DBSJ Journal, Vol.12,No.2
October 2013

Bloom Filter を利用した暗号化 SLCA 検索手法の提案

Searching Encrypted SLCA by Using Bloom Filter

中村伸一♥ 山本博章♦

Shin-ichi NAKAMURA♥ Hiroaki YAMAMOTO♦

XML 文書は XPath や XQuery を利用することで検索を行うことができる．しかし，XPath や XQuery を利用するには，XPath や XQuery の文法や検索する XML 文書の構造を知っている必要があり，初心者には難しいものとなっている．そのため，初心者にも簡単に XML 文書の検索が行えるように，キーワードによる XML 文書の検索手法について研究が進められている．本研究は，キーワードによる XML 文書の検索に焦点を当て，与えられたキーワードに対し暗号化された XML 文書から，その SLCA を検索するための手法を提案する．我々の手法では権限のない管理者は検索するキーワードや結果を知ることができない．本稿では，提案手法の安全性について議論し，さらに，実験的に検索性能についても評価する．

An XML document search can be performed by using XQuery and XPath. However, we must know the grammar of XPath or XQuery and the structure of an XML document. Furthermore it is difficult for beginners to use XQuery and XPath. Therefore, search methods on an XML document using keywords have been studied so that beginners can easily search an XML document. In this paper, we propose a method to find Smallest Lowest Common Ancestors (SLCAs) from an encrypted XML document. In the proposed method, administrators without access permission to the XML document can know neither keywords nor search results. We discuss security on the proposed method, and also evaluate search performance by some experiments.

1. はじめに

近年，音声や動画配信，クラウド，商取引，Web 等の利用によって，インターネット上でやり取りされる情報量は増加し続けている．様々な形式の情報がインターネット上でやり取りされるが，その一つである XML は情報交換を目的とした形式として広く利用されている．XML は XPath や XQuery を利用することで詳細な検索を行うことができる．しかし，XPath や XQuery を利用するには，XPath や XQuery の文法や検索する XML 文書 の構造を知っている必要があり，初心者には難しいものとなっている．

そのため，初心者にも簡単に XML 文書 の検索が行えるように，キーワードによる XML 文書の検索手法について研究が進められている．しかし，今までに提案されている手法は，XML 文書を平文でデータベースに登録するため，データベースの管理者に検索内容や結果が知られてしまい，管理者から情報漏えいが発生する問題がある．

そのため，我々はキーワードによる XML 文書の検索手法である The Indexed Lookup Eager Algorithm[1](以下 IL)について，管理者に検索内容や結果がわからない暗号化に対応した手法を提案する．

提案手法は，事前に XML 文書から Bloom Filter と暗号化されたデータで構成されたテーブルを構築し，リレーショナルデータベース(以下 RDB)に登録する．検索時は RDB 上で Bloom Filter を利用したキーワードによる XML 文書の検索を行い，検索結果の Bloom Filter に対応する暗号化されたデータを RDB から取得する．取得した暗号化されたデータを復号して検索結果を作成する．この手法により，セキュリティ上構築が難しい情報環境でもキーワードによる XML 文書の検索環境を構築できる．管理者に検索内容や結果がわからないため，Amazon Simple DB[2]等のインターネットを経由したデータベースサービスのような，利用者によるセキュリティ管理が難しい情報環境でも，安心してキーワードによる XML 文書の検索環境を構築できる．

本論文は次のように構成されている．1 章のまえがきに続き，2 章は関連研究について述べる．3 章は提案手法について前提となる知識について述べる．4 章は IL の具体的なアルゴリズムについて述べる．5 章は提案手法を利用した暗号化 XML 文書検索システムと具体的なアルゴリズムについて述べ，また，提案手法の安全性について議論する．6 章は 5 章で述べた暗号化 XML 文書検索システムによる実験と考察について述べる．7 章はまとめと今後の課題について述べる．

2. 関連研究

2.1 暗号化データに対するキーワード検索手法

金子[3]らは，Non-ShuffledBF と ShuffledBF を組み合わせることで安全性を損なわずに RDB を高速に検索できる Semi-ShuffledBF と呼ばれる手法を提案したが，検索時に選択されるタプル数に合わせて Bloom Filter の処理を変えることでさらに高速化した手法を提案している．Watanabe[4]らは，データと鍵でハッシュ化するだけでは同じデータは同じ結果となり出現数からデータが推測される可能性があると指摘し，暗号化したデータをさらにハッシュ化することでデータの推測を防ぐ手法を提案している．

Yan[5]らは，疑似乱数ビットを文書に対するインデックスに処理することで，計算量や通信量が少ないモバイル機器でも暗号化したままで検索できる手法を提案している．Liu[6]らは，検索可能公開鍵暗号である PEKS に対して，クラウド上の文書に対するモバイル機器のキーワード検索を想定した計算量や通信量が少ない検索可能公開鍵暗号である CKPS と呼ばれる手法を提案している．

Wang[7]らは，XML 文書の要素の位置を 0 から 1 の実数で表現する DSI を利用して，暗号化した要素名と DSI のインデックスと DSI と対応する暗号化データのインデックスを RDB に登録することで，暗号化された XML 文書を検索する手法を提案している． Bertino[8]らは，XML 文書中の部分木を異なる鍵で暗号化することで利用者に対するアクセス制御を行う手法を提案している．Chang[9]らは，暗号化された XML 文書全体を入手して復号し検索するのは効率的ではないと指摘し，XML 文書のスキーマを利用して検索結果に必要な暗号

♥学生会員 信州大学大学院総合工学系研究科
s09t203@shinshu-u.ac.jp
♦正会員 信州大学工学部
yamamoto@cs.shinshu-u.ac.jp

化された部分のみ入手するようにXQueryを翻訳することで，暗号化された部分を減らす手法を提案している．Li[10]らは，事前にXML文書からアクセス制御の情報を追加したインデックスを作成することで検索結果にアクセス制御が反映される手法を提案している．Unay[11]らは，暗号化されたXML文書検索に関する手法について，暗号化標準と暗号化インデックスとパスやDeweyOrderの暗号化の視点でまとめたものを報告している．

2.2 検索手法の比較

提案手法と論文[3]～[10]について手法の比較を表1に示す．項目について，データは扱うデータの種類，検索は検索処理をクライアントかサーバのどちらで実行するか，条件は検索条件の指定方法，暗号化はデータに対して暗号化する範囲,暗号化(XML)はXML文書に対して暗号化する範囲の詳細，取得データはクライアントが取得するデータ，アクセス制御は利用者へのアクセス制御が可能かを示す．

提案手法は，XML文書の内容や構造を暗号化する範囲としてキーワードによる検索を行う．クライアントはサーバからXML文書中のキーワードに合致するデータの一部を取得し，サーバとクライアントで検索を実行する．利用者へのアクセス制御は対応しない．

論文[7]～[10]のようなRDBに暗号化したXML文書を登録して検索する手法は，条件に合う要素を取得する手法のため，SLCAを取得するにはクライアントで処理する必要があり，SLCAを検索する仕組みは考慮されていない．

暗号化ILは，XML文書を暗号化したデータに対して，データと同様に暗号化したキーワードと一致するデータのみ取得し，復号したデータにILを適用する手法である．

表1 手法の比較

Table 1 Comparison of the Technique

論文	ﾃﾞｰﾀ	検索	条件	暗号化	暗号化(XML) 内容	暗号化(XML) 構造	取得ﾃﾞｰﾀ	ｱｸｾｽ制御
金子[3]	RDB	ｻｰﾊﾞ	SQL	すべて			条件に合致するﾃﾞｰﾀのみ	不可
Watanabe[4]	RDB	ｻｰﾊﾞ	SQL	すべて			条件に合致するﾃﾞｰﾀのみ	不可
Yan[5]	ﾃｷｽﾄ	ｻｰﾊﾞ	ｷｰﾜｰﾄﾞ	すべて			条件に合致するﾃﾞｰﾀのみ	不可
Liu[6]	ﾃｷｽﾄ	ｻｰﾊﾞ	ｷｰﾜｰﾄﾞ	すべて			条件に合致するﾃﾞｰﾀのみ	不可
Wang[7]	XML文書	ｻｰﾊﾞ	XPath	一部	○	×	条件に合致するﾃﾞｰﾀのみ	不可
Bertino[8]	XML文書	ｻｰﾊﾞ	XQuery, XPath	一部	○	×	条件に合致するﾃﾞｰﾀのみ	可能
Chang[9]	XML文書	ｻｰﾊﾞ	XQuery	一部	○	×	条件に合致するﾃﾞｰﾀのみ	不可
Li[10]	XML文書	ｻｰﾊﾞ	ｷｰﾜｰﾄﾞ	一部	○	×	条件に合致するﾃﾞｰﾀのみ	可能
暗号化IL	XML文書	ｸﾗｲｱﾝﾄ	ｷｰﾜｰﾄﾞ	すべて	○	○	ｷｰﾜｰﾄﾞに合致するﾃﾞｰﾀすべて	不可
提案手法	XML文書	ｻｰﾊﾞ，ｸﾗｲｱﾝﾄ	ｷｰﾜｰﾄﾞ	すべて	○	○	ｷｰﾜｰﾄﾞに合致するﾃﾞｰﾀの一部	不可

3. 準備

XML文書は要素をノードとする木構造で表すことができ，木の各ノードは要素名でラベル付けされている．〈要素名〉の開始タグと〈/要素名〉の終了タグで挟まれるテキストも要素と同様にノードとして表すことができ，各ノードはテキストの内容でラベル付けされている．パスは根からノードまで辿る経路に出現するノードである．XML文書を検索するには，要素やテキストの位置関係（先祖，子孫，親，子，兄弟等）を把握するため，要素やテキストの位置を把握する必要がある．XML文書の要素やテキストの位置の表現に関して様々な手法があるが,ILはDeweyOrderを利用している.DeweyOrderは木の深さごとに″.″で区切り，同じ親の要素である兄弟の左端を0とし，右に向かって順に1,2,...と番号を割り当てたものである．本稿では，DeweyOrderはテキストを持つノードのDeweyOrderを意味する．

XML文書の図1(a)を同等の木で表現し，木の各ノードに要素名やテキストの内容とDeweyOrderを追加し，テキストのノードは下線を追加したものを図1 (b)に示す．

XQueryやXPathを利用した検索手法は,検索結果がXQueryやXPathが指定するXML文書中の特定の部分となるため，検索手法に関わらず検索結果は同じとなる．一方で，キーワードを利用したXML文書の検索手法は，検索結果に対する視点が異なるため，同じキーワードで検索しても検索手法によって結果が異なる.本稿で暗号化に対応する手法であるILは，XML文書中ですべてのキーワードを含む最小部分木の根である Lowest Common Ancestor(以下 LCA)の中で，子孫に LCAを持たない Smallest Lowest Common Ancestor(以下 SLCA)を検索する．提案手法の説明の簡略化のため，本稿ではXML文章中のテキストそのものをキーワードとし，キーワードと同じテキストと言った場合，キーワードと内容が完全に一致するテキストを意味するものとする．

Bloom Filterは要素が集合に属しているか調べるデータ構造である．Bloom Filterは長さmのビット配列と，k個のハッシュ関数を用いる．ハッシュ関数はビット配列の長さの範囲(0,...,m-1)の整数を出力する．要素を集合に登録するには，要素名をハッシュ関数で数値に変換し，数値に対応するビット配列のビットを1にする．要素が集合に含まれるか判定するには，要素名をハッシュ関数で数値に変換し，数値に対応するビット配列のビットが1であれば要素は集合に含まれる(図2)．Bloom Filterは誤って集合に属していると判断する偽陽性の可能性があるが，提案手法はクライアントでBloom Filterの判断が正しいか確認している．

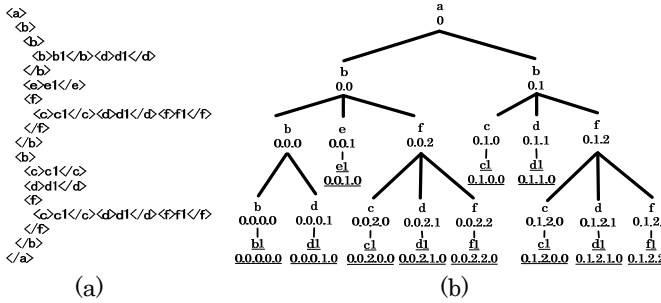

図1 XML文書の木構造表現

Fig.1 Tree Representation of the XML Document

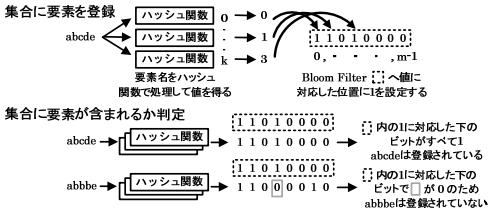

図2 ブルームフィルタ

Fig.2 Bloom Filter

4. The Indexed Lookup Eager Algorithmの手法

4.1 概要

The Indexed Lookup Eager Algorithm[1]は，3つのキーワードに対するSLCAを取得する場合，初めに1番目と2番目のキーワードに対するSLCAを取得し，次に先ほどのSLCAと3番目のキーワードに対するSLCAを取得する.このようにキーワードに対して順番にSLCAを繰り返し取得することで，すべてのキーワードに対するSLCAを取得する．SLCAはキーワードと同じテキストから見て，左側と右側で最も近い次の

キーワードと同じテキストの LCA が子孫となる方を SLCA としている．ただし，子孫にすでに検索した SLCA は含まないものとする．図 1(a)の XML 文書からキーワード c1, d1, e1 の SLCA を取得する場合は，以下の(1)～(4)の手順で行う．IL はテキストの内容毎にテキストの DeweyOrder の集合で構成されたデータから，キーワードと同じテキストすべての DeweyOrder の集合を取得する．2.2 節で述べた暗号化 IL はテキストの内容と DeweyOrder の集合を暗号化したデータから，暗号化したキーワードと同じテキストの内容に対応する DeweyOrder の集合を取得し，復号した DeweyOrder の集合に IL を適用する．

(1) 1つ目のキーワード(c1)と2つ目のキーワード(d1)と同じ内容のテキストを検索する(図 3○)．
(2) (1)で得たテキストに対する SLCA を求める(図 3□)．
(3) 3 つ目のキーワード(e1)と同じ内容のテキストを検索する(図 3 点線○)．
(4) (2)と(3)の要素に対する SLCA がキーワード c1, d1, e1 に対する SLCA となる(図 3 点線□)．

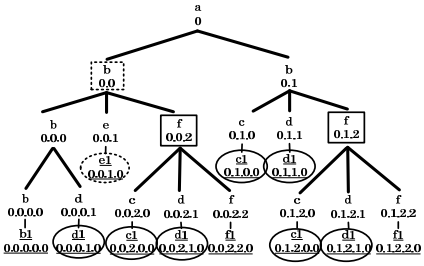

図 3 IL の手法

Fig.3 Method of IL

4.2 従来のキーワードによる XML 文書検索の問題点

IL はキーワードや検索結果の暗号化を行っていない．そのため，管理者がキーワードや検索結果を入手することで，情報漏えいが発生する問題がある(図 4)．

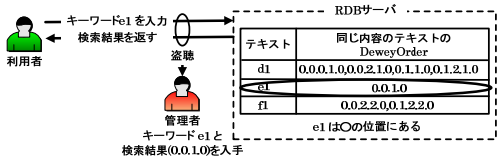

図 4 管理者による情報漏えい

Fig.4 Information Leakage by the Administrator

5 暗号化 XML 文書検索システム

4.2 節で述べた問題点に対応した，XML 文書を暗号化したままで検索する暗号化 XML 文書検索システム(以下システム)を以下に提案する．

5.1 概要

システムの概要を図 5 に示す．前処理として XML 文書を RDB サーバに登録するため，利用者は XML 文書と暗号化に使用する鍵(以下 K)をクライアントに入力する．クライアントは，K を用いて XML 文書の暗号化を行い，暗号化したデータを RDB サーバに登録する．検索は，利用者がキーワードと K をクライアントに入力する．クライアントは 5.2 節の手法に基づき，検索結果を返す．なお，RDB サーバからクライアントがデータを取得した場合，K によるデータの復号が行われる．

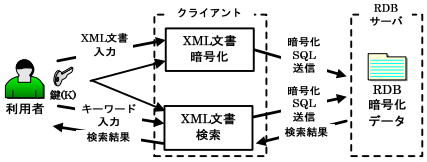

図 5 暗号化 XML 文書検索システムの構成

Fig.5 Configuration of Encrypted XML Document Search System

5.2 提案手法

5.2.1 概要

4.1 節で述べたように IL が SLCA を検索するには，キーワードと同じテキストに対して，左側と右側で最も近い次のキーワードと同じテキストが必要となる．IL はキーワードと同じテキストを，XML 文書全体を範囲として RDB サーバから取得する．提案手法は，1 番目のキーワードと同じテキストを，XML 文書全体を範囲として RDB サーバから暗号化したままで検索し取得するが，2 番目以降のキーワードは前のキーワードと同じテキストの親から根に向かったパス上の要素を根とする部分木を範囲として，次のキーワードと同じテキストを RDB サーバから暗号化したままで検索し取得する．提案手法は検索する範囲が XML 文書全体より狭いため，IL よりテキストの取得を減らすことができる．検索したテキストから 4.1 節で述べた手法で SLCA を取得する．

部分木の範囲でキーワードと同じテキストの検索をする場合は，事前に XML 文書の要素毎に要素名とパス上の要素の位置を Bloom Filter に登録する(図 6①)．検索時はキーワードと範囲とする部分木の根から XML 文書の根までのパス上の要素の位置から Bloom Filter を作成し(図 6②)，作成した Bloom Filter と同じデータが RDB サーバの Bloom Filter に含まれるか判定する(図 6③)．

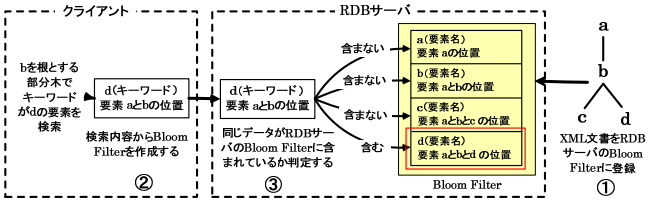

図 6 部分木のキーワード検索

Fig.6 Keyword Search in the Subtree

クライアントと RDB サーバとの間で以下のデータのやり取りを行う．

(1) Bloom Filter と暗号化された情報で構成されたデータテーブルを作成する(図 7(a)①)．
(2) キーワードと同じテキストを取得するため，テキストの内容とパスに関する情報から Bloom Filter を作成し，作成した Bloom Filter と同じデータが含まれるデータテーブルの Bloom Filter に対応した暗号化された情報を取得する(図 7(b)①②④)．
(3) (2)で取得した暗号化された情報から IL と同じ手法で SLCA を取得する(図 7(b)③)．

ハッシュ化の際，ハッシュから平文を推測されないようにするため，平文に鍵を追加している．ハッシュ化は，ハッシュ化する平文のデータを HD，鍵を K，H_{hash} をハッシュ関数，

HD′をハッシュ化データとして式(1)に従って行っている．

$$HD' = H_{hash}(K, HD) \quad (1)$$

暗号化の際，平文と暗号文から鍵を推測されないようにするため，平文にランダム文字列を追加している．暗号化は暗号化する平文のデータをED，ランダム文字列をS，鍵をK，暗号化関数をE_{enc}，暗号化データをED′として式(2)，復号は復号関数をE_{dec}として式(3)に従って行っている．

$$ED' = E_{enc}(K, ED, S) \quad (2)$$
$$ED = E_{dec}(K, ED') \quad (3)$$

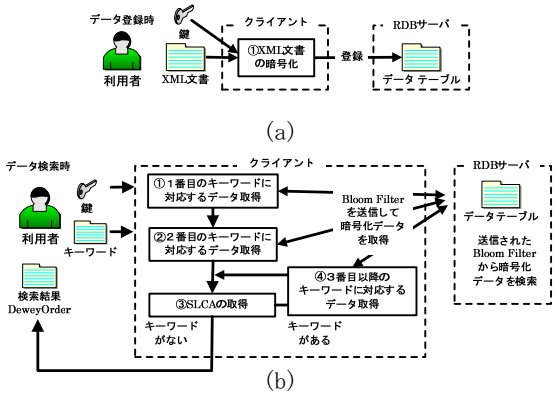

図 7 提案手法

Fig.7 Proposed Method

5.2.2 XML 文書の暗号化

XML 文書を RDB サーバ に登録するため，データテーブル(以下 DTBL)を構築する．図 1(a)を XML 文書とする DTBL の構造を図 8 に示す．図 8 の EncryptedData 列の括弧内は，そこに格納されている暗号化データの元の値(要素名，テキスト，DeweyOrder，位置情報)を示している．

DTBL は，5.2.1 節に従ってハッシュ化された要素名やテキストの内容と位置情報と DeweyOrder が登録された Bloom Filter，5.2.1 節に従って暗号化されたデータである EncryptedData で構成されている．位置情報はパス上の要素やテキスト毎に要素名やテキストの内容と深さを”:”でつなげたものである．DeweyOrder が 0.0.2.0 の位置情報は a:0,b:1,f:2,c:3 であり，それぞれを Bloom Filter に登録する．DeweyOrder は Bloom Filter がすべて異なるようにするため登録している．暗号化されたデータは要素名やテキストの内容と DeweyOrder と位置情報で構成されている．紙面の関係から DTBL の一部と Bloom Filter と EncryptedData の暗号列の途中を省略する．この処理における計算量は，XML 文書の要素とテキストの合計を n とした場合 $O(n)$ である．

Bloom Filter	EncryptedData
00001...00000	mJ4c8...yhiC4 (c,0.0.2.0,a:0,b:1,f:2,c:3)
00000...01000	Mc2jK...VU4uS (d,0.0.2.1,a:0,b:1,f:2,d:3)
00000...00000	vih0I...etA== (b1,0.0.0.0.0,a:0,b:1,b:2,b:3,b1:4)
⋮	
00000...00000	/+lFj...NiY0V (b,0.0.0,a:0,b:1,b:2)
00000...01000	7fQvb...VU4uS (d,0.1.2.1,a:0,b:1,f:2,d:3)
00000...00000	LcUIB...mZz9H (f,0.1.2.2,a:0,b:1,f:2,f:3)

図 8 図 1(a)の XML 文書に対応する DTBL

Fig.8 DTBL for the XML Document in Fig. 1(a)

5.2.3 キーワードに対応するデータ取得

キーワードと同じテキストの取得を以下の(1)～(3)の手順で行う．キーワードが 3 つ以上の場合はいままでの SLCA の位置情報と(3)からキーワードに対応するデータ取得を行う．Bloom Filter は誤ってデータを取得する場合があるため，取得後にテキストの内容がキーワードと一致するか確認する．(3)で判定結果はクライアントに記憶するため，以前に作成した Bloom Filter で判定は行わない．この処理における計算量は，キーワードと同じテキストのパス上の要素を根とする部分木の要素とテキストの合計を n とした場合 $O(n)$ である．

(1) 5.2.1 節に従ってハッシュ化されたキーワードから Bloom Filter を作成し，同じデータが RDB サーバの Bloom Filter に含まれるか判定する．含まれる Bloom Filter に対応した EncryptedData を取得する．

(2) (1)で取得した EncryptedData 毎に 5.2.1 節に従って復号し DeweyOrder と位置情報を取得する．取得した EncryptedData 毎に(3)を行う．

(3) 5.2.1 節に従ってハッシュ化されたキーワードと位置情報から最も深さが大きいものを除いた残りから Bloom Filter を作成し，同じデータが RDB サーバの Bloom Filter に含まれるか判定する．含まれる Bloom Filter に対応した EncryptedData を取得する．取得した EncryptedData 毎に 5.2.1 節に従って復号し，DeweyOrder が(2)の DeweyOrder に対し左側か右側にあるか調べ，左側右側毎に取得したデータを保存する．再び左側か右側にデータがあった場合，最初に保存したデータより(2)の DeweyOrder から遠くなるため保存しない．左側と右側の両方のデータが揃うまで(3)を繰り返す．

キーワード c1 と同じテキスト(図 9①)に対して，キーワード d1 と同じテキストを検索する場合，テキストの親(図 9②)から XML 文書の根(図 9④)に向かってパス上の要素(図 9②③④) 毎に，要素を根とする部分木にキーワード d1 と同じテキストがないか検索する(図 9 矢印)．結果として，図 9 ①に対する右側のキーワード d1 と同じ内容のテキストとして，要素(図 9②)を根とする部分木に次のキーワード d1 と同じテキスト(図 9⑤)を発見し，左側のキーワード d1 と同じテキストとして，XML 文書の根(図 9④)を根とする部分木に次のキーワード d1 と同じテキスト(図 9⑦⑧)を発見する．

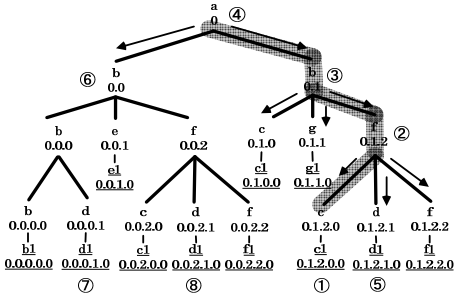

図 9 キーワードによるデータ取得

Fig.9 Obtaining Data by Keywords

5.2.4 SLCAの取得

5.2.3節で取得したデータを5.2.1節に従って復号し，EncryptedData中のDeweyOrderを利用して，ILと同様の手法でSLCAを取得する(図10)．

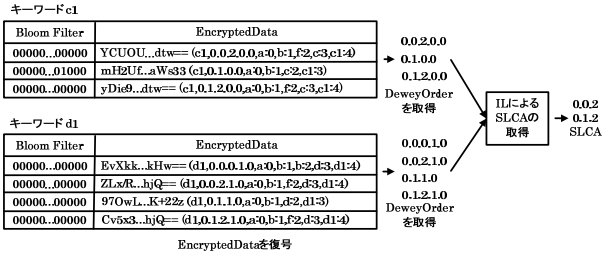

図 10 SLCA の取得

Fig.10 Obtaining SLCA

5.3 安全性について

提案手法の安全性の評価を，想定される攻撃に対する対応策を分析することで行った．

(1) RDBサーバに対する攻撃

- DTBLのEncryptedDataに対する既知平文攻撃
  攻撃を行うには，要素に関するデータがある平文が必要だが，5.2.1節で述べたように平文にランダムな文字列を追加して暗号化されており，平文は入手できない．そのため，攻撃はできない．
- DTBLのEncryptedDataに対する選択平文攻撃
  既知平文攻撃で述べたように，ランダムな文字列を追加して暗号化してあるため，平文や平文に対する暗号文も入手することができない．そのため，攻撃はできない．
- DTBLのBloom Filterに対する攻撃
  5.2.2節で述べたようにBloom Filterは要素名やテキストの内容とDeweyOrderをハッシュ関数で処理したデータを登録しているため，安全性は利用するハッシュ関数に依存する．

(2) RDBサーバとクライアント間のデータのやり取りに対する攻撃

- 送信するキーワードに対する攻撃
  5.2.1節で述べたように，RDBサーバへBloom Filterのみ送信されるため，安全性は利用するハッシュ関数に依存する．
- RDBサーバから取得するデータに対する攻撃
  RDBサーバから暗号化されたDTBLのデータのみ取得するため，(1)RDBサーバに対する攻撃で述べたように，攻撃はできない．

5.4 提案手法の改善について

Bloom Filter用のインデックステーブルを追加してRDBサーバのインデックスを利用することで検索を高速化する改善を行った．

5.2.3節で実行するSQLは，論理積でBloom Filterに1が設定されているデータを抽出し，排他的論理和で1が設定されている部分が0の場合は，データを取得しない条件としていた(図11(a)①)．しかし，実験で使用したRDBサーバは，条件に論理積や排他的論理和を使用するとRDBサーバのインデックスを使用しないことがわかった．そのため，Bloom Filterの各ビットが列と対応するテーブルDTBL_INDEXを構築し(図11(b))，条件をBloom Filterのビット位置に対応する列が1と等しいか比較するSQLとした(図11(a)②)．ただし，この手法はBloom Filterのビット数分のインデックスが必要なため，DTBL_INDEXの構築に時間がかかる．

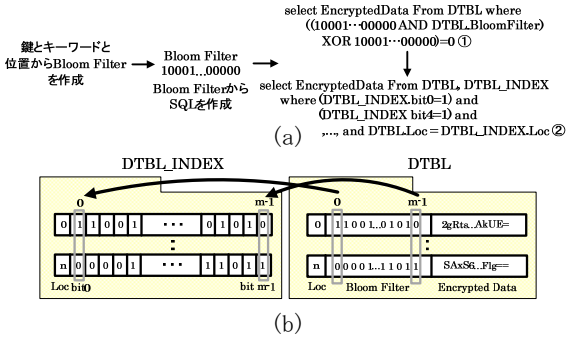

図 11 Bloom Filter のインデックス構築

Fig.11 Constructing Index for Bloom Filter

6. 実験

6.1 実験環境について

提案手法と 2.2 節で述べた暗号化 IL と IL の性能比較を行うため実験を行った．RDB サーバは，Intel Xeon 2GHz，メモリ 4GB，CentOS 6.0，クライアントは，Intel Core2Duo 2.26GHz，メモリ 2GB，Windows XP の PC を 100Mbps のネットワークで接続して実験を行った．RDB サーバは MySQL 5.1.52 を使用し，提案手法は 5.4 節で述べた手法で Bloom Filter，暗号化 IL と IL はテキストの内容でインデックスを構築した．提案手法と暗号化 IL と IL のプログラムは，プログラミング言語を Python，ハッシュ関数を SHA1，暗号化アルゴリズムは AES を採用した．

XML 文書は XMark[12]で公開されている XML 文書作成ソフトウェアで，50MB の XML 文書を作成した．XML 文書の要素とテキストの合計は 1,254,624 個であった．

Bloom Filter について，誤って判定してしまう確率を 0.1%未満と想定し，登録する要素数は要素名やテキストの内容とパス上の要素毎の位置情報と DeweyOrder を想定し(n=5)，長さ 108(m=108)のビット配列，8 個のハッシュ関数(k=8)を用いた．

XML 文書作成ソフトウェアは，オークションに関するデータを想定した XML 文書を作成する．引数の指定により XML 文書の大きさも指定できる．図 12 に作成する XML 文書の構造の概要を示す．

実験は，提案手法と 2.2 節で述べた暗号化 IL と IL に対して検索するキーワード数と XML 文書中の同じ内容のテキスト数を変化させて行った．検索したキーワードとテキストの数を表 2 に示す．表 2 の Keywords は検索したキーワード，キーワード右の括弧で囲まれた数値は XML 文書中のキーワードと同じテキストの数を示す．表 2 のキーワードに対して，結果は 3 回測定した検索時間の平均値とした．

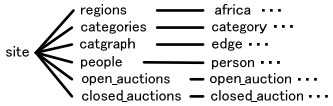

図 12 XML 文書の構造

Fig.12 Element Hierarchy of XML Document

表 2 検索するキーワード

Table 2 Keyword to Search

No.	Keywords
K1	Money order, Personal Check (632), Will ship only within country (657), United States (11369), 1(18553)
K2	Featured, Dutch (217), 2 (2353), 1 (18553)
K3	Graduate School (676), United States (11369)

6.2 実験結果

表 3 に提案手法と 2.2 節で述べた暗号化 IL と IL の検索時間を示す．項目について，RDB は RDB サーバの実行時間，Decrypt はデータを復号する実行時間，Other は RDB と Decrypt 以外の実行時間，Total は RDB と Decrypt(表 3(a)(b)のみ)と Other の実行時間，SLCA は検索した SLCA の数，Text は RDB から入手した要素やテキストの数，Data はサーバとクライアント間の通信量を示す．

提案手法(表 3(a))と 2.2 節で述べた暗号化 IL(表 3(b))と IL(表 3(c))を比較すると，IL に比べて暗号化 IL は約 1.1 倍遅く，提案手法は約 97～231 倍遅い．この理由として，IL と暗号化 IL はキーワードと同じテキストのデータを 1 回で検索している．対して，提案手法はキーワードと同じテキストの親から，パス上の要素を根とする部分木に対して検索しているため，IL や暗号化 IL と比べて遅いと考えられる．提案手法は IL や暗号化 IL と比べて遅いが，XML 文書の一部しか検索しないため Text は IL や暗号化 IL と比べて少ない．しかし，提案手法は Bloom Filter を利用して XML 文書の内容だけでなく構造も隠したままで検索するが，クライアントは EncryptedData を取得して検索結果が正しいか確認して擬陽性に対応するため，Data は IL や暗号化 IL に比べて多くなってしまう．

表 3 実験結果(検索時間)

Table 3 Experimental Results for Search Times

	RDB (ms)	Decrypt (ms)	Other (ms)	Total (ms)	SLCA	Text	Data (Bytes)
K1	62,133	553	119,400	182,086	22	17,257	19,880,060
K2	50,583	439	30,381	81,403	15	11,603	17,221,320
K3	33,872	207	29,100	63,179	240	4,913	9,544,084

(a)

	RDB (ms)	Decrypt (ms)	Other (ms)	Total (ms)	SLCA	Text	Data (Bytes)
K1	266	5	627	898	22	31,211	588,976
K2	161	10	449	620	15	21,123	390,232
K3	114	0	580	694	240	12,045	235,908

(b)

	RDB (ms)	Other (ms)	Total (ms)	SLCA	Text	Data (Bytes)
K1	193	595	788	22	31,211	441,688
K2	130	428	558	15	21,123	292,643
K3	83	569	652	240	12,045	176,903

(c)

7. おわりに

本論文では，管理者が検索内容や結果がわからない，安全なキーワードによる暗号化 XML 文書検索手法を提案した．

提案手法について，XML 文書を暗号化する範囲の攻撃に対して安全性の評価を行い安全であることを示した．XMark[12]で公開されている XML 文書作成ソフトウェアで作成した XML 文書で，IL と提案手法の手法との性能比較を行い，検索時間は IL より約 97～231 倍遅い結果となった．

今後の課題は検索時間の高速化と DTBL_INDEX を高速に構築する手法の提案である．

【文献】

[1] Yu Xu, Yannis Papakonstantinou:“Efficient Keyword Search for Smallest LCAs in XML Databases”, SIGMOD '05 Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp.527-538 (2005).

[2] “AmazonSimpleDB”, http://aws.amazon.com/jp/simpledb/.

[3] 金子 静花, 渡辺 知恵美 , 天笠 俊之:“ブルームフィルタを用いたプライバシ保護検索システム Semi-ShuffedBF の問合せ高速化についての諸検討”, 第 4 回データ工学と情報マネージメントに関するフォーラム(DEIM2012), B7-4 (2012).

[4] Watanabe C. and Arai Y:“Privacy-Preserving Queries for a DAS model using Two-Phase Encrypted Bloomfilter”, Database Systems for Advanced Applications, pp.491-495 (2009).

[5] Chang, Yan-Cheng, and Michael Mitzenmacher: “Privacy preserving keyword searches on remote encrypted data”, Applied Cryptography and Network Security, pp.442-455 (2005).

[6] Liu, Qin, Guojun Wang, and Jie Wu:“Secure and privacy preserving keyword searching for cloud storage services”, Journal of network and computer applications 35.3 pp.927-933 (2012).

[7] Wang, Hui, and Laks VS Lakshmanan:“Efficient secure query evaluation over encrypted XML databases”, Proceedings of the 32nd international conference on Very large data bases, pp.127-138 (2012).

[8] Bertino, Elisa, and Elena Ferrari:“Secure and selective dissemination of XML documents”, ACM Transactions on Information and System Security (TISSEC) 5.3 pp.290-331(2002).

[9] Chang, Tao-Ku, and Gwan-Hwan Hwang:“Developing an efficient query system for encrypted XML documents”, Journal of Systems and Software 84.8 pp.1292-1305 (2011).

[10] Xiaodong Li, Weidong Yang, Jiangang Zhu, Hao Zhu:“SXKSearch: A System of XML Keyword Search Based on Secure Access Control”, Proceedings of the 2010 International Conference on Internet Computing, pp.77-82(2010).

[11] Ozan Ünay, Taflan I. Gündem:“A survey on querying encrypted XML documents for databases as a service”, ACM SIGMOD Record 37.1 pp.12-20 (2008).

[12] “XMark - An XML Benchmark Project”, http://www.xml-benchmark.org/.

中村伸一 Shin-ichi NAKAMURA

信州大学大学院総合工学系研究科博士課程在学中．信州大学大学院工学系研究科・情報工学専攻修了．データベースに関する研究に従事．日本データベース学会学生会員．

山本博章 Hiroaki YAMAMOTO

信州大学工学部教授．東北大学大学院工学研究科博士課程修了．工学博士．アルゴリズムの開発，オートマトン，情報検索に関する研究に従事．電子情報通信学会，情報処理学会，日本ソフトウェア科学会，日本データベース学会，EATCS 各会員．