Vol.2016-CSEC-73 No.18
Vol.2016-IOT-33 No.18
2016/5/27

曖昧検索機能を持つ動的な検索可能暗号方式

大塚 元気[1]　多田 充[2,a)]

概要：2012 年，Kamara ら [2] は検索可能暗号方式 (SSE) が最低限実用的なものになるためには，(1) 検索にかかる計算量が準線形オーダーであること，(2) 適応的な選択キーワード攻撃に対し安全であること，(3) 大きすぎないインデックスサイズ及びインデックスに対するファイルの動的な追加・削除が可能であること，が必要であると述べ，これらの要件をすべて満たす SSE を提案した。しかし，この方式は検索の機能として，完全一致によるものしか扱うことができない。SSE としてより実用的なものにするためには，曖昧検索機能やレコメンド等の検索の機能を実現し，より柔軟な検索機能を実現すべきである。本論文では，上記の要件を全て満たし，その上で，フォーマットの違いやスペリングミス等の誤字に対応する機能や，ユーザの入力から関連するキーワードを予測するレコメンド機能を実現する方法を述べる。

キーワード：検索可能暗号方式 (SSE)，曖昧検索

Dynamic searchable symmetric encryption with fuzzy reference function

Genki Otsuka[1]　Mitsuru Tada[2,a)]

Abstract: In 2012, Kamara et al.[2] told that searchable symmetric encryption (SSE) systems should provide the following functions for its being minimally practical: (1) The system requires, at most, quasi-linear computational complexity for searching; (2) The system is secure against adaptive chosen-keyword attacks; (3) The system enables dynamic file-addition and file-deletion for each index of not so large size. Then they presented an SSE with all the properties given above. However, that system can deal with perfect-matching search. It is better for SSE to have the functions for flexible search such as fuzzy-matching search and showing recommendations. In this paper, we show an SSE which realizes the functions for making up for misspelling and differnent formats of data, and that for recommendation derived from words given by the user.

Keywords: Searchable symmetric encryption (SSE), Fuzzy reference

1. はじめに

クラウドストレージは大変便利なサービスである。ただ，そのセキュリティは完璧とは言えない。例えば，格納したデータを外部に漏らさないとも限らない。またそもそも，クラウドストレージのサービス提供者がそのデータを見るかも知れない。そのため，プライベートなデータ，その他機密性が要求されるデータ，人に見せたくないデータをクラウドストレージに格納する場合，当該データを暗号化して格納する必要がある。しかし暗号化するという行為はそのデータに対する基本的な処理 — 例えば文書の検索 — ですら不可能にしてしまう。そのため，クラウドストレージの利便性を保ったまま，プライベートなデータに対する検索，つまり，暗号化されたデータに対する検索を行いたいというのは自然な要求でもある。検索可能暗号とは，データを暗号化したまま検索を行う技術を指す。検索可能暗号

[1] 千葉大学大学院 理学研究科
Graduate School of Science, Chiba University, Inage, Chiba 263–8522, Japan
[2] 千葉大学 統合情報センター
Institute of Media and Information Technologies, Chiba University, Inage, Chiba 263–8522, Japan
[a)] m.tada@faculty.chiba-u.jp

の中でも，ユーザが所持し，クラウドストレージへ格納したいファイル群 $\mathbb{F} = (f_1, \ldots, f_n)$ の他に，インデックスと呼ばれるデータ構造を $\mathbb{F}$ から作成し，暗号の初期化時に，暗号化したファイル群 $\mathbb{C} = (c_1, \ldots, c_n)$ と共にデータベースに格納し，検索時にはこのインデックスを利用し検索を行うタイプの検索可能暗号を「インデックスベースの検索可能暗号」と呼ぶ。本論文では，インデックスベースの検索可能暗号に対する考察をし，単に検索可能暗号と呼んだ場合はインデックスベースのものを指すことにする。著者らが知る限り, 検索可能暗号に対する明示的な発表は 2000 年の Song ら [4] のものであり，以降様々な成果が報告されている。2012 年，Kamara ら [2] は，検索可能暗号が実用的であるためには最低でも

(C1) 検索に要する時間が準線形であること
(C2) 適応的キーワード選択攻撃に対して安全であること
(C3) インデックスのサイズが現実的であること
(C4) インデックスを動的に更新, つまりファイルの追加と削除が効率的に可能であること

のすべてが満たされるべきであると主張し，同文献において，そのすべてを満たしたインデックスベースの検索可能暗号を提案した。しかし，以上のすべてを満たすことは検索可能暗号が実用的なものとなるための最低限な要件に過ぎず，実際，Kamara らの方法では完全一致による検索のみしか行うことができない。そのため誤字や表記の揺れ，フォーマットの違いに対応せず，また我々がよく Google 等の検索エンジンで検索をするときのような，レコメンド機能 [*1] も存在しない。さらに，インデックス作成時に使用するワードをすべて定義しておかなければならず，新規単語を追加したい場合はデータベースを再度，一から構築する必要がある。本論文では Kamara らの手法 [2] を拡張し，以下のようにして，これらの問題点を解決した。これは，クラウドストレージサービスを検索可能暗号を用いて展開するにあたり，Kamara らの手法では足りない機能面の拡張に焦点を当て，検索可能暗号としての必要最低限の項目をクリアしたうえで，以下の問題を解決したものである。

(1) 誤字耐性：N-Gram[3] の技術を用い，検索時に，少しの誤字 [*2] が存在しても正しく検索できるようにした。
(2) 新規単語の追加：N-Gram の技術を用いることにより，新規単語を出現させないようにインデックスを作成できるようにした。
(3) レコメンド機能：レコメンド用のテーブルを新たに作成しすることによりレコメンド機能を実現した。特に，同じワードを表す複数の表現やフォーマットの異なる表現をレコメンドできるようにした。

本論文は以下のように構成される。第 2 章では提案手法や既存技術を説明するために必要な記号や概念の定義を行う。第 3 章では既存の方式を示すが，スペースの都合上，説明は割愛する。第 4 章では，課題解決のために必要な戦略を述べ，実際にその課題を解決するための構成を具体的に記述し，安全性に関する説明も行う。第 5 章では提案手法の問題点や，曖昧検索の具体例を述べ，第 6 章で本論文をまとめる。

2. 準備

ここでは，本論文で使用する表記法，検索可能暗号方式 (SSE)，および，インデックスへの動的な変更を可能にした Dynamic SSE (DSSE) の定義を述べる。

2.1 表記法

列 v に対して，v_i または $v[i]$ によりその i 番目の要素，$\#v$ によりその要素数を表す。同様に $\mathtt{A}$ を配列とするとき，$\mathtt{A}[i]$ でその i 番目の要素，$\#A$ でセルの数を表す。$\mathtt{L}$ がリストのとき，$\#\mathtt{L}$ でそのノード数を表す。$\mathtt{T}$ を連想配列としたとき，キーと値の組 (s, v) が $\mathtt{T}$ に含まれていれば，$\mathtt{T}[s]$ は v を表し，$\#\mathtt{T}$ は格納されている組の数を表す。キーが s となる要素が存在するとき $s \in \mathtt{T}$ と表す。

$W \subset \mathfrak{P}(\{0,1\}^*)$ で可能な語全体の (有限) 集合を表し，これを辞書とも呼ぶ。ただし，セキュリティパラメータ k に対して $\#W = \mathsf{poly}(k)$ とする [*3]。語 $w \in W$ に対して $|w|$ で w のビット長を表す。ただし，$\forall w \in W, |w| = \mathsf{poly}(k)$ とする。$f = (w_1, \ldots, w_m) \in W^m$ をファイルとしたとき，$\#f$ は f に含まれる語数，すなわち m とし，$|f|$ はファイルのビット長，つまり $\sum_{i=1}^{m} |w_i|$ とする。また，$\bar{f}$ は f の中から重複を除いたファイルとする。ただし，重複は後ろから除くとする。

ユーザのデータは $n(= \mathsf{poly}(k))$ 個のファイルの集合 $\mathbb{F}(\subset 2^W) = (f_1, \ldots, f_n)$ と見なせる。各ファイル $f_i = (w_{i,1}, \ldots, w_{i,m_i})$ は $m_i = \mathsf{poly}(k)$ 個の語を含み，一意的な識別子 (ID) をもち，これを $\mathsf{id}(f_i)$ とする。キーワード $w \in W$ に対して，$\mathbb{F}$ 中の w を含むファイルのリストを，$\mathbb{F}_w$ で表す。各 f_i を定められた暗号方式 Enc および鍵 K で暗号化したものを c_i とし，$\mathbb{C} = (c_1, \ldots, c_n)$ としたとき，w に対する $\mathbb{F}_w$ の各要素を暗号化したものを $\mathbb{C}_w$ と表記する。つまり，$\mathbb{F}_w = (f_{i_1}, \ldots, f_{i_\ell})$ とすると $\mathbb{C}_w = (c_{i_1}, \ldots, c_{i_\ell})$ となる。

2.2 検索可能暗号方式 (SSE)

辞書 W 上のインデックスベース SSE 方式とは，5 つの

[*1] 文章入力時に, 入力中の単語・文に対し, 次に入力する可能性の高い単語・文をユーザに知らせる機能。
[*2] どれだけの曖昧さを許すかについてはアプリケーションごとにパラメータとして定める。
[*3] 本論文では，x が k の多項式に満たないオーダー — 例えば $x = o(k)$ — であったとしても $x = \mathsf{poly}(k)$ のように表記する。つまり $x = \mathsf{poly}(k)$ は「x は高々 (k の) 多項式オーダー」の意味であるとする。

多項式時間アルゴリズムの組：

$$\mathrm{SSE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Trapdoor}, \mathsf{Search}, \mathsf{Dec})$$

である。各アルゴリズムは以下のように定義される。

Gen : $1^k \mapsto K$： セキュリティパラメータ k に対する秘密鍵 K を出力する確率的アルゴリズムである。

Enc : $(K, \mathbb{F}) \mapsto (\gamma, \mathbb{C})$： 鍵 K および文書 (ファイル) のコレクション $\mathbb{F}$ を入力として，暗号化した文書のコレクションおよび検索用のインデックス (γ) を出力するアルゴリズムである。

ただし，$\mathsf{Enc}(K, \mathbb{F})$ を $\mathsf{Enc}_K(\mathbb{F})$ を表記することもある。

Trapdoor : $(K, w) \mapsto t$： 秘密鍵 K およびキーワード w を入力として，トラップドア t を出力する決定的アルゴリズムである。

前項と同様に $\mathsf{Trapdoor}_K(w)$ と表記することもある。

Search : $(\gamma, t) \mapsto X$： インデックス γ およびトラップドア t を入力として，辞書順に並んだ文書の ID からなるリスト X を出力する決定的アルゴリズムである。

Dec : $(K, c_i) \mapsto f_i$： 秘密鍵 K および暗号化された文書 c_i を入力とし，復号された文書 f_i を出力する決定的アルゴリズムである。

$\mathsf{Dec}(K, c_i)$ を $\mathsf{Dec}_K(c_i)$ と表記する場合もある。

ユーザは自身の持つデータ ($\mathbb{F}$) をクラウドストレージサーバ (以下「サーバ」と称する) へ SSE を利用し格納したいと考えている。このとき，SSE の各アルゴリズムは以下のように実行される。

まず，ユーザは秘密鍵の生成 (Gen) を行う。この鍵 (K) はインデックスの作成時と検索時，そして，データの暗号化及び復号に用いる。ユーザは鍵を用い，サーバへ格納したいデータからインデックス (γ) と呼ばれる検索用の構造を生成 (Enc) する。同時に，鍵を用いファイルそのものも暗号化し，インデックスと一緒にサーバへ格納する。ユーザがファイルの検索を行う場合は，鍵および検索ワード w から，トラップドア t と呼ばれるクエリを作成 (Trapdoor) し，サーバへ送信する。クエリを受け取ったサーバはインデックスとクエリから検索処理 (Search) を行い，検索結果として，暗号化されたファイル群 ($\mathbb{C}_w$) をユーザへ返す。最後に，ユーザが検索結果のファイル群を復号 (Dec) することにより，検索は完了する。

2.3 Dynamic 検索可能暗号方式 (DSSE)

動的インデックスベース SSE 方式 (DSSE) とは，以下の 9 つの多項式時間アルゴリズムの組

$$\mathrm{DSSE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{SrchToken}, \mathsf{AddToken}, \mathsf{DelToken}, \mathsf{Search}, \mathsf{Add}, \mathsf{Del}, \mathsf{Dec})$$

であり，各アルゴリズムは以下のように定義される。ただし，Gen, Enc, Dec については前節と同様である。

SrchToken : $(K, w) \mapsto \tau_s$： 秘密鍵 K およびキーワード w を入力として，検索トークン τ_s を出力する確率的アルゴリズムである。

AddToken : $(K, f) \mapsto (\tau_a, c_f)$： 秘密鍵 K および文書 (ファイル)f を入力として，追加トークン τ_a および追加するファイルを暗号化したもの c_f を出力する確率的アルゴリズムである。

DelToken : $(K, f) \mapsto \tau_d$： 秘密鍵 K および文書 (ファイル)f を入力として，削除トークン τ_d を出力する確率的アルゴリズムである。

Search : $(\gamma, \mathbb{C}, \tau_s) \mapsto I_w$： インデックス γ，暗号文の列 $\mathbb{C}$ および検索トークン τ_s を入力として，ファイル ID からなるリスト I_w を出力する決定的アルゴリズムである。

Add : $(\gamma, \mathbb{C}, \tau_a, c) \mapsto (\gamma', \mathbb{C}')$： インデックス γ，暗号文の列 $\mathbb{C}$，追加トークン τ_a および追加する暗号文 c を入力として，新しいインデックス γ' および新しい暗号文の列 $\mathbb{C}'$ を出力するアルゴリズムである。

Del : $(\gamma, \mathbb{C}, \tau_d) \mapsto (\gamma', \mathbb{C}')$： インデックス γ，暗号文の列 $\mathbb{C}$ および削除トークン τ_d を入力として，新しいインデックス γ' および新しい暗号文の列 $\mathbb{C}'$ を出力するアルゴリズムである。

ユーザは秘密鍵の生成 (Gen) を行う。生成された鍵 K はインデックスの作成時と検索時，データの暗号化及び復号，およびファイルの追加及び削除時に用いる。ユーザは鍵 K を用い，サーバへ格納したいデータからインデックスと呼ばれる検索用の構造を生成 (Enc) する。同時に，鍵を用いファイルそのものも暗号化し，インデックスと共にサーバへ格納する。ユーザがファイルの検索を行う際は，鍵および検索ワードから，検索トークンと呼ばれるクエリを作成 (SrchToken) しサーバへ送信する。クエリを受け取ったサーバはインデックスとクエリから検索処理 (Search) を行い，検索結果として，暗号化されたファイル群をユーザへ返す。最後に，ユーザは検索結果のファイル群を復号 (Dec) することにより，検索を完了する。また，ユーザがファイルの追加，または削除を行う場合は，鍵および追加するファイルから追加トークン，または削除トークンを作成 (AddToken, DelToken) し，サーバへ送信する。トークンを受け取ったサーバはインデックスとトークンから追加 (Add) または削除 (Del) 処理を行う。

3. 既存方式 [2]

本論文では既存技術として Kamara ら [2] が考案した DSSE を取り上げ，本文中でしばしば参照し，表記についても踏襲しているが，その詳細については，スペースの都合上，割愛する。

4. 提案方式

本章では，Kamara らの方法 [2] に対する下記の課題：

- 新規単語の追加問題

- 誤字や表記の揺れの問題

の対応策として，第 4.1 節から第 4.4 節にかけて，

- N-Gram[3] を用いた新規単語を出現させない方法
- N-Gram を用いた誤字や表記の揺れを検索する方法
- レコメンドテーブルを用いて単純に単語をレコメンドする方法
- フォーマット変換をレコメンドテーブルを用いて実現する方法

を述べ，第 4.5 節において提案方式の詳細を述べ，第 4.6 節においてその安全性について述べる。

4.1 新規単語の追加問題に対する対応

既存方式では，テーブルに追加するワード全体の集合 W は予め決まっており，使用状況に応じ新しい単語を追加する場合は，テーブルを一から作りなおす必要がある。しかし，N-Gram の技術を用いることで，この問題を以下のように解決できる。ここで，ワード w の N-Gram とは，w 中の連続する長さ N の部分文字列である。例えば，w を「検索可能暗号」とすれば，その 2-Gram は「検索」「索可」「可能」「能暗」「暗号」の 5 つである。

N-Gram を利用し，新規単語の追加問題に対応する戦略は以下のようなものである。

(1) 使用する文字全体の集合を C(ただし $|C| = \mathsf{poly}(k)$) とする。
(2) 登録する文字全体の集合 W を $W := C^N$ と定義する。
(3) 文書 f をテーブルへ登録する場合，f をひとつの文字列とみなし，その N-Gram を利用する。

つまり，テーブルへ登録されるワードは，文字列の N-Gram として現れる文字列すべて，即ち，長さが N の文字列全体である。これらをすべて登録する。インデックス作成時に，文書 f の情報をテーブルへ登録する方法は，f を 1 つの文字列とみなし，現れた N-Gram に対応するリストへ f(の ID) を格納することになる。

4.2 誤字や表記の揺れへの対応

既存方式では，完全一致によるものしか扱えず，文書中や検索文字列に誤字があった場合，検索が困難になってしまう。そこで，N-Gram を用い登録されていることを仮定し，以下のような変更を加えることによりこの問題を解決できる。なお，表記の揺れへの対応については，次節で述べるレコメンド機能を用い対応する。

インデックスへの追加処理：

テーブル $\mathtt{A}_s$ のノードに暗号化したポジションの情報の列を加える。これにより，ワードが文書のどの位置に出現したかが分かるようになる。なお，ワードの出現回数を漏洩させないために，ポジションの列の長さが固定長になるようダミーデータを加える。このとき，文書中に出てきた各 N-Gram のポジション情報を記憶できる個数の上限を定める。

検索処理：：

(1) 検索語 w を N-Gram のワードに区切る。ここで $w \mapsto (g_1, \ldots, g_m)$ とし，さらに w の長さを ℓ とする。
(2) $g_1, \ldots, g_m$ を従来方式で検索する。このとき，暗号化されたポジション情報を得るので，ユーザは復号する。
(3) $g_1, \ldots, g_m$ が現れた文書およびポジションから，完全一致や曖昧検索を行う。ここで，曖昧検索で合致したと見なす基準はアプリケーションごとに自由に定めることができる。例えば，w の $\lceil m/2 \rceil$ 個以上の異なる N-Gram が 2ℓ の長さ以下の文字列に出現した文書を検索結果に含める，等である。

4.3 レコメンド機能の追加

既存方式にはレコメンド機能は存在しない。ここで，より検索をスムーズに行えるように単語間のレコメンド機能を以下のように追加する。

下準備：

(1) 暗号化されていないレコメンドテーブルを用意する。暗号化されていないレコメンドテーブルとは，ワードとワードのリストの対応表のことである。
(2) レコメンドテーブルの暗号化
   - $\mathtt{T}_s, \mathtt{A}_s$ と同様の構造をもったテーブル $\mathtt{T}_r, \mathtt{A}_r$ を用意する。
   - ワード w でレコメンドされるワード群を $w_1, \ldots, w_n$ としたとき，$\mathtt{A}_r$ にはワード $w_1, \ldots, w_n$ を暗号化したものをランダムな場所にリストとして格納し，$\mathtt{T}_r$ にはワード w のユーザ鍵でのハッシュ値に対応する場所に上記リストの先頭アドレスを格納する。

レコメンドテーブル検索時：

(1) 検索キーワードを w とする。
(2) w のユーザ鍵でのハッシュ値をサーバへ送信する。
(3) サーバはリストを復元，暗号化されたワードのリストをユーザへ返す。
(4) ユーザはリストの各ノードを復号し，レコメンド情報を得る。

4.4 フォーマット変換レコメンド機能の追加

単語間の関連だけではフォーマットの違いに対応できないため，より柔軟なレコメンド機能を実現できるよう，簡易的なフォーマット変換も行えるように拡張する。

下準備：

(1) フォーマット変換で用いるメタ文字の集合を M とする。ただし，M 内の各メタ文字は
   - 単一の文字を表す
   - メタ文字以外からメタ文字への対応が一意になる

   という条件を満たすとする。
(2) 暗号化されていないレコメンドテーブルを用意する。

このとき，検索ワードとレコメンドリストに格納するワードにメタ文字を用いてもよい。

(3) レコメンドテーブルの暗号化

- $\mathtt{T}_s, \mathtt{A}_s$ と同様の構造をもったテーブル $\mathtt{T}_r, \mathtt{A}_r$ を用意する。
- ワード w でレコメンドされるワード群を $w_1, \ldots, w_n$ としたとき，$\mathtt{A}_r$ にはワード $w_1, \ldots, w_n$ を暗号化したものをランダムな場所にリストとして格納し，$\mathtt{T}_r$ にはワード w のユーザ鍵でのハッシュ値に対応する場所に上記リストの先頭アドレスを格納する。

レコメンドテーブル検索時：

(1) 検索キーワードを w とする。

(2) w を可能な限りメタ文字で置き換えた後，w のユーザ鍵でのハッシュ値をサーバへ送信する。

(3) サーバはリストを復元，暗号化されたワードのリストをユーザへ返す。

(4) ユーザはリストを復号し，レコメンド情報を得る。

(5) レコメンド情報にメタ文字が含まれていた場合は置き換えたうえでユーザにレコメンド情報として伝える。

4.5 提案方式

ここでは提案方式の構成を述べる。なお，第 2.3 節で示した関数 (アルゴリズム) の他，曖昧検索処理を行う関数 FuzSearch，レコメンドテーブルの検索クエリを作成する関数 RecSrchToken，および，レコメンドテーブルの検索処理関数 RecSearch を追加している。提案方式において，$\textsc{Ske1} = (\mathsf{Gen1}, \mathsf{Enc1}, \mathsf{Dec1})$ および $\textsc{Ske2} = (\mathsf{Gen2}, \mathsf{Enc2}, \mathsf{Dec2})$ を適切な対称鍵暗号方式とする。また，$F : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$，$G : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$, $P : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$, および $Q : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^*$ を擬似ランダム関数とし，$H_1 : \{0,1\}^* \to \{0,1\}^*$ および $H_2 : \{0,1\}^* \to \{0,1\}^*$ をランダムオラクルとする。$z \in \mathbb{N}$ をフリーリストの初期サイズとする。使用できる通常文字の有限集合を Σ とする (このとき $W \subset \Sigma^*$ である)。s を 1 つの文書に現れることが可能な同じ N-Gram 語の個数の最大値とする。メタ文字の集合を Ω とする。ただし，Σ の任意の文字に対応する Ω のメタ文字は多くとも 1 つとする。

我々が提案するインデックスベース DSSE 方式：

$$\begin{aligned}\mathrm{DSSE} \;=\; (&\mathsf{Gen}, \mathsf{Enc}, \mathsf{SrchToken}, \mathsf{AddToken},\\ &\mathsf{DelToken}, \mathsf{Search}, \mathsf{Add}, \mathsf{Del}, \mathsf{Dec},\\ &\mathsf{FuzSearch}, \mathsf{RecSrchToken}, \mathsf{RecSearch})\end{aligned}$$

は以下のように構成される。

Gen(1^k)：

(i) $K_1 \xleftarrow{\$} \{0,1\}^k$, $K_2 \xleftarrow{\$} \{0,1\}^k$, $K_3 \xleftarrow{\$} \{0,1\}^k$, $K_4 \leftarrow \textsc{Ske1}.\mathsf{Gen1}(1^k)$, $K_5 \xleftarrow{\$} \{0,1\}^k$, $K_6 \leftarrow \textsc{Ske2}.\mathsf{Gen2}(1^k)$

(ii) $K = (K_1, K_2, K_3, K_4, K_5, K_6)$ を出力する。

Enc$(K, \mathbb{F})$：

(i) 準備：

(a) $\mathtt{A}_s, \mathtt{A}_d$ をサイズ $\#\mathbb{C}/8 + z$ の配列とする。

(b) $\mathtt{T}_s, \mathtt{T}_d$ を，それぞれ，サイズ $\#\Sigma^N + 1$，$\#\mathbb{F}$ の連想配列とする。

(c) $\mathtt{T}_r$ をサイズ t の連想配列とする。ここで，t はレコメンドを機能させたい有限個の語 $w \in (\Sigma \cup \Omega)^*$ の数である。

(d) $\mathtt{A}_r$ をサイズ $10t$ の配列とする。

(e) $\mathbf{0}$ で $(\log \#\mathtt{A}_s)$ 個の 0 の列を表すとする。

(f) *free* を W に現れていない文字列とする。

(ii) 検索用のリストを作成：

各 N-Gram $g \in \Sigma^N$ に対し，以下を実行する。

(a) 配列 $\mathtt{A}_s$ にランダムに格納された $\#\mathbb{F}_g$ 個のノード $(\mathtt{N}_1, \ldots, \mathtt{N}_{\#\mathbb{F}_g})$ からなるリスト $\mathtt{L}_g$ を構成する：

$$\begin{aligned}\mathtt{N}_i := (&\langle \mathsf{id}_i, \mathrm{addr}_s(\mathtt{N}_{i+1}),\\ &\mathsf{Enc2}_{K_6}(\langle Q_{K_5}(\mathrm{pos}_{i,g,1}), \ldots, Q_{K_5}(\mathrm{pos}_{i,g,s})\rangle)\rangle\\ &\oplus H_1(K_w, r_i), r_i)\end{aligned}$$

と定義する。ここで, id_i は, $\mathbb{F}_g$ の i 番目のファイルの ID, $\mathrm{pos}_{i,g,j}$ は, $\mathbb{F}_g$ の i 番目のファイル中での g の j 回目の出現位置，$r_i \xleftarrow{\$} \{0,1\}^k$，$K_w \leftarrow P_{K_3}(w)$，$\mathrm{addr}_s(\mathtt{N}_{\#\mathbb{F}_w+1}) = \mathbf{0}$，$\mathrm{pos}_{i,g,j}$ が存在しない場合はランダムなデータで埋める，とする。

(b) 検索テーブル $\mathtt{T}_s$ へ $\mathtt{L}_g$ の先頭ノードへのポインタを格納する：

$$\mathtt{T}_s[F_{K_1}(g)] \leftarrow \langle \mathrm{addr}_s(\mathtt{N}_1), \mathrm{addr}_d(\mathtt{N}_1^*)\rangle \oplus G_{K_2}(g)$$

ただし，$\mathtt{N}_1^*$ は $\mathtt{N}_1$ の対，つまり，$\mathtt{A}_d$ のノードであり，その 4 番目のエントリーポイントが $\mathtt{A}_s$ のノード $\mathtt{N}_1$ を指しているもの，とする。

(iii) 削除用のリストを作成：

各ファイル $f \in \mathbb{F}$ に対して以下を実行する。

(a) 削除配列 $\mathtt{A}_d$ にランダムに格納された $\#f$ 個の対ノード $(\mathtt{D}_1, \ldots, \mathtt{D}_{\#f})$ からなるリスト $\mathtt{L}_f$ を構成する：

- 各エントリ $\mathtt{D}_i$ は N-Gram g と関連付けられているため，$\mathtt{L}_g$ のあるノード $\mathtt{N}$ と関連付けられている。
- $\mathtt{N}_{+1}$ は, $\mathtt{L}_w$ 内における $\mathtt{N}$ の次のノードを表す。
- $\mathtt{N}_{-1}$ は, $\mathtt{L}_w$ 内における $\mathtt{N}$ の前のノードを表す。

このとき $\mathtt{D}_i$ を

$$\begin{aligned}\mathtt{D}_i := (&\langle \mathrm{addr}_d(\mathtt{D}_{i+1}), \mathrm{addr}_d(\mathtt{N}_{-1}^*), \mathrm{addr}_d(\mathtt{N}_{+1}^*),\\ &\mathrm{addr}_s(\mathtt{N}), \mathrm{addr}_s(\mathtt{N}_{-1}), \mathrm{addr}_s(\mathtt{N}_{+1}), F_{K_1}(g)\rangle\\ &\oplus H_2(K_f, r_i'), r_i')\end{aligned}$$

と定義する。ここで，r_i' は k ビットのランダムな列，$K_f \leftarrow P_{K_3}(f)$，$\mathrm{addr}_d(\mathtt{N}_{\#\bar{f}+1}) = \mathbf{0}$ とする。

(b) 削除テーブル $\mathtt{T}_d$ へ $\mathtt{L}_f$ の先頭ノードへのポインタを格納する：

$$\mathtt{T}_d[F_{K_1}(f)] \leftarrow \mathrm{addr}_d(\mathtt{D}_1) \oplus G_{K_2}(f)$$

(iv) レコメンド用のリストを作成：
メタ文字を用いても良い，レコメンドを機能させたい s 個の各語 $w \in (\Sigma \cup \Omega)^*$ に対して以下の (a)〜(c) を実行する。
(a) 配列 $\mathtt{A}_r$ にランダムに格納された w_r 個のノード $(\mathtt{R}_1, \ldots, \mathtt{R}_{w_r})$ からなるリスト $\mathtt{L}_w$ を構成する：
$$\mathtt{R}_i \leftarrow (\langle \mathsf{Enc1}_{K_4}(w_i), \mathrm{addr}_r(\mathtt{R}_{i+1}) \rangle \oplus H_1(K_w, r_i), r_i)$$
と定義する。ここで $r_i \xleftarrow{\$} \{0,1\}^k$, $K_w \leftarrow P_{K_3}(w)$, $\mathrm{addr}_r(\mathtt{R}_{w_r+1}) = \mathbf{0}$ とする。
(b) レコメンドテーブル $\mathtt{T}_r$ へ $\mathtt{L}_w$ の先頭ノードへのポインタを格納する：
$$\mathtt{T}_r[F_{K_1}(w)] \leftarrow \mathrm{addr}_r(\mathtt{R}_1) \oplus G_{K_2}(w)$$
(c) 配列 $\mathtt{A}_r$ の余った要素をランダムな文字列で埋める。
(v) 暗号化されていないフリーリスト $\mathtt{L}_{free}$ を作る：
$(\mathtt{F}_1, \ldots, \mathtt{F}_z)$ および $(\mathtt{F}'_1, \ldots, \mathtt{F}'_z)$ を，それぞれ，$\mathtt{A}_s$ および $\mathtt{A}_d$ からランダムに選んだ，値が格納されていない z 個のセルとする。
(a) 検索テーブル $\mathtt{T}_s$ にフリーリストの先頭の情報を格納する：
$$\mathtt{T}_s[free] \leftarrow \langle \mathrm{addr}_s(F_z), 0^{\log \#\mathtt{A}} \rangle$$
(b) 検索配列 $\mathtt{A}_s$ へノードの情報を格納する：
$\mathtt{A}_s[\mathrm{addr}_s(\mathtt{F}_i)] \leftarrow \langle 0^{\log \#\mathbb{F}}, \mathrm{addr}_s(\mathtt{F}_{i-1}), \mathrm{addr}_d(\mathtt{F}'_i) \rangle$ とする。ただし，$\mathrm{addr}_s(\mathtt{F}_0) = 0^{\log \#\mathtt{A}}$ とする。
(vi) $\mathtt{A}_s, \mathtt{A}_d$ の値が格納されていないセルをランダムな文字列で埋める。
(vii) ファイルを暗号化する。
$1 \leqq i \leqq \#\mathbb{F}$ に対して $c_i \leftarrow \textsc{Ske1}.\mathsf{Enc1}_{K_4}(f_i)$ を実行する。
(viii) $(\gamma, \mathbb{C})$ を出力する。ここで，$\gamma \leftarrow (\mathtt{A}_s, \mathtt{T}_s, \mathtt{A}_d, \mathtt{T}_d)$ および $\mathbb{C} \leftarrow (c_1, \ldots, c_{\#\mathbb{F}})$ とする。

SrchToken(K, w)：
(i) w を N-Gram へ分割し，それを $(g_1, \ldots, g_m)$ とする。
(ii) $1 \leqq i \leqq m$ に対して $\tau_{s_i} \leftarrow (F_{K_1}(g_i), G_{K_2}(g_i), P_{K_3}(g_i))$ とする。
(iii) $(\tau_{s_1}, \ldots, \tau_{s_m})$ を出力する。

Search$(\gamma, \mathbb{C}, \{\tau_{s_1}, \ldots, \tau_{s_m}\})$：
サーバ側の処理は以下の通りである。
(i) 各 τ_{s_i} に対して以下を実行する。
(a) 引数をチェックする。
(i) τ_{s_i} を $(\tau_{i,1}, \tau_{i,2}, \tau_{i,3})$ のように分解する。
(ii) $\mathtt{T}_s$ に $\tau_{i,1}$ が存在しない場合は空リストを返す。
(b) 検索テーブルのエントリポイントを探す：
$$(\alpha_1, \alpha'_1) \leftarrow \mathtt{T}_s[\tau_{i,1}] \oplus \tau_{i,2}$$
(c) $\mathtt{N}_1 \leftarrow \mathtt{A}_s[\alpha_1]$ とし，$\tau_{i,3}$ で復号する。つまり，$\mathtt{N}_1$ を (ν_1, r_1) として，$\nu_1 \oplus H_1(\tau_{i,3}, r_1)$ を計算し，これを $\mathrm{info}_{i,1}$ とする。実際，これは
$$(\mathsf{id}_1, \mathrm{addr}_s(\mathtt{N}_2), \mathsf{Enc2}_{K_6}(\langle Q_{K_5}(\mathrm{pos}_{1,g,1}), \ldots, Q_{K_5}(\mathrm{pos}_{i,g,s}) \rangle))$$
に等しい。
(d) $j \geqq 2$ に対して，$\alpha_{j+1} = \mathbf{0}$ となるまで，前項と同様に $\mathtt{N}_i$ を復号する。
(ii) すべての $\tau_{s_i} (1 \leqq i \leqq m)$ の検索結果に存在する ID とそのその出現情報を，どの τ_{s_i} の検索結果であったかを区別できる形式で出力する。

ユーザ側の処理は以下の通りである。
(i) ポジションの情報を復号する。
(ii) $\mathrm{info}_{i,j}$ を用いて，すべての $\mathrm{info}_{i,*} (1 \leqq i \leqq m)$ に現れ，かつ，$\mathrm{info}_{i,*}$ から $\mathrm{info}_{m,*}$ まで，pos の情報が 1 つずつ増加しているものが存在する ID をすべて出力する。

FuzSearch$(\gamma, \mathbb{C}, \{\tau_{s_1}, \ldots, \tau_{s_m}\})$：
Search$(\gamma, \mathbb{C}, \{\tau_{s_1}, \ldots, \tau_{s_m}\})$ とほぼ同様であり，違いは，ユーザ側の処理の (ii) である。$\mathrm{info}_{i,j}$ を用いて曖昧検索を実施するが，どのような条件を定めるかは柔軟に設定できる。第 5 章においてその構成例を述べる。

RecSrchToken(K, w)：
(i) w の各文字を可能な限りメタ文字で置き換える。
(ii) $\tau_r \leftarrow (F_{K_1}(w), G_{K_2}(w), P_{K_3}(w))$ を出力する。

RecSearch$(\gamma, \mathbb{C}, \tau_r)$：
(i) 引数のチェックを行う：τ_r を (τ_1, τ_2, τ_3) と分解し，$\mathtt{T}_r$ に τ_1 が存在しない場合は空リストを返す。
(ii) レコメンドテーブルのエントリポイントを探す：
$$(\alpha_1, \alpha'_1) \leftarrow \mathtt{T}_r[\tau_1] \oplus \tau_2$$
(iii) $\mathtt{R}_1 \leftarrow \mathtt{A}_r[\alpha_1]$ として τ_3 で復号する。つまり，$\mathtt{R}_1$ を $(\nu_1.r_1)$ と分解し，$\nu_1 \oplus H_1(\tau_3, r_1)$ を計算する。実際，これは $(\mathsf{Enc1}_{K_4}(w_1), \mathrm{addr}_r(\mathtt{R}_2))$ に等しい。
(iv) $i \geqq 2$ に対して，$\alpha_{i+1} = \mathbf{0}$ となるまで，前項と同様に $\mathtt{R}_i$ を復号する。
(v) $\{\mathsf{Enc1}_{K_4}(w_1), \ldots, \mathsf{Enc1}_{K_4}(w_r)\}$ を出力する。

AddToken(K, f)：
(i) 準備：
$(g_1, \ldots, g_{\#\bar{f}})$ を f 中の一意的な N-Gram の列とし，初出現順は f と変わらないものとする。また，
$$\tau_a \leftarrow (F_{K_1}(f), G_{K_2}(f), \lambda_1, \ldots, \lambda_{\#f})$$
とする。ただし，$1 \leqq i \leqq \#f$ に対して，$r_i \xleftarrow{\$} \{0,1\}^k$, $r'_i \xleftarrow{\$} \{0,1\}^k$ であり，さらに，
$$\lambda_i \leftarrow (F_{K_1}(g_i), G_{K_2}(g_i), \langle \mathsf{id}(f), \mathbf{0}, \mathsf{Enc2}_{K_6}(\langle Q_{K_5}(\mathrm{pos}_{g_i,1}), \ldots, Q_{K_5}(\mathrm{pos}_{g_i,s}) \rangle) \rangle \oplus H_1(P_{K_3}(g_i), r_i), r_i, \langle \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, F_{K_1}(g_i) \rangle \oplus H_2(P_{K_3}(f), r'_i), r'_i)$$
であるとする。
(ii) $c_f \leftarrow \textsc{Ske1}.\mathsf{Enc1}_{K_4}(f)$
(iii) (τ_a, c_f) を出力する。

Add(γ, c, τ_a)：

(i) τ_a を $(\tau_1, \tau_2, \lambda_1, \ldots, \lambda_{\#f})$ と分解し，$\mathtt{T}_d$ に τ_1 が存在しない場合は $\perp$ を返す。(引数のチェック)

(ii) インデックスの更新を行う。具体的には，$1 \leqq i \leqq \#\bar{f}$ に対して以下を実行する。

(a) フリーリストの最後尾 φ および対応する削除配列のエントリ φ^* を探す。つまり，$(\varphi, \mathbf{0}) \leftarrow \mathtt{T}_s[\mathit{free}]$，$(\varphi_{-1}, \varphi^*) \leftarrow \mathtt{A}_s[\varphi]$ とする。

(b) 検索テーブルのフリーエントリを最後から 1 つ手前にセットする：$\mathtt{T}_s[\mathit{free}] \leftarrow (\varphi_{-1}, \mathbf{0})$

(c) 1 つめのノード $\mathtt{N}_1$ へのポインタを復元する：$(\alpha_1, \alpha_1^*) \leftarrow \mathtt{T}_s[\lambda_i[1]] \oplus \lambda_i[2]$

(d) 新しいノードを φ へ格納し，ポインタを $\mathtt{N}_1$ を指すようにする：$\mathtt{A}_s[\varphi] \leftarrow (\lambda_i[3] \oplus \langle \mathbf{0}, \alpha_1, 0, \ldots, 0\rangle, \lambda_i[4])$

(e) 検索テーブルを更新する：$\mathtt{T}_s[\lambda_i[1]] \leftarrow (\varphi, \varphi^*) \oplus \lambda_i[2]$

(f) $\mathtt{N}_1$ の対を更新する：$(\mathtt{D}_1, r) \leftarrow \mathtt{A}_d[\alpha_1^*]$，および，$\mathtt{A}_d[\alpha_1^*] \leftarrow (\mathtt{D}_1 \oplus \langle \mathbf{0}, \varphi^*, \mathbf{0}, \mathbf{0}, \varphi, \mathbf{0}, \mathbf{0}\rangle, r)$

(g) $\mathtt{A}_s[\varphi]$ の対を更新する：$\mathtt{A}_d[\varphi^*] \leftarrow (\lambda_i[5] \oplus \langle \varphi_{-1}^*, \mathbf{0}, \alpha_1^*, \varphi, \mathbf{0}, \alpha_1, \lambda_i[1]\rangle, \lambda_i[6])$

(h) $i = 1$ ならば，削除テーブルを更新する：$\mathtt{T}_d[\tau_1] \leftarrow \langle \varphi^*, \mathbf{0}\rangle \oplus \tau_2$

(iii) ファイルを追加する: $\mathbb{C} \leftarrow \mathbb{C} \cup \{c\}$

DelToken(K, f)：

(i) $\tau_d \leftarrow (F_{K_1}(f), G_{K_2}(f), P_{K_3}(f), \mathsf{id}(f))$ を出力する。

Del$(\gamma, \mathbb{C}, \tau_d)$：

(i) τ_d を $(\tau_1, \tau_2, \tau_3, \mathsf{id})$ と分解し，$\mathtt{T}_d$ に τ_1 のエントリがなければ $\perp$ を返す。(引数のチェック)

(ii) $\mathtt{L}_f$ の先頭ノードを求める：$\alpha_1' \leftarrow \mathtt{T}_d[\tau_1] \oplus \tau_2$

(iii) $1 \leqq i \leqq \#\bar{f}$ に対して以下を実行する。

(a) $\mathtt{D}_i$ を復号する：$(\mathtt{D}_i, r) \leftarrow \mathtt{A}_d[\alpha_i']$，$(\alpha_1, \ldots, \alpha_6, \mu) \leftarrow \mathtt{D}_i \oplus H_2(\tau_3, r)$

(b) $\mathtt{D}_i$ を削除する：$\mathtt{A}_d[\alpha_i'] \xleftarrow{\$} \{0,1\}^{6\log \#\mathtt{A}+k}$

(c) フリーリストの最後のノードを探す：$(\varphi, 0^{\log \#\mathtt{A}}) \leftarrow \mathtt{T}_s[\mathit{free}]$

(d) 検索テーブルのフリーエントリを $\mathtt{D}_i$ の対を指すようにする：$\mathtt{T}_s[\mathit{free}] \leftarrow \langle \alpha_4, 0^{\log \#\mathtt{A}}\rangle$

(e) $\mathtt{D}_i$ の対の場所を解放する：$\mathtt{A}_s[\alpha_4] \leftarrow (\varphi, \alpha_i', 0, \ldots, 0)$

(f) 前方ノードの処理を行う：$\mathtt{N}_{-1}$ を $\mathtt{D}_i$ の対の前のノードとする。このとき，

$(\beta_1, \beta_2, \gamma_1, \ldots, \gamma_s, r_{-1}) \leftarrow \mathtt{A}_s[\alpha_5]$;

$\mathtt{A}_s[\alpha_5] \leftarrow (\beta_1, \beta_2 \oplus \alpha_4 \oplus \alpha_6, \gamma_1, \ldots, \gamma_s, r_{-1})$

として，$\mathtt{N}_1$ の次を指すポインタを更新し，

$(\beta_1, \ldots, \beta_6, \mu^*, r_{-1}^*) \leftarrow \mathtt{A}_d[\alpha_2]$;

$\mathtt{A}_d[\alpha_2] \leftarrow (\beta_1, \beta_2, \beta_3 \oplus \alpha_i' \oplus \alpha_3, \beta_4, \beta_5, \beta_6 \oplus \alpha_4 \oplus \alpha_6, \mu^*, r_{-1}^*)$

として，$\mathtt{N}_{-1}$ の対 ($\mathtt{D}_i$ の前のノード) を更新する。

(g) 後方ノードの処理を行う：$\mathtt{N}_{+1}$ を $\mathtt{D}_i$ の対の次のノードとする。このとき，

$(\beta_1, \ldots, \beta_6, \mu^*, r_{+1}^*) \leftarrow \mathtt{A}_d[\alpha_3]$;

$\mathtt{A}_d[\alpha_3] \leftarrow (\beta_1, \beta_2 \oplus \alpha_i' \oplus \alpha_2, \beta_3, \beta_4, \beta_5 \oplus \alpha_4 \oplus \alpha_5, \beta_6, \mu^*, r_{+1}^*)$

として，$\mathtt{N}_{+1}$ の対のポインタを更新する。

(h) $\alpha_{i+1}' \leftarrow \alpha_1$ とする。

(iv) id に一致する暗号文を $\mathbb{C}$ から削除する。

(v) $\mathtt{T}_d$ から τ_1 を削除する。

Dec(K, c)：

(i) K を $(K_1, \ldots, K_6)$ のように分解する。

(ii) $m \leftarrow \textsc{Ske1}.\mathsf{Dec1}_{K_4}(c)$ を出力する。

4.6 提案方式の安全性

スペースの都合上，概略のみ述べる。安全性に関する表記方法は [2] と同様である。提案方式の安全性に関しては，ノードにポジション情報を含めた点とレコメンドテーブルを利用した点に関して，別々に述べる。まず，ノードにポジション情報を含めた点に関しては以下が示される。

定理 1 Ske1 および Ske2 が CPA-安全であり，かつ，F, G, P, Q が擬似ランダム関数であるならば，ランダムオラクルモデルの下で，提案 DSSE 方式は，適応的選択キーワード攻撃にたいして $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4)$-安全である。 □

この定理は背理法, 即ち, 提案方式が適応的選択キーワード攻撃に対し, ランダムオラクルモデルの下で $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4)$-安全でなく，かつ，Ske1 が CPA-安全であり，さらに，F, G, P, Q が擬似ランダムであるならば，Ske2 が CPA-安全でなくなることを示すことができる。実際，その証明はシミュレータ $\mathcal{S}$ を構成することによる。$\mathcal{S}$ の構成については Kamara ら [2] で示されているものと同様であるが，Ske2 を利用する箇所，つまりノード $\mathtt{N}_i$ に関するシミュレートを行う部分に関しては，ノード情報に Ske2 の暗号文空間からランダムに取った暗号文 c を追加する。ここで仮定より，提案方式が安全でないこと，および，Kamara ら [2] の方式が安全であることが示されていることより，

$$\mathsf{Enc2}_{K_6}(\langle Q_{K_5}(\mathrm{pos}_{i,g,1}), \ldots, Q_{K_5}(\mathrm{pos}_{i,g,s})\rangle)$$

とランダムな暗号文 c を無視できない確率で識別できる攻撃者 $\mathcal{A}$ が存在する ($\mathcal{A}$ はポジション情報を任意に指定可能)。従って，$\mathcal{A}$ は $\mathsf{Enc2}_{K_6}(m)$ とランダムな暗号文 c を無視できない確率で識別できる (m は $\mathcal{A}$ が任意に指定した平文)。つまり Ske2 は CPA-安全ではなくなる。

次に，レコメンドに関する安全性を考察する。レコメンド機能に用いた構造は Curtmola ら [1] の SSE-1 と同様であり，従って，安全性もそれと同等であることが示される。

5. 考察

本章では，提案方式に対する以下の考察を行う。

変更に伴う問題点： 提案方式で Kamara ら [2] の方式から変更した点による不都合な点を述べる。

曖昧検索： N-Gram を用いた具体的な曖昧検索の方式と具体例を示す。

上記の他，レコメンドテーブルの活用法などを述べる必要があるが，紙面?の都合上割愛する。

5.1 変更に伴う問題点

ここでは，変更点に伴う問題点をいくつか挙げる。

長さ N 未満の文字列を検索できなくなる：

長さ N 未満の文字列は N-Gram に分けることが不可能なため，検索ができない。そのため，N は大きくし過ぎないほうが良い [*4]。

検索結果に間違いが生ずる：

検索結果に非常にまれにだがミスが出現することがある。なぜなら，ポジション情報は暗号化されたものとランダムなダミーデータに分かれるが，ダミーデータが偶然復号でき，他のポジション情報に対して検索結果に含めるような条件の値になっているかもしれないからである。しかし, この現象が起こることは非常に稀であり無視できる。実際，$Q_{K_5}(\mathrm{pos}_{i,g,1}),\ldots,Q_{K_5}(\mathrm{pos}_{i,g,j})$ $(j \leqq s)$ が偶然復号でき，かつ，それが他のポジション情報に対し検索結果へ含めるような条件の値になる確率は，まず，検索語 w の長さに関しては，その長さが N の場合 (つまり，ポジション情報が復号できた時点で検索結果へ含めることになる場合) が最高である。また，ポジション情報に関しては，$Q_{K_5}(\mathrm{pos}_{i,g,2}),\ldots,Q_{K_5}(\mathrm{pos}_{i,g,s})$ がダミーデータの場合には，検索結果に間違いが生じるのは $Q_{K_5}(\mathrm{pos}_{i,g,2}),\ldots,Q_{K_5}(\mathrm{pos}_{i,g,j})$ $(j \leqq s)$ がすべて復号できる場合である。このとき，その確率は $j=2$ の場合が最高である。以上より，ダミーデータにより検索結果へ間違えが生じる確率は $Q_{K_5}(\mathrm{pos}_{i,g,2})$ が偶然，ランダムに選んだデータと一致する確率を超えない。Q が擬似ランダムであるなら，その確率は無視できる。

その他：

上記以外にも，実行効率が低下する，レコメンドテーブルのサイズを予め決める必要がある，レコメンドテーブルが重複する，ユーザ負担が増加する，などの問題が挙げられるが，その詳細については割愛する。

5.2 曖昧検索

曖昧検索の構成として，具体的な例を示す。

具体的な曖昧検索の構成例：

具体的な条件として

- 長さ ℓ の検索語 w を m 個の N-Gram $(g_1,\ldots,g_m)$ に分けたとき，w の $\lceil m/2 \rceil$ 個以上の異なる N-Gram が 2ℓ の長さ以下の文字列に出現した場合，f を検索結果に含める

などが考えられる。具体的にこの条件において，$N=2$ とした場合，$w=$ “検索可能暗号” という検索語に対して，その N-Gram は $g_1=$ “検索”，$g_2=$ “索可”，$g_3=$ “可能”，$g_4=$ “能暗”，$g_5=$ “暗号” である。このとき，文字が欠落している「検索可暗号」という文字が入った文書は検索結果に含まれる。なぜなら，「検索可暗号」という文字列中には，g_1, g_2, g_5 の $3 \geqq \lceil 5/2 \rceil$ 個の 2-Gram が出現し，その文字列長は $5 \leqq 2 \times 6$ であるからである。その他，“検索可能な暗号”，“可能検索暗号” や “検索不能暗号” のような文字列を含む文書も，検索結果に含まれる。一方，“検索できる暗号” の場合，出現する 2-Gram は g_1, g_5 の 2 つであり，$2 < \lceil 5/2 \rceil$ であることから，検索結果には含まれない。

6. まとめ

現在，クラウドストレージの発達などに伴い，SSE の技術に対する研究は急速に進んでいる。しかし，検索にかかるオーダが準線形であること，適応的な選択キーワード攻撃に対し安全であること，大きすぎないインデックスサイズ，およびインデックスに対するファイルの動的な追加・削除が可能であること，といった条件を満たすものは多くない。Kamara ら [2] の方式はこれらを満たした方式であったが，より実用に近づけるために，曖昧検索の技術やレコメンド，オートコンプリート等の技術を実現する必要がある。本論文では，N-Gram を用いた曖昧検索およびレコメンド機能の実現方法を述べた。今後の課題として，サーバの負担軽減，具体的には，インデックスサイズの縮小が挙げられる。提案方式では，レコメンドテーブルをユーザの数だけ用意する必要があったが，レコメンドテーブルは 1 つで十分であり，サーバごとに 1 つのレコメンドテーブルを用いたレコメンド機能を実装すべきと思われる。また，レコメンドそのものの機能に関しても，単語やフォーマット間だけでなく，ユーザが入力する文章から残りの検索内容をレコメンドできるようにすることが望まれる。

[*4] N の増加に伴って N-Gram の数も指数関数的に増加するので，その理由からも N は大きくしない方がよいであろう。

参考文献

[1] R. Curtmola, J. Garay, S. Kamara and R. Ostrovsky: “*Searchable symmetric encryption: Improved definitions and efficiet constructions*”, Proceedings of the 13th ACM conference on Computer and Communications Security (CCS), pp.79-88, 2006.

[2] S. Kamara, C. Papamanthou and T. Roeder: “*Dynamic Searchable Symmetric Encryption*”, Proceedings of The 2012 ACM conference on Computer and Communications Security (CCS), pp.965-976, 2012.

[3] C. E. Shannon: “*A mathematical theory of communication*”, Bell System Technical Journal, vol.27, pp.379-423, 623-656, 1948.

[4] D. Song, D. Wagner and A. Perrig: “*Practical techniques for searches on encrypted data*”, In IEEE Symposium on Research in Security and Privacy, pp.44-55. IEEE Computer Society, 2000.