Computer Security Symposium 2016
11 - 13 October 2016

単純なデータ構造を用いた検索可能暗号

三好 竜司[1]　山本 博章[1]

概要：情報セキュリティの観点から，暗号化したデータを効率的に検索する暗号化検索法の開発が進められている．効率的な検索を実現するため，検索対象のデータから暗号化索引を構成することが一般的である．暗号化索引に対し，ブルームフィルタを用いた手法，転置索引を用いた手法が提案されている．また，安全性に対し，適応的，非適応的セキュリティモデルが提案されているが，一般に，適応的安全性を満たすようなシステムは，暗号化索引のサイズが大きくなってしまう．本論文では，転置索引を単純な配列で実現した効率的な手法を提案する．

キーワード：検索可能暗号，ブルームフィルタ，転置索引，共通鍵暗号方式

Symmetric Searchable Encryption Using Simple Data Structure

Ryuuji Miyoshi[1]　Hiroaki Yamamoto[1]

Abstract: From a view point of information security, researches on an encrypted search system have been done intensively. Such search systems are called symmetric searchable encryption (SSE). The main part of SSE is an encrypted index which affects security and efficiency. the existing SSEs use a Bloom filter or an inverted index to construct an encrypted index. Although a Bloom filter is simple data structure because it is a bit sequence, an inverted index is implemented by an advanced data structure such as hash or dictionary. In this paper, we propose a new secure search scheme using Bloom filters and simple arrays, that is, an array of integers and an array of strings. We can construct an efficient SSE because the proposed method is built using only simple data structures.

Keywords: symmetric searchable encryption，Bloom filter, inverted index

1. はじめに

クラウドコンピューティングが普及する中，ストレージサービス，メールサービスなど多くのサービスがネットワークを通して行なわれるようになってきた．このようなサービスでは，外部サーバに自分の情報を保存しなければならない．したがって，サーバ管理者及び第 3 者に内容を知られずに検索するには，データを暗号化して保存し，暗号化したまま検索する技術が要求される．そのため，暗号化データに対する効率的な検索手法に関する研究が活発に行なわれてきた [3], [4], [7], [8], [9], [11], [12], [13], [14], [15], [16], [17], [18]．このような暗号化データを検索する技術は，検索可能暗号と呼ばれている．暗号方式として，共通鍵暗号方式または公開鍵暗号方式を用いた手法が研究されているが，ここでは共通鍵暗号方式を用いた手法を提案する．本手法は，クライアントがデータ保有者で，データを暗号化してサーバに保存する．サーバは，クライアントから暗号化キーワードを受け取ると，暗号化したまま検索を行い，結果をクライアントに返す．このように，サーバはその暗号化キーワードと検索結果を知ることはできるが，内容はすべて暗号化されているため，それ以外の情報を知ることはできない．本論文の設定では，サーバはアルゴリズム通り検索を実施するとする．このような安全性の設定は，キーワード検索においてしばしば用いられている（例えば，文献 [4], [7], [8], [15], [18] など）．本論文の目的は，より単純なデータ構造からなる暗号化索引を開発し，

[1] 信州大学
Shinshu University

効率的な検索可能暗号方式を提案することである．

1.1 本論文の結果

提案する暗号化索引構造は，ブルームフィルタ（Bloom filter）を用いて構成する．ブルームフィルタは，Bloom [1] によって考案されたデータ構造で，集合に対し，ある要素がその集合に入っているかどうかを効率よく検査することができる．欠点としては，集合にない要素をあると判定する誤りが発生することである．ブルームフィルタの概要と応用については，Broder と Mitzenmacher [2] の中でもよくまとめられている．本論文では，ブルームフィルタと転置索引を組み合わせた手法を提案する．従来の転置索引を用いた手法では，基本的に，暗号化されたキーワードとそのキーワードに一致するドキュメント ID のリストで構成される．そのため，データ構造として単純な配列を使おうとするとサイズが大きくなりすぎる．そこで我々は，各キーワードにランダム順で順番に番号 (キーワード ID) を割り振り，ブルームフィルタによって，キーワードの ID を求める関数を実現する．これによって，転置索引はキーワード ID を添え字とする配列で実現できる．提案法は，以下の性質を持つ．

- 暗号化索引に登録するキーワード数を m としたとき，任意のクエリ q に対し，検索は $O(\log m + n_q)$ 時間で行なうことができる．ここで，n_q を q にマッチするドキュメント数である．
- 暗号化索引のサイズは，$O(n \times s_{max})$ である．ここで，n はドキュメント数，s_{max} は，最大のドキュメントに含まれるキーワード数である．
- 提案法は適応的安全である．

1.2 関連研究

Goh [7] は，最初にブルームフィルタを用いた検索法を提案した．これは，空間効率はいいが，検索に時間がかかる．山本他 [20] は，ドキュメントを 2 分木構造で管理するという考えを発展させ，各階層型ブルームフィルタを用いた効率的な検索法提案した．Suga [18] は，ブルームフィルタにキーワードの文字と位置情報を登録することにより，部分一致もできる手法を提案した．Curtomola [6] は，従来のセキュリティモデルの欠点を指摘し，新たに非適応的，適応的安全性に関するセキュリティモデルを導入した．彼らは，これらのモデルに基づいて，非適応的安全な検索可能暗号 (SSE-1) 及び適応的安全な検索可能暗号方式 (SSE-2) を提案した．Curtomola らのモデルを用いるならば，上記のブルームフィルタに基づいた検索法は，非適応的である．Curtomola らの手法は，転置索引に基づいている．転置索引は，（トラップドア，ドキュメント ID）のペアから構成される．これを，トラップドアをインデックスとした単純な配列で構成しようとすると，巨大な配列が必要となる．そのため，ハッシュ，ヒープなどのより高度なデータ構造を利用した実装が必要になる．Kamara [13] は，Curtomola らの手法を拡張し，データの更新機能を備えた適応的安全な手法を提案した．Kurosawa ら [11] も，Curtomola らの手法を発展させ，UC 安全な手法を提案している．我々は，転置索引を単純な配列で実装する手法を提案する．これにより，（トラップドア，ドキュメント ID）の保存に対し，ドキュメント ID を保存する配列を構成するだけで済む．転置索引を単純な配列で実現できることで，空間的，時間的に効率的なシステムを実現できる．提案法は，高野と山本の手法 [19] を発展させたものである．欠点として，提案法はクライアント・サーバ間の通信を 2 回行う．Curtomola や他の手法は 1 回である．表 1 に関連する手法の比較を示す．提案法の索引サイズは，Curtomola らの手法 (SSE-2) とオーダーで同じであるが，配列にドキュメント ID だけを格納すればよいため，オーダーの定数分がかなり小さくなる．また，検索時間についても，$O(\log m + n_q)$ であるが，実際，単純に配列からデータを取り出すだけなので，高速な検索が可能である．Kamara らの索引サイズは，適応的でありながら，SSE-1 と同等のオーダーを実現している．しかし，彼らはランダムオラクルを利用しており，これがかなり有効に働いているように見える．

論文の構成：本論文の構成は次のようになっている．以下，2 章で検索システムの概要，3 章でブルームフィルタの概要，4 章で提案法について述べ，5 章で提案法のセキュリティについて述べる．6 章で実験結果について述べ，7 章はまとめである．

2. 準備

今，$D = \{d_0, \ldots, d_{n-1}\}$ をドキュメントの集合とする．各ドキュメント d_i $(0 \leq i \leq n-1)$ はキーワード（文字列）の集合で，識別番号 i が割り振られている（i をドキュメント ID と呼ぶ）．また，$\mathcal{K}(D) = \cup_{d \in D} d$ と定義する．集合 d に対し，$|d|$ で集合 d の要素数を表す．論文を通し，$m = |\mathcal{K}(D)|$ と定義する．すなわち，m は異なるキーワード数を表す．任意のキーワード w に対し，$D(w) = \{i \mid d_i \text{ は } w \text{ を含む}\}$ と定義する．本論文では，キーワード w（検索キーワードはクエリとも呼ぶ）が与えられたとき，$D(w)$ を見つける問題を考える．$D(w)$ 内のドキュメント d_i は，クエリ w に一致するドキュメントと言う．ここでは，これを暗号化したまま行う方法について提案する．

検索システムの概要を図 1 に示す．システムは，クライアント（ユーザ）とサーバからなり，クライアントがドキュメントの所有者で，サーバはクライアントからの送られたクエリのトラップドアを用いて検索を行う．本システムでは，データの暗号化のために，共通鍵暗号方式を用いる．さらに我々は，検索用トラップドアの生成及びハッ

表 1 関連手法の比較. 表の中で，m はキーワード数，n はドキュメント数，n_w はキーワード w を含むドキュメント数，s_{max} はドキュメントの中で最多のキーワードを含むドキュメントのキーワード数を表す.

手法	索引サイズ	検索時間	安全性	通信ラウンド数
Curtomola [6](SSE-1)	$O(\sum_w n_w)$	$O(n_w)$	非適応的	1
Curtomola [6](SSE-2)	$O(n \times s_{max})$	$O(n_w)$	適応的	1
Kamara [13]	$O(\sum_w n_w)$	$O(n_w)$	適応的	1
提案法	$O(n \times s_{max})$	$O(\log m + n_w)$	適応的	2

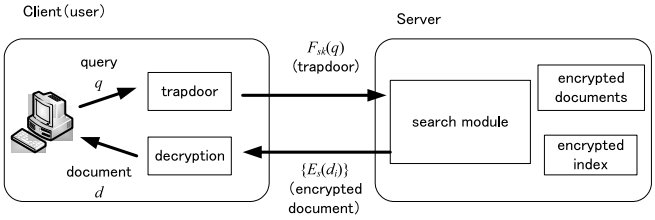

図 1 検索システムの概要

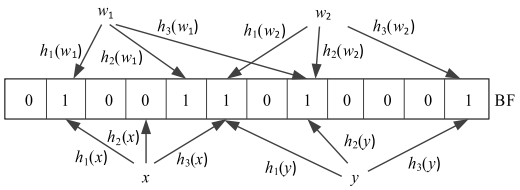

図 2 ブルームフィルタの例

シュ関数として擬似ランダム関数を使う．擬似ランダム関数 $F : \{0,1\}^{\lambda} \times \{0,1\}^{l_1} \rightarrow \{0,1\}^{l_2}$ に対し，$F(K,x)$ を $F_K(x)$ と書く．

3. ブルームフィルタ

まず，ブルームフィルタについて説明する．ブルームフィルタは，Bloom [1] によって開発されたデータ構造で，ビット列で構成され，ある要素が集合に含まれるかどうかを効率的にチェックできる．サイズ m ビットのブルームフィルタ BF の概要について説明する．今，$W = \{w_1, \ldots, w_l\}$ を l 個の語からなる集合，$h_1, \ldots, h_k$ を語から $[0, m-1]$ の整数へのハッシュ関数とする．そのとき，各 $w_i \in W$ に対し，BF 内で，$h_1(w_i)$, ..., $h_k(w_i)$ の位置にあるビットを 1 にセットする．与えられた語 w が W に入っているかどうかは，$h_1(w)$, ..., $h_k(w)$ を計算し，BF で対応する位置のビットがすべて 1 ならば $w \in W$，そうでなければ $w \notin W$ と判定する．欠点としては，W にない語 $v \notin W$ に対し，$h_1(v)$, ..., $h_k(v)$ のすべての位置のビットが 1 になってしまう場合がある．この場合は，$v \in W$ という間違った答えを得てしまう．これは偽陽性 (false positive) と呼ばれる．なお，W 内の語に対しては必ず正しい答えを返す．例を図 2 に示す．$W = \{w_1, w_2\}$，h_1, h_2, h_3 をハッシュ関数とする．そのとき，W に対する BF は，w_1 に対しては，$h_1(w_1), h_2(w_1), h_3(w_1)$ の位置のビットが 1 にセットされ，w_2 に対しては，$h_1(w_2), h_2(w_2), h_3(w_2)$ の位置のビットが 1 にセットされる．この BF に対し，語 x を与えると，$h_2(x)$ の位置のビットが 0 より，$x \notin W$ と判定される．また，語 y に対しては，3 箇所すべての位置のビットが 1 より，$y \in W$ と判定される．

4. 暗号化検索法

本論文の提案法 **SSE_KBF** は，**SSE_KBF** = ($KeyGen$, Enc, $Trapdr$, $BuildIndex$, $Search$, Dec) の 6 つの多項式時間アルゴリズムで構成される．それぞれは，次の役割を成す．

- $KeyGen(1^{\lambda})$: セキュリティパラメータ λ が与えられると，秘密鍵 $SK = (sk_1, sk_2, sk_3)$ を生成する．
- $Enc(sk_1, d)$: ドキュメント d を秘密鍵 sk_1 で暗号化するアルゴリズム．ここでは，$E_{sk_1}(d)$ と略記する．
- $Trapdr(sk_2, q)$: 秘密鍵 sk_2 を使って，検索クエリ q からトラップドア $T(q)$ を作成するアルゴリズム．擬似ランダム関数 $F_{sk_2}(q)$ を使って，$Trapdr(sk_2, q) = F_{sk_2}(q)$ とする．
- $BuildIndex(SK, D, \varepsilon)$: ドキュメントの集合 D から暗号化索引 Π = (**Kmap**, **Nmap**, **Dmap**) を作成するアルゴリズム．
- $Search(T(q))$: 暗号化索引を使って，トラップドア $T(q)$ から q を含むドキュメントを検索するアルゴリズム．
- $Dec(sk_1, c)$: 暗号化ドキュメント c を秘密鍵 sk_1 で復号化するアルゴリズム．

$KeyGen$, Enc, Dec は安全な共通鍵暗号方式を使う．したがって，以下の節では，本手法の核となる，暗号化索引を構成するアルゴリズム $BuildIndex$ と検索アルゴリズム $Search$ について述べる．

4.1 暗号化索引の構成

暗号化索引 Π = (**Kmap**, **Nmap**, **Dmap**) は，キーワードのトラップドアからその ID を求める索引 **Kmap**，キーワード ID からそのキーワードを含むドキュメント数を求める索引，及びドキュメント ID を取り出すための索引 **Dmap** とから構成される．**Kmap** はブルームフィルタで構成され，**Nmap** 及び **Dmap** は，配列で構成する．これらは，アルゴリズム 1 で与える $BuildIndex$ で同時に構成される．ここで，$x||y$ は，文字列 x と y の連接を表す．

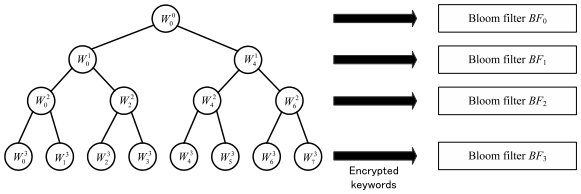

図 3 キーワード木とブルームフィルタ

4.1.1 Kmap の構成

我々は，$\mathcal{K}(D)$ のキーワードに 0 から順番に個別の番号を割り振る．番号はランダムに割り当てられ，キーワード w の番号を $ID(w)$ とする．今，ランダムに番号が割り当てられた $\mathcal{K}(D)$ を $\mathcal{K}(D) = \{w_0, \ldots, w_{n-1}\}$ とする．すなわち，各キーワード w_i $(0 \leq i \leq n-1)$ に個別の識別番号は $ID(w_i) = i$ である．そのとき，**Kmap** は，任意の $w \in \mathcal{K}$ に対し，w のトラップドアから $ID(w_i)$ を求めるための索引で，階層型ブルームフィルタを用いて実現する．具体的な実現方法について示す．今，$\lceil x \rceil$ は x 以上の最小の整数を表す．我々は，キーワードの集合を管理するため，2 分木を導入する．ここで，$h = \lceil \log_2 n \rceil$ と定義する．なお，これ以降，特に断りがなければ，対数の底として 2 を用いる．さて，$0 \leq lev \leq h$ に対し，階層 lev における $\mathcal{K}(D)$ の分割 $\mathcal{K}(D)^{lev} = \{W_0^{lev}, W_{2^\alpha}^{lev}, W_{2\cdot 2^\alpha}^{lev}, \ldots, W_{(2^{lev}-1)\cdot 2^\alpha}^{lev}\}$ を以下のように定義する．ここで，$\alpha = h - lev$ とする．

定義 1 $i = 0, 2^\alpha, 2\cdot 2^\alpha, 3\cdot 2^\alpha, \ldots, (2^{lev}-1)\cdot 2^\alpha$ に対し，$W_i^{lev} = \{w_i, w_{i+1}, \ldots, w_{i+k^\alpha-1}\}$ とする．

階層 lev において，各ノード W_i^{lev} の ID i は，0 から始め，2^α の間隔で順に割り振られていることに注意する．階層 lev の分割において，各 W_i^{lev} は，高々 2^α 個の要素からなる $\mathcal{D}$ の部分集合であり，番号 i は W_i^{lev} の識別番号（ID）と呼ばれる．

分割において，W_i^{lev} $(0 \leq lev \leq h-1)$ に対し，W_i^{lev+1},…, $W_{i+(2-1)2^{h-(lev+1)}}^{lev+1}$ を D_i^{lev} の 2 個の子供とみなせば，キーワード集合の分割は，W_i^{lev} をノードとし，W_0^0 を根とする 2 分木を構成する．この木をキーワード木と呼ぶことにする．キーワード木において，葉 W_i^h は 1 個のキーワード w_i だけからなる集合 $\{w_i\}$ となる．任意のノード W_i^{lev} に対し，W_i^{lev} が空ならば，そのノードを空ノードと言う．木の用語を使うならば，各ノードの階層 lev はそのノードの深さに対応し，木の高さが h となる．階層型は，各階層 $0 \leq lev \leq h$ に対し，1 個のブルームフィルタ BF_{lev} を用いる．したがって，全体で $h+1$ 個のブルームフィルタを用いる．

キーワード木において，各ノードの ID は，その最も左の子の ID と同じになることに注意する．これによって，検索時に，暗号化とハッシュ関数の計算を減らすことができる．

例 1 キーワード木の例を与える．$\mathcal{K}(D) = \{w_0, w_1, w_2, w_3, w_4\}$ とする．そのとき，$\mathcal{K}(D)$ に対するキーワード木は図 3 となる．図 3 で，BF_0, BF_1, BF_2, BF_3 は各階層に対するブルームフィルタを表し，各ノードは以下のようなキーワード集合になる．

$W_0^0 = \{w_0, w_1, w_2, w_3, w_4\}$,
$W_0^1 = \{w_0, w_1, w_2, w_3\}$, $W_4^1 = \{w_4\}$,
$W_0^2 = \{w_0, w_1\}$, $W_2^2 = \{w_2, w_3\}$, $W_4^2 = \{w_4\}$, $W_6^2 = \emptyset$,
$W_0^3 = \{w_0\}$, $W_1^3 = \{w_1\}$, $W_2^3 = \{w_2\}$, $W_3^3 = \{w_3\}$,
$W_4^3 = \{w_4\}$, $W_5^3 = \emptyset$, $W_6^3 = \emptyset$, $W_7^3 = \emptyset$.

アルゴリズム 1 $BuildIndex((sk_1, sk_2, sk_3), D, \varepsilon)$

$n \leftarrow |\mathcal{K}(D)|$, $h \leftarrow \lceil \log m \rceil$
Dmap の上限 $N = s_{max} \times m$ を設定する．
Kmap, **Nmap**, **Dmap** を初期化する．
$start \leftarrow 0$.
for $lev = 0$ to h **do**
  サイズ $\varepsilon \times m$ ビットのブルームフィルタ BF_{lev} を初期化する．
  for all $W_i^{lev} \in \mathcal{K}(D)^{lev} = \{W_0^{lev}, W_{2^\alpha}^{lev}, W_{2\cdot 2^\alpha}^{lev}, \ldots, W_{(2^{lev}-1)\times 2^\alpha}^{lev}\}$ **do**
    for all $w \in W_i^{lev}$ **do**
      $X \leftarrow F_{sk_2}(w)$,
      $p_1 \leftarrow F_X((i||1)), \ldots, p_k \leftarrow F_X(\ (i||k))$ を計算し，対応する BF_{lev} のビットをセットする．
      if $lev = h$ **then**
        $n_{w_i} = |D(w_i)|$ と置く．$D(w_i) = \{d_0, \ldots, d_{n_{w_i}-1}\}$ とする．
        Nmap$[i] \leftarrow (F_X(*), Esk_3(n_{w_i}||start))$.
        for $j = 0$ to $n_{w_i} - 1$ **do**
          Dmap$[\pi(start + j)] \leftarrow ID(d_j)$
        end for
        $start \leftarrow start + n_{w_i}$
      end if
    end for
  end for
  BF_{lev} を **Kmap** に加える．
end for
$ctr \leftarrow 0$.
for all $d \in D$ **do**
  Dmap における $ID(d)$ の出現回数を s' とする．
  if $s_{max} - s' > 0$ **then**
    for $j = 1$ to $s - s'$ **do**
      Dmap$[\pi(start + ctr)] \leftarrow ID(d)$.
      $ctr++$.
    end for
  end if
end for
output $\Pi = (\mathbf{Kmap}, \mathbf{Nmap}, \mathbf{Dmap})$

4.1.2 Nmap の構成

任意のキーワード $w \in \mathcal{K}$ に対し，$n_w = |D(w)|$ と置く．このとき，**Nmap**$[ID(w)]$ には，$(F_X(*), E_{sk_3}(n_w||start))$ が格納される．ここで，$X = F_{sk_2}(w)$ とする．すなわち，X は w のトラップドアであるから，$F_X(*)$ をチェックすることにより，w に対応した正しい **Nmap**$[ID(w)]$ を見ているか知ることができる．これによって，**Kmap** で発生する可能性のある偽陽性を排除することができる．

4.1.3 Dmap の構成

$N = s_{max} \times n$ とする．ここで，s_{max} は，$s_{max} =$

アルゴリズム 2 *Search*

Require: クエリ q

クライアント

1: クライアントはトラップドア $T(q) = F_{sk_2}(q)$ を作成し，サーバに送る．

サーバ

2: $calID = -1$ //直近にハッシュ値が計算された ID を保持する．
3: $stackID$ と $stacklev$ に 0 を入れる．
4: **while** $stackID \neq \emptyset$ **do**
5: $stackID$ から id を,$stacklev$ から lev を取り出す．
6: **if** $calID \neq id$ **then**
7: $p_1 \leftarrow F_{T(q)}(id||1), \ldots, p_k \leftarrow F_{T(q)}(id||k)$
8: $calID = id$
9: **end if**
10: **if** BF_{lev} 内の p_1, ... ,p_k に対応するビットが全て 1 **then**
11: **if** $lev = h$ **then**
12: $listID$ に id を追加する．
13: **else**
14: $lev = lev + 1$
15: $stackID$ に $id + 2^{h-lev}$ を入れる．
16: $stacklev$ に lev を入れる．
17: $stackID$ に id を入れる．
18: $stacklev$ に lev を入れる．
19: **end if**
20: **end if**
21: **end while**
22: **for all** $id \in listID$ **do**
23: **Nmap**$[id]$ の要素 $(F_X(*), E_{sk_3}(n_q||start))$ を取り出す．
24: **if** $F_{T(q)}(*) = F_X(*)$ **then**
25: $E_{sk_3}(n_q||start)$ をクライアントに送る
26: **end if**
27: **end for**

クライアント

28: サーバから $E_{sk_3}(n_q||start)$ を受け取ったら，$(n_q, start)$ を取り出す．
29: $j_1 = \pi(start), \ldots, j_{n_q-1} = \pi(start + n_q - 1)$ を計算し，サーバに送る．

サーバ

30: クライアントから $j_1, \ldots, j_{n_q-1}$ を受け取る．
31: **Dmap**$[j_1], \ldots,$ **Dmap**$[j_{n_q-1}]$ からドキュメント ID を取り出す．
32: 対応する暗号化ドキュメントをクライアントに送る．

$\max_{d \in D} |d|$ とする．すなわち，s_{max} は最も多くのキーワードを含むドキュメントのキーワード数である．任意の $0 \leq i \leq N-1$ に対し，配列 **Dmap**$[i]$ にはドキュメント ID が格納される．今，π を $[0, N-1]$ から $[0, N-1]$ へのランダム置換とする．キーワード w を含むドキュメントの数を n_w としたとき，**Dmap**$[\pi(start_w)], \ldots,$ **Dmap**$[\pi(start_w + n_w - 1)]$ に $D(w)$ のドキュメントの ID を順番に格納する．すべてのドキュメント ID の出現数をそろえるため，**Dmap** での出現数が s_{max} に達していないドキュメント ID は **Dmap** の空欄にランダムに埋めていく．

4.2 検索アルゴリズム

検索は，クライアントとサーバ間でのクエリ・リプライ（ラウンド）を 2 回行うことにより実行される．すなわち，クライアントはまず検索したいキーワード（クエリ）q のトラップドア $T(q)$ をサーバに送る．サーバは，トラップドアと **Kmap** を用いて，キーワードの ID $ID(q)$ を求め，**Nmap**$[ID(q)]$ の要素 $(F_X(*), E_{sk_3}(n_q, start))$ を取り出し，$E_{sk_3}(n_q, start)$ をクライアントに返す．クライアントは，$E_{sk_3}(n_q, start)$ を復号し，$(n_q, start)$ を取り出す．それから，$\pi(start), \ldots, \pi(start + n_q - 1)$ をサーバに送る．サーバは **Dmap** を用いてドキュメント ID を求め，対応する暗号化ドキュメントをクライアントに送る．検索アルゴリズムを *Search* に与える．

Search に従って，詳しく検索の流れを説明する．クライアントはクエリ q を検索したいとき，まず，検索用トラップドア $T(q) = F_{sk_2}(q)$ を作成し，それをサーバに送る．サーバは，$T(q)$ を受け取ると，*BuildIndex* で作成された **Kmap** を用いてキーワードの ID を取得する．**Kmap** の探索において，次にチェックすべきノードの ID を保持する $stackID$ と, そのノードの階層を保持する $stacklev$ の 2 個のスタックを用いて, クライアントから受け取ったトラップドア $T(q)$ をもとにキーワード木を深さ優先探索する. 具体的には, チェックするノードの階層のブルームフィルタに対して，k 個のハッシュ関数（鍵を $T(q)$，入力をチェックするノードの ID とした）のそれぞれの値が指している位置のビットが全て 1 であるかをチェックしていく. 最下層以外のブルームフィルタに対して全て 1 であった場合はそのノードの 2 つの子供の ID とその階層 lev を右の子, 左の子の順でスタックに入れ, 最下層のブルームフィルタに対してそうであった場合は, その時の ID を検索キーワードのキーワード ID とみなし, 正解の ID を格納する $listID$ に入れる. 本来, 1 つのトラップドアに対して正解となる ID は 1 個 (登録されてなければ 0 個) であるはずなので, $listID$ に複数以上の ID が格納されている場合, 偽陽性が発生していることになる. $listID$ が求まったら，**Nmap** からキーワードに関する情報を引き出す．**Nmap** には，トラップドア X に対する $F_X(*)$ が保存されている．したがって，$T(q) = X$ のとき 24 行目の if 文が成り立つ．これにより，偽陽性を防ぐことができる．さらに，**Nmap** には対応するキーワードを含むドキュメント数，**Dmap** の開始位置が暗号化されて保存されているので，この情報をクライアントに送る．

クライアントは，マッチするドキュメント数, 開始位置の暗号化を受け取るとそれを復号する．さらに，ランダム置換 π を用いて, $\pi(start), \ldots, \pi(start - n_q - 1)$ を計算し，それらをサーバに送る．サーバは，$\pi(start), \ldots, \pi(start - n_q - 1)$ を受け取ると，**Dmap** からドキュメント ID を取り出し，対応する暗号化ドキュメントをクライアントに返す．

Search の時間は，**Kmap** からキーワードの ID を求める時間は木の高さになるので $O(\log m)$，**Nmap** から情報を引き出す時間は $O(1)$，**Dmap** からドキュメント ID を取り出す時間は $O(n_q)$ となる．また，通信のラウンド数は 2

回である．

5. セキュリティの解析

我々は，Curtmola et al. [6] のセキュリティモデルに従って，提案手法が適応的セキュリティモデルにおいて安全性であることを証明する．まず， 無視可能関数を定義する．

定義 2 正の整数から正の実数への関数 f がセキュリティパラメータ λ に関して無視可能とは，もしすべての正の多項式時間関数 $p(\cdot)$ と十分大きな λ に対し, $f(\lambda) < p(\lambda)$ なるときである．

次に，適応的セキュリティモデルを定義する．２つのゲーム，$\mathbf{REAL}_{\mathcal{A}}$ と $\mathbf{SIM}_{\mathcal{A},\mathcal{S}}$ を考える．これらのゲームは, 3 人のプレーヤー, 挑戦者 $\mathcal{C}$, 攻撃者 $\mathcal{A}$, シミュレーター $\mathcal{S}$ によって実行される．以下で定義するように，$\mathbf{REAL}_{\mathcal{A}}$ は，提案手法を用いて実行され，$\mathbf{SIM}_{\mathcal{A},\mathcal{S}}$ は，攻撃者が知りうる情報だけを使って，提案法の暗号化索引，暗号化，トラップドアを模倣することでゲームが行われる．今，ドキュメントの集合 D を $D = \{d_0, \ldots, d_{n-1}\}$ とする．攻撃者が知り得る情報は，以下のものである．

- 各ドキュメントの長さ，$|d_0|, \ldots, |d_{n-1}|$, 及び暗号化されたドキュメント．
- 暗号化索引に登録するキーワード数 m と **Dmap** の上限 N.
- 検索に使われるキーワード w のトラップドアと $D(w)$（アクセスパターンと呼ばれる）. 攻撃者はトラップドアから過去に w が検索されたかどうか知ることができる（サーチパターンと呼ばれる）.

適応的セマンティックセキュリティモデルに向け，２つのゲームを定義する．

$\mathbf{REAL}_{\mathcal{A}}(\lambda)$

- 攻撃者 $\mathcal{A}$ は，ドキュメントの集合 D を任意に作成する．ここで，$D = \{d_0, \ldots, d_{n-1}\}$ とする．その後，$\mathcal{A}$ は D を挑戦者 $\mathcal{C}$ に送る．
- $\mathcal{C}$ は，$KeyGen(1^{\lambda})$ を使って，秘密鍵 $SK = (sk_1, sk_2)$ を生成し，$\Pi = (\mathbf{Kmap}, \mathbf{Nmap}, \mathbf{Dmap})$, $E_{sk_1}(D)$ を，$BuildIndex(\ SK\ , D, \varepsilon)$, を使って作成する．その後，$\mathcal{C}$ は，$(\Pi\ , E_{sk_1}(D))$ を $\mathcal{A}$ に送る．
- 以下を多項式回繰り返す．
  (1) $\mathcal{A}$ はクエリ q を $\mathcal{C}$ へ送る．
  (2) $\mathcal{C}$ はトラップドア $T(q) = Trapdr(sk_2, q)$ を作成し，$\mathcal{A}$ へ送る．
  (3) $\mathcal{A}$ はトラップドア $T(q)$ と **Kmap** から q の ID を求め，$\mathbf{Nmap}^*$ から (X, Y) を取り出し，X が $T(q)$ と無矛盾なら，Y を $\mathcal{C}$ へ送る．
  (4) $\mathcal{C}$ は Y を復号し $(n_q, start)$ を取り出し，ランダム置換 π を適用した結果 $j_1, \ldots, j_{n_q}$ を $\mathcal{A}$ へ送る．
  (5) $\mathcal{A}$ はドキュメント ID を取り出す．
- $\mathcal{A}$ は，ビット $b \in \{0, 1\}$ を出力する．

$\mathbf{SIM}_{\mathcal{A},\mathcal{S}}(\lambda)$

- 攻撃者 $\mathcal{A}$ は，D を任意に作成する．ここで，$D = \{d_1, \ldots, d_n\}$ とする．その後，$\mathcal{A}$ は D を挑戦者 $\mathcal{C}$ に送る．
- $\mathcal{C}$ は，$|d_0|, \ldots, |d_{n-1}|$, m, N をシミュレータ $\mathcal{S}$ に送る．
- $\mathcal{S}$ は，$\mathcal{C}$ からの情報を使って，暗号化索引 $\Pi^* = (\mathbf{Kmap}^*, \mathbf{Nmap}^*, \mathbf{Dmap}^*)$，暗号化ドキュメントの集合 $\{c_0^*, \ldots, c_{n-1}^*\}$ を作成する．その後，$\mathcal{S}$ は，Π^*, c_i^*, T_i^* を $\mathcal{C}$ へ送る．
- $\mathcal{C}$ は，それらを $\mathcal{A}$ に渡す.
- 以下を多項式回繰り返す．
  (1) $\mathcal{A}$ はクエリ q を $\mathcal{C}$ へ送る．$\mathcal{C}$ は，それを $\mathcal{S}$ へ送る．
  (2) $\mathcal{S}$ は，q に対するトラップドア T_q^* を作成し，$\mathcal{C}$ へ送る．
  (3) $\mathcal{C}$ は，T_q^* を $\mathcal{A}$ へ送る．
  (4) $\mathcal{A}$ はトラップドア T_q^* と $\mathbf{Kmap}^*$ から ID を求め，$\mathbf{Nmap}^*$ から (X, Y) を取り出し，$\mathcal{C}$ へ送る．
  (5) $\mathcal{C}$ は，それを $\mathcal{S}$ へ送る．
  (6) $\mathcal{S}$ は，(X, Y) に対し，$j_1, \ldots, j_{n_q}$ を作成し，$\mathcal{C}$ へ送る．
  (7) $\mathcal{C}$ は，それを $\mathcal{A}$ へ送る．$\mathcal{A}$ は，$\mathbf{Dmap}^*$ から ID を取り出す．
- $\mathcal{A}$ はビット $b \in \{0, 1\}$ を出力する．

定義 3 もし検索システムが次の条件を満たすならば，それは適応的セマンティック安全な検索システムという．確率的多項式時間で動作するすべての攻撃者 $\mathcal{A}$ に対し, 次の式を満足する確率的多項式時間シミュレータが存在する．
$|Pr(\mathbf{REAL}_{\mathcal{A}}(\lambda)$ で $\mathcal{A}$ は $b = 1$ を出力 $) - Pr(\mathbf{SIM}_{\mathcal{A},\mathcal{S}}(\lambda)$ で $\mathcal{A}$ は $b = 1$ を出力 $)|$
は無視可能である.

安全性について，共通鍵暗号及び擬似ランダム関数の安全性（例えば，文献 [10]）を仮定すると，次の定理が成り立つ．

定理 1 提案法 **SSE_KBF** は，適応的セマンティック安全である．

証明： **REAL** と区別がつかない **SIM** の構成法を段階的に示す．

(1) 暗号化ドキュメントをシミュレートする．シミュレータ $\mathcal{S}$ は，$|d_0|, \ldots, |d_{n-1}|$ を知っている．したがって，$|d_0|, \ldots, |d_{n-1}|$ と同じサイズのランダム列 $c_0^*, \ldots, c_{n-1}^*$ を生成する．**REAL** で用いられる共通鍵暗号の CPA-安全性より，$c_0, \ldots, c_{n-1}$ と $c_0^*, \ldots, c_{n-1}^*$ は区別できない．

(2) **Kmap** をシミュレートする．シミュレータ $\mathcal{S}$ は，全キーワード数 m を知る．これより，m 個のランダム列 $T_1^*, \ldots, T_m^*$ を生成する．各 T_j^* の ID を j とし，$BuildIndex$ に従って，$T_1^*, \ldots, T_m^*$ を擬似ランダム関数 F_{sk_2} の値と見なし，$\mathbf{Kmap}^*$ を構成する．F_{sk_2} の

擬似ランダム性により，**Kmap** と **Kmap*** は区別できない．

(3) **Nmap** をシミュレートする．シミュレータ $\mathcal{S}$ は，全キーワード数 m を知る．そのとき，各 i $(1 \leq i \leq m)$ に対し，**Nmap**$^*[i]$ に $(F_{T_i^*}(*), r_i)$ を格納する．ここで，r_i はランダム列である．**Nmap** は，**Nmap**$[i]$ に $(F_X(*), E_{sk_3}(n_w||start))$ を格納している．X は i 番目のキーワード w に F_{sk_2} を適用した値であること，また E_{sk_3} は CPA 安全な暗号方式であることより，**Nmap** と **Nmap*** は区別できない．

(4) **Dmap** をシミュレートする．シミュレータ $\mathcal{S}$ は，**Dmap** のサイズとドキュメント数より s_{max} を知る．$\mathcal{S}$ は次のように **Dmap*** を構成する．各ドキュメント d に対し，**Dmap*** からまだ空の場所を s_{max} 個ランダムに取り出し，その位置に d の ID を入れる．**Dmap** は，ランダム置換を用いて，各ドキュメントの ID が s_{max} 個ランダムに入れられている．したがって，**Dmap*** と **Dmap** は区別できない．

(5) 検索をシミュレートする．攻撃者がクエリ q を発したとする．シミュレータ $\mathcal{S}$ は，q に対するアクセスパターン $D(q)$ を知る．まず，$\mathcal{S}$ は，q が過去に質問されたクエリかどうかチェックする．もし新しいクエリなら，$\{T_1^*, \ldots, T_m^*\}$ からまだ用いられていないビット列 T_j^* を取り出す．もし過去に質問されたクエリならば，前に用いたビット列 T_j^* を取り出す．この T_j^* を q のトラップドアとして攻撃者に返す．攻撃者は，T_j^* と **Kmap*** で検索することにより，ID　j を得る．それから，**Nmap**$[j]$ から $(F_{T_j^*}(*), r_j)$ を取り出す．もし偽陽性により **Nmap**$[j]$ 以外の要素を取り出せば，$F_{T_j^*}(*)$ の部分が一致しない．これは，**REAL** とまったく同じ動作であるので，この時点で **REAL** と区別できない．攻撃者は，$(F_{T_j^*}(*), r_j)$ を $\mathcal{S}$ に送る，$\mathcal{S}$ は，$D(q)$ を知っているから，$D(q)$ の各ドキュメント ID id に対し，**Dmap** から id が入っている場所で，過去に使われていない位置をランダムに取り出し，攻撃者に返す．検索結果に対応するランダムな位置情報が帰ってくる．また，すべてのドキュメント ID は s_{max} 個挿入されているので，ランダム置換の結果と同じに見えるため，攻撃者は **REAL** と区別できない．

以上のようにシミュレータを構成すると，攻撃者が **REAL** と **SIM** を区別できる確率は，無視可能となる．

6. 実験

我々は提案法を実装し，実験的に評価した．評価用のデータとして，Enron Email データセット [5] を用いた．Enron Email データセットで公開されている 517431 個のメールデータを使い，各メールから高々 500 個のキーワードを取り出し暗号化索引を構成した．異なるキーワード数は，307830 個となった．提案法の実装は Windows 上で JAVA を用いて行った．共通鍵暗号方式としては鍵長 128 ビットの AES を用い，ハッシュ関数は HMAC-SHA256 を用いた．暗号化索引の実装において，**Kmap** は，各階層の BF のサイズをキーワード数の 10 倍とした．また，N 上のランダム置換は，配列 $Perm[N]$ を使って，$Perm$ にランダムに選択した置換を保存することにより実装されている．したがって，i に対する値は，$Perm[i]$ から直接取り出すことができる．ただ，ランダム置換を配列として保持する実装は，クライアントにこの配列を持たせることにもなり，今後の改善点の一つでもある．

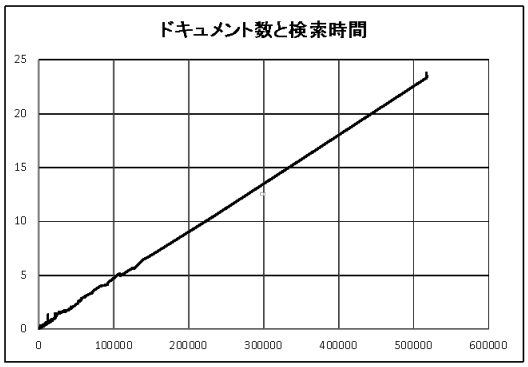

図 4　ドキュメント数と検索時間の関係：横軸が一致するドキュメント数，縦軸が検索時間 (ms) を示す

表 2　クエリにマッチするドキュメント数と検索時間との関係

ドキュメント数	検索時間 (ms)
100 以下	0.08
101〜1000	0.08〜0.13
1001〜10000	0.13〜0.5
10001〜100000	0.5〜4
100001 以上	5〜24
517431	24

暗号化索引のサイズについて考察する．クエリからキーワード ID を求める **Kmap** は階層型ブルームフィルタで構成される．今回，キーワード数は 307830 であるため，階層数は 20 である．登録キーワードの数はどの階層も同じで，307830 個である．また，各階層のブルームフィルタのサイズは，キーワード数の 10 倍のビット数である．したがって，トータルのビット数は，$307830 \times 10 \times 20 = 61566000$ ビットである．これは，約 7.7MB ほどになる．**Nmap** は，307830 の要素を持つ配列である．各要素は $(F_X(*), E_{sk_3}(n_w||start))$ である．今回，F_X, E_{sk_3} とも 128 ビットの HMAC および AES を用いているので，各要素は 32 バイトのデータである．したがって，**Nmap** のサイズは，$307830 \times 32 = 9850560$ で，約 10MB となる．配列 **Dmap** が最も大きくなる．適応的安全性を満足させるため，すべてのドキュメント ID の出現を最大値に合わせ，同一にしているからである．そのため，ドキュメント数× s_{max} の要素数の配列として実装してい

る．ここで，s_{max} は最も多くのキーワードを含むドキュメントのキーワード数である．今回，$s_{max} = 500$ である．各要素には，ドキュメント ID が入る．これは，32 ビット整数型を用いたので，4 バイトとなる．したがって，トータルサイズは，$517431 \times 500 \times 4 = 1034862000$ で，約 1GB となる．このように，暗号化索引のサイズは **Dmap** に依存する．

各クエリについては，すべてのキーワードについて，検索時間を確認した．検索時間は，トラップドアを与えてから一致するドキュメント ID を求めるまでの時間を計測した．図 4 に示すように一致するドキュメント数に比例して時間が増える．これは **Kmap** によるキーワード ID の検索は，すべてのキーワードで同じでほとんど時間がかからない．**Dmap** の検索は，クエリに一致するドキュメント数に比例するから，図 4 のような結果となる．表 2 に検索時間の概略を示す．一致ドキュメント数が 10000 以下の場合，1[ms] 以内でドキュメントを求めることができる．最悪の場合，すなわち，517431 個のすべてのドキュメントに一致する場合でも 24[ms] 程度で求めることができる．このように，提案法は高速な検索を実現している．これは，暗号化索引における **Nmap** 及び **Dmap** が単純な配列であるため，クエリにマッチするドキュメント ID を直接配列から取り出すことができることに大きく依存している．また，ランダム置換の計算も単純に配列から要素を取り出すように実装したことも影響していると思われる．

7. まとめ

本論文では，キーワードベースのブルームフィルタを用いた，適応的安全性を満足する新たな検索手法を提案した．本手法は，適応的安全性を満足させるため，クライアント・サーバ間で 2 回の通信を行うが，暗号化索引を単純な配列で実現できるため，高速な検索が可能である．今後の課題としては，さらなる索引サイズの削減及び動的に変化するデータへの対応が挙げられる．索引サイズの削減に関しては，ランダムオラクルを導入することによって，適応的安全性を維持したまま削減できるのではないかと考えている．また，提案法は，並列化の導入が比較的容易であると思われるため，並列化による検索効率の改善がある．

謝辞　本研究は，JSPS 科研費 26330154 の助成を得て行われた．

参考文献

[1] B.H. Bloom, Space/Time Trade-offs in Hash Coding with Allowable Errors, Comm. of the ACM, 13, pp.422–426, 1970.
[2] A. Broder and M. Mitzenmacher, Network Applications of Bloom Filters: A Survey, Internet Mathematics, 1, 4, pp.485–509, 2004.
[3] N. Cao, C. Wang, M. Li, K. Ren and W. Lou, Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data, Proc. of INFOCOM 2011, pp.829–837, 2011.
[4] Y.-C. Chang and M. Mitzenmacher, Privacy Preserving Keyword Searches on Remote Encrypted Data, Proc. of ACNS 2005, LNCS 3531, pp.442–455, 2005.
[5] W. W. Cohen, Enron Email Dataset, http://www.cs.cmu.edu/ enron/.
[6] R. Curtmola, J. Garay, S. Kamara and R. Ostrovsky, Searchable symmetric encryption: Improved definitions and efficient constructions, Journal of Computer Security, pp.895–934, 2011.
[7] E.-J. Goh, Secure Indexes, Stanford Univ. Technical Report, In IACR ePrint Cryptography Archive, 2003, See http://eprint.iacr.org/ 2003/216.
[8] P. Golle, J. Staddon and B. Waters, Conjuctive Keyword Searcg over Encrypted Data, Proc. of ACNS 2004, LNCS 3089, pp.31–45, 2004.
[9] H. Hacüigumüs, B. Hore, B. Iyer and S. Mehrotra, Search on Encrypted Data, Advances in Information Security, 33, pp.383-425, 2007.
[10] J. Katz and Y. Lindell, Introduction Modern Cryptography, Second Edition, CRC Press, 2015.
[11] K. Kurosawa and Y. Ohtaki, UC-Secure Searchable Symmetric Encryption, FC 2012, LNCS 7397, pp.285–298, 2012
[12] S. Kamara and C. Papamanthou, Parallel and Dynamic Searchable Symmetric Encryption, Proc. of FC'13, LNCS 7859, pp.258–274, 2013
[13] S. Kamara, C. Papamanthou, and T. Roeder, Dynamic Searchable Symmetric Encryption, Proc. of CCS'12, pp.965–976, 2012
[14] L. Liu and J. Gai, Bloom Filter Based Index for Query over Encrypted Character Strings in Database, Proc. of CSIE 2009, pp.303–307, 2009.
[15] Q. Liu, G. Wang and J. Wu, An Efficient Privacy Preserving Keyword Search Scheme in Cloud Computing, Proc. of CSE 2009, pp.715–720, 2009.
[16] R.A. Popa, C.M.S. Redfield, N. Zeldovich and H. Balakrishnan, CryptDB: Processing Queries on an Encrypted Database, Commun. ACM, 55, 9, pp.103–111, 2012.
[17] D.X. Song, D. Wagner and A. Perrig, Techniques for Searchers on Encrypted Data, IEEE Symposium on Security and Privacy, pp.44–55, 2000.
[18] T. Suga, T. Nishida and K. Sakurai, Secure Keyword Search Using Bloom Filter with Specified Character Positions, ProvSec 2012, LNCS 7496, pp.235–252, 2012.
[19] 高野匠, 山本博章, キーワード型ブルームフィルタを用いた安全で効率的な検索法, CSS 2015, 3D4-3, pp.1351–1358, 2015.
[20] 山本博章, 山下智穂, 大井篤，中村伸一，白井啓一郎，宮崎敬, 階層化的ブルームフィルタを用いた安全で効率的なキーワード検索法, 電子情報通信学会論文誌, Vol.J96-D, No.12, pp.3030–3043, 2013.