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

高野 匠 †　　山本 博章 ‡

† 信州大学大学院理工学研究科
380-8553 長野市若里 4-17-1
14tm510e@shinshu-u.ac.jp

‡ 信州大学工学部
380-8553 長野市若里 4-17-1
yamamoto@cs.shinshu-u.ac.jp

あらまし 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．この問題に対して，ブルームフィルタを用いた手法を提案されているが，従来の手法は，ドキュメントをベースにしたブルームフィルタで暗号化索引を構成していた．そのため，検索クエリにマッチするドキュメントの数が多くなると，偽陽性率も高くなり，最悪，検索時間はドキュメント数に比例した時間になる．本論文では，キーワードをベースとしたブルームフィルタを用いた新たな手法を提案し，大規模データを用いてその性能を評価する．

A Secure and Efficient Search Scheme Using a Keyword-based Bloom Filters

Shou Kouno†　　Hiroaki Yamamoto‡

†Shinshu University.
4-17-1, Wakasato, Nagano-shi, Nagano 380-8553, JAPAN
14tm510e@shinshu-u.ac.jp

‡Shinshu University
4-17-1, Wakasato, Nagano-shi, Nagano 380-8553, JAPAN
yamamoto@cs.shinshu-u.ac.jp

Abstract From a view point of information security, researches on an encrypted search system have been done intensively. For this problem, we developed a method which makes use of hierarchical Bloom filters based on documents. Therefore, the existing method runs in time proportional to the number of documents at the worst case. In this paper, we propose a new secure search scheme using hierarchical Bloom filters based keywords and evaluate its performance using a large data set.

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, 15, 14].

このような背景のもと，本論文は，ドキュメントに対するキーワード検索を暗号化された世界で効率的に行うための新たな手法を提案する．

このような暗号化検索法では，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，サーバに暗号化キーワードを与えて検索を行うため，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．しかし，内容はすべて暗号化されているため，それ以外の情報を漏らすことはない．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, 12, 15] など）．

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．Goh [7] は，最初にブルームフィルタを用いた手法を提案した．Goh は，各ドキュメントに対し検索用のブルームフィルタを用意し，そのドキュメントに含まれるキーワードを暗号化して暗号化索引を構成する手法を提案した．山本他 [16] 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，2 分木の各階層にブルームフィルタを割り当てることにより，効率的な検索を可能にする安全な索引構造を提案した．彼らは，理論的に検索時間を解析するとともに実験的に性能を評価した．Suga [15] 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，部分一致もできるような手法を提案した．ブルームフィルタ以外を用いた手法としては転置索引がある．代表的なものは Curtomola [6] によるものである．

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うことが難しい．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．以下の特長を持つ．

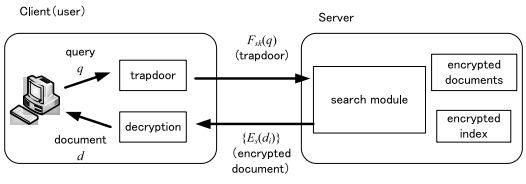

図 1: 検索システムの概要

- 暗号化索引に登録するキーワード数を n としたとき，検索は $O(\log n)$ 時間で行なうことができる．
- コンパクトな暗号化索引を実現できる．

本論文の構成は次のようになっている．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 章で提案法である階層型暗号化索引の構成法及び検索アルゴリズムについて述べ，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 $(0 \leq i \leq n-1)$ はテキストで，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）．各ドキュメント d_i に対し，$\mathcal{K}(d_i)$ を，d_i から抜き出したキーワードの集合とする．キーワードとは語（すなわち，文字列）である．また，$\mathcal{K}(D) = \cup_{d \in D} \mathcal{K}(d)$ と定義し，$|\mathcal{K}(D)|$ で $\mathcal{K}(D)$ の要素数を表す．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（クエリと呼ばれる）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．

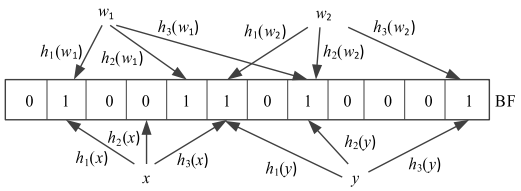

図 2: ブルームフィルタの例

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検索キーワード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 $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^m$ 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 $SK = (sk_1, sk_2)$ 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 $\Pi = (\mathbf{Kmap}, \mathbf{Dmap})$ 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 $\Pi = (\mathbf{Kmap}, \mathbf{Dmap})$ は，キーワードのトラップドアから，そのIDを求める索引 $\mathbf{Kmap}(\cdot)$ とキーワードのIDからそのキーワードを含むドキュメントIDを取り出すための配列 $\mathbf{Dmap}[]$ とから構成される．$\mathbf{Kmap}$ 及び $\mathbf{Dmap}$ は，アルゴリズム1で与える $BuildIndex$ で同時に構成する．ここで，$x||y$ は，文字列 x と y の連接を表す．

4.1.1 Kmapの構成

我々は，各キーワードに0から順番に個別の番号を割り振り，それを検索のために使う．今，$Trap(\mathcal{K}(D)) = \{Trapdr(sk, w) \mid w \in \mathcal{K}(D)\}$ をすべてのキーワードのトラップドアの集合とする．$\mathcal{K}(D) = \{w_0, \ldots, w_{n-1}\}$ に対し，各キーワード w_i $(0 \leq i \leq n-1)$ に個別の識別番号 $ID(w_i) \in \{1, \ldots, n-1\}$ をランダムに割り当てる．そのとき，$\mathbf{Kmap}$ は，$Trap(\mathcal{K}(D))$ 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．また，$\lceil x \rceil$ 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

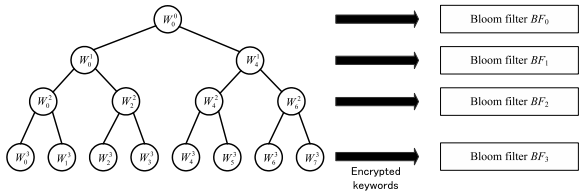

図 3: キーワード木とブルームフィルタ

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

4.1.2 Dmapの構成

キーワード w を持つドキュメントのリスト $D(w)$ は，暗号化され配列 $\mathbf{Dmap}[ID(w)]$ に格納される．このとき，$D(w)$ は w のトラップドア $F_{sk}(w)$ をキーとして，暗号化されて登録される．これによって，検索結果を，検索のトラッ

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

$n \leftarrow |\mathcal{K}(D)|$, $h \leftarrow \lceil \log n \rceil$
Dmap の上限 N を設定する.
Kmap, **Dmap** を初期化する.
for $lev = 0$ to h **do**
  サイズ $\varepsilon \times n$ 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 $lev = 0$ **then**
        $D(w_i)$ を $D_1(w_i), \ldots, D_l(w_i)$ に等分に分割する.
        $n \geq j_1, \ldots, j_{l-1} \leq N$ で未使用な数をランダムに生成する.
        Dmap$[i] \leftarrow E_X(D_1(w_i)||\ j_1)$
        for $e = 1$ to $l - 2$ **do**
          Dmap$[j_e] \leftarrow E_X(D_{e+1}(w_i)||\ j_{e+1})$
        end for
        Dmap$[j_{l-1}] \leftarrow E_X(D_l(w_i)||*)$
      end if
    end for
  end for
  BF_{lev} を **Kmap** に加える.
end for
output $\Pi = (\mathbf{Kmap}, \mathbf{Dmap})$

プドアで復号化することにより，偽陽性を排除することができる.

4.2 検索アルゴリズム

検索アルゴリズム $Search(X)$ をアルゴリズムに与える．$Search(X)$ は，検索用トラップドア X を受け取ると，$BuildIndex$ で作成された階層型 BF**Kmap** を用いて，キーワードの ID を取得し，次に，**Dmap** を用いて，キーワードの ID からドキュメントの ID を得る.

アルゴリズム 2 $Search(X)$

$calID = -1$ //直近にハッシュ値が計算された ID を保持する.
$stackID$ と $stacklev$ に 0 を入れる.
while $stackID \neq \emptyset$ **do**
  $stackID$ から id を,$stacklev$ から lev を取り出す.
  if $calID \neq id$ **then**
    $p_1 \leftarrow F_X(id||1), \ldots, p_k \leftarrow F_X(id||k)$
    $calID = id$
  end if
  if BF_{lev} 内の p_1, ... ,p_k に対応するビットが全て 1 **then**
    if $lev = h$ **then**
      $listID$ に id を追加する.
    else
      $lev = lev + 1$
      $stackID$ に $id + 2^{h-lev}$ を入れる.
      $stacklev$ に lev を入れる.
      $stackID$ に id を入れる.
      $stacklev$ に lev を入れる.
    end if
  end if
end while
for all $id \in listID$ **do**
  $C \leftarrow$ **Dmap**$[id]$
  $Dec(X, C)$ で，C が正しく復号化されたならば，それを出力
end for

Search は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 $X = F_{sk_2}(w)$ 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 X，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．キーワード木の左の子ノードをチェックする際はノードの左の子は親ノードと同じ ID を持つため, ハッシュ関数の計算が不要であり, 計算時間が短縮できる. しかし偽陽性発生率を減らすために, キーワード木の最下層に対応するブルームフィルタは他の層のブルームフィルタより大きく作られている関係で, 最下層のブルームフィルタをチェックする際は左の子ノードに対しても再度ハッシュ関数の計算をする必要がある.

$Search(X)$ は，前段階で得られたキーワード ID のリスト *listID* を用いて，**Dmap** からドキュメント ID のリスト $D(w)$ を取り出す．ブルームフィルタは偽陽性を発生する可能性があるため，*listID* には複数の ID が含まれる可能性がある．しかし，**Dmap** のドキュメント ID のリストは, トラップドア X を鍵として暗号化されているため，X に対応しないキーワード ID で取り出した **Dmap** のデータは正常に復号化できない．よって，*listID* が複数以上の ID を保持していたとしても，X に対応するドキュメント ID だけが正しく復号化される．したがって, 正解となるキーワード ID の $D(w)$ のみが得られる.

$Search(X)$ の時間は，n を **Kmap** への登録キーワード数とすれば，キーワード ID を求める時間は木の高さになるので $O(\log n)$，ドキュメント ID を得る時間は，キーワード IDid から **Dmap**$[id]$ を直接取り出すことができるので, **Dmap**$[id]$ のサイズに比例する時間となる.

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，安全性を証明する．彼らは，非適応型と適応型のセマンティックセキュリティモデルを提案したが，非適応型にしたがってセキュリティモデルを定義する．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.

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 $D = \{d_1, \ldots, d_n\}$ 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，$|d_1|, \ldots, |d_n|$, 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 n と **Dmap** の上限 N.
- 検索に使われるキーワード w の ID $ID(w)$ と w を含むドキュメント ID（アクセスパターンと呼ばれる）. 攻撃者は $ID(w)$ 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）.

非適応型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，$(D, \mathcal{Q})$ を任意に作成する．ここで，$D = \{d_1, \ldots, d_n\}$ はドキュメントの集合，$\mathcal{Q} = \{q_1, \ldots, q_m\}$ はクエリの集合である．その後，$\mathcal{A}$ は $(D, \mathcal{Q})$ 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，$((\mathbf{Kmap}, \mathbf{Dmap}), E_{sk_1}(D), Trap(Q))$ を，$BuildIndex(\ SK\ , D, \varepsilon)$, $Trapdr(Q)$ を使って作成する．その後, $\mathcal{C}$ は，$((\mathbf{Kmap}, \mathbf{Dmap}), E_{sk_1}(D),\ Trap(Q))$ を $\mathcal{A}$ に送る.

- $\mathcal{A}$ 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，$(D, \mathcal{Q})$ を任意に作成する．ここで，$D = \{d_1, \ldots, d_n\}$ はドキュメントの集合，$\mathcal{Q} = \{q_1, \ldots, q_m\}$ はクエリの集合である．その後，$\mathcal{A}$ は $(D, \mathcal{Q})$ を挑戦者 $\mathcal{C}$ に送る．
- $\mathcal{C}$ は，$|d_1|, \ldots, |d_n|$, $D(q_1), \ldots, D(q_m)$，n, N をシミュレータ $\mathcal{S}$ に送る．ここで，$|d_i|$ は d_i の長さ, $D(q_i) = \{j \mid d_j$ は q_i を含む $\}$.
- $\mathcal{S}$ は，$\mathcal{C}$ からの情報を使って，暗号化索引 $\Pi^* = (\mathbf{Kmap}^*, \mathbf{Dmap}^*)$ 暗号化ドキュメントの集合 $\{c_1^*, \ldots, c_n^*\}$, トラップドア T_1^*, $\ldots$, T_m^* 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.
- $\mathcal{A}$ 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，それは非適応型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，非適応型セマンティック安全である．

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，これらのメールデータから取り出した異なる 478520 個のキーワードで暗号化索引を構成した．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．実験は，**Kmap** は，最下位層の BF だけ，登録キーワード数の 20 倍のサイズとし，他の階層は 10 倍のサイズとした．**Dmap** は，簡単のため，$D(w)$ の分割を行わないで，そのまま保存した．各クエリについては，キーワードをランダムに 100 個選び，クエリに用いた．評価は，クエリから ID を求める時間 (**Kmap** の探索時間）とキーワードの ID からドキュメント ID を求める時間 (**Dmap** の探索時間）に分け，平均時間を計測した．表 1 に実験結果を示す．クエリからキーワード ID を求める時間は，平均で 444.78[ms] であるが，どのクエリも対応する ID は一つしかないため，ほとんど同じ時間でである．一方，キーワード ID からドキュメント ID を求める時間は，暗号化されたドキュメント ID のリストをファイルから読み出し，復号化している．そのため，平均時間は，1165.12[ms] であるが，暗号化されたドキュメント ID のサイズによってかなり時間に違いが出てくる．最も速い場合と遅い場合で 20 倍ほどの差が出る．

表 1: 100 クエリの平均検索時間（ms）

クエリから キーワード ID	キーワード ID から ドキュメント ID
444.78	1165.12

次に，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 478520 であるため，階層数は 21 である．登録キーワードの数はどの階層も同じで，478520 個である．また，各階層のブルームフィルタのサイズは，最下位層だけキーワード数の 20 倍で，他は 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，$478520 \times 10 \times 21 = 100489200$ ビットである．これは，約 12.5MB ほどになる．キーワード ID による転置索引 **Dmap** は，約 210MB 程になった．

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使うことにより，効率的な検索手法を提案した．今後の課題としては，セキュリティの強化，部分一致検索など柔軟な検索への対応が挙げられる．

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. Haciügumüs, 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] 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.

[12] 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.

[13] 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.

[14] D.X. Song, D. Wagner and A. Perrig, Techniques for Searchers on Encrypted Data, IEEE Symposium on Security and Privacy, pp.44–55, 2000.

[15] 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.

[16] 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.