し,  $(\text{sk}^*, \text{pk}^*) \leftarrow \text{KGen}(\text{crs})$  をサンプルし,  $\text{pp}_{|\mathcal{S}|} :=$

$\text{Reg}^{[\text{aux}]}(\text{crs}, \text{id}, \text{pk}^*)$  を計算することによって  $(\text{id}, \text{pk}^*)$  を登録する. また,  $\text{hsk}^* := \text{Upd}^{[\text{aux}]}(\text{pp}, \text{id})$  も計算する.  $\text{Chal}$  は  $\text{id}^* := \text{id}$  をセットし,  $(\text{pp}_{|\mathcal{S}|}, \text{aux}, \text{sk}^*, \text{pk}^*)$  を  $\mathcal{A}$  に返す.

- **暗号化クエリ:** このクエリでは,  $\mathcal{A}$  はメッセージ  $\text{m}_t \in \mathcal{M}$  を指定する.  $\text{id}^* = \perp$  の場合,  $\text{Chal}$  は  $\perp$  を返す. そうでない場合,  $\text{Chal}$  はカウンター  $t := t + 1$  をインクリメントし,  $\text{ct}_t \leftarrow \text{Enc}(\text{crs}, \text{pp}, \text{id}^*, \text{m}_t)$  を計算し,  $(t, \text{ct}_t)$  を  $\mathcal{A}$  に返す.
- **復号クエリ:** このクエリでは,  $\mathcal{A}$  はインデックス  $j \in [t]$  を指定する.  $\text{Chal}$  は  $\text{m}'_j \leftarrow \text{Dec}(\text{sk}^*, \text{hsk}^*, \text{ct}_j)$  を計算する.  $\text{m}'_j = \text{GetUpd}$  の場合,  $\text{Chal}$  は  $\text{hsk}^* := \text{Upd}^{[\text{aux}]}(\text{pp}, \text{id})$  を実行することによって補助復号鍵を更新し,  $\text{m}'_j \leftarrow \text{Dec}(\text{sk}^*, \text{hsk}^*, \text{ct}_j)$  を再計算する.  $\text{m}'_j \neq \text{m}_j$  ならば, ゲームは停止し  $b = 1$  を出力する.  $\mathcal{A}$  のクエリが終了し, (復号クエリの結果として) ゲームが停止していない場合, ゲームは  $b = 0$  を出力する.

$N = |\mathcal{S}|$  とする. 任意の  $\lambda \in \mathbb{N}$  と多項式回のクエリを行う任意の攻撃者  $\mathcal{A}$  に対して以下が成り立つとき, RBE 方式は正当性, コンパクト性, (弱) 効率性を満たすという.

- **正当性:** 上記のゲームで  $\Pr[b = 1] = \text{negl}(\lambda)$  となる.
- **コンパクト性:** 任意の  $i \in [N]$  に対して,  $|\text{pp}_i| = \text{poly}(\lambda, \log i)$  となる. また, (ゲームのすべての時点において)  $|\text{hsk}^*| = \text{poly}(\lambda, \log N)$  となる.
- **(弱) 効率性<sup>\*1</sup>:** 上記のゲームにおいて,  $\text{Chal}$  は更新アルゴリズム  $\text{Upd}$  を高々  $O(\log N)$  回しか呼び出さない. 各呼び出しは, RAM 計算モデルにおいて  $\text{poly}(\log N)$  時間で実行されること.

**定義 2.3** (安全性). セキュリティパラメータ  $\lambda$ , 攻撃者  $\mathcal{A}$ , ビット  $b \in \{0, 1\}$  に対して,  $\mathcal{A}$  とチャレンジヤ  $\text{Chal}$  との間の安全性ゲームを以下のように定義する.

- **セットアップ:**  $\text{Chal}$  は  $\text{crs} \leftarrow \text{Setup}(1^\lambda)$  をサンプルし  $\mathcal{A}$  に渡す. また,  $\text{Chal}$  は補助情報  $\text{aux} := \perp$ , 公開パラメータ  $\text{pp} := \perp$ , ID の集合  $\mathcal{S} := \emptyset$ , チャレンジ ID  $\text{id}^* := \perp$  を初期化する.
- **クエリフェーズ:**  $\mathcal{A}$  は以下のクエリを行う.
  - **非ターゲット ID 登録クエリ:** このクエリでは,  $\mathcal{A}$  は

<sup>\*1</sup> 本稿では, 弱効率性のみを考慮しているが, 一部の研究 [7], [8], [10] では登録アルゴリズムの  $\text{Reg}$  の実行時間が  $\text{poly}(\log N)$  であるという効率性要件を考慮している.

- $\text{IDid} \in \mathcal{ID}$  と公開鍵  $\text{pk}$  を指定する.  $\text{id} \in \mathcal{S}$  の場合,  $\text{Chal}$  は  $\perp$  を返す. そうでない場合,  $\text{Chal}$  は  $\text{id}$  を  $\mathcal{S}$  に追加し,  $\text{pp}_{|\mathcal{S}|} := \text{Reg}^{[\text{aux}]}(\text{crs}, \text{id}, \text{pk})$  を計算することによって  $(\text{id}, \text{pk})$  を登録し,  $(\text{pp}, \text{aux})$  を  $\mathcal{A}$  に返す.
- ターゲット ID 登録クエリ: このクエリでは,  $\mathcal{A}$  は  $\text{IDid}$  を指定する.  $\text{id}^* \neq \perp$  または  $\text{id}^* \in \mathcal{S}$  の場合,  $\text{Chal}$  は  $\perp$  を返す. そうでない場合,  $\text{Chal}$  は  $\text{id}$  を  $\mathcal{S}$  に追加し,  $(\text{sk}^*, \text{pk}^*) \leftarrow \text{KGen}(\text{crs})$  をサンプルし,  $\text{pp} := \text{Reg}^{[\text{aux}]}(\text{crs}, \text{id}, \text{pk}^*)$  を計算することによって  $(\text{id}, \text{pk}^*)$  を登録する. また,  $\text{hsk}^* := \text{Upd}^{[\text{aux}]}(\text{pp}, \text{id})$  も計算する.  $\text{Chal}$  は  $\text{id}^* := \text{id}$  をセットし,  $(\text{pp}, \text{aux}, \text{pk}^*)$  を  $\mathcal{A}$  に返す.
  - **チャレンジフェーズ**:  $\mathcal{A}$  は二つのメッセージ  $\text{m}_0^*, \text{m}_1^* \in \mathcal{M}$  と  $\text{IDid} \in \mathcal{ID}$  を指定する.  $\text{id} \in \mathcal{S} \setminus \{\text{id}^*\}$  の場合, ゲームは停止する. そうでない場合,  $\text{Chal}$  はチャレンジ暗号文  $\text{ct}^* \leftarrow \text{Enc}(\text{crs}, \text{pp}, \text{id}, \text{m}_b^*)$  を返す.
  - **出力フェーズ**: 最後に,  $\mathcal{A}$  は  $b' \in \{0, 1\}$  を出力する. 任意の  $\lambda \in \mathbb{N}$  と任意の PPT 攻撃者  $\mathcal{A}$  に対して, 上記のゲームで  $|\Pr[b' = 1 | b = 0] - \Pr[b' = 1 | b = 1]| = \text{negl}(\lambda)$  が成り立つとき, RBE 方式は安全性を満たすという.

### 3. 分解可能なラコニック暗号

本章では, 分解可能なラコニック暗号を導入する.

**定義 3.1** (分解可能なラコニック暗号). メッセージ空間が  $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$  である分解可能なラコニック暗号 (dLE) は以下の PPT アルゴリズムからなる.

- $\text{Setup}(1^\lambda, 1^\ell) \rightarrow (\text{crs}, \text{dig}, \text{st})$ : セットアップアルゴリズムはセキュリティパラメータ  $\lambda$  とインデックス長  $\ell$  を入力として, 共通参照文字列  $\text{crs}$  とダイジェスト  $\text{dig}$  と状態  $\text{st}$  を出力する.  $\text{crs}$  は暗黙的に乱数空間  $\mathcal{R} = (\mathcal{R}^i, \mathcal{R}^d)$  を定義する. ここで,  $\mathcal{R}^i$  はダイジェスト非依存暗号化のための乱数空間であり,  $\mathcal{R}^d$  はダイジェスト依存暗号化のための乱数空間である.
- $\text{KGen}(\text{crs}) \rightarrow (\text{sk}, \text{pk})$ : 鍵生成アルゴリズムは共通参照文字列  $\text{crs}$  を入力として, 鍵ペア  $(\text{sk}, \text{pk})$  を出力する.
- $\text{IsValid}(\text{crs}, \text{pk}) \rightarrow b$ : 鍵検証アルゴリズムは共通参照文字列  $\text{crs}$  と公開鍵  $\text{pk}$  を入力として,  $\text{pk}$  が有効かどうかを示すビット  $b \in \{0, 1\}$  を決定的に出力する.
- $\text{MemUpd}^{[\text{st}]}(\text{crs}, \text{id}, \text{pk}) \rightarrow \text{dig}$ : メンバー更新アルゴリズムは共通参照文字列  $\text{crs}$  とインデックス  $\text{id} \in \{0, 1\}^\ell$  と公開鍵  $\text{pk}$  を入力として, 状態  $\text{st}$  への読み書き可能なランダムアクセスを通じて (更新された) ダイジェスト  $\text{dig}$  を決定的に出力する.
- $\text{Enc}(\text{crs}, \text{dig}, \text{id}, \text{m}; \text{r}^i, \text{r}^d) \rightarrow \text{ct} = (\text{ct}^i, \text{ct}^d)$ : 暗号化アルゴリズムは以下の二つのアルゴリズムに分解される.
- $\text{Enc}^i(\text{crs}, \text{id}; \text{r}^i) \rightarrow \text{ct}^i$ : ダイジェスト非依存暗号化アルゴリズムは共通参照文字列  $\text{crs}$  とインデックス  $\text{id} \in \{0, 1\}^\ell$  と乱数  $\text{r}^i \in \mathcal{R}^i$  を入力として, ダイジェス

ト非依存暗号文  $\text{ct}^i$  を出力する.

- $\text{Enc}^d(\text{crs}, \text{dig}, \text{id}, \text{m}; \text{r}^i, \text{r}^d) \rightarrow \text{ct}^d$ : ダイジェスト依存暗号化アルゴリズムは共通参照文字列  $\text{crs}$  とインデックス  $\text{id} \in \{0, 1\}^\ell$  とメッセージ  $\text{m} \in \mathcal{M}$  と乱数  $(\text{r}^i, \text{r}^d) \in \mathcal{R}^i \times \mathcal{R}^d$  を入力として, ダイジェスト依存暗号文  $\text{ct}^d$  を出力する.
- $\text{WGen}^{[\text{st}]}(\text{crs}, \text{id}) \rightarrow \text{wit}$ : 証拠生成アルゴリズムは共通参照文字列  $\text{crs}$  とインデックス  $\text{id} \in \{0, 1\}^\ell$  を入力として, 状態  $\text{st}$  への読み取り可能なランダムアクセスを通じて証拠  $\text{wit}$  を出力する.
- $\text{Dec}(\text{sk}, \text{wit}, \text{ct}) \rightarrow \text{m} \in \mathcal{M} \cup \{\perp\}$ : 復号アルゴリズムは秘密鍵  $\text{sk}$  と証拠  $\text{wit}$  と暗号文  $\text{ct}$  を入力として,  $\text{m} \in \mathcal{M} \cup \{\perp\}$  を出力する. ここで,  $\perp$  は復号エラーを表す特殊記号である.

さらに, 上記の全てのアルゴリズムの実行時間は遅くとも  $\text{poly}(\lambda, \ell)$  である.

注釈 3.2 (従来のラコニック暗号 [3] との違い). 以下に従来のラコニック暗号 [3] と分解可能なラコニック暗号との違いをまとめる.

- 最も重要な変更点は暗号化アルゴリズムをダイジェスト非依存部分とダイジェスト依存部分に分解できる分解可能性である. この変更に伴い, 暗号文のコンパクト性 (定義 3.4) とマルチダイジェスト安全性 (定義 3.5) を導入する. 暗号文のコンパクト性はダイジェスト依存暗号文が十分にコンパクトであることを要求する. マルチダイジェスト安全性は, チャレンジ暗号文からメッセージの情報が漏洩しないことを要求する. ここで, チャレンジ暗号文は, 一つのダイジェスト非依存暗号文と, 複数のダイジェストと共に乱数によって生成された複数のダイジェスト依存暗号文からなる.
- [12] に着想を得て, 鍵検証アルゴリズムを追加した. この変更に伴い, 鍵生成アルゴリズムによって生成された公開鍵は鍵検証アルゴリズムによる検証に通るという完全性 (定義 3.3) を要求する.
- 正当性ゲーム (定義 3.4) において, チャレンジ暗号文を攻撃者が選択した時刻固有のダイジェストから生成するように変更する.

**完全性, 正当性, コンパクト性.** ここでは, dLE 方式の完全性, 正当性, コンパクト性を定義する.

**定義 3.3** (完全性). 任意の  $\lambda, \ell \in \mathbb{N}$ , 任意の  $(\text{crs}, \text{dig}, \text{st}) \leftarrow \text{Setup}(1^\lambda, 1^\ell)$ , 任意の  $(\text{sk}, \text{pk}) \leftarrow \text{KGen}(\text{crs})$  に対して  $\text{IsValid}(\text{crs}, \text{pk}) = 1$  が成り立つとき, dLE 方式は完全性を満たすという.

**定義 3.4** (正当性とコンパクト性). パラメータ  $\lambda, \ell \in \mathbb{N}$  と攻撃者  $\mathcal{A}$  に対して,  $\mathcal{A}$  とチャレンジャ  $\text{Chal}$  との間の正当性ゲームを以下のように定義する.

- **セットアップ:**  $\text{Chal}$  は  $(\text{crs}, \text{dig}, \text{st}) \leftarrow \text{Setup}(1^\lambda, 1^\ell)$  をサンプルし, それらを  $\mathcal{A}$  に渡す.  $\text{Chal}$  はカウンター  $\text{ctr} := 0$  と辞書  $\text{Dict}_H$  を初期化する.
  - **クエリフェーズ:**  $\mathcal{A}$  は以下のクエリを行う.
    - 正直な更新クエリ: このクエリでは,  $\mathcal{A}$  はインデックス  $\text{id} \in \{0, 1\}^\ell$  を指定する. これに対し,  $\text{Chal}$  は  $\text{ctr} := \text{ctr} + 1$  をインクリメントし,  $(\text{sk}, \text{pk}) \leftarrow \text{KGen}(\text{crs})$  を生成し,  $\text{dig} := \text{MemUpd}^{[\text{st}]}(\text{crs}, \text{id}, \text{pk})$  を実行し,  $\mathcal{A}$  に  $(\text{dig}, \text{st}, \text{sk}, \text{pk})$  を返す.
    - 悪意のある更新クエリ: このクエリでは,  $\mathcal{A}$  はインデックス  $\text{id} \in \{0, 1\}^\ell$  と公開鍵  $\text{pk}$  を指定する. これに対し,  $\text{Chal}$  はまず  $\text{isValid}(\text{crs}, \text{pk}) = 1$  をチェックする. チェックに失敗した場合,  $\perp$  を出力する. そうでない場合,  $\text{Chal}$  は  $\text{ctr} := \text{ctr} + 1$  をインクリメントし,  $\text{dig} := \text{MemUpd}^{[\text{st}]}(\text{crs}, \text{id}, \text{pk})$  を実行し,  $\mathcal{A}$  に  $(\text{dig}, \text{st})$  を返す. さらに,  $\text{Chal}$  は  $\text{id}$  を集合  $\mathcal{C}_M$  に追加し,  $(\text{dig}_{\text{ctr}}, \text{st}_{\text{ctr}}) := (\text{dig}, \text{st})$  を格納する.
  - **チャレンジフェーズ:** ある時点で,  $\mathcal{A}$  はインデックス  $\text{id}^* \in \{0, 1\}^\ell$  とメッセージ  $\text{m}^* \in \mathcal{M}$  と時刻  $t \in [\text{ctr}]$  を指定する.  $\text{id}^* \notin \text{Dict}_H$  の場合,  $\text{Chal}$  は  $b = 1$  を出力する. そうでない場合,  $\text{Chal}$  は以下のように処理する.
    - (1)  $\text{ct}^* = (\text{ct}^i, \text{ct}^d) \leftarrow \text{Enc}(\text{crs}, \text{dig}_t, \text{id}^*, \text{m}^*)$  を計算する.
    - (2)  $(\text{sk}^*, \text{pk}^*) \leftarrow \text{Dict}_H[\text{id}^*]$  を取得する.
    - (3)  $\text{wit}^* := \text{WGen}^{[\text{st}_t]}(\text{crs}, \text{id}^*)$  を生成する.
    - (4)  $\text{m} := \text{Dec}(\text{sk}^*, \text{wit}^*, \text{ct}^*)$  を計算する.
 最後に,  $\text{Chal}$  は  $\text{m} = \text{m}^*$  ならば  $b = 1$  を, そうでないなら  $b = 0$  を出力する.
- 任意の  $\lambda, \ell \in \mathbb{N}$  と任意の攻撃者  $\mathcal{A}$  に対して以下が成り立つとき, dLE 方式は正当性とコンパクト性を満たすという.
- **正当性:** 上記のゲームで  $\Pr[b = 1] = \text{negl}(\lambda)$  となる.
  - **コンパクト性:** ゲーム内の全ての時刻において  $|\text{dig}| = \text{poly}(\lambda, \ell)$  と  $|\text{wit}^*| = \text{poly}(\lambda, \ell)$  と  $|\text{ct}^d| = \text{poly}(\lambda, \log \ell)$  が成り立つ.

**安全性.** ここでは, dLE 方式の安全性を定義する.

**定義 3.5** ( $L$ -ダイジェスト安全性).  $b \in \{0, 1\}$  とする. パラメータ  $\lambda, \ell, L \in \mathbb{N}$  と攻撃者  $\mathcal{A}$  に対して,  $\mathcal{A}$  とチャレンジ  $\text{Chal}$  との間の安全性ゲームを以下のように定義する.

- **セットアップ:**  $\text{Chal}$  は  $(\text{crs}, \text{dig}, \text{st}) \leftarrow \text{Setup}(1^\lambda, 1^\ell)$  をサンプルし, それらを  $\mathcal{A}$  に渡す.  $\text{Chal}$  はカウンター  $\text{ctr} := 0$  と集合  $\mathcal{C}_M := \emptyset$  を初期化する.
- **クエリフェーズ:**  $\mathcal{A}$  は以下のクエリを行う.
  - 正直な更新クエリ: このクエリでは,  $\mathcal{A}$  はインデックス  $\text{id} \in \{0, 1\}^\ell$  を指定する. これに対し,  $\text{Chal}$  は  $\text{ctr} := \text{ctr} + 1$  をインクリメントし,  $(\text{sk}, \text{pk}) \leftarrow \text{KGen}(\text{crs})$  を生成し,  $\text{dig} := \text{MemUpd}^{[\text{st}]}(\text{crs}, \text{id}, \text{pk})$  を

実行し,  $\mathcal{A}$  に  $(\text{dig}, \text{st}, \text{sk}, \text{pk})$  を返す. さらに,  $\text{Chal}$  は  $(\text{dig}_{\text{ctr}}, \text{st}_{\text{ctr}}) := (\text{dig}, \text{st})$  を格納する.

- 悪意のある更新クエリ: このクエリでは,  $\mathcal{A}$  はインデックス  $\text{id} \in \{0, 1\}^\ell$  と公開鍵  $\text{pk}$  を指定する. これに対し,  $\text{Chal}$  はまず  $\text{isValid}(\text{crs}, \text{pk}) = 1$  をチェックする. チェックに失敗した場合,  $\perp$  を出力する. そうでない場合,  $\text{Chal}$  は  $\text{ctr} := \text{ctr} + 1$  をインクリメントし,  $\text{dig} := \text{MemUpd}^{[\text{st}]}(\text{crs}, \text{id}, \text{pk})$  を実行し,  $\mathcal{A}$  に  $(\text{dig}, \text{st})$  を返す. さらに,  $\text{Chal}$  は  $\text{id}$  を集合  $\mathcal{C}_M$  に追加し,  $(\text{dig}_{\text{ctr}}, \text{st}_{\text{ctr}}) := (\text{dig}, \text{st})$  を格納する.

- **チャレンジフェーズ:** ある時点で,  $\mathcal{A}$  はインデックス  $\text{id}^* \in \{0, 1\}^\ell$  とメッセージ  $\text{m}_0^*, \text{m}_1^* \in \mathcal{M}$  と  $L$  個の時刻  $t_1, \dots, t_L \in [\text{ctr}]$  を指定する.  $\text{id}^* \in \mathcal{C}_M$  の場合,  $\text{Chal}$  は  $0$  を出力する. そうでない場合,  $\text{Chal}$  は  $\text{r}^i \leftarrow \$ \mathcal{R}^i$  と  $\text{r}_1^d, \dots, \text{r}_L^d \leftarrow \$ \mathcal{R}^d$  をサンプルし,

$$\text{ct}^i := \text{Enc}^i(\text{crs}, \text{id}^*; \text{r}^i),$$

$$\text{ct}_i^d := \text{Enc}^d(\text{crs}, \text{dig}_{t_i}, \text{id}^*, \text{m}_b^*; \text{r}_i^i, \text{r}_i^d), \forall i \in [L],$$

を計算し,  $(\text{ct}^i, \text{ct}_1^d, \dots, \text{ct}_L^d)$  を  $\mathcal{A}$  に返す.

- **出力フェーズ:** 最後に,  $\mathcal{A}$  は  $b' \in \{0, 1\}$  を出力する. 任意の  $\lambda, \ell = \text{poly}(\lambda), L = \text{poly}(\lambda) \in \mathbb{N}$  と任意の PPT 攻撃者  $\mathcal{A}$  に対して, 上記のゲームで  $|\Pr[b' = 1 | b = 0] - \Pr[b' = 1 | b = 1]| = \text{negl}(\lambda)$  が成り立つとき, dLE 方式は  $L$ -ダイジェスト安全性を満たすという.

Döttling ら [3] が提案した通常のラコニック暗号方式は, そのまま分解可能性を満たす. したがって, 下記の定理が成り立つ.

**定理 3.6** ([3]). LWE 仮定の元で  $\text{poly}(\lambda)$ -ダイジェスト安全性を満たす dLE 方式が存在する.

ページ数の都合上, 方式の記述や証明は省略する.

## 4. 一般的構成法

本章では, dLE 方式からコンパクトな暗号文を持つ RBE 方式への一般的変換を示す.

### 4.1 構成

$\Pi_{\text{dLE}} = (\tilde{\text{Setup}}, \tilde{\text{KGen}}, \tilde{\text{isValid}}, \tilde{\text{MemUpd}}, \tilde{\text{Enc}} = (\tilde{\text{Enc}}^i, \tilde{\text{Enc}}^d), \tilde{\text{WGen}}, \tilde{\text{Dec}})$  をメッセージ空間  $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$  の dLE 方式とする. ID 空間  $\mathcal{ID} = \{0, 1\}^\ell$  とメッセージ空間  $\mathcal{M}$  の RBE 方式  $\Pi_{\text{RBE}} = (\text{Setup}, \text{KGen}, \text{Reg}, \text{Enc}, \text{Upd}, \text{Dec})$  を以下のように構成する. ここで,  $\ell$  は  $\Pi_{\text{dLE}}$  のインデックス長である. 構成では, 以下の記述を用いる.

- $\Pi_{\text{RBE}}$  方式は, 内部的に  $\Pi_{\text{dLE}}$  方式のダイジェストの複数バージョンを維持する.
- 補助情報  $\text{aux} = (\text{ctr}, \text{Dict}_1, \text{Dict}_2, \text{pp}, \tilde{\text{st}})$  は, 以下の要素からなる.
  - カウンター  $\text{ctr} \geq 0$  は, システム内の登録済みユーザーの現在の数を示す.

- 辞書  $\text{Dict}_1$  は、カウンター  $\text{ctr}$  を  $\text{ctr}$  番目の登録済み ID/鍵ペア  $(\text{id}, \text{pk})$  にマッピングする.
- 辞書  $\text{Dict}_2$  は、 $\text{IDid}$  を  $\text{id}$  と関連付けられた補助復号鍵の集合にマッピングする.
- 現在の公開パラメータ  $\text{pp} = (\text{ctr}, \tilde{\text{dig}}_1, \dots, \tilde{\text{dig}}_{\text{hw}(\text{ctr})})$ において、 $\text{ctr}$  は現在のカウンターであり、 $\tilde{\text{dig}}_1, \dots, \tilde{\text{dig}}_{\text{hw}(\text{ctr})}$  は  $\Pi_{\text{dLE}}$  のダイジェストである. ダイジェストのうち、 $\tilde{\text{dig}}_1$  が最も古いバージョンであり、 $\tilde{\text{dig}}_{\text{hw}(\text{ctr})}$  が最も新しいバージョンである.
- $\tilde{\text{st}}$  は最も新しいダイジェスト  $\tilde{\text{dig}}_{\text{hw}(\text{ctr})}$  対応する状態である.
- 自然数  $x \in \mathbb{N}$  に対して、 $x$  の二進表現において最下位ビットが 1 である位置を出力する関数を  $\delta$  とする.

## 構成.

- $\text{Setup}(1^\lambda) \rightarrow \text{crs}:$ 
  - (1)  $\ell = O(\lambda)$  を設定し、 $(\tilde{\text{crs}}, \tilde{\text{dig}}, \tilde{\text{st}}) \leftarrow \tilde{\text{Setup}}(1^\lambda, 1^\ell)$  をサンプルする.
  - (2)  $\text{aux} := (0, \emptyset, \emptyset, (0, \tilde{\text{dig}}), \tilde{\text{st}})$  を初期化する.
  - (3)  $\text{crs} := \tilde{\text{crs}}$  を出力する.
- $\text{KGen}(\text{crs}) \rightarrow (\text{sk}, \text{pk}):$ 
  - (1)  $\text{crs} = \tilde{\text{crs}}$  とする.
  - (2)  $(\text{sk}, \text{pk}) := (\tilde{\text{sk}}, \tilde{\text{pk}}) \leftarrow \tilde{\text{KGen}}(\tilde{\text{crs}})$  を出力する.
- $\text{Reg}^{[\text{aux}]}(\text{crs}, \text{id}, \text{pk}) \rightarrow \text{pp}:$ 
  - (1)  $\text{crs} = \tilde{\text{crs}}$ ,  $\text{pk} = \tilde{\text{pk}}$ ,  $\text{aux} = (\text{ctr}, \text{Dict}_1, \text{Dict}_2, \text{pp}, \tilde{\text{st}})$ ,  $\text{pp} = (\text{ctr}, \tilde{\text{dig}}_1, \dots, \tilde{\text{dig}}_{\text{hw}(\text{ctr})})$  とする.
  - (2)  $\tilde{\text{isValid}}(\tilde{\text{crs}}, \tilde{\text{pk}}) = 0$  なら  $\text{pp}$  を出力する.
  - (3)  $\text{ctr} := \text{ctr} + 1$  と  $\text{Dict}_1[\text{ctr}] := (\text{id}, \text{pk})$  を更新する.
  - (4)  $\tilde{\text{dig}} := \tilde{\text{MemUpd}}^{[\tilde{\text{st}}]}(\tilde{\text{crs}}, \text{id}, \tilde{\text{pk}})$  を計算する.
  - (5) 各  $i \in [2^{\delta(\text{ctr})}]$  に対して以下を実行する.
    - (a)  $(\text{id}, \tilde{\text{pk}}) \leftarrow \text{Dict}_1[\text{ctr} + 1 - i]$  を取得する.
    - (b)  $\tilde{\text{wit}} := \tilde{\text{WGen}}^{[\tilde{\text{st}}]}(\tilde{\text{crs}}, \text{id})$  を生成する.
    - (c)  $(\text{ctr}, \tilde{\text{wit}})$  を  $\text{Dict}_2[\text{id}]$  に追加する.
  - (6)  $\text{pp} := (\text{ctr}, \tilde{\text{dig}}_1, \dots, \tilde{\text{dig}}_{\text{hw}(\text{ctr})-1}, \tilde{\text{dig}})$  を出力する.
- $\text{Enc}(\text{crs}, \text{pp}, \text{id}, \text{m}) \rightarrow \text{ct}:$ 
  - (1)  $\text{crs} = \tilde{\text{crs}}$ ,  $\text{pp} = (\text{ctr}, \tilde{\text{dig}}_1, \dots, \tilde{\text{dig}}_{\text{hw}(\text{ctr})})$  とする.
  - (2)  $\tilde{\text{r}}^i \leftarrow \$ \tilde{\mathcal{R}}^i$  と  $\tilde{\text{r}}_{\text{hw}(\text{ctr})}^d \leftarrow \$ \tilde{\mathcal{R}}^d$  をサンプルする.
  - (3) 以下を計算する.
 
$$\begin{aligned} \tilde{\text{ct}}^i &\leftarrow \tilde{\text{Enc}}^i(\tilde{\text{crs}}, \text{id}; \tilde{\text{r}}^i), \\ \tilde{\text{ct}}_i^d &\leftarrow \tilde{\text{Enc}}^d(\tilde{\text{crs}}, \tilde{\text{dig}}_i, \text{id}, \text{m}; \tilde{\text{r}}^i, \tilde{\text{r}}_i^d), \quad \forall i \in [\text{hw}(\text{ctr})]. \end{aligned}$$
  - (4)  $\text{ct} := (\text{ctr}, \tilde{\text{ct}}^1, \tilde{\text{ct}}_1^d, \dots, \tilde{\text{ct}}_{\text{hw}(\text{ctr})}^d)$  を出力する.
- $\text{Upd}^{[\text{aux}]}(\text{pp}, \text{id}) \rightarrow \text{hsk}:$ 
  - (1)  $\text{aux} = (\text{ctr}, \text{Dict}_1, \text{Dict}_2, \text{pp}, \tilde{\text{st}})$  とする.
  - (2)  $\text{hsk} := \{(\text{ctr}_i, \tilde{\text{wit}}_i)\}_{i \in [d]} \leftarrow \text{Dict}_2[\text{id}]$  を出力する.
- $\text{Dec}(\text{sk}, \text{hsk}, \text{ct}) \rightarrow \text{m}:$ 
  - (1)  $\text{sk} = \tilde{\text{sk}}$ ,  $\text{hsk} = \{(\text{ctr}_i, \tilde{\text{wit}}_i)\}_{i \in [d]}$ ,  $\text{ct} = (\text{ctr}_{\text{enc}}, \tilde{\text{ct}}^1, \tilde{\text{ct}}_1^d, \dots, \tilde{\text{ct}}_{\text{hw}(\text{ctr}_{\text{enc}})}^d)$  とする.

- (2)  $\text{ctr}_{\text{enc}} < \text{ctr}_1$  なら  $\perp$  を出力する.
- (3)  $(\text{ctr}_{\text{enc}} \oplus \text{ctr}_i) < 2^{\delta(\text{ctr}_i)}$  であるカウンター  $\text{ctr}_i$  を探す.
- (4) そのようなカウンターが存在しない場合、 $\text{GetUpd}$  を出力する.
- (5)  $\text{m} := \tilde{\text{Dec}}(\tilde{\text{sk}}, \tilde{\text{wit}}_i, (\tilde{\text{ct}}^i, \tilde{\text{ct}}_{\text{hw}(\text{ctr}_i)}^d))$  を出力する.

## 4.2 正当性、コンパクト性、効率性

$\Pi_{\text{RBE}}$  の正当性、コンパクト性、および効率性が、 $\Pi_{\text{dLE}}$  の正当性およびコンパクト性から導かることを示す.

**定理 4.1** (正当性).  $\Pi_{\text{dLE}}$  が完全性と正当性を満たすならば、 $\Pi_{\text{RBE}}$  は正当性を満たす.

証明.  $\Pi_{\text{RBE}}$  に対する正当性ゲームにおける攻撃者  $\mathcal{A}$  とチャレンジャー  $\text{Chal}$  を考える. 共通参照文字列を  $\text{crs} = \tilde{\text{crs}} \leftarrow \tilde{\text{Setup}}(1^\lambda, 1^\ell)$  とする.  $(\text{sk}^*, \text{pk}^*) = (\tilde{\text{sk}}^*, \tilde{\text{pk}}^*) \leftarrow \tilde{\text{KGen}}(\tilde{\text{crs}})$  を、 $\text{Chal}$  がチャレンジ  $\text{IDid}^*$  に対するターゲット ID 登録クエリへの応答としてサンプルした鍵ペアとし、 $(\text{id}^*, \text{pk}^*)$  が登録された際のカウンターを  $\text{ctr}^*$  とする. さらに、 $\text{aux} = (\text{ctr}, \text{Dict}_1, \text{Dict}_2, \text{pp}, \tilde{\text{st}})$  を、 $\mathcal{A}$  がターゲット ID 登録クエリをした後の任意の時点において  $\text{Chal}$  が維持する補助情報とする. 構成から、 $\text{pp} = (\text{ctr}, \tilde{\text{dig}}_1, \dots, \tilde{\text{dig}}_{\text{hw}(\text{ctr})})$  と表すことができ、 $\text{ctr} \geq \text{ctr}^*$  である.

正当性ゲームにおいて、任意の攻撃者  $\mathcal{A}$  に対して  $b = 1$  となる確率が無視できることを示す必要がある.  $\text{m}_t \in \mathcal{M}$  を  $\mathcal{A}$  によって行われた  $t$  番目の暗号化クエリとし、 $\text{ct}_t \leftarrow \text{Enc}(\text{crs}, \text{pp}, \text{id}^*, \text{m}_t)$  をその結果の暗号文とする. 任意のインデックス  $j \in [t]$  に対する復号クエリを考える. ここで、 $\text{Chal}$  は  $\text{m}'_j \leftarrow \text{Dec}(\text{sk}^*, \text{hsk}^*, \text{ct}_j)$  を計算する.

- 構成から、 $\text{ct}_j = (\text{ctr}_{\text{enc}}, \tilde{\text{ct}}_j^1, \tilde{\text{ct}}_{j,1}^d, \dots, \tilde{\text{ct}}_{j,\text{hw}(\text{ctr}_{\text{enc}})}^d)$  と  $\text{hsk}^* = \{(\text{ctr}_i, \tilde{\text{wit}}_i)\}_{i \in [d]}$  となる.
- $\text{ctr}_{\text{enc}} < \text{ctr}_1$  ならば、 $\text{Chal}$  は  $\perp$  を返す.
- $(\text{ctr}_{\text{enc}} \oplus \text{ctr}_i) < 2^{\delta(\text{ctr}_i)}$  となるカウンターを探す.
- そのようなカウンターが存在しない場合、 $\text{Chal}$  は  $\text{hsk}^* := \text{Upd}^{[\text{aux}]}(\text{crs}, \text{id}^*)$  を計算する. (後で示すように、更新された補助復号鍵  $\text{hsk}^*$  は上記の条件を満たすカウンター  $\text{ctr}_i$  を常に含む.)
- $\text{Chal}$  は  $\text{m}'_j \leftarrow \tilde{\text{Dec}}(\tilde{\text{sk}}^*, \tilde{\text{wit}}_i, (\tilde{\text{ct}}_j^i, \tilde{\text{ct}}_{j,\text{hw}(\text{ctr}_i)}^d))$  を返す. 次に、3つの失敗ケースを検証する.
- ケース 1:  $\text{ctr}_{\text{enc}} < \text{ctr}_1$ . 構成から、 $\text{ctr}_1 = \text{ctr}^*$  である. ここで、 $\text{ctr}^*$  は  $(\text{id}^*, \text{pk}^*)$  が登録された際のカウンターである. 正当性ゲームにおいて、 $\text{Chal}$  はターゲット  $\text{IDid}^*$  が登録されるまでメッセージを暗号化しない. したがって、 $\text{ctr}_{\text{enc}} \geq \text{ctr}^* = \text{ctr}_1$  は常に成立する. したがって、このケースは起こり得ない.
- ケース 2: 適切なカウンター  $\text{ctr}_i$  が存在しない. 補助復号鍵の更新メカニズムにより、任意のカウンター  $\text{ctr}_{\text{enc}}$  に対して、 $\text{Dict}_2$  には少なくとも一つの

$(ctr_i, \tilde{wit}_i)$  が存在し,  $ctr_i \leq ctr_{enc} < ctr_i + 2^{\delta(ctr_i)}$  を満たす. これにより,  $ctr_{enc}$  は  $\delta(ctr_i)$  番目のビット以降で  $ctr_i$  と同じ接頭辞を持つ. したがって, このケースは最新の  $hsk^*$  を取得することによって解決される.

- Case 3:  $\Pi_{dLE}$  の復号に失敗する. 最後に,

$$m_j \neq m'_j \leftarrow \tilde{Dec}(sk^*, \tilde{wit}_i, (\tilde{ct}_j^i, \tilde{ct}_{j, hw(ctr_i)}^d)).$$

となるケースを検討する.  $\Pi_{dLE}$  の正当性から, 証拠  $\tilde{wit}_i$  と, ダイジェスト依存暗号文  $\tilde{ct}_{j, hw(ctr_i)}^d$  に用いられたダイジェスト  $\tilde{dig}_{hw(ctr_i)}$  が同期しているならば, このケースは無視できる確率でしか発生しない. 言い換えると, 復号が成功するかどうかは, 以下の要素の整合性に依存している.

- 証拠  $\tilde{wit}_i$ :  $ctr_i$  番目の登録時に,  $\tilde{WGen}^{\tilde{s}t}(c\bar{r}s, id^*)$  の実行により生成される. ここで,  $\tilde{s}t$  は  $ctr_i$  時点での状態である.
  - ダイジェスト依存暗号文  $\tilde{ct}_{j, hw(ctr_i)}^d$  に用いられたダイジェスト  $\tilde{dig}_{hw(ctr_i)}$ : カウンターが  $ctr_i$  のときの状態  $\tilde{s}t$  を用いて暗号化時に計算される.
- したがって,  $\tilde{dig}_{hw(ctr_i)}$  が  $ctr_i$  番目の登録における  $\tilde{MemUpd}$  の実行によって生成されることを示せば十分である. 条件  $(ctr_{enc} \oplus ctr_i) < 2^{\delta(ctr_i)}$  は,  $ctr_{enc}$  と  $ctr_i$  が  $\delta(ctr_i)$  番目のビット以降で同じ接頭辞を持つことを保証する. 構成上, これにより  $\tilde{dig}_{hw(ctr_i)}$  が  $ctr_i$  番目の登録における  $\tilde{MemUpd}$  の実行によって生成されることが保証される.

したがって, 復号失敗確率は無視できる.  $\square$

**定理 4.2** (コンパクト性).  $\Pi_{dLE}$  がコンパクト性を満たすならば,  $\Pi_{RBE}$  もコンパクト性を満たす.

証明. 各要件を個別に証明する.

- **公開パラメータのコンパクト性:** 構成から, 公開パラメータ  $pp$  は, 現在の登録ユーザ数を示すカウンター  $ctr$  と  $\Pi_{dLE}$  の  $hw(ctr)$  個のダイジェスト  $\tilde{dig}_1, \dots, \tilde{dig}_{hw(ctr)}$  からなる.  $\Pi_{dLE}$  のコンパクト性と  $\ell = O(\lambda)$  であることから,  $|\tilde{dig}_i| = \text{poly}(\lambda, \ell) = \text{poly}(\lambda)$  である.  $hw(ctr) \leq \log(ctr)$  であるため,  $|pp| = \text{poly}(\lambda, \log(ctr))$  となる.
- **補助復号鍵のコンパクト性:**  $hsk = \{(ctr_i, \tilde{wit}_i)\}_{i \in [d]}$  をある  $id$  に対する補助復号鍵とする. 構成から, 任意の  $i \in [d-1]$  に対して,  $ctr_i < ctr_{i+1}$ かつ  $ctr_{i+1} - 2^{\delta(ctr_{i+1})} + 1 \leq ctr_i$  である. 後者は,  $ctr_{i+1} - 2^{\delta(ctr_{i+1})} + 1 \leq ctr_i$  が成立する際に  $(ctr_{i+1}, \tilde{wit}_{i+1})$  が  $\text{Dict}_2[id]$  に追加されるからである. 次に,

$$ctr_{i+1} - 2^{\delta(ctr_{i+1})} + 1 \leq ctr_i < ctr_{i+1} \quad (1)$$

が  $\delta(ctr_i) < \delta(ctr_{i+1})$  であることを意味することを示す. 関数  $\delta$  の定義から, ある奇数  $k_i$  に対して

$ctr_{i+1} = k_{i+1} \cdot 2^{\delta(ctr_{i+1})}$  と書ける. したがって, 不等式 (1) は以下のように表すことができる.

$$(k_{i+1} - 1) \cdot 2^{\delta(ctr_{i+1})} + 1 \leq ctr_i \leq k_{i+1} \cdot 2^{\delta(ctr_{i+1})} - 1.$$

これは,  $ctr_i \equiv 1 \pmod{2^{\delta(ctr_{i+1})}}$  であることを意味する. もし  $\delta(ctr_i) \geq \delta(ctr_{i+1})$  ならば,  $ctr_i$  は  $2^{\delta(ctr_i)}$  で割り切ることになり, 上記と矛盾する. したがって,  $\delta(ctr_i) < \delta(ctr_{i+1})$  である. 上記の議論から,

$$1 \leq \delta(ctr_1) < \delta(ctr_2) < \dots < \delta(ctr_d) \leq \log N$$

が成り立つ. したがって,  $d \leq \log N$  である. また,  $hsk$  は最大で  $\log N$  組のカウンターと証拠のペアからなる.  $\Pi_{dLE}$  のコンパクト性と  $\ell = O(\lambda)$  であることから,  $|\tilde{wit}_i| = \text{poly}(\lambda, \ell) = \text{poly}(\lambda)$  となる. したがって,  $|hsk| = \text{poly}(\lambda, \log N)$  となる.

$\square$

さらに,  $\Pi_{dLE}$  の(暗号文の)コンパクト性により, 上記の変換は  $|ct| \leq |\tilde{ct}^i| + O(\log N) \cdot \text{poly}(\lambda, \log \ell)$  を達成する. **定理 4.3** (効率性).  $\Pi_{dLE}$  がコンパクト性を満たすならば,  $\Pi_{RBE}$  は(弱)効率性を満たす.

証明. 各要件を個別に証明する.

- **更新の実行時間:** 構成から,  $Upd$  の操作は単に補助復号鍵を探すだけである. 定理 4.2 より,  $|hsk| = \text{poly}(\lambda, \log N)$  である. 補助情報には, 各 ID を対応する補助復号鍵にマッピングする辞書  $\text{Dict}_2$  が保持されているため, 更新操作は RAM 計算モデルにおいて  $\text{poly}(\lambda, \log N)$  の時間で実装可能である.
- **更新回数:**  $hsk = \{(ctr_i, \tilde{wit}_i)\}_{i \in [d]}$  をある  $id$  に対する補助復号鍵とする. 構成と定理 4.2 から,  $\delta(ctr_i) < \delta(ctr_{i+1})$  が起きるたびに  $\text{Dict}_2[id]$  に新しい要素が追加されるため, 登録された  $IDid$  に対して  $Upd$  アルゴリズムを呼び出す必要がある. したがって, 任意の ID に対して, その  $i$  番目の更新は,  $(i-1)$  番目の更新後に  $2^i$  個の新しい ID が登録された際に発生する. 登録は最大で  $N$  回であるため, 任意の ID に対する更新回数は最大  $\log N$  である.

$\square$

### 4.3 安全性

$\Pi_{RBE}$  の安全性が  $\Pi_{dLE}$  のマルチダイジェスト安全性から導かれる事を示す.

**定理 4.4** (安全性).  $\Pi_{dLE}$  が  $\log N$ -ダイジェスト安全性を満たすならば,  $\Pi_{RBE}$  は安全性を満たす.

証明.  $\mathcal{A}$  を  $\Pi_{RBE}$  の PPT 攻撃者,  $ctr_{enc}$  を  $\mathcal{A}$  がチャレンジエリをした際のカウンター,  $ct^*$  をチャレンジ暗号文とする. 構成から,  $ct^* = (ctr_{enc}, \tilde{ct}^i, \tilde{ct}_1^d, \dots, \tilde{ct}_{hw(ctr_{enc})}^d)$  と

なる。ここで、 $hw(ctr_{enc}) \leq \log N$  である。 $\mathcal{A}$  を用いて、 $\Pi_{dLE}$  の  $\log N$ -ダイジェスト安全性を破る PPT 攻撃者  $\mathcal{B}$  を構成する。

- **セットアップ:** ( $\Pi_{dLE}$  に対する) チャレンジャーから  $(\tilde{crs}, \tilde{dig}, \tilde{st})$  を与えられて、 $\mathcal{B}$  は次のような動作する。
  - (1)  $\mathcal{B}$  はカウンター  $ctr := 0$ , 辞書  $Dict_1, Dict_2 := \emptyset$ , 公開パラメータ  $pp := (0, \tilde{dig})$ , 補助情報  $aux = (ctr, Dict_1, Dict_2, pp, \tilde{st})$  を初期化する。
  - (2) さらに、 $\mathcal{B}$  は内部的に順序付きリスト  $S := \perp$  を維持し、 $Reg$  の実行中に生成されるダイジェストと状態を追跡する。
  - (3) 最後に、 $\mathcal{B}$  は  $crs := \tilde{crs}$  を  $\mathcal{A}$  に渡す。
- **クエリフェーズ:** このフェーズでは、 $\mathcal{B}$  は、 $\mathcal{A}$  が実行するクエリを次のようにシミュレートする。
  - **非ターゲット ID 登録クエリ:**  $\mathcal{A}$  が  $IDid$  と公開鍵  $pk = \tilde{pk}$  を指定した時、 $\tilde{IsValid}(crs, pk) = 0$  ならば  $\mathcal{B}$  は現在の公開パラメータ  $pp$  を出力する。そうでないならば、次のように動作する。
    - (1)  $ctr := ctr + 1$  と  $Dict_1[ctr] := (id, \tilde{pk})$  を更新する。
    - (2)  $(id, \tilde{pk})$  について悪意のある更新クエリを行い、 $(\tilde{dig}, \tilde{st})$  を得る。
    - (3)  $pp := (ctr, \tilde{dig}_1, \dots, \tilde{dig}_{hw(ctr)-1}, \tilde{dig})$  を返す。
    - (4) さらに、 $S[ctr] := (\tilde{dig}, \tilde{st})$  を更新する。
  - **ターゲット ID 登録クエリ:**  $\mathcal{A}$  が  $IDid$  を指定した時、 $\mathcal{B}$  は次のように動作する。
    - (1)  $id$  について正直な更新クエリを行い、 $(\tilde{dig}, \tilde{st}, \tilde{pk})$  を得る。
    - (2)  $ctr := ctr + 1$  と  $Dict_1[ctr] := (id, \tilde{pk})$  を更新する。
    - (3)  $pp := (ctr, \tilde{dig}_1, \dots, \tilde{dig}_{hw(ctr)-1}, \tilde{dig})$  と  $pk := \tilde{pk}$  をセットする。
    - (4)  $\mathcal{A}$  に  $(pp, aux, pk)$  を返す。
    - (5) さらに、 $S[ctr] := (\tilde{dig}, \tilde{st})$  を更新する。
- **チャレンジフェーズ:**  $\mathcal{A}$  がチャレンジ  $IDid^*$  とメッセージのペア  $m_0^*, m_1^* \in M$  を指定した後、 $\mathcal{B}$  は次のようにしてチャレンジ暗号文  $ct^* = (ctr_{enc}, \tilde{ct}^i, \tilde{ct}^d_1, \dots, \tilde{ct}^d_{hw(ctr_{enc})})$  を構成する。
  - (1)  $pp = (ctr_{enc}, \tilde{dig}_1, \dots, \tilde{dig}_{hw(ctr_{enc})})$  を現在の公開パラメータとする。
  - (2)  $t_1, \dots, t_{hw(ctr_{enc})} \in [ctr_{enc}]$  を、それぞれ  $\tilde{dig}_1, \dots, \tilde{dig}_{hw(ctr_{enc})}$  が生成された時刻とする。
  - (3)  $(t_1, \dots, t_{hw(ctr_{enc})}, id^*, m_0^*, m_1^*)$  についてチャレンジクエリを行い、 $(\tilde{ct}^i, \tilde{ct}^d_1, \dots, \tilde{ct}^d_{hw(ctr_{enc})})$  を得る。
  - (4)  $ct^* = (ctr_{enc}, \tilde{ct}^i, \tilde{ct}^d_1, \dots, \tilde{ct}^d_{hw(ctr_{enc})})$  を  $\mathcal{A}$  に返す。
- **出力フェーズ:** ゲーム終了時、 $\mathcal{A}$  は  $b' \in \{0, 1\}$  を出力し、 $\mathcal{B}$  も同様に  $b'$  を出力する。

上記のシミュレーションによると、非ターゲット ID 登録クエリでチャレンジ  $IDid^*$  が使用されない場合、 $\mathcal{B}$  も  $id^*$  に対する悪意のある更新クエリを実行しない。したがって、

$\mathcal{B}$  のチャレンジクエリによってゲームが終了することはない。また、 $\mathcal{B}$  は  $\mathcal{A}$  に対する RBE の安全性ゲームを完全にシミュレートしている。したがって、 $\mathcal{A}$  が  $\Pi_{RBE}$  の安全性ゲームに勝利する確率と同じ確率で、 $\mathcal{B}$  は  $\Pi_{dLE}$  の  $\log N$ -ダイジェスト安全性ゲームに勝利する。  $\square$

上記の変換と定理 3.6 を組み合わせることによって、表 1 を満たす LWE ベースの RBE 方式が得られる。

## 謝辞

本研究は JSPS 科研費 JP25051030 の助成を受けたものです。

## 参考文献

- [1] S. Chiku, K. Hara, K. Hashimoto, T. Tomita, and J. Shikata. How to apply fujisaki-okamoto transformation to registration-based encryption. *CANS 2024*, 2024.
- [2] K. Cong, K. Eldefrawy, and N. P. Smart. Optimizing registration based encryption. *IMACC 2021*, 2021.
- [3] N. Döttling, D. Kolonelos, R. W. F. Lai, C. Lin, G. Malavolta, and A. Rahimi. Efficient laconic cryptography from learning with errors. *EUROCRYPT 2023*, 2023.
- [4] J. Dujmovic, G. Malavolta, and W. Qi. Registration-based encryption in the plain model. *PKC 2025*, 2025.
- [5] D. Fiore, D. Kolonelos, and P. de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. *ASIACRYPT 2023*, 2023.
- [6] D. Francati, D. Frioli, M. Maitra, G. Malavolta, A. Rahimi, and D. Venturi. Registered (inner-product) functional encryption. *ASIACRYPT 2023*, 2023.
- [7] S. Garg, M. Hajiabadi, M. Mahmoody, and A. Rahimi. Registration-based encryption: Removing private-key generator from IBE. *TCC 2018*, 2018.
- [8] S. Garg, M. Hajiabadi, M. Mahmoody, A. Rahimi, and S. Sekar. Registration-based encryption from standard assumptions. *PKC 2019*, 2019.
- [9] N. Glaeser, D. Kolonelos, G. Malavolta, and A. Rahimi. Efficient registration-based encryption. *ACM CCS 2023*, 2023.
- [10] R. Goyal and S. Vusirikala. Verifiable registration-based encryption. *CRYPTO 2020*, 2020.
- [11] M. Hajiabadi, M. Mahmoody, W. Qi, and S. Sarfaraz. Lower bounds on assumptions behind registration-based encryption. *TCC 2023*, 2023.
- [12] S. Hohenberger, G. Lu, B. Waters, and D. J. Wu. Registered attribute-based encryption. *EUROCRYPT 2023*, 2023.
- [13] S. Katsumata, K. Kwiatkowski, F. Pintore, and T. Prest. Scalable ciphertext compression techniques for post-quantum KEMs and their applications. *ASIACRYPT 2020*, 2020.
- [14] M. Mahmoody and W. Qi. Online mergers and applications to registration-based encryption and accumulators. *ITC 2023*, 2023.
- [15] M. Mahmoody, W. Qi, and A. Rahimi. Lower bounds for the number of decryption updates in registration-based encryption. *TCC 2022*, 2022.
- [16] P. Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015.