DEIM Forum 2017 H6-4

暗号化データベースにおけるデータの秘匿性を保証した検索手法

篠塚　千愛† 　渡辺知恵美†† 　北川　博之†††

† 筑波大学大学院システム情報工学研究科　〒 305–8573 つくば市天王台 1 丁目 1–1
†† 筑波大学システム情報系情報工学域　〒 305–8573 つくば市天王台 1 丁目 1–1
††† 筑波大学計算科学研究センター　〒 305–8573 つくば市天王台 1 丁目 1–1
E-mail: †sn@kde.cs.tsukuba.ac.jp, ††chiemi@cs.tsukuba.ac.jp, †††kitagawa@cs.tsukuba.ac.jp

あらまし　DBaaS によるデータ提供では，データ提供者や利用者のプライバシを保証するため，データとクエリを秘匿したまま検索を行ないたい．そこで我々はこれまでに，暗号化されたデータとクエリを秘密計算で照合する検索フレームワークを提案している．しかし，データを m 分探索して検索するので，検索を繰り返すにつれて探索段階と選択確率からデータの並びを推測できてしまい，データの秘匿性が低下してしまう．本研究では，探索対象データの選択にランダム性を与え，探索段階ごとの選択確率の違いを取り除くことで解決を図る．

キーワード　プライバシ保護，暗号化データベース，秘匿検索，秘密計算

1. はじめに

近年，データ提供のための新たなプラットフォームとして，Database as a Service（DBaaS）が注目されている．DBaaS とはインターネットを介してデータベース機能を提供するクラウドコンピューティングサービスで，Amazon Relational Database Service [1]，Google Cloud Bigtable [2] などが有名である．データ提供者にとって DBaaS によるデータ提供は，サーバの購入や DBMS の構築に加え，バックグラウンドのデータベース保守を委託できるという利点がある．

しかしながら，格納データや検索クエリにはしばしば機密情報が含まれており，DBaaS のようにデータベース管理を第三者に委託する場合には注意が必要となる．管理者はデータベース内のデータやクエリログを自由に閲覧できるため，利用者のプライバシを侵害する恐れがあるからである．

信頼できない管理者にデータやクエリを秘匿する手法として暗号化データベースが提案されている [7,12]．暗号化データベースでは検索可能暗号を利用することにより，格納データを暗号化したままで管理や検索が可能にしている．しかし，暗号化データベースにおける検索処理は時間計算量が大きく，データへの高速なアクセスの保障を損ねてしまうため，データ提供への応用には適さない．そこで本研究では，DBaaS によるデータ提供事例を想定し，高速な検索処理が可能な暗号化データベースの実現を目指す．

索引（index）は，データベースのアクセス効率の向上に有効であるが，暗号化データベースのようにデータの内容を秘匿したい場合には妨げとなってしまいかねない．例えば，B+木索引の最下層ではキー値の順序を保存した並びで全データが格納されている．ここで，キー値自体が暗号化されていたとしても，平文時の順序関係が保持されていると，一部のキー値の平文値が漏洩した場合に残りのキー値の平文値を推測されやすくなることが指摘されている [5]．検索にかかる時間が短縮できるからといって，データの安全性が低下することは暗号化データベースとして望ましくない．

我々はこれまでに，索引のキー値と順序関係を分散管理し，秘密計算技術 [4] によって暗号化データベースにおいてデータとクエリを秘匿しつつ索引検索を実現する Oblivious Secure Index Traversal（OSIT）フレームワークを提案している [16,17]．OSIT では，まず索引を暗号化してクライアントとサーバで分散管理する．これにより，クライアントは検索対象の属性値に基づく各レコードの順序関係を知っており，サーバは各レコードの属性値を暗号化した情報を知っていることになる．データを検索する際は，クライアントは自身の所有する索引を m 分探索して，クエリ条件に該当するレコードを発見する．ただし，クライアントとサーバはそれぞれ不完全な状態で索引を所有しており，各レコードの属性値に関する情報をもたないクライアント単独では索引を探索できない．そこで，クライアントはクエリと比較したいレコードの属性値をサーバに問い合わせ，秘密計算を用いたプロトコルによって双方の索引の情報を明かさずにそれらの大小関係を比較する．

しかし，サーバ管理者はクライアントからのアクセスログに基づいて検索対象の属性値に基づく各レコードの順序関係を推測できてしまう．そこで本研究では，アクセスログから各レコードの順序関係を推測できない索引探索手法を提案する．

本論文の構成は以下の通りである．まず，第 2 節で関連する研究について述べる．第 3 節で提案手法を説明する上で必要となる前提知識を，第 4 節で OSIT フレームワークを紹介する．第 5 節で提案手法について述べ，第 6 節で評価実験について述べる．最後に，第 7 節で本稿のまとめと今後の課題を述べる．

2. 関連研究

暗号化データベースは，格納データを暗号化したままで管理や検索が可能なシステムで，2002 年に Hacigümüş ら [7] によって提案されて以来，現在までに多くの研究がなされている．

[7] では，予め格納データを一定の範囲ごとに分割してから暗号化しておき，サーバで分割データに対する検索を行ない，クライアントで検索結果を復号して精査する検索手法が提案されている．また，Hore らによってデータ値の統計的推測が困難な分割データの生成手法が後に提案されている [8]．

初期の暗号化データベースでは集約演算を始めとした復号を必要とする検索に対応できなかったが，Mykletun ら [10] や Ge ら [6] は準同型暗号スキームを利用し，集約演算や k–近傍探索に対応した暗号化データベースを実現した．さらに 2011 年には，Popa がこれらの手法を組み合わせ，より複雑な検索に対応できる暗号化データベース CryptDB [12] を提案した．CryptDB では，RND や DET 暗号，Paillier 暗号 [11] などの複数の暗号化スキームを用いて，格納データを多層的に暗号化する．この多層的な暗号化により，格納データの機密性を保証するとともに，各暗号化スキーム層ごとに選択・射影・結合・集約といった異なる演算をサポートすることができる．CryptDB はオープンソースパッケージとして公開されており，今なお実用的な暗号化データベースとして大いに注目されている．また，Stephen らにより CryptDB の検索処理最適機構を改良した Monomi [13] が提案されている．

2.1 秘密計算を利用した索引検索手法

CryptDB などの多くの暗号化データベースでは，暗号化されたすべてのデータとクエリとを一つ一つ照合して検索を行なうため，大量のデータを扱う場合に時間計算量が大きくなってしまう．この問題に対し，一般的なデータベースと同様に索引検索（Index Search）が実現できれば，パフォーマンスの向上を見込むことができる．しかし，暗号化データベースで索引検索を実現するためには，格納データの秘匿性を低下させずに暗号化されたデータとクエリを比較しなければならない．

これに対して，Wang ら [14] は R 木に非対称内積保存暗号スキーム [15] を R 木に適用して暗号化されたデータからクエリの近傍点を発見するアルゴリズムを提案し，これを用いて暗号化データベース上で索引による多次元範囲検索を実現している．また，Hu らの Oblivious Index Traversal フレームワーク [9] では，暗号化データベースを対象に B+木による索引検索を実現している．この手法では，索引の各エントリを Paillier 暗号で暗号化しておき，クライアントとサーバが協力して暗号化されたエントリとクエリの大小関係を秘密計算 [4] する．

3. 事前準備

3.1 DBaaS によるデータ提供

本研究では DBaaS によるデータ提供事例を，データ提供者，サーバ管理者，クライアントで成立するモデルとして扱う．データ提供者は，DBaaS で利用可能なクラウド上のデータベースで，収集・解析したデータを管理する．サーバ管理者は，データ提供者に代わってバックグラウンドのデータベース管理を行なう．クライアントは，クエリ q をデータベースに問い合わせ，取得したデータ量に応じてデータ提供者に課金する．

DBaaS によるデータ提供では，データ提供者のデータプライバシ及びクライアントのクエリプライバシが保証されることが望ましい．

3.1.1 データプライバシ

データの機密を秘匿したいという要求をデータプライバシと呼ぶ．DBaaS を利用する際，データ提供者が自身の資産であるデータをサーバ管理者に秘匿したいと考えることは自然である．また，データ提供を行なう際には，課金対象とならないデータに関する情報をクライアントになるべく与えたくないと考えるであろう．

3.1.2 クエリプライバシ

クエリの機密を秘匿したいという要求をクエリプライバシと呼ぶ．サーバ管理者はクエリログを自由に閲覧できるため，過去に発行されたクエリの内容から興味関心や行動履歴といったクライアント個人に関する情報を把握し得る．このような個人情報の漏洩を恐れて，クライアントは発行したクエリの内容をサーバ管理者に秘匿したいと考えることがある．

3.1.3 攻撃者モデル

本研究では，攻撃者としてサーバ管理者とクライアントを想定する．サーバ管理者が攻撃者である場合，攻撃者は格納データの内容やクライアントの個人情報を入手しようとする．クライアントが攻撃者である場合，攻撃者は不正な問合せを行なうことで課金対象とならないようにデータを取得しようとする．いずれの攻撃者も semi-honest モデルとして扱い，サーバ管理者とクライアントは結託しないものとする．

3.2 加法準同型暗号

平文 m_1, m_2 の暗号文 $E(m_1), E(m_2)$ が与えられたとき，平文や秘密鍵なしに $E(m_1+m_2)$ を導出できる性質を加法準同型性といい，加法準同型性をもつ暗号を加法準同型暗号という．加法準同型暗号には lifted-Elgamal 暗号や Paillier 暗号 [11] などがあり，本稿では Paillier 暗号を使用する．

3.2.1 Paillier 暗号

Paillier 暗号は，以下の 4 つの機能（鍵生成・暗号化・復号・準同型演算）を有する．

［鍵生成］秘密鍵 $sk=(p,q)$ および公開鍵 $pk=(n,g)$ を生成する．p,q は任意に選ばれた大きな素数で，n はそれらの積である．g は，任意に選ばれた $k\in\mathbb{Z}_n$ に対して，$g=1+kn \bmod n^2$ を満たす値である．

［暗号化］平文 m と公開鍵 pk を入力として，暗号文 $E(m)=g^m\cdot r^n \bmod n^2$ を出力する．ただし，r は $r\in\mathbb{Z}^*_{n^2}$ を満たす乱数である．この r により二つの同じ平文から必ずしも同じ暗号文が生成されるとは限らない．

［復号］暗号文 $E(m)$ と秘密鍵 sk を入力として，平文 m を出力する．

［準同型演算 1］二つの暗号文 $E(m), E(m')$ を入力として，暗号文 $E(m+m')$ を出力する．$E(m')=g^{m'}\cdot s^n \bmod n^2$ として，$E(m+m')$ は式 1 のように計算できる．

$$E(m)\cdot E(m')=g^{m+m'}\cdot(r\cdot s)^n \bmod n^2=E(m+m') \quad (1)$$

［準同型演算 2］暗号文 $E(m)$ と定数 a を入力として，暗号文 $E(am)$ を出力する．$E(am)$ は式 2 のように計算できる．

$$E(m)^a=g^{am}\cdot r^n \bmod n^2=E(am) \quad (2)$$

4. OSIT フレームワーク

本節では，索引を用いて効率的に暗号化データベースを検索するための Oblivious Secure Index Traversal（OSIT）フレームワークを紹介する．本フレームワークの主要な特徴として，データ提供者は検索のための索引を構築し，それをサーバとクライアントで分散管理する．また，秘密計算を用いたプロトコルによって双方の索引の情報を明かさずに索引検索を実現する．

4.1 問題定義

暗号化データベース $D = \{r_i\}$ に関して，レコード $r_i = (A_1, A_2, ..., A_j)$，検索対象となる数値属性を $\mathcal{A}$ とする．また，クライアントが発行できるクエリを $\mathcal{A}$ に対する範囲検索クエリ Q とし，式 3 で定義する．クライアントは検索範囲の下限値 min 及び上限値 max を指定すると，範囲内に $\mathcal{A}$ 値が収まるようなすべてのレコードを取得できる．

$$Q(min, max) := \{r_i \in D | min \leqq r_i.\mathcal{A} \leqq max\} \qquad (3)$$

本研究では，D, min, max が与えられたとき，Q を高速に回答するという問題を考察する．ただし，3.1 節で示したプライバシを保証するために，Q を導出する際には以下の三つの要件を満たさなければならない．

要件 1 — サーバ管理者は，データベースに格納されているデータの内容を推測できない．

要件 2 — サーバ管理者は，クエリ内容を推測できない．

要件 3 — クライアントは，発行したクエリ条件に該当しないレコードの情報を得られない．

4.2 暗号化索引

OSIT フレームワークでは索引に起因する情報漏洩を防ぐために，暗号化索引の構築ならびにクライアントとサーバによる暗号化索引の分散管理する．以降では，OSIT フレームワークで使用する索引を暗号化索引と呼ぶ．

4.2.1 索引のデータ構造

暗号化データベース D に対する暗号化索引 I を，キー i とエントリ $\langle E(e_i), E(r_i)\rangle$ を紐付けるハッシュ索引とし，式 4 で定義する．ただし，e_i はレコード r_i 中の属性 $\mathcal{A}$ の値であり（つまり $e_i = r_i.\mathcal{A}$），i は全レコードを e の順に並べたときのインデックスを表す．

$$I\colon i \to h(i) \to \langle E(e_i), E(r_i)\rangle^+ \qquad (4)$$

$E(\cdot)$ は暗号化関数を表し，e_i は加法準同型暗号スキームで，r_i は任意の暗号化スキームでそれぞれ暗号化されているものとする．ここで，暗号化索引のサイズ $|I| = N$ とする．

4.2.2 索引の構築と配置

暗号化索引の構築と配置は，データ提供者が予め行なう．

データ提供者は最初に全レコードについて $\langle e_i, r_i\rangle$ のペアを列挙し，e の値が小さい順に並べたときのインデックス i を求める．そして，$\langle E(e_i), E(r_i)\rangle$ を計算して，ハッシュテーブルの $h(i)$ をキーとするエントリに割り当てる．このとき，i の衝突が発生しないようにハッシュ関数 $h(\cdot)$ を設定する．

続いて，構築した暗号化索引をクライアントとサーバで分散管理するために，I を順序情報 SI とエントリ情報 EI に分離する．SI, EI をそれぞれ式 5，式 6 で定義する．

$$SI\colon i \to h(i) \qquad (5)$$

$$EI\colon h(i) \to \langle E(e_i), E(r_i)\rangle^+ \qquad (6)$$

ここでは，i と $h(i)$ の対応を SI としてクライアントに配置する．一方，$h(i)$ と $\langle E(e_i), E(r_i)\rangle^+$ の対応を EI としてサーバに配置する．$\langle E(e_i), E(r_i)\rangle^+$ について，集合自体を 1 つのデータとして暗号化することで $E(e_i)$ が同じ値をとるレコード数もサーバに秘匿することができる．

Algorithm 1 *ObliviousSecureIndexTraversal* (client side)

Require: $N, h(\cdot), min, max$
Ensure: クエリに該当するレコード集合 $Result = Q(min, max)$

```
Result ⇐ {}
i_min ⇐ getRecordIndex(N, min, "≦")
i_max ⇐ getRecordIndex(N, max, ">")
for i : i_min to i_max do
    h(i) に対応する E(r_i) を取得
    r_i ⇐ D(E(r_i))
    Result.add(r_i)
end for
```

Algorithm 2 *getRecordIndex* (client side)

Require: N, クエリ条件値 q, クエリタイプ $type$
Ensure: 該当するレコードのインデックス i_q

```
1:  i_q ⇐ null
2:  探索領域 (l, u) ⇐ (1, N)
3:  E(q) をサーバに送信
4:  while l < (u − 1) do
5:      (l, u) を m 分割する i を (m − 1) 個選択
6:      h(i_1), h(i_2), ..., h(i_{m−1}) をサーバに送信
7:      procedure at server
8:          for j : i_1 to i_{m−1} do
9:              h(j) に対応する E(e_j) を取得
10:             E(c) ⇐ {E(e_j) · E(q)^{−1}}^r
11:             E(c) をクライアントに送信
12:         end for
13:     end procedure
14:     E(c_1), E(c_2), ..., E(c_{m−1}) を復号
15:     c_1, c_2, ..., c_{m−1} に基づき (l, u) を更新
16: end while
17: if l ≠ N or u ≠ 1 then
18:     if type is "≦" then
19:         i_q ⇐ u
20:     else if type is ">" then
21:         i_q ⇐ l
22:     end if
23: end if
```

4.3 索引検索アルゴリズム

分散配置された暗号化索引 I の探索を含む検索全体の流れを Algorithm1 に記す．一般的なデータベースでは索引をサーバ側で探索するが，OSIT フレームワークでは I をクライアント側で探索する．ここで，クライアントはデータ提供者から I の

構築時に使用した暗号化スキームの暗号化関数 $E(\cdot)$ 及び対応する復号関数 $D(\cdot)$ を事前に与えられているものとする．クライアントは，クエリ条件 $[min, max)$ を指定して I を探索することで，min より大きい最小の $\mathcal{A}$ 値をとるレコードのインデックス i_{min} と，max より小さい最大の $\mathcal{A}$ 値をとるレコードのインデックス i_{max} を発見する．ただし，I のエントリ情報 EI を知らないため単独では探索ができないので，EI をもつサーバと協力して探索を実行する．最後に，i_{min} 番目から i_{max} 番目のレコードをサーバに問い合わせ，結果を復号することでクエリに該当するレコード集合 $Q(min, max)$ を得られる．

分散配置された暗号化索引 I を探索するための秘密計算プロトコルを Algorithm2 に記す．I を探索するには，サーバで管理される属性値 e とクライアントが指定するクエリの条件値 q の大小を比較する必要が生じる．クライアントもサーバも互いに自身のもつ値を明かしたくないので，両者が協力して秘密計算をして大小関係を導出する．このプロトコルは，クライアントが e_i, q の大小関係を知るためには，$e_i - q$ の正負が分かればよいというアイディアに基づく．まず，4.1 節の要件 2 を満たすために q を暗号化した上でサーバに送信する．次に，索引の探索領域 (l, u) 内で比較したいエントリを $(m-1)$ 個選択し，$h(i_1), h(i_2), ..., h(i_{m-1})$ をサーバに送信する．サーバでは $h(i)$ に対応するエントリの $E(e_i)$ を取り出し，$E(c) = \{E(e_j) \cdot E(q)^{-1}\}^r$ を計算する．Paillier 暗号は加法準同型性をもつため，この結果は $E(r(e_i - q))$ となる．なお，クライアントに $E(e_i - q)$ を明かしてしまうと e_i が知られてしまって要件 3 を満たさないので，$E(e_i - q)$ に乱数 r を掛けて正確な差を分からなくしている．ただし，真の差とで正負が反転しないよう r は正値とする．クライアントはサーバから返された $E(c)$ を復号すると，c の正負から比較したいエントリとクエリの大小関係を得られる．

5. サーバでの平文値推測の防止

OSIT フレームワークでは暗号化索引の順序情報とエントリ情報を分散管理することで，暗号化されたエントリそのものと平文時の順序関係を共存させないようにしていた．しかし，クライアントがどのエントリにアクセスしたかを知ることができる．サーバ側で順序関係を推測できてしまう．

[例 1] クライアントはサイズ N の暗号化索引を二分探索するとする．まず最初に，クライアントは索引中央に位置するエントリ $e_{\frac{1}{2}N}$ にアクセスする．このときサーバでは，クライアントが二分探索を開始した直後であると考えると，$e_{\frac{1}{2}N}$ が索引の $\frac{1}{2}N$ 番目に位置すると推測できる．次に，クライアントはエントリ $e_{\frac{1}{4}N}$ にアクセスするとする．このときサーバでは，クライアントが二分探索を行っていることを考えると，$e_{\frac{1}{4}N}$ が索引の $\frac{1}{4}N$ 番目あるいは $\frac{3}{4}N$ 番目のいずれかに位置すると推測できる（サーバは $e_{\frac{1}{2}N}$ とクエリの大小関係を知ることはできないため，これ以上の絞り込みはできない）．

文献 [5] で指摘されている通り，サーバにエントリ値の順序関係を知られてしまうと，たとえエントリ値が暗号化されていたとしても選択平文攻撃に弱くなる．この問題に対して，アクセスするエントリの選択規則の改善手法を提案する．

5.1 アクセスするエントリの選択規則の改善

提案手法では，クライアントは暗号化索引を m 分探索するために，1 往復の通信ラウンドで k 個のエントリにアクセスする．ただし，k は後述する式 16 に基いて決定される．

まず初回の問合せでは，クライアントはランダムに選択した k 個のエントリにアクセスする．ただし，k 個のエントリは重複を許す．ここで得られた各エントリとクエリの大小関係 $E(c_1), E(c_2), ..., E(c_k)$ を基に，2 回目以降の問合せで m 分探索する索引領域 (l, u) を決定する．これにより，クライアントは検索時に毎回異なる索引領域を m 分探索することになる．

2 回目以降の問合せでは，クライアントは (l, u) を m 分割する $m-1$ 個のエントリに加え，(l, u) 外の索引領域からランダムに $k-(m-1)$ 個のエントリを選択して，サーバにクエリとの大小関係を問い合わせる．ランダムに選択されるエントリにもアクセスすることで，m 分探索のためにアクセスされたエントリをサーバに分からなくしている．なお，得られる $E(c_1), E(c_2), ..., E(c_{m-1}), ..., E(c_k)$ のうち，ランダムに選択したエントリとの大小関係は (l, u) の更新に影響しないため復号処理を省略できる．

5.2 安全性評価

提案手法を用いた場合，クライアントからのアクセスログに基づきサーバ管理者がエントリの順序を推測できないことを示す．

まず，索引としてソートされた配列 A を考える．

$$A = (e_1, ..., e_i, ..., e_N) \quad (7)$$
$$\forall i \forall j, i < j \text{ ならば } e_i < e_j$$

A に対して，e_i の値を暗号化し，A 上での順序関係がわからないようにした集合を A' として式 8 で示す．

$$A' = \{e'_1, e'_2, ..., e'_N\} \quad (8)$$

ここで，e'_i は配列 A 上のいずれかのエントリを暗号化したものであり，A と A' 間で各エントリの添字の順序には関係がない．以降では，A' はサーバで管理されるものとする．

任意の $e' \in A'$ について，e' を復号したエントリが A 上のどの位置であっても同様に確からしいとき，$1 \leqq j \leqq N$ のすべての j に対して e' が e_j を暗号化したものである確率は $\frac{1}{N}$ である．これと同等であれば A 上の位置を知ることができないとみなせる．このことを根拠として，定義 1 を導く．

[定義 1]（エントリの位置推測の防止）クライアントがアクセスしたエントリリスト $AccessLog$ を式 9 とする．

$$AccessLog = (e'_{q1}, ..., e'_{qk} | e'_{qi} \in A') \quad (9)$$

ここで $e' \in AccessLog$ が，あるエントリ $e_i \in A$ を暗号化したものである確率が，$1 \leqq i \leqq N$ のすべての i において高々 $\frac{1}{N}$ であるとき，サーバ管理者は e' の A 上での位置を推測できない．

[定理 1]（提案手法によるエントリの位置推測の防止）提案手法において，サーバ管理者は $AccessLog$ に含まれる任意のエントリ e' に対し，A 上での位置を推測できない．

証明. 提案手法ではクライアントが 1 往復の通信ラウンドで k 個のエントリにアクセスする．以降の議論では，クライアントがアクセスしたエントリ $e'_j \in A'$ が $e_i \in A$ を暗号化したものである確率を $P(e'_j, e_i)$ で表す．

1 ラウンド目はランダムに選択したエントリにアクセスするので，$e'_{q1}, e'_{q2}..., e'_{qk} \in AccessLog$ について，e'_{qj} があるエントリ $e_i \in A$ を暗号化したものである確率は式 10 の通りである．

$$P(e'_{qj}, e_i) = \frac{1}{N} \tag{10}$$

続いて，2 ラウンド目以降のアクセスについて考える．2 ラウンド目以降は，1 ラウンド目で決定された探索領域を m 分探索するようにエントリを選択する．まず通常の手法で A を m 分探索をする場合を考える．A の e_1 から e_N の間で均等な位置にある $m-1$ 個のエントリ $e_{\lceil \frac{N}{m} i \rceil} (1 \leqq i \leqq m-1)$ にアクセスする．そのため，$e'_j (1 \leqq j \leqq m-1)$ が $e_{\lceil \frac{N}{m} i \rceil} (1 \leqq i \leqq m-1)$ である確率は式 11 で表される．

$$\forall i \forall j \in [1, m-1],\ P(e'_j, e_{\lceil \frac{N}{m} i \rceil}) = \frac{1}{m} > \frac{1}{N} \tag{11}$$

ところで，提案手法の 2 ラウンド目の場合は 1 ラウンド目で探索する領域が狭められている上に領域の開始位置がランダムに決められているためサーバ側には探索領域がわからない．そのため，2 ラウンド目で $e_i (1 \leqq i \leqq N)$ を暗号化したものにアクセスした確率を考えるには，すべてのあらゆる探索領域（考え得るすべての領域長，開始位置）で m 分探索することを考える必要がある．このとき，$P(e', e_i)$ は式 12 で表される．

$$P(e', e_i) \tag{12}$$
$$= \frac{m-1}{k} \frac{1}{N-m+2} \sum_{l=m-1}^{N} \frac{1}{N-l+1} \sum_{e'=1}^{N-l+1} f(l, s, i)$$

ここで $f(l, s, i)$ は，探索領域の領域長を l，開始位置を s としたとき，領域を m 分割する分岐点が均等になる部分の確率が $\frac{1}{m-1}$ となることを表す．

$$f(l, s, i) = \begin{cases} \frac{1}{m-1} & if\ i = \frac{l}{m} j + s\ (\exists j, 1 \leqq j \leqq m-1) \\ 0 & if\ otherwise \end{cases}$$

式 12 中の $\sum_{e'=1}^{N-l+1} f(l, e', i)$ は，m 分探索する領域のスタート地点を 1 から $N-l$ に動かす間に $m-1$ 回 $f(l, e', i)$ が $\frac{1}{m-1}$ になるときを最大値にとる．すべての広さの探索範囲でこうなることはないが，上限として以下の値を設定できる．

$$P(e', e_i) \leqq \frac{1}{N-m+2} \sum_{l=m-1}^{N} \frac{1}{N-l+1} \tag{13}$$
$$\leqq \frac{1}{N-m+2} \int_{1}^{N-m+2} \frac{1}{x} dx \tag{14}$$
$$= \frac{\log(N-m+2)}{N-m+2} \tag{15}$$

ここで，k の値が式 16 を満たせば，提案手法における 2 サイクル目以降のアクセスで $P(e'_j, e_i) \leqq \frac{1}{N}$ となる．

$$k \geq \frac{N(m-1)\log(N-m+2)}{N-m+2} \tag{16}$$

以上から，提案手法で常に $P(e'_j, e_i) \leqq \frac{1}{N}$ が成立する． □

5.3 コストモデルの導出

本節では，提案手法によって暗号化索引を m 分探索する際に，最適な m を決定するための索引探索のコストモデルを提案する．

本稿では，($t_{comm}, t_{comp}, t_{dec}$) をそれぞれ，1 往復の通信に必要なコスト，1 個のレコードとクエリの比較に必要な秘密計算のコスト，1 個の暗号化された比較結果の復号に必要なコストとし，1 往復の通信による 1 個のレコードとクエリの比較コストを定義 2 に示す．

[定義 2]（1 往復の通信ラウンドによる単一エントリとクエリの比較コスト）1 往復の通信に必要な通信コストが t_{comm} であり，1 個のエントリとクエリの比較に必要な秘密計算コストが t_{comp}，1 個の暗号化された比較結果の復号コストが t_{dec} であるとき，1 往復の通信ラウンドによる 1 個のエントリとクエリの比較コスト Cost を式 17 で定義する．

$$\text{Cost} = t_{comm} + t_{comp} + t_{dec} \tag{17}$$

提案手法で暗号化索引を m 分探索する場合，1 往復の通信ラウンドで k 個のエントリとクエリを比較する．このとき，初回の問合せではランダムに選択したエントリすべてとクエリの比較結果を得たいので，$t_{comm}, t_{comp}, t_{dec}$ は送信エントリ数 k に依存する．ところが，2 回目以降の問合せでは m 分探索に必要なエントリとクエリの比較結果のみを得られればよいので，t_{comm}, t_{comp} は送信エントリ数 k に，t_{dec} は m 分探索のための比較エントリ数 $m-1$ に依存する．よって，定義 2 に従って 1 往復の通信ラウンドによる k 個のエントリとクエリの比較コストを定理 2 に示す．

[定理 2]（1 往復の通信ラウンドによる k 個のエントリとクエリの比較コスト）提案手法により暗号化索引を m 分探索する場合，1 往復の通信ラウンドによる k 個のエントリとクエリの比較コストは式 18 のようになる．ただし，式中の m は $2 \leqq m$ とする．

$$\text{Cost}_{first}(m, k) = k(t_{comm} + t_{comp} + t_{dec}) \tag{18}$$
$$\text{Cost}_{otherwise}(m, k) = k(t_{comm} + t_{comp}) + (m-1)t_{dec}$$

提案手法では初回の通信ラウンドでエントリをランダムに選択し，それらの比較結果に基づいて決定された索引領域を 2 回目以降の通信ラウンドで m 分探索する．故に検索毎に m 分探索する索引領域の広さが変化するが，ここでは説明を簡単にするために初回の通信ラウンドで均等にエントリを選択して 2 回目以降の通信ラウンドで m 分探索する索引領域の広さを固定してコストモデルを導出する．暗号化索引のサイズが N であるとき，2 回目以降の索引探索に必要な通信ラウンド数は $\lceil \log(N-k)/\log(m) \rceil$ であるので，2 回目以降の探索開始から終了までの比較コストは $\text{Cost}_{otherwise}(m, k)$ に $\lceil \log(N+1)/\log(m) \rceil$ をかけたものになる．これを提案手法の検索コストモデルとし，定理 3 に示す．

[定理 3]（提案手法のコストモデル）探索対象の暗号化索引のサイズが N であるとき，提案手法における検索コストモデルは，最悪のケースで上限として式 19 のようになる．

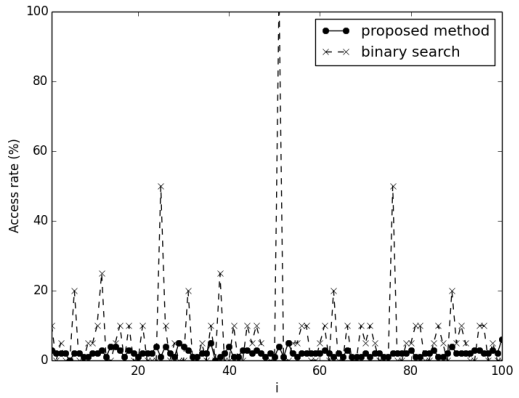

図 1　各エントリのアクセス確率の比較

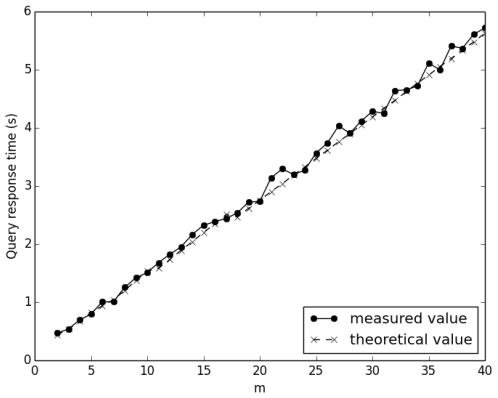

図 2　クエリ応答時間の実測値とコストモデルによる理論値の比較

$$\begin{aligned}&\textsc{TotalCost}(N,m,k) \qquad (19)\\ &\quad = \textsc{Cost}_{first}(m,k)\\ &\quad + \lceil \log(N-k)/\log(m) \rceil \textsc{Cost}_{otherwise}(m,k)\end{aligned}$$

5.4　各種パラメータの決定

提案手法では，式 16 ならびに式 19 に基づいて k, m を決定する．クライアントは，まず式 16 から様々な m に対する k の下限値を得る．続いて，($t_{comm}, t_{comp}, t_{dec}$) と先程導出した m, k の組合せのうち式 19 が最小となる m, k を得る．

6.　評 価 実 験

6.1　実 験 設 定

すべての実験は，Mac OS X，Intel Core i5 @ 1.70GHz CPU，4GB RAM で構成されたマシン（クライアント端末）と，Ubuntu 12.04，Intel Xeon L563 @ 2.13GHz CPU，2GB RAM で構成されたマシン（サーバ）を使用して実施した．プログラムは Java で実装し，JRE 64bit version 1.8 上で実行した．なお，Paillier 暗号の実装は公開されているプログラム [3] を使用した．

6.2　アクセスされるエントリの一様性の検証

提案手法における各エントリのアクセス確率を検証する．本実験では，$m = 2$ とした場合の各エントリのアクセス確率について，提案手法と通常の二分探索を比較する．各エントリのアクセス確率は，索引サイズ $N = 100$ のデータセットに対して，様々なクエリごとに 50 回ずつ検索を行ない，そのときのアクセス回数から算出した．なお，式 16 に基づき，$k = 10$ と設定した．比較結果を図 1 に示す．通常の二分探索では二分木のようなプロットが得られ，提案手法ではより水平に近い線がプロットされた．

6.3　コストモデルの妥当性の検証

様々な m に対するクエリ応答時間を実際に測定して，提案したコストモデルの妥当性を検証する．ここで，本実験環境下では t_{comm} は 9.60 ミリ秒，t_{comp} は 0.02 ミリ秒，t_{dec} は 0.08 ミリ秒となっており，これに基づいて式 19 から様々な m に対するクエリ応答時間の理論値を算出する．

本実験では，m を 2 から 40 の間で変化させたときのクエリ応答時間について，実測値と理論値を比較する．また，k は式 16 に基づいて設定した．実測値は，索引サイズ $N = 100,000$ のデータセットに対して各 m ごとに 10 回ずつ検索を行ない，平均クエリ応答時間を算出した．比較結果を図 2 に示す．実測値と理論値の傾向がほぼ一致しており，式 19 で定めたコストモデルが妥当であると言える．また，$m = 2$ のときにクエリ応答時間が最短となっていることがわかる．

7.　お わ り に

本稿では，Oblivious Secure Index Traversal（OSIT）フレームワークにおいて，サーバ管理者による格納データ（暗号文）の平文値推測の防止を目的とし，アクセスログからデータ間の順序関係を推測できない索引探索手法を提案した．続いて，提案手法によってサーバ管理者がアクセスログを利用して暗号化されたエントリの平文時の順序関係を推測することが困難であることを理論的および実験的に示した．また，提案手法のコストモデルを定義し，実験を通して妥当性を検証した．

今後の課題として，様々な索引サイズと通信環境の組み合わせにおけるクエリ応答時間の評価が挙げられる．提案手法では格納データの秘匿性を確保するため，クライアントは実際の探索に必要なエントリ以外にも複数のエントリについてサーバに問い合わせなければならない．問い合わせるべきエントリ数は索引サイズに応じて増加するが，通信コストが大きい環境では現実的な時間で検索できなくなる可能性がある．

謝　　辞

本研究の一部は JSPS 科研費 基盤研究 (C) JP16K00149 および　平成 28 年度共同研究（SKY 株式会社） (CPE27116K)「機械学習の適用による SKYSEA Client View のログ及び資産情報からの例外的状況の自動検出」の助成を受けたものです．

文　　献

[1] Amazon Relational Database Service. https://aws.amazon.com/rds/.
[2] Google Cloud Bigtable. https://cloud.google.com/bigtable/.
[3] Google Code Archive (thep). https://code.google.com/

archive/p/thep/.

[4] Ian F Blake and Vladimir Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 515–529. Springer, 2004.

[5] Alexandra Boldyreva, Nathan Chenette, Younho Lee, AdamO ' neill. Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 224–241. Springer, 2009.

[6] Tingjian Ge and Stan Zdonik. Answering aggregation queries in a secure system model. In *Proceedings of the 33rd international conference on Very large data bases*, pp. 519–530. VLDB Endowment, 2007.

[7] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pp. 216–227. ACM, 2002.

[8] Bijit Hore, Sharad Mehrotra, and Gene Tsudik. A privacy-preserving index for range queries. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pp. 720–731. VLDB Endowment, 2004.

[9] Haibo Hu, Jianliang Xu, Xizhong Xu, Kexin Pei, Byron Choi, and Shuigeng Zhou. Private search on key-value stores with hierarchical indexes. In *2014 IEEE 30th International Conference on Data Engineering*, pp. 628–639. IEEE, 2014.

[10] Einar Mykletun and Gene Tsudik. Aggregation queries in the database-as-a-service model. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 89–103. Springer, 2006.

[11] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223–238. Springer, 1999.

[12] Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: processing queries on an encrypted database. *Communications of the ACM*, Vol. 55, No. 9, pp. 103–111, 2012.

[13] Stephen Tu, M Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing analytical queries over encrypted data. In *Proceedings of the VLDB Endowment*, Vol. 6, pp. 289–300. VLDB Endowment, 2013.

[14] Peng Wang and Chinya V Ravishankar. Secure and efficient range queries on outsourced databases using rp-trees. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pp. 314–325. IEEE, 2013.

[15] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 139–152. ACM, 2009.

[16] 篠塚千愛, 渡辺知恵美, 北川博之. Daas におけるデータとクエリ双方のプライバシー保護を実現する効率的な秘匿検索. 第 7 回データ工学と情報マネジメントに関するフォーラム（DEIM フォーラム 2015）, 2015.

[17] 篠塚千愛, 渡辺知恵美, 北川博之. 秘密計算による秘匿検索フレームワーク osit-bs における検索コストと最適化. 第 8 回データ工学と情報マネジメントに関するフォーラム（DEIM フォーラム 2016）, 2016.