より少ない漏洩の下で安全な動的検索可能暗号への変換手法

渡邉 洋平[1,2,a)]　大原 一真[3]　岩本 貢[1]　太田 和夫[1,3]

概要：動的検索可能暗号 (Dynamic Searchable Symmetric Encryption: Dynamic SSE) は，各操作において多少の情報漏洩を許す代わりに，それ以外の情報を漏らすことなく効率的な検索を実現する暗号化方式である．特に，多くの構成法において，更新時に “その文書ファイルに含まれる異なるキーワード数の漏洩” が許されている．この漏洩を許しても重要な安全性要件であるフォワード安全性は達成可能であり，実際この漏洩を用いた具体的な攻撃は未だに知られていないものの，今後の漏洩悪用攻撃の進展によっては致命的な漏洩となり得る．本稿では，任意のフォワード安全な Dynamic SSE 方式を，この漏洩を許さない強フォワード安全な方式へと拡張する手法を 2 つ提案する．まず，十分な数のダミーエントリを一緒に登録する素朴な手法を示し，次により少ないダミーエントリの数で強フォワード安全性を達成する方法を示す．具体的には，検索キーワードが特定の確率分布で選ばれると仮定し，かつ多少の検索エラーを許すことで，後者は素朴な手法に比べて大幅に少ないダミーエントリ数で上記漏洩を防ぐことができる．

Generic Transformations for Making Dynamic SSE Secure with Smaller Leakage

Yohei Watanabe[1,2,a)]　Kazuma Ohara[3]　Mitsugu Iwamoto[1]　Kazuo Ohta[1,3]

Abstract: Dynamic Searchable Symmetric Encryption (Dynamic SSE) enables a client to efficiently and securely perform database operations such as update and search. To provide efficient operations, dynamic SSE leaks some “inconsequential” information. In particular, many existing dynamic SSE schemes allow leakage of the number of distinct keywords contained in each newly-inserted file when the file insertion is performed. This leakage does not affect forward privacy, and there is still no concrete attacks using it. However, the future progress of leakage-abuse attacks may turn the leakage into a fatal one. In this paper, we show two transformations that lift forward privacy of any dynamic SSE scheme to strong forward privacy, which does not allow the leakage. The first one is a naïve construction that requires so many dummy entries, and the second one achieves strong forward privacy by allowing small search error and assuming a specific probability distribution of search keywords.

1. はじめに

データベースを安全かつ効率的に運用するための暗号技術として，**検索可能暗号** (Searchable Symmetric Encryption: SSE) [6], [15] が知られている．特に，データベースの基本操作である更新と検索の両方をサポートする**動的検索可能暗号** (Dynamic SSE) について，幅広く研究が進められてきた (例えば，[3], [5], [7], [11], [12], [16])．Dynamic SSE は多少の情報の漏洩を許すことで効率的な更新・検索処理を実現するが，その漏えいが重要な情報，例えば検索キーワードやデータベースに保存された暗号化ファイルの情報を漏らすようでは意味がない．そこで，攻撃の観点から Dynamic SSE で許す漏えい情報が本当に “取るに足らない” ものかどうかを調べる研究が盛んに行われており，こういった攻撃を**漏洩悪用攻撃** (Leakage-Abuse Attacks) [2], [4], [9], [13], [14], [20] と呼ぶ．これらの攻撃は非現実的な仮定が必要となるものが多いが，以下の 2 つの攻撃は比較的現実的な条件で攻撃が可能であることが知られている．

[1] 電気通信大学, The University of Electro-Communications
[2] 国立研究開発法人情報通信研究機構, NICT
[3] 国立研究開発法人 産業技術総合研究所, AIST
[a)] watanabe@uec.ac.jp

ファイル挿入攻撃 [20]. 攻撃者 (サーバ) がクライアントに対してデータベースに追加するファイルを指定できる状況を想定して行う攻撃．例えば届いたメールを自動的に (Dynamic SSE を通じて) サーバに登録するようなメールシステムがそのような状況の一例として考えられる．**フォワード安全性** [3], [16] は “これまでの検索履歴によらず，新たに追加されるファイルが含んでいるキーワードの情報が漏れない” ことを保証するものであり，ファイル挿入攻撃の脅威を軽減できることが知られている．また，渡邉ら [18] はより強い (より漏洩が少ない) 安全性要件である強フォワード安全性を導入している．

BKM 攻撃 [2]. Blackstone ら [2] はこれまでの漏洩悪用攻撃を改良し，これまでに比べ十分弱い仮定で攻撃できることを示した．詳細は割愛するが，BKM 攻撃が示唆するのは，漏洩悪用攻撃の研究が進むに従って，これまでに問題ないと考えられていた漏洩情報が攻撃に利用できるようになる可能性が十分あるということである．

1.1 本稿の貢献

以上より，フォワード安全性を十分満たす漏洩情報であっても，今後攻撃に利用できる可能性は否定できない．そこで本稿では，任意の Dynamic SSE 方式をより少ない漏洩情報で実現できるように変換する手法を提案する．具体的には，任意のフォワード安全な Dynamic SSE 方式を可能な限り効率を落とさず強フォワード安全な方式へと拡張する方法を 2 つ提案する．

フォワード安全性を強フォワード安全性に引き上げるということは，ファイル挿入時に漏洩を許していた “当該ファイルが含む異なるキーワードの数” が漏れないようにするということである．従って，提案する 2 つの手法の基本的な構成アイディアは次の通りである：ファイル挿入時は，一般的にファイルが含む異なるキーワードごとに情報をデータベースに追加することから，同時に十分多くのダミー情報も一緒に登録することで各ファイルが含む異なるキーワード数を隠す．2 つの提案手法の違いは追加するダミーの数である．1 つ目の方式は，当該ファイルとサイズが同じファイルが含み得る最大キーワード数を計算し，その数までダミーを追加する素朴な方式である．2 つ目の方式はそれよりも少ないダミー数で強フォワード安全性を達成することを目指したものであり，少しの検索エラーを許し，また検索キーワードの確率分布をあらかじめ固定することで，大幅なダミー数の削減を達成する．

1.2 記法

任意の自然数 $a, b \in \mathbb{N}$ $(a \leq b)$ に対して，$\{a, \ldots, b\}$ を $[a, b]$ と書く．$a = 1$ の時は単に $[b]$ と書く．$\|$ は要素同士の連結を表す．論文を通し，κ をセキュリティパラメータとする．任意の非対話アルゴリズム A に対し，$\mathsf{out} \leftarrow \mathsf{A}(\mathsf{in})$ は A が in を入力に取り out を出力することを表す．本稿では，クライアントとサーバ間の対話アルゴリズムを考え，$\langle \mathsf{out}_{\mathrm{C}}, \mathsf{out}_{\mathrm{S}} \rangle \leftarrow \langle \mathsf{A}_{\mathrm{C}}(\mathsf{in}_{\mathrm{C}}), \mathsf{A}_{\mathrm{S}}(\mathsf{in}_{\mathrm{S}}) \rangle$ は次を意味する：クライアントが in_{C} を入力して A_{C} を，サーバが in_{S} を入力して A_{S} をそれぞれ実行し，それぞれ $\mathsf{out}_{\mathrm{C}}$ と $\mathsf{out}_{\mathrm{S}}$ を出力として得る．簡単のために，上記を $(\mathsf{out}_{\mathrm{C}}; \mathsf{out}_{\mathrm{S}}) \leftarrow \mathsf{A}(\mathsf{in}_{\mathrm{C}}; \mathsf{in}_{\mathrm{S}})$ と書く．必要に応じてトランスクリプト trans を $\langle (\mathsf{out}_{\mathrm{C}}; \mathsf{out}_{\mathrm{S}}), \mathsf{trans} \rangle \leftarrow \mathsf{A}(\mathsf{in}_{\mathrm{C}}; \mathsf{in}_{\mathrm{S}})$ のように書く．関数 $\mathsf{negl}(\cdot)$ が任意の多項式 $\mathsf{poly}(\cdot)$ に対してある定数 $\kappa_0 \in \mathbb{N}$ が存在して，全ての $\kappa \geq \kappa_0$ に対して $\mathsf{negl}(\kappa) < 1/\mathsf{poly}(\kappa)$ が成り立つ時，$\mathsf{negl}(\cdot)$ は無視可能関数であるという．

2. 動的検索可能暗号

2.1 SSE 用の記法

D 個の要素からなるキーワードの集合 (辞書とも呼ぶ) を $\Lambda := \{0,1\}^{\log D}$ とする．既存研究と同様に，各文書ファイル f_{id} はそれぞれ内容とは無関係な識別子 (文書番号) $\mathsf{id} \in \{0,1\}^{\ell}$ を有し (ℓ は高々 κ の多項式)，各文書ファイル f_{id} は識別子 id とファイル中の異なるキーワードの集合 $\mathcal{W}_{\mathsf{id}} \subset \Lambda$ からなるものとする．本稿では，時系列の整理にカウンタ t を用いる．t は 0 に初期化され，更新・検索の度にインクリメントされる．t におけるデータベース $\mathsf{DB}^{(t)} := \{(\mathsf{id}_i, \mathcal{W}_i)\}_{i=1}^{n^{(t)}}$ は文書ファイルの集合で表され，$n^{(t)}$ は t の時点でサーバに保存されている文書ファイルの数を表す．$\mathsf{ID}^{(t)} := \{\mathsf{id} \mid (\mathsf{id}, \mathcal{W}_{\mathsf{id}}) \in \mathsf{DB}^{(t)}\}$ を $\mathsf{DB}^{(t)}$ 内の識別子の集合とする．

2.2 モデル

Dynamic SSE のモデル [3], [7], [8] を示す．まずクライアントは Setup アルゴリズムを実行し，秘密鍵 k，初期ステート情報 $\sigma^{(0)}$，初期暗号化データベース $\mathsf{EDB}^{(0)}$ を得て，$\mathsf{EDB}^{(0)}$ をサーバに送る．t の時点でデータベースの更新を行いたい場合，クライアントとサーバは対話アルゴリズム $\mathsf{Update} = (\mathsf{Update}_{\mathrm{C}}, \mathsf{Update}_{\mathrm{S}})$ を実行する．具体的には，クライアントは $\mathsf{Update}_{\mathrm{C}}$ に k，$\sigma^{(t)}$，ラベルとそれに対応する入力 $(\mathsf{op}, \mathsf{in})$ を入力し，ステート情報を更新，$\sigma^{(t+1)}$ を得る．サーバは $\mathsf{Update}_{\mathrm{S}}$ に $\mathsf{EDB}^{(t)}$ を入力し，更新された暗号化データベース $\mathsf{EDB}^{(t+1)}$ を得る．この時，$(\mathsf{op}, \mathsf{in})$ は行いたい更新操作によって以下のように変わる：

$$(\mathsf{op}, \mathsf{in}) = \begin{cases} (\mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) & \text{挿入の場合,} \\ (\mathsf{del}, \mathsf{id}) & \text{削除の場合,} \\ (\mathsf{add}, (\mathsf{id}, w)) & \text{変更 (追加) の場合,} \\ (\mathsf{del}, (\mathsf{id}, w)) & \text{変更 (削除) の場合.} \end{cases}$$

挿入及び削除はファイル中の各キーワードに対して変更を行うことで実現可能であるため，変更操作のみを扱うこと

Search Correctness: $\mathsf{Exp}^{\Sigma}_{\mathcal{A}}(\kappa, Q)$

1: $(k, \sigma^{(0)}, \mathsf{EDB}^{(0)}) \leftarrow \mathsf{Setup}(1^{\kappa})$
2: $\mathsf{st}_{\mathsf{A}} := \{(k, \sigma^{(0)}, \mathsf{EDB}^{(0)})\}$
3: **for** $t = 1$ **to** Q **do**
4: $\quad \mathsf{query} \leftarrow \mathcal{A}_t(\mathsf{st}_{\mathsf{A}})$
5: $\quad$ **if** $\mathsf{query} = (\mathtt{upd}, \mathsf{op}, \mathsf{in})$ **then**
6: $\quad\quad \langle(\sigma^{(t)}; \mathsf{EDB}^{(t)}), \mathsf{trans}^{(t)}\rangle$
$\quad\quad\quad \leftarrow \mathsf{Update}(k, \mathsf{op}, \mathsf{in}, \sigma^{(t-1)}; \mathsf{EDB}^{(t-1)})$
7: $\quad$ **if** $\mathsf{query} = (\mathtt{srch}, q)$ **then**
8: $\quad\quad \langle(\sigma^{(t)}, \mathcal{X}_q^{(t-1)}; \mathsf{EDB}^{(t)}), \mathsf{trans}^{(t)}\rangle$
$\quad\quad\quad\quad \leftarrow \mathsf{Search}(k, q, \sigma^{(t-1)}; \mathsf{EDB}^{(t-1)})$
9: $\quad \mathsf{st}_{\mathsf{A}} \leftarrow (\sigma^{(t)}, \mathsf{trans}^{(t)})$
10: $q^{\star} \leftarrow \mathcal{A}_{Q+1}(\mathsf{st}_{\mathsf{A}})$
11: $\langle(\sigma^{(Q+1)}, \mathcal{X}_q^{(Q)}; \mathsf{EDB}^{(Q+1)}), \mathsf{trans}^{(Q+1)}\rangle$
$\quad\quad\quad \leftarrow \mathsf{Search}(k, q^{\star}, \sigma^{(Q)}; \mathsf{EDB}^{(Q)})$
12: **if** $\mathcal{X}_q^{(Q)} = \mathsf{ID}_q^{(Q)}$ **then**
13: $\quad$ **return** 1
14: **else**
15: $\quad$ **return** 0

図 1　試行 $\mathsf{Exp}^{\Sigma}_{\mathcal{A}}$.

も多いが，本稿ではデータベースに対する更新処理 (挿入・削除・変更) に則り，上記 4 種類の処理全てを考える．t の時点でキーワード $q \in \Lambda$ を検索する場合，クライアントとサーバは対話アルゴリズム $\mathsf{Search} = (\mathsf{Search}_{\mathrm{C}}, \mathsf{Search}_{\mathrm{S}})$ を実行する．クライアントは $\mathsf{Search}_{\mathrm{C}}$ に k，$\sigma^{(t)}$，q を入力し，更新されたステート情報 $\sigma^{(t+1)}$ 及び検索結果 $\mathcal{X}_q^{(t)}$ を得る．サーバは $\mathsf{Search}_{\mathrm{S}}$ に $\mathsf{EDB}^{(t)}$ を入力し，暗号化データベースを更新，$\mathsf{EDB}^{(t+1)}$ を得る．

定義 1 (Dynamic SSE)**.** 辞書 Λ に対する *Dynamic SSE* Σ は次の 3 つのアルゴリズム $\Sigma := (\mathsf{Setup}, \mathsf{Update}, \mathsf{Search})$ からなる．

- $(k, \sigma^{(0)}, \mathsf{EDB}^{(0)}) \leftarrow \mathsf{Setup}(1^{\kappa})$*:* κ を入力に取り，秘密鍵 k，初期ステート情報 $\sigma^{(0)}$ 初期暗号化データベース $\mathsf{EDB}^{(0)}$ を出力する非対話確率的アルゴリズム．
- $(\sigma^{(t+1)}; \mathsf{EDB}^{(t+1)}) \leftarrow \mathsf{Update}(k, \mathsf{op}, \mathsf{in}, \sigma^{(t)}; \mathsf{EDB}^{(t)})$*:* $\mathsf{Update}_{\mathrm{C}}$ 及び $\mathsf{Update}_{\mathrm{S}}$ からなる対話アルゴリズム．$\mathsf{Update}_{\mathrm{C}}$ は k，ラベルと対応する入力の組 $(\mathsf{op}, \mathsf{in}) \in \{(\mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})), (\mathsf{add}, (\mathsf{id}, w)), (\mathsf{del}, \mathsf{id}), (\mathsf{del}, (\mathsf{id}, w))\}$，$\sigma^{(t)}$ を入力に取り，更新されたステート情報 $\sigma^{(t+1)}$ を出力する．$\mathsf{Update}_{\mathrm{S}}$ は $\mathsf{EDB}^{(t)}$ を入力に取り，更新された暗号化データベース $\mathsf{EDB}^{(t+1)}$ を出力する．
- $(\sigma^{(t+1)}, \mathcal{X}_q^{(t)}; \mathsf{EDB}^{(t+1)}) \leftarrow \mathsf{Search}(k, q, \sigma^{(t)}; \mathsf{EDB}^{(t)})$*:* $\mathsf{Search}_{\mathrm{C}}$ 及び $\mathsf{Search}_{\mathrm{S}}$ からなる対話アルゴリズム．$\mathsf{Search}_{\mathrm{C}}$ は k，検索キーワード $q \in \Lambda$，$\sigma^{(t)}$ を入力に取り，更新されたステート情報 $\sigma^{(t+1)}$ 及び検索結果 $\mathcal{X}_q^{(t)}$ を出力する．$\mathsf{Search}_{\mathrm{S}}$ は $\mathsf{EDB}^{(t)}$ を入力に取り，更新された暗号化データベース $\mathsf{EDB}^{(t+1)}$ を出力する．

検索正当性． 上記 Σ は検索正当性を満たすことを要求する．すなわち，クライアントは任意のキーワードについて正しい検索結果を得ることができる．形式的な定義は以下の通り．

定義 2 (検索正当性)**.** Σ を *Dynamic SSE* とし，任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して，図 *1* の試行を考える．$\Pr[\mathsf{Exp}^{\Sigma}_{\mathcal{A}}(\kappa, Q)] \geq 1 - \mathsf{negl}(\kappa)$ を満たす無視可能関数 $\mathsf{negl}(\kappa)$ が存在する時，Σ は検索正当性を満たすという．

2.3 安全性

$\mathcal{L}$-適応的安全性． 多くの既存研究同様，シミュレーションベースで安全性を定義する．Dynamic SSE における情報漏洩は**漏洩関数** $\mathcal{L} := (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Upd}}, \mathcal{L}_{\mathsf{Srch}})$ として特徴づけられる．直感的には，$\mathcal{L}_{\mathsf{Setup}}$，$\mathcal{L}_{\mathsf{Upd}}$，$\mathcal{L}_{\mathsf{Srch}}$ は，それぞれセットアップ時，更新時，検索時に漏洩する情報を表す．$\mathcal{L}$-適応的安全性とは，そのような漏洩関数 $\mathcal{L}$ で示される情報漏洩を許したうえでそれ以上の情報は一切漏らさないという，Dynamic SSE における標準的な安全性である．具体的には，確率的多項式時間アルゴリズム $\mathsf{D} = (\mathsf{D}_1 \ldots, \mathsf{D}_{Q+1})$ とクライアント間の試行 $\mathsf{Real}^{\Sigma}_{\mathsf{D}}$ と D とシミュレータ $\mathsf{S} = (\mathsf{S}_0, \ldots, \mathsf{S}_Q)$ 間の試行 $\mathsf{Ideal}^{\Sigma}_{\mathsf{D},\mathsf{S},\mathcal{L}}$ の 2 つで定義される (各試行は図 2 参照)．

定義 3 ($\mathcal{L}$-適応的安全性)**.** Σ を *Dynamic SSE* 方式とする．任意の確率的多項式時間アルゴリズム D に対し，$\left|\Pr\left[\mathsf{Real}^{\Sigma}_{\mathsf{D}}(\kappa, Q) = 1\right] - \Pr\left[\mathsf{Ideal}^{\Sigma}_{\mathsf{D},\mathsf{S},\mathcal{L}}(\kappa, Q) = 1\right]\right| \leq \mathsf{negl}(\kappa)$ を満たす確率的多項式時間アルゴリズム S が存在するならば，Σ は $\mathcal{L}$-適応的安全であるという．

(強) フォワード安全性． これまでの検索可能暗号研究および漏洩悪用攻撃 [2], [9] を通じ，“どういった情報漏洩を許すか” のある程度の基準が形成されている．特に，更新時の漏洩関数 $\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, \mathsf{in})$ によって以下の**フォワード安全性** [3] が定義される．一言で言えば，フォワード安全性とは，“新たに追加されたファイルが過去の検索キーワードを含むかどうかの情報を漏らさない” ことを保証する．更に，渡邉ら [18] はより漏洩を抑えた強フォワード安全性を導入している．フォワード安全性は “ファイル f_{id} が含む異なるキーワード数に関する情報を漏洩することを許すが (下記 (2) 式参照)，強フォワード安全性はその漏洩を許さない (下記 (3) 式参照)．

定義 4 ((強) フォワード安全性 [3], [18])**.** Σ を $\mathcal{L}$-適応的安全な *Dynamic SSE* 方式とする *($\mathcal{L} = (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Upd}}, \mathcal{L}_{\mathsf{Srch}})$)*. $\mathcal{L}_{\mathsf{Upd}}$ が以下を満たす時，Σ はフォワード安全性を満たすという：

$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, w)) = \mathcal{L}'(\mathsf{add}, \mathsf{id}), \quad (1)$$

$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = \mathcal{L}''(\mathsf{add}, (\mathsf{id}, |\mathcal{W}_{\mathsf{id}}|, |f_{\mathsf{id}}|)). \quad (2)$$

Real Experiment: $\mathsf{Real}^{\Sigma}_{\mathsf{D}}(\kappa, Q)$

$(k, \sigma^{(0)}, \mathsf{EDB}^{(0)}) \leftarrow \mathsf{Setup}(1^{\kappa})$
$\mathsf{st}_{\mathsf{D}} := \{\mathsf{EDB}^{(0)}\}$
for $t = 1$ **to** Q **do**
  $\mathsf{query} \leftarrow \mathsf{D}_t(\mathsf{st}_{\mathsf{D}})$
  if $\mathsf{query} = (\mathtt{upd}, \mathsf{op}, \mathsf{in})$ **then**
    $\langle(\sigma^{(t)}; \mathsf{EDB}^{(t)}), \mathsf{trans}^{(t)}\rangle$
      $\leftarrow \mathsf{Update}(k, \mathsf{op}, \mathsf{in}, \sigma^{(t-1)}; \mathsf{EDB}^{(t-1)})$
  if $\mathsf{query} = (\mathtt{srch}, q)$ **then**
    $\langle(\sigma^{(t)}, \mathcal{X}_q^{(t-1)}; \mathsf{EDB}^{(t)}), \mathsf{trans}^{(t)}\rangle$
      $\leftarrow \mathsf{Search}(k, q, \sigma^{(t-1)}; \mathsf{EDB}^{(t-1)})$
  $\mathsf{st}_{\mathsf{D}} \leftarrow (\mathsf{EDB}^{(t)}, \mathsf{trans}^{(t)})$
$b \leftarrow \mathsf{D}_{Q+1}(\mathsf{st}_{\mathsf{D}})$
return b

Ideal Experiment: $\mathsf{Ideal}^{\Sigma}_{\mathsf{D},\mathsf{S},\mathcal{L}}(\kappa, Q)$

$(\mathsf{EDB}^{(0)}, \mathsf{st}_{\mathsf{S}}) \leftarrow \mathsf{S}_0(\mathcal{L}_{\mathsf{Setup}}(\kappa))$
$\mathsf{st}_{\mathsf{D}} := \{\mathsf{EDB}^{(0)}\}$
for $t = 1$ **to** Q **do**
  $\mathsf{query} \leftarrow \mathsf{D}_t(\mathsf{st}_{\mathsf{D}})$
  if $\mathsf{query} = (\mathtt{upd}, \mathsf{op}, \mathsf{in})$ **then**
    $\langle(\mathsf{st}'_{\mathsf{S}}; \mathsf{EDB}^{(t)}), \mathsf{trans}^{(t)}\rangle$
      $\leftarrow \mathsf{S}_t(\mathsf{st}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Upd}}(\sigma^{(t-1)}, \mathsf{EDB}^{(t-1)}, \mathsf{op}, \mathsf{in}); \mathsf{EDB}^{(t-1)})$
  if $\mathsf{query} = (\mathtt{srch}, q)$ **then**
    $\langle(\mathsf{st}'_{\mathsf{S}}; \mathsf{EDB}^{(t)}), \mathsf{trans}^{(t)}\rangle$
      $\leftarrow \mathsf{S}_t(\mathsf{st}_{\mathsf{S}}, \mathcal{L}_{\mathsf{Srch}}(\sigma^{(t-1)}, \mathsf{EDB}^{(t-1)}, q); \mathsf{EDB}^{(t-1)})$
  $\mathsf{st}_{\mathsf{D}} \leftarrow (\mathsf{EDB}^{(t)}, \mathsf{trans}^{(t)})$
  $\mathsf{st}_{\mathsf{S}} := \mathsf{st}'_{\mathsf{S}}$
$b \leftarrow \mathsf{D}_{Q+1}(\mathsf{st}_{\mathsf{D}})$
return b

図 2 試行 $\mathsf{Real}^{\Sigma}_{\mathsf{D}}$ 及び試行 $\mathsf{Ideal}^{\Sigma}_{\mathsf{D},\mathsf{S},\mathcal{L}}$.

ただし，$\mathcal{L}', \mathcal{L}''$ はステートレスな関数である．また，更に $\mathcal{L}_{\mathsf{Upd}}$ が (2) 式に代わって以下を満たす時，Σ は強フォワード安全性を満たすという：

$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = \mathcal{L}'''(\mathsf{add}, (\mathsf{id}, |f_{\mathsf{id}}|)). \quad (3)$$

ただし，$\mathcal{L}'''$ はステートレスな関数である．
(1) 式を満たせば自明に (2) 式を実現することが可能である一方で，(3) 式を満たせるとは限らない．また，以降では次に示すより具体的な漏洩関数を考える：

$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, w)) = \mathsf{id}, \quad (4)$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = (\mathsf{id}, |\mathcal{W}_{\mathsf{id}}|, |f_{\mathsf{id}}|), \quad (5)$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = (\mathsf{id}, |f_{\mathsf{id}}|). \quad (6)$$

実際，既存のフォワード安全な方式 (例えば [3], [7], [12], [17], [19]) は (4) 式および (5) 式の漏洩の下でフォワード安全，既存の強フォワード安全な方式 [18], [19] は (6) 式の漏洩の下で強フォワード安全である．

3. 素朴な手法

3.1 アイディア

変換のアイディアは “ダミーも一緒に登録することで各ファイル f_{id} に含まれる異なるキーワードの数を隠す” というシンプルなものである．本手法では，$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = (\mathsf{id}, |f_{\mathsf{id}}|)$ のみを用いて挿入処理をシミュレートするために必要なダミーの数を Curtmola ら [6] が導入した概念である max で定める．ここで，ファイル f_{id} の max ($\mathsf{max}_{\mathsf{id}}$ と書く) は f_{id} と同じサイズのファイルが含み得る最大のキーワード数である．本稿では，$\mathsf{max}_{\mathsf{id}}$ を計算するアルゴリズムを $\mathsf{CompMax}(\Lambda, |f_{\mathsf{id}}|)$ と書く．

3.2 構成法

上記のアイディアを基に，任意のフォワード安全な Dynamic SSE 方式 $\widehat{\Sigma} = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Update}}, \widehat{\mathsf{Search}})$ を強フォワード安全性な方式 $\Sigma = (\mathsf{Setup}, \mathsf{Update}, \mathsf{Search})$ に変換する素朴な手法を示す．

$\mathsf{Setup}(1^{\kappa})$: 以下を実行する．

1. $\widehat{\mathsf{Setup}}(1^{\kappa})$ を実行し，$(k, \hat{\sigma}^{(0)}, \widehat{\mathsf{EDB}}^{(0)})$ を得る．
2. $(k, \sigma^{(0)}, \mathsf{EDB}^{(0)}) \coloneqq (k, \hat{\sigma}^{(0)}, \widehat{\mathsf{EDB}}^{(0)})$ を出力する．

$\mathsf{Update}(k, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}}), \sigma^{(t)}; \mathsf{EDB}^{(t)})$: $(\sigma^{(t)}, \mathsf{EDB}^{(t)})$ を $(\hat{\sigma}^{(t')}, \widehat{\mathsf{EDB}}^{(t')})$ とし，カウンタ c を 0 に初期化する．以下を実行する．

1. $\mathsf{CompMax}(\Lambda, |f_{\mathsf{id}}|)$ を実行し，$\mathsf{max}_{\mathsf{id}}$ を得る．
2. 各キーワード $w \in \mathcal{W}_{\mathsf{id}}$ に対し，以下を実行する．
   2-a. $\widehat{\mathsf{Update}}(k, \mathsf{add}, (\mathsf{id}, 0\|w), \hat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\hat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．
   2-b. c をインクリメントする．
3. 各 $\beta \in [\mathsf{max}_{\mathsf{id}} - |\mathcal{W}_{\mathsf{id}}|]$ に対し，以下を実行する．
   3-a. $\widehat{\mathsf{Update}}(k, \mathsf{add}, (\mathsf{id}, 1\|\beta), \hat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\hat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．
   3-b. c をインクリメントする．
4. $(\sigma^{(t+1)}; \mathsf{EDB}^{(t+1)}) := (\hat{\sigma}^{(t'+\mathsf{max}_{\mathsf{id}})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{max}_{\mathsf{id}})})$ を出力する．

$\mathsf{Update}(k, \mathsf{del}, \mathsf{id}, \sigma^{(t)}; \mathsf{EDB}^{(t)})$: まず $(\sigma^{(t)}, \mathsf{EDB}^{(t)})$ を $(\hat{\sigma}^{(t')}, \widehat{\mathsf{EDB}}^{(t')})$ とし，カウンタ c を 0 に初期化する．クライアントはサーバに (暗号化された状態で) 保存していた f_{id} を取り出し，以下を実行する．

1. $\mathsf{CompMax}(\Lambda, |f_{\mathsf{id}}|)$ を実行し，$\mathsf{max}_{\mathsf{id}}$ を得る．

2. 各キーワード $w \in \mathcal{W}_{\mathsf{id}}$ に対し，以下を実行する．

   2-a. $\widehat{\mathsf{Update}}(k, \mathsf{del}, (\mathsf{id}, 0\|w), \widehat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\widehat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．

   2-b. c をインクリメントする．

3. 各 $\beta \in [\mathsf{max}_{\mathsf{id}} - |\mathcal{W}_{\mathsf{id}}|]$ に対し，以下を実行する．

   3-a. $\widehat{\mathsf{Update}}(k, \mathsf{del}, (\mathsf{id}, 1\|\beta), \widehat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\widehat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．

   3-b. c をインクリメントする．

4. $(\sigma^{(t+1)}; \mathsf{EDB}^{(t+1)}) := (\widehat{\sigma}^{(t'+\mathsf{max}_{\mathsf{id}})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{max}_{\mathsf{id}})})$ を出力する．

$\mathsf{Update}(k, \mathsf{op}, (\mathsf{id}, w), \sigma^{(t)}; \mathsf{EDB}^{(t)})$ **for** $\mathsf{op} \in \{\mathsf{add}, \mathsf{del}\}$**:** $\widehat{\Sigma}$ の対応する Update アルゴリズムを実行する．

$\mathsf{Search}(k, q, \sigma^{(t)}; \mathsf{EDB}^{(t)})$**:** $(\sigma^{(t)}, \mathsf{EDB}^{(t)})$ を $(\widehat{\sigma}^{(t')}, \widehat{\mathsf{EDB}}^{(t')})$ とし，以下を実行する．

1. $(\sigma^{(t)}, \mathsf{EDB}^{(t)})$ を $(\widehat{\sigma}^{(t')}, \widehat{\mathsf{EDB}}^{(t')})$ とする．
2. $\widehat{\mathsf{Search}}(k, 0\|q, \widehat{\sigma}^{(t')}; \widehat{\mathsf{EDB}}^{(t')})$ を実行し，得られた $(\widehat{\sigma}^{(t'+1)}, \mathcal{X}_q^{(t)}; \widehat{\mathsf{EDB}}^{(t'+1)})$ を $(\sigma^{(t+1)}, \mathsf{EDB}^{(t+1)})$ とする．
3. $(\sigma^{(t+1)}, \mathcal{X}_q^{(t)}; \mathsf{EDB}^{(t+1)})$ を出力する．

検索正当性． Σ の検索正当性は $\widehat{\Sigma}$ のものに従う．

安全性． 上記構成法は強フォワード安全性を有する．

定理 1. $\widehat{\Sigma}$ が $\widehat{\mathcal{L}}$-適応的安全かつフォワード安全で，かつ (4) 式および (5) 式を満たすならば *($\widehat{\mathcal{L}} = (\widehat{\mathcal{L}}_{\mathsf{Setup}}, \widehat{\mathcal{L}}_{\mathsf{Upd}}, \widehat{\mathcal{L}}_{\mathsf{Srch}})$)*，上記構成法による Σ は以下の漏洩関数 $\mathcal{L} = (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Upd}}, \mathcal{L}_{\mathsf{Srch}})$ の下で $\mathcal{L}$-適応的安全性を満たす：

$$\mathcal{L}_{\mathsf{Setup}}(\kappa) = \widehat{\mathcal{L}}_{\mathsf{Setup}}(\kappa),$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add},) = \mathsf{id},$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = (\mathsf{id}, |f_{\mathsf{id}}|),$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{del}, (\mathsf{id}, w)) = \widehat{\mathcal{L}}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{del}, (\mathsf{id}, w)),$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{del}, \mathsf{id}) = \left(\widehat{\mathcal{L}}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{del}, (\mathsf{id}, w)), \mathsf{id}\right),$$
$$\mathcal{L}_{\mathsf{Srch}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, q) = \widehat{\mathcal{L}}_{\mathsf{Srch}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, q).$$

従って，Σ は強フォワード安全性を満たす．

証明. $\widehat{\Sigma}$ は $\widehat{\mathcal{L}}$ の下で $\widehat{\mathcal{L}}$-適応的安全性を満たすことから，任意の確率的多項式時間アルゴリズム D に対し，$|\Pr[\mathsf{Real}_{\mathsf{D}}^{\widehat{\Sigma}}(\kappa, Q) = 1] - \Pr[\mathsf{Ideal}_{\mathsf{D},\mathsf{S},\widehat{\mathcal{L}}}^{\widehat{\Sigma}}(\kappa, Q) = 1]| \leq \mathsf{negl}(\kappa)$ であるような S が存在する．Setup，Search，及び変更処理に対する Update は $\widehat{\Sigma}$ の各アルゴリズムと同じであり，漏洩関数も同じであるため，S は問題なく各アルゴリズムをシミュレートすることができる．以下では挿入・削除に関するシミュレートについて述べる．

まず，$\mathsf{Update}(k, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}}), \sigma^{(t)}; \mathsf{EDB}^{(t)})$ のシミュレートについて述べる．$\mathsf{Real}_{\mathsf{D}}^{\widehat{\Sigma}}(\kappa, Q)$ では，各 $w \in \mathcal{W}_{\mathsf{id}}$ に対して $\widehat{\mathsf{Update}}(k, \mathsf{add}, (\mathsf{id}, 0\|w), \sigma^{(t')}; \mathsf{EDB}^{(t')})$ を実行し，また各 $\beta \in [\mathsf{max}_{\mathsf{id}} - |\mathcal{W}_{\mathsf{id}}|]$ に対して $\widehat{\mathsf{Update}}(k, \mathsf{add}, (\mathsf{id}, 1\|\beta), \sigma^{(t')}; \mathsf{EDB}^{(t')})$ を実行する．S は，上記の手順を以下のようにシミュレートする．S は $\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = (\mathsf{id}, |f_{\mathsf{id}}|)$ を得て，$\mathsf{CompMax}(\Lambda, |f_{\mathsf{id}}|)$ から $\mathsf{max}_{\mathsf{id}}$ を計算する．その後，id を用いた $\widehat{\mathsf{Update}}(k, \mathsf{add}, (\mathsf{id}, \cdot), \sigma^{(t')}; \mathsf{EDB}^{(t')})$ のシミュレートを $\mathsf{max}_{\mathsf{id}}$ 回行えばよい．

次に，$\mathsf{Update}(k, \mathsf{del}, \mathsf{id}, \sigma^{(t)}; \mathsf{EDB}^{(t)})$ のシミュレートについて述べる．S は id が追加されたときの漏洩情報を参照し，$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t')}, \mathsf{EDB}^{(t')}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = (\mathsf{id}, |f_{\mathsf{id}}|)$，対応する $\mathsf{max}_{\mathsf{id}}$ を再計算する．後は挿入処理と同様に $\widehat{\mathsf{Update}}(k, \mathsf{del}, (\mathsf{id}, \cdot), \sigma^{(t')}; \mathsf{EDB}^{(t')})$ のシミュレートを $\mathsf{max}_{\mathsf{id}}$ 回行えばよい．

従って，D に対する S は $|\Pr[\mathsf{Real}_{\mathsf{D}}^{\Sigma}(\kappa, Q) = 1] - \Pr[\mathsf{Ideal}_{\mathsf{D},\mathsf{S},\mathcal{L}}^{\Sigma}(\kappa, Q) = 1]| \leq \mathsf{negl}(\kappa)$ を満たし，これは全ての D に対して成り立つ． □

4. 提案手法

前節の素朴な手法では，追加される各ファイル f_{id} ごとに $\mathsf{max}_{\mathsf{id}}$ 個のアドレスを生成することになるため，非常に効率が悪い．本節では，より効率的にフォワード安全な Dynamic SSE 方式を強フォワード安全な方式にする手法を提案する．

4.1 構成の概要

基本的なアイディアは，前節の手法では必ず f_{id} に関わる情報が $\mathsf{max}_{\mathsf{id}}$ 個となるようダミーを埋めていたところを，乱数 μ_{id} ($\leq \mathsf{max}_{\mathsf{id}}$) を選び，$\mu_{\mathsf{id}}$ 個となるまでダミーを埋めるように変更することである．しかし，このように変更するだけでは次の理由で安全性証明が難しい．試行 $\mathsf{Real}_{\mathsf{D}}^{\Sigma}(\kappa, Q)$ において，クエリ $(\mathsf{upd}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}}))$ に対して，μ_{id} が取り得る範囲は $[|\mathcal{W}_{\mathsf{id}}|, \mathsf{max}_{\mathsf{id}}]$ である．一方で，試行 $\mathsf{Ideal}_{\mathsf{D},\mathsf{S},\mathcal{L}}^{\Sigma}(\kappa, Q)$ では，シミュレータ S は強フォワード安全性のため $|\mathcal{W}_{\mathsf{id}}|$ が分からず，μ_{id} として選び得る範囲が $[\mathsf{max}_{\mathsf{id}}]$ となってしまう．従って，D は多くのキーワードを含むような ($|\mathcal{W}_{\mathsf{id}}|$ が $\mathsf{max}_{\mathsf{id}}$ に近いような) f_{id} をクエリすることでシミュレータは無視できない確率で $\mu_{\mathsf{id}} < |\mathcal{W}_{\mathsf{id}}|$ となるような μ_{id} を選んでしまうため，安全性を破ることができる．

そこで我々は Kamara らの**刈り取り技法** [10] を応用し，$\mu_{\mathsf{id}} < |\mathcal{W}_{\mathsf{id}}|$ となる状況を許すアプローチを取る．具体的には，(どんな μ_{id} とも独立な) しきい値 λ を用意し，これを秘密鍵 k 及びセットアップ時の漏えい情報 $\mathcal{L}_{\mathsf{Setup}}(\kappa)$ として含めることで，両試行における μ_{id} を選ぶ範囲を $[\lambda, \mathsf{max}_{\mathsf{id}}]$ に統一する．試行 $\mathsf{Real}_{\mathsf{D}}^{\Sigma}(\kappa, Q)$ においても $\mu_{\mathsf{id}} < |\mathcal{W}_{\mathsf{id}}|$ の状況を許すということは，いくつかのキーワードはデータベース

に登録されないということを意味する．これはキーワードベクトル ($\mathcal{W}_{\mathsf{id}}$ の要素を並べたもの) $\mathbf{w}_{\mathsf{id}} = (w_1, \ldots, w_{|\mathcal{W}_{\mathsf{id}}|})$ の末端が刈り取られることを意味することから，便宜的に**刈取**と呼ぶ．そのため，もしクライアントが刈り取られたキーワードを検索してしまうと正しい検索結果を得ることができず，すなわち “全てのキーワードに関して高い確率で正しい検索結果を得ることができる” という検索正当性 (定義 2) を達成することができない．そこで我々は次の (P_q, δ)-検索正当性を導入する．

定義 5 ((P_q, δ)-検索正当性)**.** Σ を *Dynamic SSE* 方式とする．任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対し，図 *1* の試行 $\mathsf{Exp}^{\Sigma}_{\mathcal{A}}$ のステップ *10* で $\mathcal{A}$ がある確率分布 P_q に従って $q^\star$ を選ぶよう変更したものを考える (*$\mathsf{Exp}^{\Sigma}_{\mathcal{A},P_q}$* とする)．この時，$\Pr[\mathsf{Exp}^{\Sigma}_{\mathcal{A},P_q} = 1] \geq \delta$ を満たすならば，Σ は (P_q, δ)-検索正当性を満たすという．

すなわち，検索キーワードの確率分布 P_q を事前に固定した上で，その確率が十分低いキーワードは検索に失敗しても構わないとする定義である．言い換えれば，辞書の確率上位のキーワードからその確率を足し，累計 δ となるまでに登場したキーワードは必ず検索に成功する．検索に成功しないキーワードがあるため通常の検索正当性に比べて弱い概念だが，実際の検索システムにおいてもインデックスサイズ削減のために検索に不要なキーワードを削除することはあることから，ある程度自然な定義だと考えている．

上記の刈り取り技法を用いて，(P_q, δ)-検索正当性を次のように達成する．まず，各ファイル f_{id} 中のキーワード $\mathbf{w}_{\mathsf{id}}$ をその確率が高い順番に並び替え，$\mathbf{w}'_{\mathsf{id}}$ とする．この処理をランク関数 Rank として書く．刈り取りが起きる場合 (すなわち $\mu_{\mathsf{id}} < |\mathcal{W}_{\mathsf{id}}|$ の場合)，キーワードは確率が低い順に (キーワードベクトルの末尾から) 刈り取られていく．各キーワードベクトルの先頭 λ 個のキーワードは刈り取られることはないため，確率上位 λ 位までのキーワードは必ず検索に成功する．一方で，刈り取られる可能性のあるキーワード (確率 λ 位以降のキーワード) は検索に用いられる確率が低いため，結果として提案構成法は高い確率で正しい検索結果を得ることができる．

4.2 構成法

任意のフォワード安全な Dynamic SSE 方式 $\widehat{\Sigma} = (\widehat{\mathsf{Setup}}, \widehat{\mathsf{Update}}, \widehat{\mathsf{Search}})$ を強フォワード安全性な方式 $\Sigma = (\mathsf{Setup}, \mathsf{Update}, \mathsf{Search})$ に変換する提案手法を示す．変更および検索は素朴な手法と同じであるため，以下では省略する．

$\mathsf{Setup}(1^\kappa)$**:** 以下を実行する．

1. $\widehat{\mathsf{Setup}}(1^\kappa)$ を実行し，$(k', \widehat{\sigma}^{(0)}, \widehat{\mathsf{EDB}}^{(0)})$ を得る．確率分布 P_q を基に λ を設定し，$\mathcal{M} \coloneqq \emptyset$ とする．
2. $(k, \sigma^{(0)}, \mathsf{EDB}^{(0)}) \coloneqq ((k', \lambda), (\widehat{\sigma}^{(0)}, \mathcal{M}), \widehat{\mathsf{EDB}}^{(0)})$ を出力する．

$\mathsf{Update}(k, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}}), \sigma^{(t)}; \mathsf{EDB}^{(t)})$**:** $(\sigma^{(t)}, \mathsf{EDB}^{(t)})$ を $((\widehat{\sigma}^{(t')}, \mathcal{M}), \widehat{\mathsf{EDB}}^{(t')})$ とし，カウンタ c を 0 に初期化する．以下を実行する．

1. $\mathsf{Rank}(\mathbf{w}_{\mathsf{id}})$ を実行し，$\mathbf{w}'_{\mathsf{id}}$ を得る．
2. $\mathsf{CompMax}(\Lambda, |f_{\mathsf{id}}|)$ を実行し，$\mathsf{max}_{\mathsf{id}}$ を得た後，μ_{id} を $[\lambda, \mathsf{max}_{\mathsf{id}}]$ の範囲から一様ランダムに選ぶ．
3. もし $\mu_{\mathsf{id}} \geq |\mathcal{W}_{\mathsf{id}}|$ であれば以下を実行する．そうでなければ本ステップは行わない．まず，各 $i \in [|\mathcal{W}_{\mathsf{id}}|]$ に対して以下を実行する．
   (3-a) $\widehat{\mathsf{Update}}(k', \mathsf{add}, (\mathsf{id}, 0 \| \mathbf{w}_{\mathsf{id}}[i]), \widehat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\widehat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．
   (3-b) c をインクリメントする．
   次に各 $\beta \in [\mu_{\mathsf{id}} - |\mathcal{W}_{\mathsf{id}}|]$ に対し，以下を実行する．
   (3-c) $\widehat{\mathsf{Update}}(k', \mathsf{add}, (\mathsf{id}, 1 \| \beta), \widehat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\widehat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．
   (3-d) c をインクリメントする．
4. もし $\mu_{\mathsf{id}} < |\mathcal{W}_{\mathsf{id}}|$ であれば以下を実行する．そうでなければ本ステップは行わない．各 $i \in [\mu_{\mathsf{id}}]$ に対して以下を実行する．
   (4-a) $\widehat{\mathsf{Update}}(k', \mathsf{add}, (\mathsf{id}, 0 \| \mathbf{w}_{\mathsf{id}}[i]), \widehat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\widehat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．
   (4-b) c をインクリメントする．
5. $\mathcal{M}$ に $(\mathsf{id}, \mu_{\mathsf{id}})$ を追加し，$(\sigma^{(t+1)}; \mathsf{EDB}^{(t+1)}) \coloneqq ((\widehat{\sigma}^{(t'+\mu_{\mathsf{id}})}, \mathcal{M}); \widehat{\mathsf{EDB}}^{(t'+\mu_{\mathsf{id}})})$ を出力する．

$\mathsf{Update}(k, \mathsf{del}, \mathsf{id}, \sigma^{(t)}; \mathsf{EDB}^{(t)})$**:** まず $(\sigma^{(t)}, \mathsf{EDB}^{(t)})$ を $((\widehat{\sigma}^{(t')}, \mathcal{M}), \widehat{\mathsf{EDB}}^{(t')})$ とし，カウンタ c を 0 に初期化する．$\mathcal{M}$ から $(\mathsf{id}, \mu_{\mathsf{id}})$ を取り出しておく．クライアントはサーバに (暗号化された状態で) 保存していた f_{id} を取り出し，以下を実行する．

1. $\mathsf{Rank}(\mathbf{w}_{\mathsf{id}})$ を実行し，$\mathbf{w}'_{\mathsf{id}}$ を得る．
2. $\mathsf{CompMax}(\Lambda, |f_{\mathsf{id}}|)$ を実行し，$\mathsf{max}_{\mathsf{id}}$ を得る．
3. もし $\mu_{\mathsf{id}} \geq |\mathcal{W}_{\mathsf{id}}|$ であれば以下を実行する．そうでなければ本ステップは行わない．まず，各 $i \in [|\mathcal{W}_{\mathsf{id}}|]$ に対して以下を実行する．
   (3-a) $\widehat{\mathsf{Update}}(k', \mathsf{del}, (\mathsf{id}, 0 \| \mathbf{w}_{\mathsf{id}}[i]), \widehat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\widehat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．
   (3-b) c をインクリメントする．
   次に各 $\beta \in [\mu_{\mathsf{id}} - |\mathcal{W}_{\mathsf{id}}|]$ に対し，以下を実行する．
   (3-c) $\widehat{\mathsf{Update}}(k', \mathsf{del}, (\mathsf{id}, 1 \| \beta), \widehat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を

実行し，$(\widehat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．
(3-d) c をインクリメントする．

4. もし $\mu_{\mathsf{id}} < |\mathcal{W}_{\mathsf{id}}|$ であれば以下を実行する．そうでなければ本ステップは行わない．各 $i \in [\mu_{\mathsf{id}}]$ に対して以下を実行する．
(4-a) $\widehat{\mathsf{Update}}(k', \mathsf{del}, (\mathsf{id}, 0 \| \mathbf{w}_{\mathsf{id}}[i]), \widehat{\sigma}^{(t'+\mathsf{c})}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c})})$ を実行し，$(\widehat{\sigma}^{(t'+\mathsf{c}+1)}; \widehat{\mathsf{EDB}}^{(t'+\mathsf{c}+1)})$ を得る．
(4-b) c をインクリメントする．

5. $\mathcal{M}$ から $(\mathsf{id}, \mu_{\mathsf{id}})$ を削除し，$(\sigma^{(t+1)}; \mathsf{EDB}^{(t+1)}) := ((\widehat{\sigma}^{(t'+\mu_{\mathsf{id}})}, \mathcal{M}); \widehat{\mathsf{EDB}}^{(t'+\mu_{\mathsf{id}})})$ を出力する．

検索正当性． Λ の要素を確率分布 P_q に従い確率が大きい順に $w_1, w_2, \ldots, w_D$ とする．以下を満たすように λ を設定すれば (P_q, δ)-検索正当性を達成するのに十分である：

$$\sum_{i=1}^{\lambda} P_q(w_i) \geq \delta > \sum_{i=1}^{\lambda-1} P_q(w_i).$$

安全性． 上記構成法は強フォワード安全性を満たす．

定理 2. $\widehat{\Sigma}$ が $\widehat{\mathcal{L}}$-適応的安全かつフォワード安全で，かつ (4) 式および (5) 式を満たすならば $(\widehat{\mathcal{L}} = (\widehat{\mathcal{L}}_{\mathsf{Setup}}, \widehat{\mathcal{L}}_{\mathsf{Upd}}, \widehat{\mathcal{L}}_{\mathsf{Srch}}))$，上記構成法による Σ は以下の漏洩関数 $\mathcal{L} = (\mathcal{L}_{\mathsf{Setup}}, \mathcal{L}_{\mathsf{Upd}}, \mathcal{L}_{\mathsf{Srch}})$ の下で $\mathcal{L}$-適応的安全性を満たす：

$$\mathcal{L}_{\mathsf{Setup}}(\kappa) = \left(\widehat{\mathcal{L}}_{\mathsf{Setup}}(\kappa), \lambda\right),$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, w)) = \mathsf{id},$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{add}, (\mathsf{id}, \mathcal{W}_{\mathsf{id}})) = (\mathsf{id}, |f_{\mathsf{id}}|),$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{del}, (\mathsf{id}, w)) = \widehat{\mathcal{L}}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{del}, (\mathsf{id}, w)),$$
$$\mathcal{L}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{del}, \mathsf{id}) = \left(\widehat{\mathcal{L}}_{\mathsf{Upd}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, \mathsf{del}, (\mathsf{id}, w)), \mathsf{id}\right),$$
$$\mathcal{L}_{\mathsf{Srch}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, q) = \widehat{\mathcal{L}}_{\mathsf{Srch}}(\sigma^{(t)}, \mathsf{EDB}^{(t)}, q).$$

従って，強フォワード安全性を満たす．

証明． 基本的には定理 1 と同様に証明が可能である．定理 1 の証明との違いは，挿入処理のシミュレートにおいて，$\mathsf{max}_{\mathsf{id}}$ 回 $\widehat{\mathsf{Update}}$ をシミュレートしていたところを，一様ランダムに $\mu_{\mathsf{id}} \in [\lambda, \mathsf{max}_{\mathsf{id}}]$ を選び，$\widehat{\mathsf{Update}}$ のシミュレートを μ_{id} 回行う点である． □

4.3 Zipf 分布を適用した具体例

本節では，確率分布 P_q の具体例として Zipf 分布 [21] を適用した例を示す．Zipf 分布は情報検索分野では主要な裾の重い分布のひとつとして知られている．

定義 6 (Zipf 分布 [21])**.** 集合 $\Lambda = \{w_1, \ldots, w_D\}$ の要素を値に取る確率変数 q の確率分布 P_q が，j 番目に大きい生起確率を持つキーワードの生起確率が以下を満たしている時，Λ は $\mathcal{Z}_{D,c}$-*distributed* であるという：

$$\frac{1}{j^c \cdot H_{D,c}}.$$

ここで，$H_{D,c}$ は調和級数 $\sum_{i=1}^{D} i^{-c}$ である．

c は任意に定めることのできる定数であり同様に解析可能なため，以降は簡単のため $c = 1$ の場合を考える．

Zipf 分布を仮定した (P_q, δ)-検索正当性． Zipf 分布を仮定することで，より具体的に (P_q, δ)-検索正当性を達成するために必要な λ の下界を見積もることができる．ただし，以下に記す λ の下界は十分性のみしか議論していないため，改善の余地があることに留意されたい．

定理 3 (λ の下界)**.** 前節で構成した $\Sigma' = (\mathsf{Setup}, \mathsf{Update}, \mathsf{Search})$ は，Λ が $\mathcal{Z}_{D,1}$-*distributed* であり，かつ λ が以下の不等式を満たしていれば (P_q, δ)-検索正当性を満たす：

$$\lambda \geq e^{H_{D,1}\delta} - 1.$$

証明． Suc を $\mathcal{X}_q^{(t)} = \mathsf{ID}^{(t)}$ が成り立つならば 1，そうでなければ 0 である確率変数とする．この時，以下が成り立つ．

$$\begin{aligned}
&\Pr[\text{The client obtains a correct search result for } q] \\
&= \Pr[(q = w_1 \wedge \mathsf{Suc} = 1) \vee \cdots \vee (q = w_D \wedge \mathsf{Suc} = 1)] \\
&= \sum_{i=1}^{D} \Pr[q = w_i \wedge \mathsf{Suc} = 1] \\
&\geq \sum_{i=1}^{\lambda} \Pr[q = w_i \wedge \mathsf{Suc} = 1] \\
&= \sum_{i=1}^{\lambda} \Pr[q = w_i] \cdot \Pr[\mathsf{Suc} = 1 \mid q = w_i] \\
&= \sum_{i=1}^{\lambda} \frac{1}{H_{D,1} \cdot i} \qquad (7) \\
&\geq \frac{1}{H_{D,1}} \left(\log_e(\lambda + 1) + \frac{\lambda}{2(\lambda + 1)}\right) \qquad (8) \\
&\geq \frac{1}{H_{D,1}} \log_e(\lambda + 1).
\end{aligned}$$

ここで，(7) 式は Zipf 分布と Λ の先頭 λ 個のキーワードは刈り取られることはないという事実から成り立ち，(8) 式は調和級数 $\sum_{i=1}^{\lambda} 1/i$ の性質から成り立つ．

従って，ある固定した δ に対して λ が以下の不等式を満たすのであれば，Dynamic SSE Σ は (P_q, δ)-検索正当性を満たす．

$$\frac{1}{H_{D,1}} \log_e(\lambda + 1) \geq \delta.$$

従って，$\lambda \geq e^{H_{D,1}\delta} - 1$ を得る． □

次に，扱うデータ群の具体例として，Enron データセット [1] を用いる．データセットの詳細は表 1, 2 の通り．データセットは 214,874 個のキーワードを含んでいることから $D = 214{,}874$ とした．対応する調和級数は $H_{D,1} = 12.855$

表 1　Enron データセット.

ファイル数	キーワード数	サイズ (KB)
517,401	214,874	2,413,971

表 2　Enron データセットの統計情報.

パラメータ	最大	最小	平均
$\lvert f_{\mathsf{id}}\rvert$ (bytes)	2,011,957	398	4,445
$\mathsf{max}_{\mathsf{id}}$	251,495	58	343.7
$\lvert\mathcal{W}_{\mathsf{id}}\rvert$	59,148	12	77.1
$\mathsf{max}_{\mathsf{id}} - \lvert\mathcal{W}_{\mathsf{id}}\rvert$	192,347	46	266.6

である．例えば，$\delta = 0.93$ と設定した時，定理 3 より $\lambda \geq 155{,}616$ であるから，$\lambda := 155{,}616$ とすればよい．

なお，方式では μ_{id} を $[\lambda, \mathsf{max}_{\mathsf{id}}]$ から一様ランダムに選んでいるところは，任意の確率分布 (便宜的に P_μ と書く) に従って選ぶように変更しても安全性に問題はない (ただし，その確率分布は漏洩関数に含める必要がある)．λ に近い値ほど確率が高くなるような P_μ を選ぶことでデータベースのサイズをより小さくすることが可能だが，一方でそれは $\mu_{\mathsf{id}} < \lvert\mathcal{W}_{\mathsf{id}}\rvert$ となる確率が大きくなる，すなわちキーワードが刈り取られる確率も大きくなることも意味する．ただし，定理 3 ではどんな場合でも刈り取られない先頭 λ キーワードについてのみ議論しているため，どんな P_μ を選んでも (P_q, δ)-検索正当性は満たす．

また経験的な方法として，データセットに応じて λ を適宜調整するという方法もある．表 2 によれば，$\lvert\mathcal{W}_{\mathsf{id}}\rvert$ の平均は 77.1 であり，これはたとえ $\lambda = 1{,}000$ などの小さい λ を設定したとしても刈り取りがほとんど起こらないだろう，ということを示唆している．多くの場合に正しい検索結果が得られると考えられるが，この方法は必ずしも (P_q, δ)-検索正当性は満たすとは限らない．

謝辞　情報検索に関する専門用語についてご助言いただいた中井雄士氏に感謝いたします．本研究は JSPS 科研費 JP18K11293，JP18H05289 の助成，および文部科学省の卓越研究員事業の支援を受けたものです．

参考文献

[1] Enron Email Dataset (May 7, 2015 Version) (2015).
[2] Blackstore, L., Kamara, S. and Moataz, T.: Revisiting Leakage Abuse Attacks, *Network and Distributed System Security Symposium, NDSS 2020*, The Internet Society (2020).
[3] Bost, R.: $\Sigma o\varphi o\varsigma$: Forward Secure Searchable Encryption, *ACM SIGSAC Conference on Computer and Communications Security, CCS 2016*, New York, NY, USA, ACM, pp. 1143–1154 (2016).
[4] Cash, D., Grubbs, P., Perry, J. and Ristenpart, T.: Leakage-Abuse Attacks Against Searchable Encryption, *ACM SIGSAC Conference on Computer and Communications Security, CCS 2015*, New York, NY, USA, ACM, pp. 668–679 (2015).
[5] Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C. and Steiner, M.: Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation, *Network and Distributed System Security Symposium, NDSS 2014*, The Internet Society (2014).
[6] Curtmola, R., Garay, J., Kamara, S. and Ostrovsky, R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions, *ACM Conference on Computer and Communications Security, CCS 2006*, New York, NY, USA, ACM, pp. 79–88 (2006).
[7] Etemad, M., Küpçü, A., Papamanthou, C. and Evans, D.: Efficient Dynamic Searchable Encryption with Forward Privacy, *Proceedings of Privacy Enhancing Technologies (PoPETs)*, Vol. 2018(1), pp. 5–20 (2018).
[8] Ghareh Chamani, J., Papadopoulos, D., Papamanthou, C. and Jalili, R.: New Constructions for Forward and Backward Private Symmetric Searchable Encryption, *ACM SIGSAC Conference on Computer and Communications Security, CCS 2018*, New York, NY, USA, ACM, pp. 1038–1055 (2018).
[9] Islam, M. S., Kuzu, M. and Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation, *Network and Distributed System Security Symposium, NDSS 2012* (2012).
[10] Kamara, S. and Moataz, T.: Computationally Volume-Hiding Structured Encryption, *Advances in Cryptology – EUROCRYPT 2019* (Ishai, Y. and Rijmen, V., eds.), Cham, Springer International Publishing, pp. 183–213 (2019).
[11] Kamara, S., Papamanthou, C. and Roeder, T.: Dynamic Searchable Symmetric Encryption, *ACM Conference on Computer and Communications Security, CCS 2012*, New York, NY, USA, ACM, pp. 965–976 (2012).
[12] Kim, S. K., Kim, M., Lee, D., Park, J. H. and Kim, W.-H.: Forward Secure Dynamic Searchable Symmetric Encryption with Efficient Updates, *ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*, New York, NY, USA, ACM, pp. 1449–1463 (2017).
[13] Ning, J., Xu, J., Liang, K., Zhang, F. and Chang, E.: Passive Attacks Against Searchable Encryption, *IEEE Transactions on Information Forensics and Security*, Vol. 14, No. 3, pp. 789–802 (2019).
[14] Rompay, C. V., Molva, R. and Önen, M.: A Leakage-Abuse Attack Against Multi-User Searchable Encryption, *Proceedings of Privacy Enhancing Technologies (PoPETs)*, Vol. 2017, No. 3, pp. 168–178 (2017).
[15] Song, D. X., Wagner, D. and Perrig, A.: Practical techniques for searches on encrypted data, *IEEE Symposium on Security and Privacy, S&P 2000*, pp. 44–55 (2000).
[16] Stefanov, E., Papamanthou, C. and Shi, E.: Practical Dynamic Searchable Encryption with Small Leakage, *Network and Distributed System Security Symposium, NDSS 2014*, The Internet Society (2014).
[17] 渡邉: フォワード安全かつ検索時通信量が最適な動的検索可能暗号, *SCIS 2020* **予稿集**, 3B3-2 (2020).
[18] 渡邉, 岩本, 太田: 効率的でフォワード安全な動的検索可能暗号, *SCIS 2019* **予稿集**, 3C1-3 (2019).
[19] 渡邉, 大原, 岩本, 太田: (強) フォワード安全な動的検索可能暗号の効率的な構成, *CSS 2019* **予稿集**, pp. 1203–1210 (2019).
[20] Zhang, Y., Katz, J. and Papamanthou, C.: All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption, *USENIX Security 2016*, Austin, TX, USENIX Association, pp. 707–720 (2016).
[21] Zipf, G. K.: The Pshcho-Bilology Language (1935).