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

篠塚 千愛† 渡辺 知恵美‡ 北川 博之‡

†筑波大学システム情報工学研究科
305-8573 つくば市天王台 1-1-1
sn@kde.cs.tsukuba.ac.jp

‡筑波大学システム情報系
305-8573 つくば市天王台 1-1-1
chiemi@cs.tsukuba.ac.jp
kitagawa@cs.tsukuba.ac.jp

あらまし クラウドデータベースには，第三者であるクラウドの管理者に対してデータとクエリを秘匿したまま検索可能なことに加え，高速な処理性能が要求される．しかし，安全性と検索処理コストはトレードオフの関係にあり，双方の実現は困難である．我々はこれまでに，データやクエリのプライバシを十分に保証しながら高速な検索処理を可能とするフレームワーク OSIT-bs を提案した．OSIT-bs では，検索に索引を使用し，それを秘密計算技術を利用したプロトコルで探索することで，安全で高速な検索処理を実現する．本研究では，OSIT-bs に基づき安全で高速な部分一致検索処理を検討し，OSIT-bs の有用性を示す．

Secure Partial Matching Query for String Attributes Using Encrypted Suffix Array

Chisato Shinozuka† Chiemi Watanabe‡ Hiroyuki Kitagawa‡

†Graduate School of Systems and Information Engineering, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
sn@kde.cs.tsukuba.ac.jp

‡Faculty of Engineering, Information and Systems University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
chiemi@cs.tsukuba.ac.jp, kitagawa@cs.tsukuba.ac.jp

Abstract Search services using Database-as-a-Service (DBaaS) are required both guarantee of strong privacy against DBaaS service provider and high query performance. However, achieving them is difficult since they are a trade-off relationship each other. To date, we proposed OSIT-bs framework which provides secure range query processor using encrypted index and its traversal algorithm without decrypting items. However, OSIT-bs framework supports only range query for numeric attribute values. In this paper, we apply OSIT-bs framework to partial matching query of string attribute values. To process partial matching queries, we define encrypted suffix array index.

1 はじめに

Database as a Service (DBaaS) が様々なプロバイダから提供され，DBaaS を利用した検索サービスが多く提供されている．DBaaS は，インターネットを介してデータの保管・運用といった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あり，Amazon

SimpleDB[1] や Google Cloud SQL[2]，Microsoft SQL Server[3] 等がある．DBaaS を利用することで，データ所有者は簡単な初期設定を済ませた後クラウド上のデータベースにデータを預けるだけで検索サービスを提供可能になる．他にも，データ所有者が自身でサーバを購入・維持するよりも安価にデータベースを利用可能なこと，専門知識を有する管理者にデータベース管理を委託可能なことが DBaaS の利点である．

しかしながら，DBaaS に委託するデータに機密情報が含まれる場合，データ所有者は DBaaS 管理者に対してデータを秘匿したい．この要求をデータプライバシという．同様に，検索クエリに個人に関する情報が含まれる場合，クライアントは DBaaS 管理者に対してクエリを秘匿したい．この要求をクエリプライバシという．

DBaaS を利用した検索サービスでは，データプライバシとクエリプライバシの保護に加えて，高い処理性能も要求される．しかし，安全性と処理コストはトレードオフの関係にあり，双方の実現は難しい．例えば，順序保存暗号 [4] でデータを暗号化する場合，高速な検索処理が期待できるが，暗号化前の値同士の大小関係が漏洩することから安全性は十分でない [10]．反対に，検索可能暗号の一種である完全準同型暗号によるキーワード検索 [5] は強固な安全性を保証するが，すべての暗号化データとクエリとを 1 つ 1 つ照合するために処理コストは非常に大きい．

我々はこれまでに，安全性と効率性を兼ね備えた秘匿検索のための実現する OSIT-bs フレームワークを提案した [6]．本フレームワークでは，索引を使用して高速な検索処理を実現する．索引のエントリ情報と構造情報を分離し，エントリ情報を暗号化してクラウド上のサーバに，構造情報をクライアントに配置する．また，索引の探索には大小比較演算のための紛失通信プロトコルである GT-SCOT[8] を使用してデータプライバシとクエリプライバシの保護を実現する．[6] では OSIT-bs フレームワークの初段階として，配列を使用した範囲検索処理を実現した．本研究では，索引に接尾辞配列を使用する検索処理に OSIT-bs フレームワークを適用して，文字列属性に対する安全で高速な部分一致検索演算を提案するとともに，OSIT-bs フレームワークの有用性を示す．

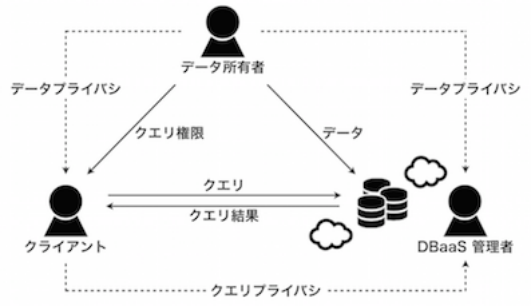

図 1: データ管理と検索の流れ

本稿は次のように構成される．まず，2 節で本研究の問題設定を述べ，3 節で本研究で使用する接尾辞配列，暗号化スキーム，および，GT-SCOT プロトコルを紹介する．次に，4 節で OSIT-bs フレームワークについて述べ，続く 5 節で OSIT-bs フレームワークに基づく文字列属性の部分一致検索手法を提案する．最後に，6 節で本稿のまとめと今後の課題について述べる．

2 問題設定

本研究では，DBaaS を利用した検索サービスを想定し，安全性と効率性を兼ね備えた文字列属性の部分一致検索演算の実現を目的とする．

2.1 データ管理と検索の流れ

DBaaS を利用した検索サービスは，データ所有者，DBaaS 管理者，クライアントの三者によって成立する．図 1 にサービスモデルを示す．データ所有者は，DBaaS で提供されるクラウド上のデータベースサーバにて検索対象となるデータを保管し，検索サービスの基盤となるデータベースの管理を DBaaS 管理者に委託する．また，クライアントには検索サービスの利用権限を与える．一方，利用権限をもつクライアントは，クラウド上のデータベースサーバに対してクエリを発行して問合せを行ない，必要なデータを取得できる．

2.2 DBaaS 利用者のプライバシ要求

DBaaS 管理者は，データ所有者にデータベース管理を委託されており，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．しかしながら，データベースに保管するデータに機密情報が含まれる場合，DBaaS 管理者がデータベース管理によりデータを取得したり，クライアントが検索結果以上にデータを取得したりすることをデータ所有者は避けたい．この要求をデータプライバシという．データ所有者は，DBaaS 管理者とクライアントに対するデータプライバシをもつ．同様に，クエリにクライアントの個人に関する情報が含まれる場合，DBaaS 管理者がクエリログに基づいてクライアントの情報を取得することをクライアントは避けたい．この要求をクエリプライバシという．これらのプライバシ要求を図 1 に示す．本研究では，満たすべき安全性として，DBaaS 管理者に対してデータプライバシとクエリプライバシを保証する方法を議論する．

3 前提知識

本節では，3.1 節にて文字列属性の部分一致検索を行なうための接尾辞配列を紹介する．続いて，3.2 節と 3.3 節にて，索引の探索アルゴリズムに使用する準同型暗号と GT-SCOT プロトコルを紹介する．

3.1 接尾辞配列

接尾辞配列は，長さ n の文字列 T の n 個の接尾辞を辞書順に並び替えたときの，各接尾辞と T 中での開始位置の組合せを順次格納した配列である．検索対象文字列 q について，二分探索アルゴリズムを利用することで T 中での q の出現位置を高速に検索できる．接尾辞配列では，q の出現位置を検索することとは，先頭が q で始まる接尾辞を求めることである．

［例 1］$T =$ “jellyfish” のとき，T の接尾辞配列 SA は表 1 のようになる．ここで，$q =$ “fish” の T 中での出現位置の検索を考える．検索には二分探索アルゴリズムを利用する．まず，SA 全体を探索範囲とし，中央のエントリ SA_4 と q を比較する．$q =$ “fish” は，SA_4 の接尾辞 “jellyfish” よりも辞書的に前であるので，現在の探索範囲の前半部分 SA_0–SA_3 を新たな探索範囲とする．同様にして，中央のエントリ SA_1 と q を比較する．$q =$ “fish” は，SA_1 の接尾辞 “fish” と一致するので，探索を終了する．

表 1: $T =$ “jellyfish” の接尾辞配列 SA

	SA_i	
i	接尾辞	開始位置
0	ellyfish	1
1	fish	5
2	h	8
3	ish	6
4	jellyfish	0
5	llyfish	2
6	lyfish	3
7	sh	7
8	yfish	4

3.2 準同型暗号

準同型暗号は，2 つの平文に対する演算を，それらの暗号文に対する演算に変換可能な準同型性を有する暗号スキームである．即ち，$D(\cdot)$ が復号化関数であり，$m_1 = D(c_1)$ および $m_2 = D(c_2)$ を満たす任意の平文 m_1, m_2 および暗号文 c_1, c_2 について，以下の式 (1) が成立する暗号を準同型暗号と呼ぶ．ただし，式中の $\odot$ には加法演算子や乗法演算子のような演算子が入る．

$$D(c_1 \odot c_2) = m_1 \odot m_2. \tag{1}$$

準同型暗号の例として，RSA 暗号，Paillier 暗号 [7]，エルガマル暗号，GM 暗号がある．本稿では準同型暗号に Paillier 暗号を使用する．Paillier 暗号では，平文 m についての暗号化関数 $E(\cdot)$ は公開鍵 $pk = (n, g)$ と n 以下の乱数 r を用いて式 (2) のように表せる．

$$E(m) = g^m \cdot r^n \bmod n^2. \tag{2}$$

$E(\cdot)$ は乱数を含むため，同じ平文 m であっても常に同じ暗号文が取得できるとは限らない．また，Paillier 暗号は式 (3) のように 2 つの平文 m_1，m_2 の暗号文から $m_1 + m_2$ の暗号文を計算できる加法準同型性を有する．

$$\begin{aligned} & E(m_1) \cdot E(m_2) \\ = \ & (g^{m_1} \cdot r_1^n \bmod n^2) \cdot (g^{m_2} \cdot r_2^n \bmod n^2) \\ = \ & g^{m_1+m_2} \cdot (r_1 \cdot r_2)^n \bmod n^2 \\ = \ & E(m_1 + m_2). \end{aligned} \tag{3}$$

3.3 GT-SCOT プロトコル

GT-SCOT (Strong Conditional Oblivious Transfer for “Greater Than” predicate) プロトコル [8] は，大小関係を秘密に秘密に計算するための条件付き紛失通信プロトコル [9] の一種である．受信者と送信者が互いに相手の持つ値を知らないまま大小関係を計算でき，受信者のみが比較結果を知ることができる．

受信者が値 x を，送信者が値 y を持つとし，GT-SCOT プロトコルによる 2 値の比較手順を示す．まず，受信者は，プロトコルによる大小関係の計算結果となる，$x < y$ であることを示す値 s_0 と，$x > y$ であることを示す値 s_1 を決定する．次に，x を暗号化した値 $E(x)$，s_0，s_1 を送信者に送信する．ただし，$E(\cdot)$ は加法準同型暗号の暗号化関数である．続いて，送信者は $E(x)$，y，s_0，s_1 に対して，x と y の大小関係として $E(s_0)$ または $E(s_1)$ が得られる計算アルゴリズムを適用し，計算結果を受信者に返す．受信者は復号化関数 $D(\cdot)$ を適用することで，s_0 もしくは s_1 を取得できる．このように受信者は大小関係を知ることができるが，y そのものを知ることはできない．一方，送信者は暗号化された x のみしか知ることができない．また，送信者側で大小比較を行なっているものの，比較結果は暗号化された状態で導出されるため，送信者は大小関係を知ることができない．

4 OSIT-bs フレームワーク

我々はこれまでに，安全性と効率性を兼ね備えた秘匿検索を実現する OSIT-bs (Oblivious Secure Index Traversal - binary search) フレームワークを提案している [6]．4 節では，OSIT-bs フレームワークと，本フレームワークによる範囲検索処理について説明する．図 2 に OSIT-bs フレームワークのシステム構成を示す．データ所有者は，DBaaS で提供される 2 台のサーバを利用し，一方をデータと索引を保管するデータサーバ，他方を大小比較をする計算サーバとする．クライアントは，データサーバおよび計算サーバとの三者間で検索処理を行なう．本フレームワークは主に次の三つの特徴をもつ．一つ目は，検索を効率的に行なうために索引を使用することである．二つ目は，索引のエントリ情報と構造情報を分離して管理することでより強固にデータプライバシを保証をすることである．三つ目は，安全なプロトコルで分離された索引を探索することである．4.1 節にて索引の配置を，4.2 節にて索引の探索アルゴリズムを紹介する．

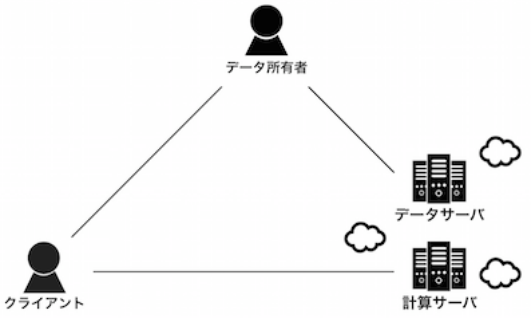

図 2: OSIT-bs フレームワークのシステム構成

4.1 索引の配置

OSIT-bs フレームワークでは，データ所有者がデータと索引を暗号化し，安全に配置することでデータプライバシを保証する．まず，データを暗号化してデータサーバに配置する．[6] では範囲検索を実現するため，検索対象となるキー k とデータサーバ上のレコードへのポインタ $I(k)$ の組合せ $\langle k, I(k) \rangle$ で構成されるエントリ e を複数もつ配列を索引とする．データ所有者はこの配列を各エントリの k 値でソートした上で，各エントリの $k, I(k)$ をそれぞれ加法準同型暗号 $E(\cdot)$ で暗号化する．以降，この索引を暗号化

索引と呼ぶ．その後，図 3 のように暗号化索引をエントリ情報と構造情報に分離し，エントリ情報をデータサーバに，構造情報をクライアントに配置する．まず，ソートされた配列の各エントリに対し，配列上の位置を引数にしたハッシュ値を付与する．例えば，あるエントリ $e_2 = \langle E(k_2), E(I(k_2))\rangle$ が配列上の 2 番目にあり，ハッシュ関数を $h(\cdot)$ としたとき，e_2 に対して $h(2)$ の値を付与する．このようにしてハッシュ値を付与したエントリを暗号化索引のエントリ情報とし，データサーバに配置する．また，ハッシュ関数 $h(\cdot)$ とエントリ数 N を暗号化索引の構造情報とし，クライアントに配置する．

暗号化索引の分離・分散配置はデータプライバシを保証する．データサーバに格納されるエントリ情報は暗号化されている上に，順序関係も秘匿されている．順序関係を保持する場合，たとえエントリ情報が暗号化されていても選択平文攻撃に弱いことが知られている [10]．以上から，データ所有者に対するデータプライバシを保証できる．一方，クライアントではエントリの構造情報は持っているものの，エントリそのものは持たない．従って，クライアントに対するデータプライバシを保証できる．

4.2 索引の探索アルゴリズム

OSIT-bs フレームワークでは，分離・分散配置された暗号化索引を安全なプロトコルで探索することで，データプライバシとクエリプライバシを保証する．クライアントは，データサーバおよび計算サーバとの三者間で検索処理を行なう．検索処理を行なう前に，データ所有者はクライアントに暗号化索引に適用した加法準同型暗号の暗号鍵と復号鍵を，計算サーバに暗号化索引に適用した加法準同型暗号の暗号鍵をそれぞれ提供する．

以降では，下限値 l と上限値 h を範囲条件とするクエリ $q = \langle l, h\rangle$ に該当するレコードを検索することを考える．クライアントは，暗号化索引を探索して l 以上で最小のキー値 k をもつエントリ e_l と，h 以下で最大のキー値 k をもつエントリ e_h を求める．そして，$e_l - e_h$ 間のエントリの各 $I(k)$ に対応するレコードを取得することで，q の範囲検索を実現する．

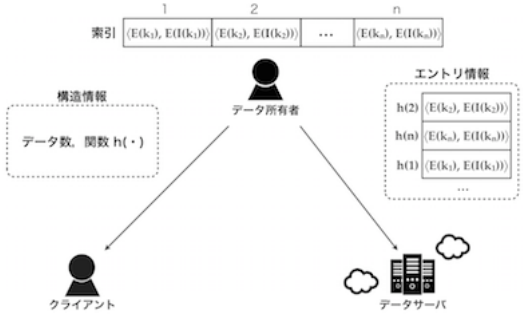

図 3: 暗号化索引の分離・分散配置

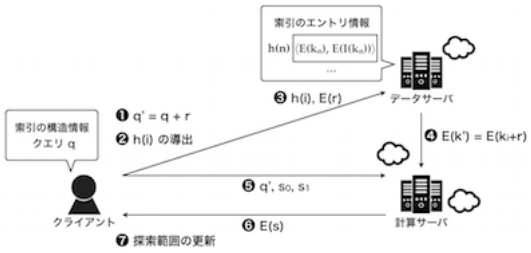

図 4: 暗号化索引の探索

しかしながら，クライアント単独では暗号化索引を探索できない．クライアントは暗号化索引のエントリ情報を持たず，クエリ値と各エントリのキー値の大小関係を求められないためである．そこで，OSIT-bs フレームワークでは，暗号化索引の探索のためにクエリ値 q と比較対象となるエントリの配列上の位置をクラウドのサーバに送信し，クラウドから比較結果を受け取るという処理を繰り返す．ここで，クエリプライバシを保証するため，クエリ値は DBaaS 管理者から秘匿する必要がある．また，データプライバシを保証するため，各エントリのキー値はクラウド上では復号化できないようにしたい．OSIT-bs フレームワークでは，図 4 と以下の手順で示すようにクエリ値と各エントリのキー値の大小比較を行ない，クエリプライバシとデータプライバシの双方を保証する．

1. ［クライアント］クエリ値 q に対して乱数 r を加えた摂動クエリ値 $q' = q + r$ を求める．
2. ［クライアント］比較したいエントリの配

列上の位置 i のハッシュ値 $h(i)$ を求める．

3. ［クライアント］乱数 r を暗号化して $E(r)$ を求め，手順 2 で求めたハッシュ値 $h(i)$ とともにデータサーバに送信する．

4. ［データサーバ］$h(i)$ が付与されたエントリ e_i の暗号化されたキー値 $E(k_i)$ に対して，乱数 r を加えた $E(k_i')$ を式 (4) のようにして求め，計算サーバに送信する．

$$E(k_i') = E(k_i + r) = E(k_i) \cdot E(r). \quad (4)$$

5. ［クライアント］摂動クエリ値 q' と GT-SCOT プロトコルにて出力結果となる値 s を計算サーバに送信する．

6. ［計算サーバ］GT-SCOT プロトコルを用いて q' と $E(k_i')$ の大小関係を求め，暗号化された結果 $E(s)$ をクライアントに送信する．

7. ［クライアント］$E(s)$ を復号化し，得られた大小関係をもとに探索範囲を絞り込む．

暗号化索引の探索アルゴリズムには二分探索を採用する．暗号化索引の探索では，クエリ値と各エントリのキー値の大小比較がクライアントとサーバ間の通信を必要とするため，探索時間に影響する．従って，比較回数を削減することで探索時間の短縮が可能になる．n 分探索では，$n = 2$ のときに探索終了までの比較回数が最小となる．

さらに，OSIT-bs フレームワークでは，二分探索を行なう場合に探索範囲の中央値に摂動値を付与し，毎回少しずれた位置を参照するようにする．通常の二分探索では，常に最初に選ばれるエントリの k が中間値であることに加え，その後の探索からもデータサーバに暗号化索引の構造情報が漏洩するためである．

5 OSIT-bs フレームワークによる文字列属性の部分一致検索手法

5 節では，OSIT-bs フレームワークに基づく安全で高速な文字列属性の部分一致検索手法を提案する．接尾辞配列を索引に使用して OSIT-be フレームワークを適用する．5.1 節にて提案手法が対象とするデータとクエリの設定を，5.2 節にて索引の生成を，5.3 節にて索引の探索アルゴリズムを紹介する．

表 2: 提案手法が対象とするデータ例

id	施設名
0	計算科学研究センター
1	プラズマ研究センター
2	外国語センター
3	体育センター
4	農林技術センター
5	陸域環境研究センター
6	留学生センター
7	アドミッションセンター
8	学術情報メディアセンター

5.1 データとクエリ

対象とするデータベースは，文字列属性を含むレコードを多数もち，文字列属性に対する検索を行える．クライアントはデータベースに対して，文字列属性値に検索したい文字列 q を含むレコード群を求める．

［例 2］表 2 は，施設情報に関するデータベースを示す．このデータベースには複数のレコードが格納されており，各レコードは，id と，文字列で表現される属性で構成される．このデータベースに対して，クライアントが q =“研究センター”に関する問合せを行なった場合，検索結果として施設名に “研究センター”を含む，$id = 0, 1, 5$ のレコードを取得できる．

5.2 索引の生成

提案手法でも，索引の生成はデータ所有者が行なう．はじめに，各レコード毎に検索対象となる文字列属性値 T の接尾辞 suf とデータサーバ上の各レコードへのポインタ $I(T)$ の組合せ $\langle suf, I(T)\rangle$ で構成されるエントリの配列を生成

する．ただし，suf は加法準同型暗号で暗号化できるように文字コードで数値に変換された接尾辞である．また，どの接尾辞も文字数を統一させた上で文字コードによる数値変換を行なう．これにより，数値変換前後で元の接尾辞の辞書順序を保持できる．続いて，データ所有者はレコード毎に生成された配列を統合し，各エントリの suf でソートを行なって接尾辞配列を得る．最後に，ソート後の各エントリの $suf, I(T)$ を加法準同型暗号でそれぞれ暗号化して，提案手法の暗号化索引とする．

5.3 索引の配置と探索アルゴリズム

データ所有者はまず，全レコードを暗号化してデータサーバに配置する．続いて，暗号化索引の分離・分散配置を行なう．ここでは，エントリ情報はソートされた各エントリのインデックスのハッシュ値とエントリ本体のペアの集合となり，構造情報は接尾辞配列とハッシュ関数となる．OSIT-bs フレームワークに基づき，データ所有者はエントリ情報をデータサーバに，構造情報をクライアントに配置する．提案手法でも，暗号化索引の分離・分散配置はデータプライバシを保証する．

索引の探索について，通常の接尾辞配列の探索アルゴリズムは二分探索であることから，OSIT-bs フレームワークが提供する探索アルゴリズムの適用は容易である．提案手法においても，クライアントは図 4 のようにして暗号化索引を探索できる．また，提案手法でも安全な探索プロトコルを用いて暗号化索引を探索するため，データプライバシとクエリプライバシを保証する．

6 おわりに

本稿では，DBaaS を利用した検索サービスを想定し，文字列属性の部分一致検索演算の安全性と効率性の両立を検討した．我々は，暗号化された索引を安全に探索できる OSIT-bs フレームワークを接尾辞配列に対して適用することを提案し，安全性と効率性を兼ね備えた文字列属性の部分一致検索手法を実現するとともに，OSIT-bs フレームワークの有用性を示した．

今後の課題は三つある．一つ目は，本稿で提案した OSIT-bs フレームワークに基づく文字列属性の部分一致検索手法の実験的評価を行なうことである．二つ目は，並列計算による探索の更なる高速化を検討することである．安全性を考慮しない通常の接尾辞配列の探索と比較した場合，通信を必要とする提案手法の探索アルゴリズムの計算量は非常に大きい．しかし，GT-SCOT プロトコルによる大小比較演算は，複数の計算サーバで並列に行なうことが可能である．なお，並列計算を行なう場合は，n 個の比較演算を同時に行えるので，二分探索よりも n 分探索を採用する方が高速化を期待できる．三つ目は，グルーピングや集約演算，他のレコードとの結合といった複雑な演算に対する OSIT-bs フレームワークの応用を検討することである．

謝辞

本研究の一部は，文部科学省 “実社会ビックデータ利活用のためのデータ統合・解析技術の研究開発”によるものです．

参考文献

[1] Amazon SimpleDB, `http://aws.amazon.com/simpledb/` .

[2] Google Cloud SQL, `https://cloud.google.com/sql/` .

[3] Microsoft SQL Server, `http://www.microsoft.com/ja-jp/server-cloud/products/sql-server/Explore.aspx` .

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. SIGMOD’04 Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pages 563–574 (2004).

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. Advances in Cryptology – EUROCRYPT 2004, Lecture Notes in Computer Science Volume 3027, pages 506–522 (2004).

[6] 篠塚 千愛, 渡辺 知恵美, 北川 博之. DaaS 環境におけるデータとクエリ双方のプライバシ保護を実現する効率的な秘匿検索. DEIM Forum 2015 G2-6 (2014).

[7] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. Advances in Cryptology – EUROCRYPT'99, pages 223–238 (1999).

[8] I. F. Blake and V. Kolesnikov. Strong Conditional Oblivious Transfer and Computing on Intervals. Advances in Cryptology – ASIACRYPT 2004, pages 515-529 (2004).

[9] G. D. Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. Advances in Cryptology — EUROCRYPT '99 Lecture Notes in Computer Science Volume 1592, pages 74–89 (1999).

[10] A.Boldyreva, N.Chenette, Y.Lee, and A.O 'Neill. Order-preserving symmetric encryption. Advances in Cryptology – EUROCRYPT 2009, Lecture Notes in Computer Science Volume 5479, pages 224–241 (2009).