問い合わせ内容を秘匿する 秘密分散型データベースシステムの検索効率に関する改良

山本 茉莉衣† 稲葉 宏幸‡

京都工芸繊維大学 大学院 工芸科学研究科 情報工学専攻
〒 606-8585 京都府京都市左京区松ケ崎橋上町 1
†myamamoto10@sec.is.kit.ac.jp, ‡inaba@kit.ac.jp

あらまし 秘密分散法によって保護されたデータベースに対して，ブルームフィルタを用いたプライバシ保護検索手法を適用することにより，安全かつ高速な検索を行う手法を提案する．2013 年に，著者らは問い合わせ内容を秘匿する秘密分散型データベースシステムを提案した．しかし，この手法では全てのデータに対して検索内容と一致するかどうかを確認しなければならず，検索効率が悪いという問題があった．本稿では，ブルームフィルタを生成する際に工夫することで，二分木構造を用いて検索を行う手法を提案する．

Improvements on Search Efficiency of Query Hiding Database System Using Secret Sharing Scheme

Marie Yamamoto† Hiroyuki Inaba‡

Dept. of Information Science, Kyoto Institute of Technology
Hashigami-cho 1, Matsugasaki, Sakyo-ku, Kyotoshi, Kyoto, 606-8585 JAPAN
†myamamoto10@sec.is.kit.ac.jp, ‡inaba@kit.ac.jp

Abstract In this paper, we propose a privacy preserving query method using Bloom filter for the database system that is protected by a secret sharing scheme. In 2013, we proposed a query hiding database system using secret sharing scheme. However, searching efficiency of this method is poor, because it is necessary to check whether a query matches the content for all of the data. In this paper, we propose a fast serching method by introducing binary tree structure into Bloom filter.

1 はじめに

近年，情報化社会の進展に伴い，個人情報の漏洩に対する意識が高まっている．ユーザが企業による個人情報の取り扱いに不安を感じ，個人情報保護法が施行されてからも情報漏洩事件は相次いでいる．特に個人情報は電子データ化され，単一のデータベースなどでまとめて管理されることが多い．そのため攻撃者は一度の攻撃に成功すれば多くの個人情報の入手が可能であり，一度の流出でも多くの被害が生じてしまう．

また，データベースサーバの管理は外部業者に委託されていることも多いため，問い合わせに応じてデータを一時復元する際にデータが露見しやすいという問題と，問い合わせ内容からデータの一部が推測される可能性があるという問題が生じる．

前者の問題に関しては，暗号化や秘密分散法を用いることでデータを復元することなく検索を効率よく安全に行う手法が多く提案されてい

る [1].

後者の問題に関しては，ブルームフィルタを利用する手法が知られている [2, 3]．ブルームフィルタとはある要素が集合に含まれているかどうかを効率的に調べるデータ構造で，これを検索用索引として利用し，検索時にはハッシュ値のみをやり取りすることによって問い合わせ内容を秘匿する．従来，この手法を暗号化データベースシステムに適用した例は報告されているが，秘密分散法を利用したデータベースシステムに適用する手法は知られていなかった．

秘密分散法は複数のサーバからデータが流出しない限りデータが復元されず，リスクの分散が可能であるという特徴を有しており，著者らは秘密分散法を利用したデータベースシステムにおいて検索内容を秘匿する手法を提案している [4]．これは，ブルームフィルタを用いて検索用索引を作成し，検索を行う手法である．これにより，サーバ側でデータの復元を行わずにデータの検索が可能になった．

しかし，この手法では検索時に全てのデータの索引を調べる必要があり，検索効率が良いとは言えない．そこで本稿では，この手法をブルームフィルタの生成方法を工夫することで改良し，二分木構造を用いて検索を行う手法を提案する．

2　関連研究

本節では，検索用索引として利用するブルームフィルタ [5] について簡単に説明した後，関連研究として，検索可能暗号を用いて二分木構造で検索を行う手法 [3] と，秘密分散法を用いて全検索を行う手法 [4] を紹介する．

2.1　ブルームフィルタ

ブルームフィルタとは 1970 年に Burton H. Bloom が考案した空間効率のよい確率的データ構造であり，要素が集合に含まれているかどうかを効率良く調べることができる．

ブルームフィルタを構成するためには，まず m ビットのビット列 $b_0, b_1, \cdots, b_{m-1}(b_i \in \{0,1\})$ を用意し，その初期値を全て 0 に設定する．ブルームフィルタに要素を追加するには，追加する文字列 x に対して，d 個のハッシュ関数 $H_1(x)$, $H_2(x), \cdots, H_d(x)(0 \leq H_i(x) < m(i = 1, 2, \cdots, d))$ を適用し，d 個のハッシュ値 $h_1, h_2, \cdots, h_d$ を得る．そして，それぞれのアドレスに対応するビットの値を 1 に設定する．すなわち，

$$b_{h_i} \leftarrow 1(i = 1, 2, \cdots, d) \tag{1}$$

とする．

また，ある要素がブルームフィルタに含まれているかどうかを判定するには，まず，先ほどと同様に d 個のハッシュ関数を利用して d 個のアドレス $h_1, h_2, \cdots, h_d$ を得る．そして，ブルームフィルタ内でそれぞれのアドレスが対応するビットが 1 であるかを確認し，対応する全てのビットが 1 である場合は，その要素は含まれていることになり，ひとつでも対応するビット内で 0 が含まれている場合は，その要素は含まれていないということになる．

しかしハッシュ値が衝突することにより，要素が含まれていないにもかかわらず含まれていると判断される場合がある．これを偽陽性という．

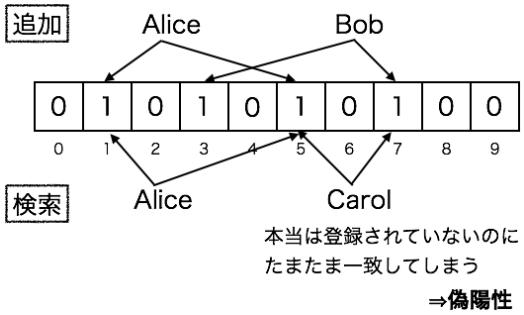

図 1: ブルームフィルタへの要素の追加・検索

2.2　単一のブルームフィルタを用いた高速な検索可能暗号方式

文献 [3] では，検索可能暗号で保護された文書データを，二分木の特性を利用して高速に検索する手法を提案している．

また，以下の説明では，文書番号 i の n ビットバイナリ表現を”0”,”1”からなる文字列で表

現したものを i のバイナリ文字列と呼び，バイナリ文字列 s の先頭から j 文字分のプレフィックスを $s^{(j)}$，文字列 s と t を連接した文字列を $s|t$ と表記する．

まず，準備として次のことを行う．

[準備]

- ユーザが所有する文書集合を $\mathbb{D} = \{D_i | i = 0, 1, \cdots, N-1\}$ とする
- 文書 D_i に含まれるキーワードの集合を $\mathbb{K}_i$ とする
- ユーザは $\mathbb{D}$ に含まれる各文書を暗号化関数 $E_k(\cdot)$ により暗号化し，暗号化文書集合 $\mathbb{E}$ を求める．つまり，$E_i = E_k(D_i)$ として，$\mathbb{E} = \{E_i | i = 0, 1, \cdots, N-1\}$ となる (k は暗号化鍵)
- ユーザは $\mathbb{K}_i$ $(i = 0, 1, \cdots, N-1)$ から検索用ブルームフィルタ $\mathbb{B}$ を作成する
- ユーザは $\mathbb{E}$ と $\mathbb{B}$ をサーバに送信し，保管を依頼する

ユーザは全ての文書 D_i について $K \in \mathbb{K}_i$ である全てのキーワードを $\mathbb{B}$ に登録する．$K \in \mathbb{K}_i$ を登録する手順は次の通りである．

1. 鍵付きハッシュ関数 $W_k(\cdot)$ を用いてキーワード K のハッシュ値 $w = W_k(K)$ を求める
2. 文書番号 i のバイナリ文字列を s とする
3. $j = 1, 2, \cdots n$ について登録用ハッシュ値 $H_l(w, s^{(j)})(l = 1, 2, \cdots, d)$ を求め，$H_l(w, s^{(j)})(l = 1, 2, \cdots, d)$ をブルームフィルタ $\mathbb{B}$ に登録する

[検索]

ユーザは検索キーワード K' を選び，ハッシュ値 $w' = W_k(K')$ を求める．w' をサーバに送信し，検索を依頼する．サーバは以下に示す手順により検索を行い，該当する文書をユーザに返信する．

1. $s = NULL$（空文字列）とし，文書 ID 集合 $\mathbb{I} = \{s\}$ と初期化する
2. ステップ 3 を n 回繰り返す
3. 文書 ID 集合 $\mathbb{I}$ に含まれる全ての要素 $s \in \mathbb{I}$ について以下の操作を行う．ただし，$\mathbb{I} = \phi$（空集合）であればステップ 4 に移る
   (a) 検索ハッシュ値 $x_l = H_l(w', s|\text{"0"})(l = 1, 2, \cdots, d)$ および $y_l = H_l(w', s|\text{"1"})(l = 1, 2, \cdots, d)$ を求める
   (b) $\boldsymbol{x} = (x_1, x_2, \cdots x_d)$ および $\boldsymbol{y} = (y_1, y_2, \cdots y_d)$ が $\mathbb{B}$ に登録されているかどうかに従って以下のいずれかを実行する
   (I) $\boldsymbol{x} \in \mathbb{B} \cap \boldsymbol{y} \notin \mathbb{B} \Rightarrow s \leftarrow s|\text{"0"}$
   (II) $\boldsymbol{x} \notin \mathbb{B} \cap \boldsymbol{y} \in \mathbb{B} \Rightarrow s \leftarrow s|\text{"1"}$
   (III) $\boldsymbol{x} \in \mathbb{B} \cap \boldsymbol{y} \in \mathbb{B} \Rightarrow$ s を $s|$"0" に更新するとともに $s|$"1" を新たに $\mathbb{I}$ に加える
   (IV) $\boldsymbol{x} \notin \mathbb{B} \cap \boldsymbol{y} \notin \mathbb{B} \Rightarrow$s を $\mathbb{I}$ から削除する
4. $\mathbb{I} = \phi$ であれば,「該当文書なし」と回答する．そうでなければ，$\mathbb{I}$ に含まれる全てのバイナリ文字列について対応する文書番号の暗号化文書をユーザに送信する

2.3 全検索を行う問い合わせ内容を秘匿する秘密分散型 DBS

文献 [4] の手法では，元のテーブルが，$id, value$ の 2 つの属性で構成されており，l 行のデータ $(id_1, id_2, \cdots, id_l$ および $value_1, value_2, \cdots, value_l)$ を格納するものとしている．このとき，j 番目の分散テーブルは id の分散情報 ${id_1}^{(j)}, {id_2}^{(j)}, \cdots, {id_l}^{(j)}$ と $value$ の分散情報 ${value_1}^{(j)}, {value_2}^{(j)}, \cdots, {value_l}^{(j)}$ に加え，各テーブルが持つ全ての行データごとに検索用索引となるブルームフィルタを持つ．元テーブルから各分散テーブルを生成する流れは図 2 の通りである．

検索用索引にはブルームフィルタを用いる．j 番目の分散テーブルにおいて，ある行データに対するブルームフィルタの生成手順は次のとおりである．

1. 行データが有するすべての属性に対して，d 種類のハッシュ関数 $H_i(\cdot)(i = 1, 2, \cdots, d)$ を適用し，ハッシュ値 $h_i = H_i(keyword)$ を計算

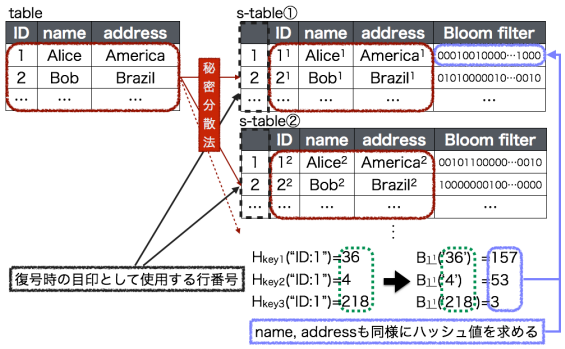

図 2: 分散フェーズの概要

2. ハッシュ値 h_i に対して，j 番目の分散テーブルにおける ID の値 $id^{(j)}$ を鍵としたハッシュ関数 $B_{id^{(j)}}(\cdot)$ を適用しハッシュ値 $b_i = B_{id^{(j)}}(h_i)$ を計算

3. ブルームフィルタに b_i を保存

ステップ 1〜3 をすべての属性に対して行ったものがその行の検索用索引となる．

なお，検索語 $(keyword)$ は検索したい単語の属性名と単語の値を「属性名:単語」のように組み合わせたものとする．例えば $ID = 1$ のデータを検索する場合の検索語は $keyword = "ID : 1"$ となる．

問い合わせはステップ 3 で生成した検索語を用いて行う必要があるため，データの一部分のみ（例えば名前に対する苗字や，住所のうち都道府県のみによる検索）で検索を行う可能性がある場合は，それらも含めて検索語を生成する必要がある．

また，検索は予め登録されている索引を用いて行う．手順は以下のとおりである．

1. ユーザは検索したい検索語に対して，ハッシュ関数 $H_i(keyword)$ を適用し，ハッシュ値 h_i を各サーバに送る

2. 受け取ったハッシュ値 h_i に対して，各行の $id^{(j)}$ の値を鍵として用い，生成手順のステップ 2 と同様にハッシュ値 b_i を計算

3. ブルームフィルタによるマッチングを行う

4. すべての行に対してステップ 2 およびステップ 3 を行い，一致したデータをユーザに送信する

ステップ 1〜4 が各テーブルにおいて実行され，ユーザは返信された分散データから元データを復元する．

3 提案手法

2.3 節で述べた既存手法では，復元に必要な分散テーブルのすべての行の探索を行うことになり，検索効率が良いとは言えない．そこで本論文では，二分木構造を用いて検索を行うことで検索効率を改善する．

3.1 提案手法の概要

本稿で対象とするデータベースのデータ構造として以下に述べるものを想定する．2.3 節で述べた分散情報に加え，”0”,”1”からなる $n(n \geq \log l)$ 桁のバイナリ文字列 ${label_1}^{(j)}, {label_2}^{(j)}, \cdots, {label_l}^{(j)}$，復元時の目印となる行番号 ${line_1}^{(j)}, {line_2}^{(j)}, \cdots, {line_l}^{(j)}$ と検索用索引をもつ．

次にシステムの詳細な処理について，データの格納，データの問い合わせ，データの復号の 3 つにわけて説明する．説明の際には以下の記号を用いる．

- ブルームフィルタ:$\mathbb{B}$
- 検索語：$keyword$
- 鍵付きハッシュ関数：$H_{key}(\cdot)$
- 検索語に対するハッシュ値: $h = H_{key}(keyword)$
- d 種類のハッシュ関数:$B_i(\cdot)(i = 1, 2, \cdots, d)$
- バイナリ文字列 ${label_a}^{(j)}$ の先頭から m 文字分のプレフィックス:${label_1}^{(j)(m)}(1 \leq m \leq n)$
- ブルームフィルタ登録時に使用するハッシュ値:$b_{(i,m)} = B_i(h, {label_a}^{(j)(m)})(1 \leq x \leq l)$

ハッシュ値を求める際に必要な鍵 key はユーザが決定し，保有するものとする．

3.2　データの格納

この節ではデータの格納時の処理について説明する．検索用索引としてブルームフィルタ $\mathbb{B}$ を用いる．

j 番目の分散テーブルの a 行目にデータを格納する手順は次のようになる．

1. 元テーブルのデータから，属性ごとに秘密分散法を用いて分散情報を生成する
2. ステップ 1 で生成した分散情報のうち，j 番目の分散情報をテーブルに格納する
3. 元テーブルの属性値から検索語 $keyword =$ "$ID : id_a$"および"$VALUE : value_a$" を生成する
4. ステップ 3 で求めたすべての $keyword$ に対して，ハッシュ関数 $H_{key}(\cdot)$ を適用し，ハッシュ値 h を計算する
5. ステップ 4 で求めたハッシュ値 h とバイナリ文字列 ${label_a}^{(j)(m)}(m = 1, 2, \cdots, n)$ に対して，ハッシュ関数 $B_i(\cdot)$ を適用し dn 個のハッシュ値 $b_{(i,m)}(i = 1, 2, \cdots, d)$ を計算する
6. $\mathbb{B}$ の $b_{(i,m)}$ 番目のビットを 1 にする

ステップ 1〜6 をすべての分散テーブルのすべての行に対して行う．行番号を格納することで，検索結果が複数あった場合や偽陽性が発生したときにデータの対応関係がわからなくなることを防いでいる．

元テーブルから各分散テーブルを生成する例を図 3，ブルームフィルタ $\mathbb{B}$ の作成方法を図 4 に示し，解説する．この例では属性として ID の他に $name$ と $address$ があると仮定している．図 3 では元テーブル table から分散テーブル s-table①，s-table②$\cdots$ を生成するものとし，$d = 2$ としている．

[データ格納の例]

1. table の各属性値に対して秘密分散法を適用し，分散情報を生成する．例えば $Alice$ からは $Alice^1, Alice^2, \cdots, Alice^t$ の t 個の分散情報が生成される

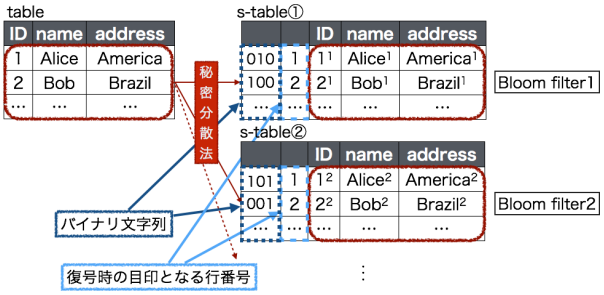

図 3: 提案手法のデータの格納方法の概要

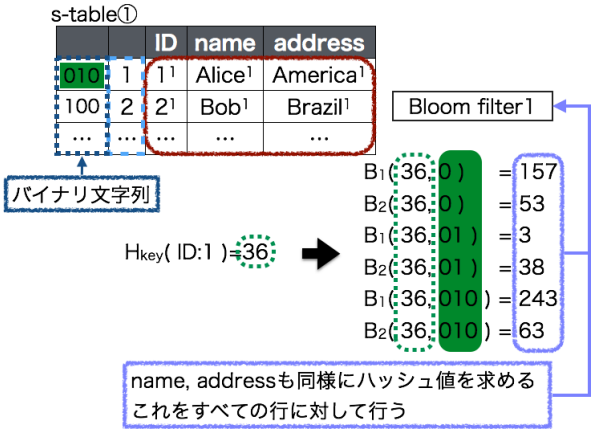

図 4: 提案手法のブルームフィルタ作成方法の概要

2. ステップ 1 で生成した分散情報 $1^1, Alice^1, America^1$ を s-table①に，$1^2, Alice^2, America^2$ は s-table②に $\cdots$ と順に格納する．このときこれらの分散情報に加え，行番号を元にしたハッシュ値を格納する
3. 検索語として，$ID : 1, name : Alice, address : America$ を生成する
4. ステップ 3 で生成した検索語のひとつ，$ID : 1$ に対してハッシュ値 H_{key}("$ID : 1$") を求め，ハッシュ値 36 を得たとする
5. ステップ 4 で求めたハッシュ値とバイナリ文字列'010' の 1 文字目のみ，2 文字目まで，3 文字目までを組み合わせたものに対してハッシュ関数を適用する．その結果，$B_1('36','0'), B_2('36','0'), B_1('36','01'), B_2('36','01'), B_1('36','010'), B_2('36','010')$ よりハッシュ値 157,53,3,38,243,63 を得る

6. $\mathbb{B}$ の 3,38,53,63,157,243 番目のビットを 1 にする

7. ステップ 4〜6 を他の検索語 $name : Alice$, $address : America$ についても同様に行う

3.3 データの問い合わせ

この節ではデータの問い合わせ時の処理について説明する．あらかじめ登録されているブルームフィルタ索引を用いて検索を行う．j 番目の分散サーバに対する問い合わせ手順は以下のとおりである．

1. ユーザは検索したい検索語 $keyword$ に対して，ハッシュ関数 $H_{key}(keyword)$ を計算し，ハッシュ値 h をサーバに送る

2. $s = NULL$ とし，$LABEL$ 集合 $\mathbb{L} = \{s\}$ と初期化する

3. $x_i = B_i(h, s|"0")$ および $y_i = B_i(h, s|"1")$ を計算し，ブルームフィルタによるマッチングを行う

4. $\boldsymbol{x} = (x_1, x_2, \cdots x_d)$ および $\boldsymbol{y} = (y_1, y_2, \cdots y_d)$ が $\mathbb{B}$ に登録されているかどうかに従って以下のいずれかを実行する
   (I) $\boldsymbol{x} \in \mathbb{B} \cap \boldsymbol{y} \notin \mathbb{B} \Rightarrow s \leftarrow s|"0"$
   (II) $\boldsymbol{x} \notin \mathbb{B} \cap \boldsymbol{y} \in \mathbb{B} \Rightarrow s \leftarrow s|"1"$
   (III) $\boldsymbol{x} \in \mathbb{B} \cap \boldsymbol{y} \in \mathbb{B} \Rightarrow s$ を $s|"0"$ に更新するとともに $s|"1"$ を新たに $\mathbb{L}$ に加える
   (IV) $\boldsymbol{x} \notin \mathbb{B} \cap \boldsymbol{y} \notin \mathbb{B} \Rightarrow$s を $\mathbb{L}$ から削除する

5. $\mathbb{L}$ に含まれる全てのバイナリ文字列について $label =$ "s" となるデータをユーザに送信する．ただし，$\mathbb{L} = \phi$ のときは一致データがないことを報告する

上記手順をユーザが任意に選んだ k 個の分散サーバについて繰り返せばよい．

例として図 3 の s-table①に対して，ID が 1 のタプルを問い合わせる手順を図 5 に示し，解説する．なお $d = 2$ としている．

[データの問い合わせの例]

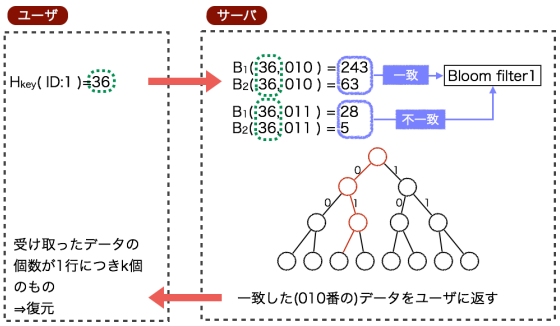

図 5: 提案手法の問い合わせ方法の概要

1. 検索語，$keyword = ID : 1$ に対してハッシュ関数 $H_{key}("ID : 1")$ を適用し，得られたハッシュ値 36 を s-table①に送る

2. $B_1('36','0'), B_2('36','1')$ を計算し，ブルームフィルタとのマッチングを行う

3. $B_1('36','0')$ のみが一致するので，バイナリ文字列"0"に"0"および"1"を連結し，$B_1('36','00'), B_2('36','01')$ を計算し，ブルームフィルタとのマッチングを行う

4. 同様の計算を行い，最終的なバイナリ文字列を LABEL として持つデータ (label="010") をユーザに送信する

上記手順をユーザが任意に選択した k 個のすべての分散テーブルに対して繰り返すことで，ユーザは欲しいデータ (この例では $ID = 1$ のデータ) を得ることができる．

3.4 データの復元

この節ではデータの復号時の処理について説明する．3.3 節の処理により k 個のテーブルからデータを得た後の手順は次のようになる．

1. 1 つの行番号につき k 個のデータがあるかどうかを確認する

2. k 個未満のデータしかない行番号のデータは削除する

3. ステップ 2 で残ったデータについて行番号ごとに復元を行う

ブルームフィルタには，偽陽性の問題があるため，一部の分散テーブルにおいて偽陽性が発生すると誤ったデータを検索結果として返してしまう場合がある．この問題点への対策として，ステップ 1~3 を行うことで偽陽性が発生したデータを削除する．

例として，3 つの分散テーブルを用意し，2 つのテーブルのデータを集めることでデータの復元が可能な (2,3) しきい値法を用いた場合の手順を説明する．$ID = 1$ のデータを検索した時，分散テーブル s-table①で偽陽性が発生し $ID = 1$(行番号 1) と $ID = 3$(行番号 3) の分散情報が，s-table②からは $ID = 1$(行番号 1) の分散情報が返還されたと想定する．

[データの復元の例]

1. すべての行番号につき 2 つのデータがあるかどうかを確認する
2. 3 行目の値 (s-table②から返ってきた $ID = 3$ の分散情報) は 1 つしかデータがないので削除する
3. 残ったデータ (s-table①からの $ID = 1$ の分散情報と s-table②からの $ID = 1$ の分散情報) に対し，復元を行う

4 評価

4.1 計算量

提案手法の計算量を 2.3 節で述べた既存手法と比較することで評価を行う．次のように記号を定義する．

- 使用するハッシュ関数の個数:d
- テーブルの行数:$l = 2^h$
- 復元に使うテーブルの個数:k
- 属性の数:c
- 二分木の深さ:h

それぞれの手法における計算回数，ブルームフィルタのメモリ (ビット"1"の個数) は表 1 のとおりである．ただし，計算回数はブルームフィルタにおける 1 回のマッチングを 1 回として数えるものとする．また，テーブルの行数 l は $2^{h-1} < l \leq 2^h$ の値をとるが，簡単のため $l = 2^h$ としている．

表 1: 既存手法と提案手法の比較

	既存手法	提案手法
計算回数 (最良)	$2^h k$	$2hk$
計算回数 (最悪)	$2^h k$	$(2^{h+1} - 2)k$
メモリ	cdl	$cdhl$

既存手法はどのような検索であっても，全ての行を探索する必要がある．そのため計算回数は一定であり，$2^h k$ となる．

一方，提案手法では二分木探索を行うので，最良の場合 (一致するデータが 1 件の場合) は $2hk$ となる．しかしこの手法では，一致する件数が多いほど，また一致するものが二分木上で分散して配置されているほど，二分木全体を探索する必要があり，計算回数は多くなる．例えば，図 6 のような場合，(a)(b) 共に一致データの個数は同じであるにも関わらず，(b) の場合は (a) の場合よりも比較回数が多くなる．このような場合では全検索を行う既存手法より計算回数が多くなる．

一致するものが最も分散して配置されている状態を想定する時，一致件数が $\frac{1}{2} \times 2^h$ 以上の場合は全て探索することになり，計算回数は $2^{h+1} - 2$ となる．一致件数が $\frac{1}{2} \times 2^h$ 未満のとき，一致件数が $\frac{1}{2}$ 倍になるごとに計算回数は減少する．一致件数が $\frac{1}{4} \times 2^h$ のとき，最下層の計算回数のみ $\frac{1}{2}$ 倍になるので，計算回数は $3 \cdot 2^{h-1} - 2$ となり，一致件数が $\frac{1}{8} \times 2^h$ のとき，最下層の計算回数は $\frac{1}{4}$ 倍であり，最下層の一つ上の層の計算回数が $\frac{1}{2}$ 倍になるので，計算回数は $2^h - 2$ となる．従って，提案手法が既存手法より有利になるのは一致件数が $\frac{1}{8} \times 2^h$ 以下の場合となる．

4.2 安全性

提案手法の安全性について述べる．

サーバがユーザから受け取るのは，秘密分散法によって分散されたデータとハッシュ値のみ

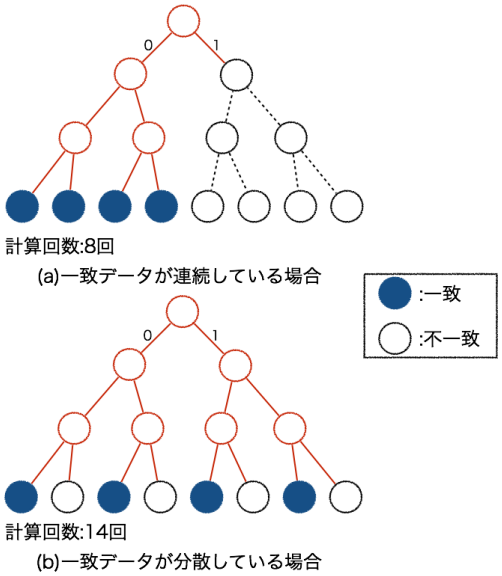

図 6: 計算回数の比較

である．検索時もデータを復元することなく，ハッシュ値のみを用いて探索を行うので，サーバ側にデータが漏えいすることはない．

ユーザがサーバに送信するのはハッシュ値のみであるため，サーバの管理者や，不正にログを取得した第三者は問い合わせ内容からデータの内容を推測することはできない．また，秘密分散法を用いているため，暗号化を用いたシステムに比べて鍵の紛失の心配がなく，1 つのサーバが攻撃を受けても情報が漏洩することはない．

本論文ではサーバ同士の結託，k 個以上のサーバからのデータ漏えいはないものと想定しているが，これらの事態が想定される場合には，各サーバに格納される行番号を暗号化することで対処が可能である．行番号が暗号化されている場合，その暗号鍵を持っていなければデータの対応関係がわからないため復元は難しいためである．

また，ブルームフィルタには，偶然にハッシュ値が一致してしまうことによって起こる偽陽性という問題があるが，その発生確率は非常に小さい．さらに提案手法では，一部のサーバのデータに偽陽性が発生してもそのデータは破棄されるため，偽陽性が問題になるのはすべてのサーバの同じデータに偽陽性が発生した場合のみである．この可能性は非常に小さく，実用上問題にならない．

5 むすび

個人情報はデータベースで管理されることが多く，一度の流出でも多くの被害が生じる．そこで本報告では，情報漏えい対策技術のひとつである秘密分散法に注目した．さらに，問い合わせ内容からデータベースの情報が露見することを防ぐプライバシ保護検索手法を適用することで，秘密分散法を用いた問い合わせ内容を秘匿するデータベースシステムを提案した．

また，提案手法では二分木で検索を行うことで，一致件数が少ない場合の検索効率を改善した．しかし，一致するものの割合が多い場合には全検索を行う場合よりも効率が悪くなってしまう場合がある．対策としては検索内容によって検索方式を切り替える方法などが考えられるが，詳細な手法については今後の課題とさせて頂く．

参考文献

[1] 櫻井友二，斉藤泰一，“情報漏えい防止データベースシステムの提案”，SCIS2006 予稿集, 1F2-5, 2006.

[2] 金子静花，渡辺知恵美，天笠俊之，“Semi-ShuffleBF:ブルームフィルタを用いた安全かつより高速なプライバシ保護検索手法の提案”，DEIM Forum2011 予稿集, C5-3, 2011.

[3] 木村俊介，稲葉宏幸：“単一のブルームフィルタを用いた高速な検索可能暗号方式とその評価”，信学技報, vol. 114, no. 494, SITE2014-79, pp. 227-232, 2015.

[4] 山本茉莉衣，稲葉宏幸：“問い合わせ内容を秘匿する秘密分散型データベースシステムの提案”，電子情報通信学会 2014 年総合大会講演論文集 D-9-18, 2014.

[5] B.H.Bloom，“Space/time trade-offs in hash coding with allowable errors,” Commun. of ACM, vol.13, no.7, pp.422-426, 1970.