秘匿検索フレームワークOSITを利用したデータ提供サービス

秋山　賢人† 　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
†† 産業技術大学院大学産業技術研究科
††† 筑波大学計算科学研究センター
E-mail: †k_akiyama@kde.cs.tsukuba.ac.jp, ††chiemi@acm.org, †††kitagawa@cs.tsukuba.ac.jp

あらまし　クラウドサービスの発達により，DBaaS を利用してデータ所有者の持つ膨大なデータをクラウドに保管し，第三者に検索を許可することでデータ提供サービスが可能になりつつある．データ提供を行う上でサーバに保存されたデータやクライアントの発行するクエリのプライバシ保護は重要である．我々はこれまでにこれらのプライバシを保護しつつ，索引を利用して高速に検索を行うことが可能な暗号化データベースシステム OSIT を提案している．データ所有者は，OSIT 上でサービス提供することで安全かつ高速なサービスを行うことが可能になる．その一方で，データ所有者がデータ提供サービスを行うためにはクライアントにどの程度検索を許可するかということも検討しなければならない．これについても，われわれは検索によりクライアントが得る情報をエントロピーで定量化する手法を提案したが，定量化を誰が行うかも検討しなければならない．データ所有者，クライアント，サーバが定量化を行うことを検討したが，いずれも実用的なシステムにするためには困難であることが分かった．そこで本稿では，エントロピー計算を信頼できるサーバに委託し，定量化された値を利用してクライアントの検索を制御する OSIT でのデータ提供サービスモデルを提案する．

キーワード　暗号化データベースシステム, プライバシ保護, Database as a Service

1. はじめに

クラウドコンピューティングの発達により, DBaaS(Database as a Service) が広く提供されている．DBaaS とはネットワークを介してデータベースの機能を提供するサービスであり, データ所有者がサービスプロバイダの管理するサーバにデータの管理を委託する事ができる．DBaaS を利用したデータ提供サービスでは，データ所有者がサーバにデータを保存し，クライアントにデータへのアクセス権を提供する．これにより，クライアントはサーバとの間で問合せを行うことが可能となる．代表的な DBaaS には, Amazon RDS (注1), Google Cloud Bigtable (注2) などがある．DBaaS では, サービスプロバイダがデータ所有者に代わってデータベース管理を行うことで，データ所有者の負担軽減が可能である．また，データ量に応じたデータベースを柔軟に拡張することも可能である．

しかし, データ所有者が機密データを預ける場合, 機密データやクエリの漏洩により引き起こされるプライバシの侵害は問題となる．機密データやクエリに関するプライバシを保護するために, データやクエリに関するプライバシの要求がある．

データプライバシに関する要求

・データ所有者はサーバ管理者に対し, 預けるデータの中身を知られたくない.

クエリプライバシに関する要求

・クライアントはデータ所有者およびサーバ管理者に対し, 発行したクエリの内容を秘匿したい.

これらの要求を満たすため，CryptDB [20]，Monomi [22], SDB [23] などの暗号化データベースシステムが提案されている．暗号化データベースシステムでは，データ所有者がデータの機密性を保障するためにデータを暗号化してクラウドサーバへ保存する．クライアントはクエリを検索可能暗号で暗号化し，データとクエリを暗号化したまま安全に検索を行える．

篠塚らは, データとクエリのプライバシを保護しつつ暗号化索引を利用した効率的な検索を可能にする検索可能暗号フレームワーク OSIT(Oblivious Secure Index Traversal) [26] を提案している．OSIT は, 索引を暗号化してクライアントとサーバで分散管理し，クライアントがサーバ上の索引を探索することで効率的な検索を実現している．また，準同型暗号でデータとクエリを暗号化し，暗号化したまま検索を行うことで安全性を保証している．

クラウドを利用したデータ提供サービスを考えた場合，サービスプロバイダに対してデータやクエリの機密性を保障するだけでなく，クライアントに対してのデータ保護の保障が必要となる場合がある．それは，データ所有者がクライアントに対して無制限にデータを提供するのではなく，クライアントに検索権の範囲内でサービスを利用してもらうということである．OSIT では，サービスプロバイダからデータを秘匿しつつ高速な検索を行うことができるが，クライアントの検索を制限する仕組みは検討されていない．Deep ら [5] や Koutris ら [13] [15]

(注1)：https://aws.amazon.com/jp/rds
(注2)：https://cloud.google.com/bigtable/?hl=ja

は，クライアントの発行したクエリベースの価格設定関数を提案している．これらの研究ではクライアントが受け取るクエリ結果に価格を設定し，価格を支払うことでクライアントにデータが提供されるという仕組みとなっており，クライアントに渡る情報を定量化して把握することで，クライアントによる検索を制御することができると考えられる．先行研究 [27] では, クライアントが問合せによって得た知識をエントロピーで定量化する手法を提案した．しかし，エントロピーによる定量化を誰が行い, クライアントの検索を制御するのか検討していない．

そこで本研究では，OSIT を利用したデータ提供サービスのモデルを提案する．本研究では [27] で提案した定量化のための計算をデータ所有者, クライアント, サービスプロバイダの誰が行うのか検討した．そして, 定量化した値を基にクライアントが持つ検索権の範囲内で問合せを行うことができるデータ提供サービスのモデルを OSIT 上で検討する．

本研究の貢献を以下に示す．

- OSIT を利用したサービス提供のモデルを提案した．OSIT を利用することで安全かつ効率的に検索を行えるだけでなく，クライアントが問合せにより得る情報をエントロピーで定量化することで，定量化された値に基づきクライアントの検索権を超えた場合には検索を制御可能である．
- 実データを用いた実験により，クライアントが問合せで得る情報をエントロピーで定量化した．また，繰り返し問合せを行うことでエントロピーが単調に減少し，クライアントに全てのデータが渡った場合にエントロピーが 0 になることを示した．更に，OSIT とクエリ結果の開示制御の処理時間を比較し，全体の処理時間に占める開示制御の処理時間の割合が少なく実用的であることを示した．

2. 関 連 研 究

2.1 暗号化データベースシステムに関する研究

暗号化データベースシステムに関する研究は Hacigümüs ら [10] に提案されて以降, 数多く取り組まれている．Hacigümüs らはクライアントとサーバの問合せ処理モデルを定義した．Hore ら [11] は Hacigümüs ら [10] の手法を基に, 統計による値の推測を困難にするサーバ側のデータ生成法を提案した．暗号化データベースに利用される暗号化手法として，Agrawal ら [1] は数値属性の順序関係を保存可能な順序保存暗号を提案した．また，Mykletun ら [18],Ge ら [7] により, 準同型暗号を利用した集約演算を可能にするサーバ側でのデータ暗号化手法や, k-近傍探索なども提案されている．Popa ら [20] は，これらの手法を組合せて暗号化データベースシステム CryptDB を提案した．CryptDB は, RND, DET, Paillier 暗号 [19] などの異なる暗号化手法を用いて機密データを幾重にも暗号化することで機密性を保証し，暗号化手法ごとに異なるクエリの処理を行うことで, キーワード検索，結合演算，大小比較など様々な演算をサポートしている．Stephen らは，CryptDB の検索処理最適化機構を改良した Monomi [22] を提案している．

近年では，準同型暗号や検索可能暗号を利用した検索手法が注目を集めている．準同型暗号には, 暗号化したまま暗号文同士の加算を行える Paillier 暗号 [19] のような加法準同型暗号，乗算を行える ElGamal [6] 暗号のような乗法準同型暗号，加算と乗算の両方を行える完全準同型暗号がある．完全準同型暗号には，Gentry らが提案した整数上完全準同型暗号 [8] がある．しかし，Gentry らの手法は計算時にビットごとの演算が必要となるため実用的でないことも知られている [9]．検索可能暗号にはキーワード検索が可能なキーワード検索公開鍵暗号 (PEKS) [4] が Boneh らによって提案されている．

暗号化データベースシステムにおいて, 関係代数のようにクエリを構成する各演算の間に相互運用性 (interoperability) を持たせることも課題となっている．Wong ら [23] は相互運用性のある加法準同型暗号を提案し，暗号化データベースシステム SDB を提案した．SDB では演算の中間結果をクライアントに転送することなく, サーバ上で暗号化したまま演算の出力結果を次の演算の入力にすることが可能である．

これまでに述べた暗号化データベースシステムは高い安全性を保証する一方で, 大規模データに対して検索を行う場合には良いパフォーマンスを示さない．データ数が大きい場合, これらの手法では各レコードが検索条件に一致するかを確認しなければならないため，検索に時間がかかる．

この問題に対し, Hu ら [12] はデータベース索引を用いた安全な検索フレームワーク OIT を提案した．OIT では，クライアントが所有する索引をクライアント側で探索することで安全性を確保している．しかし, クライアントに対するプライバシ要件が侵害されるとクライアントに大きな負荷がかかるという問題がある．篠塚ら [26] は，データとクエリを保護しつつ暗号化索引を利用して効率的な検索を行う OSIT フレームワークを提案している．OSIT はクライアントがサーバ上の索引を探索し，数値属性に対する範囲検索と完全一致検索に加えて，文字列検索 [25] が可能である．

2.2 データ提供に関する研究

データ提供に関する研究として，クラウド上でデータの販売を行うことを想定した研究が多く取り組まれている [14] [17] [16]. Balazinska ら [2] はクラウド上でのデータ販売における課題を示した．Koutris ら [13] はクエリベースのデータの価格設定フレームワークを提案している．その後，Koutris ら [15] は実験的に価格設定フレームワークの評価を行った．近年では，Deep ら [5] がシャノンエントロピー，最小エントロピーなどの様々なエントロピーを利用した理論的な価格設定方法を提案している． [13], [5] では，クライアントが受け取るクエリ結果に価格を設定し，価格を支払うことでクライアントにデータが提供されるという仕組みとなっており，クライアントに渡る情報を定量化して把握することで，クライアントによる検索を制御することができる．

3. 前 提 知 識

3. 節では，本稿を説明するための基本知識について述べる．DBaaS 環境における OSIT のシステムモデル，攻撃者モデルについて述べる．

3.1 システムモデル

OSIT のシステムモデルについて述べる．OSIT はデータ所有者，サービスプロバイダ，クライアントの 3 者からなる．データ所有者は，自身の持つ機密データをサービスプロバイダが管理するサーバに保存する．機密データは重要な資産であるため，データ所有者はサービスプロバイダにデータの内容を知られたくない．そこで，データ所有者はサーバへ保存する前に機密データを暗号化する．サービスプロバイダはデータ所有者に委託された機密データを管理するとともに，クエリ実行のために強力な計算リソースを提供する．

クライアントはクエリ q を発行し，サーバからデータを取得する．サービスプロバイダはクエリログを知ることができるが，クライアントにとってクエリの内容を知られることは望ましくない．クエリの内容を知られることを防ぐため，クライアントはクエリを暗号化し，クエリ処理を暗号化したまま行う．

3.2 攻撃者モデル

OSIT の攻撃者モデルについて述べる．OSIT において，クライアントとサービスプロバイダは semi-honest [24] であり，結託しないと仮定する．また，攻撃者にはサービスプロバイダを想定し，サービスプロバイダはサーバに保存されたデータ，索引，クエリログからレコードに含まれる属性値の順序関係を推定しようとする．攻撃者が属性値の順序関係を入手した場合，選択平文攻撃 [3] により属性値の推定が可能なため，攻撃者に索引のエントリの順序を推定されることは望ましくないと言える．OSIT は以下の定義を満たすことで高い安全性を提供する．

定義 1. 索引のエントリに関する識別不可能性

N を索引のエントリ数, $E = \{e_1, \ldots, e_i, \ldots, e_N\}$ をエントリ集合，$E' = \{e'_1, \ldots, e'_j, \ldots, e'_N\}$ を暗号化されたエントリ集合とする（ただし，E' と E' に含まれる要素の順序は対応していない）．攻撃者が暗号化されたエントリの順序を正しく推定できる確率が以下の式を満たす時，*OSIT* は安全である．

$$P(D(e'_j) = e_i) \leqq \frac{1}{N} \tag{1}$$

$P(D(e'_j), e_i)$ は復号された e'_j と e_i が同じエントリである確率である．

4. Oblivious Secure Index Traversal (OSIT)

OSIT [26] は暗号化索引を利用して安全かつ効率的に検索を行う暗号化データベースシステムである．OSIT では，クライアントとサーバが索引を分散管理し，クライアントがサーバ上の索引を探索するという特徴を持つ．これにより，完全な索引の構造を知らないまま索引探索を行うことができる．以降で，OSIT での索引の構築方法及び索引探索方法について述べる．

4.1 OSIT における索引の構築

図 1 に OSIT での索引の構築及び管理方法を示す．データ所有者は 1 つ以上の属性を持つリレーショナルテーブルを持つとし，テーブルをもとに暗号化索引を構築する．

まず，データ所有者は索引を作る属性を決め，*key* と *value* のペア $\langle k, v\rangle$ を作成する．k は属性値を，v は属性値が k であるレコード集合を表す．続いて，データ所有者は k の値で昇順ソートした配列を索引とし，$\langle k, v\rangle$ にソートされた配列上での位置を示すアドレス *addr* を加える．よって索引の構造としては $\langle addr, k, v\rangle$ のようになり，データ所有者は k と v を Paillier 暗号などの加法準同型暗号 $E(\cdot)$ で暗号化し，暗号化索引とする．図 1 において, 索引のエントリ数は N，*key* 値は $k_1, \ldots, k_i, \ldots, k_N$ $(k_1 < \ldots < k_i < \ldots < k_N)$ である．

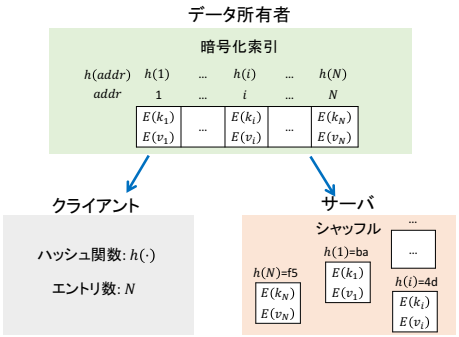

図 1　暗号化索引の管理.

暗号化索引はクライアントとサーバで分散管理する．データ所有者は索引のエントリの順序関係をサーバに秘匿するため暗号化索引にある *addr* の値を SHA-256 などの暗号学的ハッシュ関数でハッシュする．図 1 では，データ所有者は $h(1), \ldots, h(i), \ldots, h(N)$ を計算し”ba”, ..., ”4d”, ..., ”f5”を得る．そして，データ所有者は暗号化索引のエントリ情報 $\langle h(addr), E(k), E(v)\rangle$ について，順序関係を秘匿するためシャッフルしてサーバに送り，索引の構造情報であるハッシュ関数 $h(\cdot)$ とエントリ数 N をクライアントに送る．データ所有者はクライアントに索引探索を行うための $E(\cdot)$ の暗号鍵, 復号鍵も送る．

4.2 OSIT における索引探索

ここからは OSIT での索引の探索方法について述べる．OSIT での索引探索の基本的な考え方は暗号化索引の m 分探索である．しかし，m 分探索をそのまま適用すると，サーバに索引の順序関係が分かってしまう．例えば，エントリ数を N とし，$m = 2$ の場合を考えると，クライアントはクエリ q と *key* を比較するため最初の探索で $\frac{1}{2}N$ 番目のエントリ $e_{\frac{1}{2}N}$ にアクセスする．2 分探索をしているので, サーバはアクセスしたエントリについて索引上の位置が $\frac{1}{2}N$ 番目であると推測することができる．2 回目の探索では，クライアントは $\frac{1}{4}N$ 番目のエントリ $e_{\frac{1}{4}N}$ または $\frac{3}{4}N$ 番目のエントリ $e_{\frac{3}{4}N}$ にアクセスする．すると，サーバは $\frac{1}{4}N$ 番目または $\frac{3}{4}N$ 番目のエントリであると推測することができる．Boldyreva ら [3] はサービスプロバイダがエントリの順序関係を知っている場合，索引のエントリが暗号化されていたとしても選択平文攻撃に脆弱であると述べている．

この問題に対処するために，OSIT では安全な m 分探索アルゴリズムを提案している（アルゴリズムについては [26] を参照）．図 2 はクエリ値 q と i 番目のエントリの *key* 値 k_i との比較方法を図示している．図では，クライアントは i 番目のエントリとクエリ値 q を比較するため $h(i)$=ba を計算して暗号化されたクエリ値 $E(q)$ とともにサーバに送る．サーバは $E(q)$ と *key* 値 $E(k_i)$ の大小関係を式 2 で暗号化したまま計算し，ク

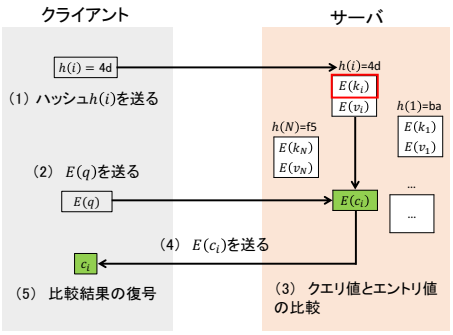

図 2　OSIT での索引探索.

ライアントに大小比較結果 $E(c_i)$ を送る．クエリ値と key 値の大小比較演算は Paillier 暗号 [19] の性質に基づき計算される．$E(c)$ を大小比較結果, $E(k)$ を暗号化した key 値, $E(q)$ を暗号化したクエリ値, r をランダムな正の整数とすると，クエリ値と key 値の大小比較は式 2 で計算できる．

$$E(c) = (E(k) \cdot E(q)^{-1})^r = E(r(k-q)) \qquad (2)$$

サーバで式 2 を計算することで q や k をサーバに明かさずに計算でき，クライアントで復号することでクライアントのみが q と k の大小比較結果を知ることができる．その後，クライアントは比較結果の復号を行うことで索引を探索し, 問合せ結果に該当する索引のアドレスを取得する．

5. クライアントが問合せで得る情報の定量化手法

我々はこれまでに, 問合せでクライアントが得る情報をエントロピーで定量化する手法を提案している [27]．提案手法ではクライアントの検索を制御するために [27] で提案した定量化手法を利用するが，検索対象が大規模データの場合，[27] の定量化法は計算時間がかかってしまうため，本稿では計算を改善したものについて説明する．OSIT ではクライアントがサーバ上の索引を探索して検索を行う．簡単のため，今回は問合せ前，クライアントはレコード数，属性値の最大値 max，最小値 min，「SELECT age FROM db WHERE age $= x$」のような問合せを実行した場合を考えると，クライアントは索引の探索によって x が索引上の何番目の key 値に該当するか知ることができ，検索結果から x を満たすレコード数を知ることができる．これにより，OSIT でクライアントが各属性に対して持つ知識をヒストグラムで表すことができる．例えば，図 3 のようにレコード数 n，最小値 min, 最大値 max の属性 age に対して「SELECT age FROM db WHERE age $= x$」$(x = a_1, ..., a_k)$ を問合せることを考えると，問合せ前は図 3 左側のように $min \sim max$ のビンに n 件のレコードが入っている状態であり，クライアントはどんな分布であるか分からない．続いて,「age= a_1」を問合せ後の age のヒストグラムを考えると図 3 中央のようになる．n_1 は $min \sim (a_1 - 1)$ に含まれるレコード数，r_1 は a_1 に含まれるレコード数，n_2 は $max \sim (a_1 + 1)$ に含まれるレコード数である．問合せを繰り返して「age= a_k」を問合せ後の age のヒストグラムを考えると図 3 右側のようになる．r_k は a_k に含

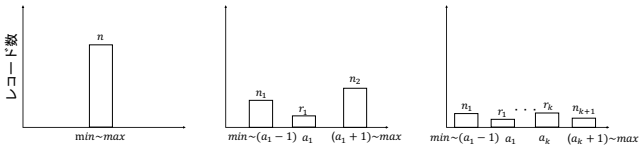

図 3　問合せ前後の age のヒストグラム.

まれるレコード数，n_{k+1} は $max\ (a_k + 1)$ に含まれるレコード数である．

このように OSIT において繰り返し問合せを行うことで，ヒストグラムが分割されて詳細になっていくことが分かる．また, 問合せを繰り返すにつれて明らかになっていないビンに含まれるレコード数が変化することが分かる．従って, 明らかになっていないビンに対してヒストグラムの作り方の総数を考えると作り方の総数が減少し, クライアントは検索していない属性値に対してもヒストグラムの形が推定できると考えられる．

これにより，検索が行われていない属性値からなるヒストグラムについて，クライアントがヒストグラムを当てる確率をクライアントの得た情報としてエントロピーで定量化する．クライアントが問合せを繰り返すことで不確定部分のヒストグラムの作り方の総数は減少し，エントロピーは単調に減少すると考えられる．この性質を利用してサービス提供前にクライアントが閾値を設定することでエントロピーの値と閾値を比較して検索を制御できると考える．

以下，ヒストグラムの総数 T と k 回問合せした時にあるヒストグラムであると分かる確率 p の求め方について述べる．様々な分布を持つ属性に対して評価を行えるように, 検索を行っていない属性値については一様にレコードが分布していると仮定する．k 回問合せした際に問合せた属性値の値を $a_1, \ldots, a_k$（$a_1 < \ldots < a_k$）とし，分割された各部分に含まれるレコード数を $n_1, \ldots, n_{k+1}$（$n_1, \ldots, n_{k+1}$ は不確定部分のヒストグラムについて属性値の異なり数の比率と同様の比率で不確定部分のレコード総数から分配される）とすると図 3 右側のようにヒストグラムを分割することができるので，k 回問合せした時の不確定部分のヒストグラムの作り方の総数 T は $a_0 = min - 1$, $a_{k+1} = max + 1$ とおくと以下の式 3 で表せる．

$$T = \prod_{i=0}^{k} \frac{(n_{i+1} + a_{i+1} - a_i - 1)!}{n_{i+1}!(a_{i+1} - a_i - 1)!} \qquad (3)$$

ここで, n_{i+1} は a_i と a_{i+1} の間にあるビンに入るレコード数を, $a_{i+1} - a_i - 1$ は a_{i+1} と a_i の間にある属性値の異なり数 (ビンの数) を表す.

なお, ヒストグラムの作り方を計算する際, サーバ側から見た場合, 各ビンに割り当てられるレコードの組合せを考えるとヒストグラムの作り方は式 4 のように計算することが可能である.

$$T = \prod_{i=0}^{k} (a_{i+1} - a_i - 1)^{n_{i+1}} \qquad (4)$$

しかし, クエリ結果受け取り時にクライアント側ではどのレコードの組合せでヒストグラムが構成されているかを知ることができない．そのため, 式 3 でヒストグラムの総数を考える．

本研究では, 不確定部分のヒストグラムが作られる可能性は全て同様に確からしいと想定する．よって，k 回問合せした時にあるヒストグラムである確率 p は以下の通りになる．

$$p = \frac{1}{T} \tag{5}$$

従って，エントロピーは以下の式 6 で表せる．

$$H = T \times -\frac{1}{T} log(\frac{1}{T}) = log(T) \tag{6}$$

エントロピーを計算する際，式 6 は，$\alpha_i = n_{i+1}$，$\beta_i = a_{i+1} - a_i - 1$ とおいて以下の式 7 のように変形して計算することができる．

$$\begin{aligned} log(T) &= log(\prod_{i=0}^{k} \frac{(\alpha_i + \beta_i)!}{\alpha_i! \beta_i!}) \\ &= \sum_{i=0}^{k} log(\frac{(\alpha_i + \beta_i)!}{\alpha_i! \beta_i!}) \\ &= \sum_{i=0}^{k} (\sum_{j=1}^{\alpha_i+\beta_i} log(j) - \sum_{j=1}^{\alpha_i} log(j) - \sum_{j=1}^{\beta_i} log(j)) \\ &= \sum_{i=0}^{k} (\sum_{j=\alpha_i+1}^{\alpha_i+\beta_i} log(j) - \sum_{j=2}^{\beta_i} log(j)) \end{aligned} \tag{7}$$

各問合せ時に式 7 を計算することで，クライアントが問合せで得る情報を定量化することができる．また，式 7 は階乗計算を行わないため高速に計算を行えると考えられる

6. 提 案 手 法

本研究では OSIT を利用したデータ提供サービスのモデルを提案する．クラウドを利用したデータ提供を利用する事を考えた時，データを安全に提供するために以下の要求がある．

- サービスプロバイダに対してデータやクエリの機密性を保障する．
- クライアントに対して無制限にデータを提供したくない．

OSIT では 1 つ目の要求を満たしているが，2 つ目の要求を満たすためのクライアントの検索の制御する仕組みは検討していない．

本研究では，どのように 2 つ目の要求を満たし，クライアントによる検索を制御するか検討する．まず, 本研究で提案する OSIT を利用したデータ提供サービスがどのようなものであるか概要を説明する．OSIT を利用してデータ提供サービスを行う際, データ所有者がサーバに預けた機密データに対し, クライアントとサーバの間で問合せを行う．サービスの利用手順としてはクライアントは事前に検索権である閾値をサービス利用開始前に設定する．そして, 問合せ結果取得前に 5. 節で説明した方法でクライアントが問合せで得た知識を定量化し, 定量化した値が設定した閾値を超えている場合, クライアントに問合せ結果が返されるといったサービスを提供する．

ここからは，クエリ結果の開示制御を誰が行うべきか，OSIT を利用したデータ提供のシステムモデルについて述べる．

6.1 クライアントが問合せで得る情報によるクエリ結果の開示制御を誰が行うかの検討

OSIT 上でデータ提供サービスを行う際, クライアントのクエリに対してサーバが結果を返すかどうかの判定をするため, サーバでクライアントが問合せで得る情報の計算と判定をするのが自然である．しかし, 本システムが満たすべきプライバシ保護要件を含めて考える必要がある．そこで, 我々はサーバに加えて, クライアントおよびデータ所有者の三者に対し, クライアントが問合せで得る情報の計算と制御を行うのに適するか否かの検討を行った．

表 1　サーバ, クライアント, データ所有者によるクライアントが得た情報の計算及び検索制御における利点と欠点のまとめ．

サーバ	利点	高い計算能力を持つマシンで計算できるため，高速に定量化を行うことができる．
	欠点	定量化した値により, サーバにクライアントがどの程度情報を得ているか知られてしまう．
クライアント	利点	クライアントの得た情報に基づき定量化を行える．
	欠点	クライアントはデータ所有者，サーバに知られず定量化の結果を改竄することができる．
データ所有者	利点	クライアントの得た情報を基にクライアントの検索を制御する判定を正しく行うことができる．
	欠点	検索の制御をデータ所有者が行わなければならないため, データ所有者の負担が増えてしまう．

サーバがエントロピー計算を行う場合，高い計算能力を持ったマシンで計算できるため，高速に定量化を行えるという利点がある．しかし，サーバが定量化を行う場合，定量化した値からクライアントがどの程度情報を得ているか知られてしまうという問題がある．この問題に対しては，サーバで暗号化したままエントロピー計算を行うという解決策が考えられるが，式 7 にも示すように，エントロピーを計算するために log の計算を行わなければならない．log の計算を暗号化したまま行うことは計算量的に高価 [21] であるため，実用的なデータ提供サービスを行う場合には適切でないと考えられる．続いて，クライアントが計算を行う場合，クライアントが問合せで得た情報に基づき定量化を行えるという利点がある．しかし，クライアントで計算することでデータ所有者，サーバに知られず定量化結果を改竄できるという問題がある．その問題に関して，定量化結果が正しく計算された値か改竄された値かを判定しなければならないという問題もあり，クライアントの検索を制御するために正しい値であると証明するための余計な計算コストがかかると想定される．データ所有者がエントロピー計算を行う場合, クライアントの得た情報を基にクライアントの検索を制御する判定を正しく行える．しかし，データ所有者は負担を軽減するためにサービスプロバイダに機密データの管理を委託しているので，データ所有者にとって望ましくないことだと言える．

表 1 に上記で検討した 3 者の利点及び欠点をまとめる．これらを踏まえて，我々はデータの定量化を誰が行うかを検討した．本研究ではデータ所有者によって権限を移譲された信頼されたサービスプロバイダの所有する開示制御サーバでエントロピー

計算を行うことでこの問題を解決する．信頼された権限委譲者は，エントロピーの計算を正しく行う他，データ所有者に代わりサービスの提供を行うとする．これにより，高い計算能力を持つマシンでエントロピー計算を行うことができる他，定量化した値を利用してクライアントが持つ検索権の範囲内であるかどうかの判定も行うことができる．

6.2 OSIT を利用したデータ提供のシステムモデル

5. 節, 6.1 小節を踏まえて，提案手法ではデータ所有者から信頼された権限委譲者がエントロピーを計算することで検索の制御を行う．信頼された権限委譲者は，エントロピーの計算を正しく行う他，データ所有者に代わりサービスの提供を行う．

まず OSIT におけるデータ提供サービスシステムモデルを図 4 に示す．図 4 では OSIT のモデルに加えてデータ所有者に信頼された権限委譲者がいる．権限委譲者はデータ所有者に代わり，問合せによりクライアントが得た情報を開示制御サーバで定量化することで検索の制御を行う他，クライアントがデータ提供サービスを利用するための検索権を管理する．

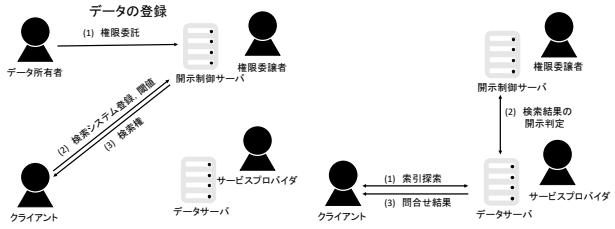

図 4　OSIT を利用したデータ提供サービスのモデル．

ここからは，提案したデータ提供サービスのモデルでどのように処理を行うかについて述べる．サービスの提供手順としては，サービス提供のためのデータを登録するフェーズと問合せを行うフェーズの 2 つの手順が必要である．

まず，図 4 左側でクライアントが検索システムを利用するためのデータ登録フェーズについて説明する．データ所有者が OSIT を用いてデータ提供を行う際, (1) で権限委譲者に提供サービスを代わりに行ってもらうために暗号化，復号に利用する暗号鍵 $E(\cdot)$，復号鍵 $D(\cdot)$，ハッシュ関数 $h(\cdot)$，索引のエントリ数 N を送る．暗号鍵及び復号鍵はクライアントがクエリを暗号化、クエリ値とエントリの大小比較結果を復号するために必要である．更に，復号鍵は開示制御サーバで問合せによりクライアントの得た情報を定量化するために用いられる．ハッシュ関数 $h(\cdot)$，エントリ数 N もクライアントが問合せを行うために用いられる．これらの情報に加え，エントロピーを開示制御サーバで計算するために属性値の最大値 max，最小値 min，レコード数 R を開示制御サーバに送る．続いて, (2) でクライアントは，検索権として閾値 θ を設定し，自身のアカウント情報とともに開示制御サーバへ登録する．閾値は，定量化されたクライアントの得た情報が自身の検索権の範囲内であるか判定を行うために利用する．クライアントの登録が終わったら（3）で開示制御サーバから検索権としてクライアントに検索権として暗号鍵，復号鍵，ハッシュ関数，エントリ数を与える．

続いて，図 4 右側，アルゴリズム 1 に登録後の検索処理について示す．まず, (1) でクライアントとサーバの間で索引探索を行い，クライアントは索引の探索で得た問合せ結果に該当する索引のアドレス $h(i)$ をサーバに送る（2 行目）．続いて, (2) で開示制御サーバでクライアントが問合せで得た情報を定量化するため，データサーバは暗号化されたクエリ値 $E(q)$ とアドレス $h(i)$ に該当する暗号化されたレコード数 $E(R_i)$ を送る（4 行目）．そして，式 7 によりエントロピーの計算を行い，クライアントが登録した閾値 θ 以上であった場合, (3) でデータサーバからクライアントへ問合せ結果を転送する（6-17 行目）．

Algorithm 1 問合せ結果の開示制御アルゴリズム．

Input: $h(i)$：問合せ結果に該当する索引のアドレス, a：検索されたクエリ値を格納するリスト，R：問合せ結果以外のレコード数
Output: $eval$: クライアントに検索結果を返すかどうかの判定結果

1: // クライアントでの処理
2: $h(i)$ をデータサーバに送る
3: // データサーバでの処理
4: $E(q)$ と $h(i)$ に対応する検索結果のレコード数 $E(R_i)$ を開示制御サーバに送る
5: // 開示制御サーバでの処理
6: $q = D(E(q))$
7: **if** $!a.contains(q)$ **then**
8: $\quad R- = D(E(R_i))$
9: $\quad a.add(q)$
10: $\quad sort(a)$ // 昇順ソート
11: $\quad entropy = calcEntropy(a, R)$
12: **else** {}
13: $\quad entropy = calcEntropy(a, R)$
14: **end if**
15: **if** $entropy < \theta$ **then**
16: $\quad eval = false$
17: **end if**

アルゴリズム 2 にエントロピーの計算方法を示す．検索されたクエリ値と問合せを行っていない属性値の総レコード数を入力とする．2-8 行目では問合せていない属性値のビンに対し, 属性値の異なり数と同じ割合でレコード数を分配するための計算を行う．9-13 行目で 2-8 行目の計算結果に基づきレコードを分配し,14-22 行目で式 7 に基づきエントロピーを計算する．

7. 評価実験

7. 節では，6. 節で提案した OSIT を利用したデータ提供サービスにおいて問合せでクライアントが得る情報をエントロピーで定量化できているか実験により評価する．加えて，OSIT の処理及びクエリ結果の開示制御にかかる実行時間，式 6 と式 7 をそれぞれ利用した場合のクエリ結果の開示制御の処理時間についても評価を行う．

7.1 実験設定

本実験には，クライアントに Mac OSX, Intel Core i7 @ 3GHz CPU, 16GB RAM で構成された PC を, データサーバに Ubuntu, Intel Core i7-2600 @ 3.40GHz CPU, 16GB RAM で構成された PC を，開示制御サーバに Mac OSX, Intel Core i7 @ 3GHz CPU, 16GB RAM で構成された PC を使用した．

Algorithm 2 *calcEntropy*(a, R).

Input: a：検索されたクエリ値を格納するリスト，R：問合せ結果以外のレコード数
Output: *entropy*: エントロピーの値

```
1: // 開示制御サーバでの処理
2: for i = 0 to a.size() − 1 do
3:     count[i] = a[i + 1] − a[i] − 1 // 属性値の異なり数を計算
4: end for
5: sum=0
6: for i = 0 to count.length − 1 do
7:     sum+ = count[i]
8: end for
9: for i = 0 to count.length − 1 do
10:    n[i] = R × count[i] / sum
11:    R− = n[i]
12:    sum− = count[i]
13: end for
14: entropy = 0
15: for i = 0 to count.length − 1 do
16:    for j = n[i] + 1 to n[i] + count[i] do
17:        entropy+ = log(j)
18:    end for
19:    for j = 2 to count[i] do
20:        entropy− = log(j)
21:    end for
22: end for
```

実装は Java で行い，データセットは Adult Data Set (注3) を用いた．

実験では (1)「SELECT age FROM Adult WHERE age $= x$」，(2)「SELECT hours-per-week FROM Adult WHERE hours-per-week $= x$」の 2 つの属性に対する問合せを考える．x の値を変化させ繰り返し問合せを行うことでエントロピーの値の変化を確認する．x は属性値の最小値から最大値までの範囲の全ての整数から 1 回ずつランダムに選択され，繰り返し問合せを行う．本実験では問合せ結果以外のデータは一様分布に従っていると仮定し，開示制御サーバはレコード数，属性値の最小値，最大値を知っているとする．

7.2 問合せでクライアントが得た情報の定量化

エントロピーの式 7 で，問合せでクライアントが得た情報を定量化できるか評価する．具体的には，クライアントに結果が開示されるほど検索対象になっていない不確定部分について，ヒストグラムの作り方の総数が減ることから問合せを行うほどエントロピーが減少し，全てのデータがクライアントに渡った時にエントロピーが 0 になると考えられるので確認する．

図 5 に属性 age，hours-per-week に対し繰り返し問合せを行った時のエントロピーの値の変化を示す．縦軸はエントロピーの値，横軸は問合せ実行回数を表す．図 5 からどちらの属性に対しても問合せによりクライアントに開示されるレコードが増加するに従い，エントロピーの値が単調減少することを確認できた．図 5 右側では 41 回目から 51 回目の問合せの間でエントロピーの値が急激に減少しているが，問合せでクライアントに開示されるレコード数が多かったためであると考えられる．このように，属性ごとの分布の違いにより減少の傾向は異なるが，全てのレコードが開示された場合にエントロピーの値が 0 になることも確認できた．

しかし，　今回は全ての属性値に対してレコード数が一様に分布していると仮定しているため，例えば k 回問合せした時に図 6 のような分布だとクライアントが推定できる場合には仮定

(注3)：http://archive.ics.uci.edu/ml/datasets/Adult

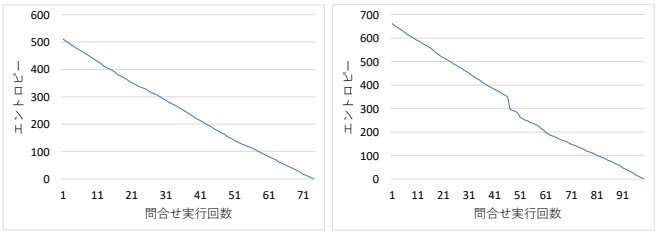

図 5　繰り返し問合せによる属性 age,hours-per-week のエントロピー変化．

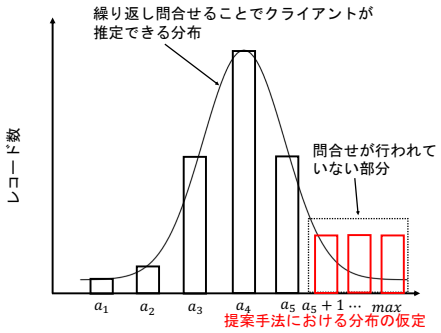

図 6　繰り返し問合せた後でクライアントが推定できる分布と提案手法でおいた仮定の分布の差．

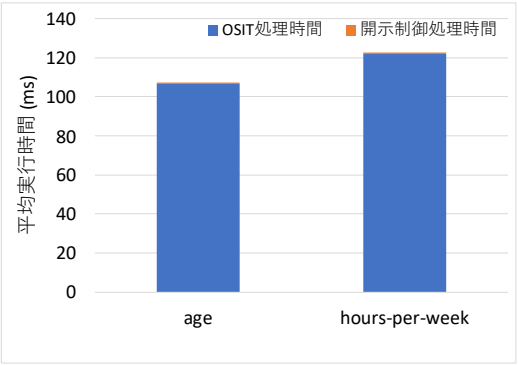

図 7　属性 age, hours-per-week に対する問合せにおける，OSIT の問合せ処理及びクエリ結果開示制御の平均処理時間．

した分布と異なるためエントロピーの値が異なると考えられる．これに対し，問合せ回数に応じて柔軟にエントロピーの計算式を変更することも考えられるが，計算時間が増加することが想定される．そのため，軽量な計算で柔軟にエントロピー計算を変更する手法の検討が必要である．

7.3 問合せ及びクエリ結果の開示制御の実行時間評価

続いて，OSIT における問合せ処理時間とクエリ結果の開示制御処理時間について評価を行う．本実験ではエントロピーが 0 になるまで属性 age，hours-per-week に対して問合せを行う．

図 7 に各属性に対する問合せにおける，OSIT の問合せ処理と開示制御の平均処理時間を示す．図 7 からクエリ結果開示制御の処理時間は OSIT の処理時間と比較すると全体の処理時間に占める割合が極めて少ないことが分かる．クエリ結果開示制御の処理時間は，どちらの問合せにおいても全体の処理時間の 0.7%程度であることから，実用的であると考えられる．

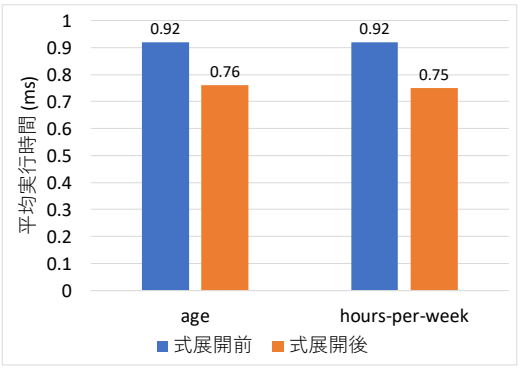

図 8　属性 age, hours-per-week に対する問合せについて，クエリ結果開示制御を展開前の式 6, 展開後の式 7 で行った場合の平均処理時間.

また，図 8 に属性 age, hours-per-week に対する問合せについて，展開前の式 6 と展開後の式 7 でクエリ結果開示制御を行った場合の平均実行時間を示す．図 8 から，どちらの属性についても展開後の式 7 を用いてクエリ結果の開示制御処理を行うことで，約 1.2 倍高速に処理できた．これは，式展開により階乗計算がなくなったからだと考えられる．展開後の式 7 を利用した開示制御処理は，より大きなデータセットに対しても効率的な処理を行えると期待される．

8. 結　　論

本研究では，クラウド上で OSIT を利用したデータ提供サービスのシステムモデルを提案した．クライアントが問合せで得た情報をエントロピーで定量化し，定量化した値に基づきクライアントの検索を制御するシステムを検討した．そして，安全に定量化を行うために，データ所有者にサービスを行う権限を委譲された信頼されたサービスプロバイダが管理するクエリ結果の開示制御サーバをモデルに組み込んだ．信頼された開示制御サーバでは，クライアントがサービス利用時に登録した閾値と計算したエントロピーの値を比較し，エントロピーが閾値を下回った場合にはクライアントに結果を開示しないという制御アルゴリズムを実行する．また，実データを用いた実験では，問合せを繰り返すことでエントロピーが単調減少し，クライアントに全ての結果が渡った時にエントロピーが 0 になることも示した．これに加えて，全体の処理時間における OSIT での問合せ処理時間とクエリ結果の開示制御処理時間についても比較を行い，全体の処理時間に占める開示制御処理時間の割合が極めて少なく，実用的であることを示した．そして，クエリ結果開示制御の処理において，エントロピーの計算式を展開することで展開前よりも約 1.2 倍高速になることも示した．

今後の課題としては，問合せを行うにつれてデータが開示されることで，分布の想定が変化していくと考えられるのでエントロピー計算の式も柔軟に変化させることができる手法の検討が挙げられる．また，本研究では，信頼されたサービスプロバイダを取り入れることにより問合せでクライアントの得た情報を定量化し，検索の制御を行うことができる一方で，1 人のサービスプロバイダが複数人のクライアントが持つ情報を定量化できてしまう．そこで，信頼されていない場合についても安全に検索の制御を行う仕組みも検討したい．

謝　　辞

本研究の一部は NICT 高度通信・放送研究開発委託研究「欧州との連携による公共ビッグデータの利活用基盤に関する研究開発」, JSPS 科研費 JP16K00149 の助成を受けたものです．

文　　献

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *Proc. ACM SIGMOD, 2004*, pp. 563–574, 2004.
[2] M. Balazinska, B. Howe, and D. Suciu. Data markets in the cloud: An opportunity for the database community. *PVLDB*, Vol. 4, No. 12, pp. 1482–1485, 2011.
[3] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *Proc. EUROCRYPT 2009*, pp. 224–241, 2009.
[4] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Proc. EUROCRYPT 2004*, pp. 506–522, 2004.
[5] S. Deep and P. Koutris. The design of arbitrage-free data pricing schemes. In *Proc. ICDT 2017*, pp. 12:1–12:18, 2017.
[6] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, Vol. 31, No. 4, pp. 469–472, 1985.
[7] T. Ge and S. B. Zdonik. Answering aggregation queries in a secure system model. In *Proc. VLDB, 2007*, pp. 519–530, 2007.
[8] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. ACM STOC 2009*, pp.169–178, 2009.
[9] C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *Proc. EUROCRYPT 2012*, pp.465–482,2012.
[10] H. Hacigümüs, B. R. Iyer, Chen Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proc. ACM SIGMOD, 2002*, pp.216–227, 2002.
[11] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *(e)Proc. VLDB 2004*, pp.720–731, 2004.
[12] H. Hu, J. Xu, X. Xu, K. Pei, B. Choi, and S. Zhou. Private search on key-value stores with hierarchical indexes. In *Proc. ICDE 2014*, pp. 628–639, 2014.
[13] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Query-based data pricing. In *Proc. ACM PODS 2012*, pp.167–178, 2012.
[14] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Querymarket demonstration: Pricing for online data markets. *PVLDB* , Vol. 5, No. 12, pp. 1962–1965, 2012.
[15] P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. Toward practical query pricing with querymarket. In *Proc. ACM SIGMOD 2013*, pp.613–624, 2013.
[16] C. Li and G. Miklau. Pricing aggregate queries in a data marketplace. In *Proc. WebDB 2012*, pp.19–24, 2012.
[17] B. R. Lin and D. Kifer. On arbitrage-free pricing for general data queries. *PVLDB*, Vol. 7, No. 9, pp. 757–768, 2014.
[18] E. Mykletun and G. Tsudik. Aggregation queries in the database-as-a-service model. In *Proc. Data and Applications Security XX, 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, 2006*, pp.89–103, 2006.
[19] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. EUROCRYPT 1999*, pp.223–238, 1999.
[20] R. A. Popa, Catherine M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: processing queries on an encrypted database. *Commun. ACM*, Vol. 55, No. 9, pp.103–111, 2012.
[21] Raphael S Ryger, O. Kardes, and Rebecca N Wright. On the lindell-pinkas secure computation of logarithms: From theory to practice. *P3DM 2008*, p. 21, 2008.
[22] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. *PVLDB*, Vol. 6, No. 5, pp. 289–300, 2013.
[23] W. K. Wong, B. Kao, D. W. L. Cheung, R. Li, and S. M. Yiu. Secure query processing with data interoperability in a cloud database environment. In *Proc. SIGMOD 2014*, pp. 1395–1406, 2014.
[24] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, Vol. 2. Cambridge University Press, 2004.
[25] 篠塚 千愛, 渡辺 知恵美, 北川 博之. クラウド環境における暗号化索引を用いた文字列属性の部分一致検索手法. コンピュータセキュリティシンポジウム 2015 論文集, pp.979-986(2015).
[26] 篠塚 千愛, 渡辺 知恵美, 北川 博之. 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. DEIM Forum 2017 H6-4 (2017).
[27] 秋山賢人, 渡辺知恵美, 北川博之. 暗号化データベースシステムにおけるクエリベースのデータ販売スキーム. 情報処理学会論文誌データベース (TOD76 テクニカルノート), Vol. 10, No. 4, pp. 31–35, 2017.