論文 / 著書情報
Article / Book Information

論題(和文)	プロキシ暗号を用いた被災者情報アクセス制御手法の評価
Title(English)	
著者(和文)	杉原 弘祐, 三上 英明, 横田 治夫
Authors(English)	Kousuke Sugihara, Hideaki Mikami, Haruo Yokota
出典(和文)	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論文集, , , E7-3
Citation(English)	, , , E7-3
発行日 / Pub. date	2016, 3

プロキシ暗号を用いた被災者情報アクセス制御手法の評価

杉原　弘祐†　　三上　英明†　　横田　治夫†

† 東京工業大学工学部情報工学科　〒 152–8552 東京都目黒区大岡山 2-12-1
E-mail: †{sugihara,mikami}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし　大規模災害における被災者情報等を管理する際には, 家族, 医療チーム, 地方自治体職員, ボランティア等の異なるクラスのユーザに対して, 個人情報の持ち主の意向に沿ったポリシーでプライバシィを保護しながら, 情報を提供することが求められる. 先行研究では, プロキシ暗号を利用することで, サーバ管理者を信頼することなく, ユーザのクラスに対応したアクセス制御を行い,RDF 形式の被災者情報を格納, 検索する手法の提案を行っている. 本研究では, 先行研究で復号することなく検索するために確定的暗号で行っていた部分を疑似確率的暗号化することで秘匿性を高めるとともに, 実際の被災状況の情報に近いベンチマークを用いて提案手法の評価を行う.

キーワード　プロキシ暗号, アクセス制御, 被災者情報, 確定的暗号, 確率的暗号

1. はじめに

災害は, 被災地における損失や個々人の生活に大きな影響を与え, その影響は長期にわたることも少なくない. 近年では災害時や災害後の復旧及び復興を目的とした情報技術活用が注目され, 被災地に関する情報や安否情報などの情報を共有を支援するためのシステムが利用されている. このようなシステムにおける被災者情報には個人情報等のプライバシーに関わるものが含まれる場合がある. そのため不正利用を防止するためにデータを暗号化し, データへのアクセス制御を行う必要がある. 被災者情報管理システムでは, 利用者として主に家族, 医療チーム, 地方自治体職員, ボランティア等の異なるクラスのユーザが考えられ, 個人情報の持ち主であるユーザの移行に沿ったポリシーでプライバシィを保護しながら, 情報を提供することが求められる.

一方で, 現在までに様々なデータ記述法が開発され, その一例として RDF 形式が知られている.RDF 形式により表現されたデータはグラフ構造で表すことができ, 情報の結合などができる点から情報管理が容易に行えるとされている.

先行研究において児玉ら [1] による共有情報に対するアクセス制御としてクラスを単位とした暗号化を行う手法が提案され, プロキシ再暗号化を用いることで, RDF 形式の情報をププロキシサーバで復号化することなく再暗号化することを可能とし, アクセス制御の柔軟性を維持させることを可能とした. しかしながらこの手法に用いた暗号化は確定的暗号化であるため, 頻度分析攻撃への耐性が懸念される. 頻度分析攻撃への対策として確率的暗号を利用することが考えられるが, 暗号化されたデータへの検索にかかる実行時間の増加が懸念される.

そこで本研究では, 先行研究において確定的暗号で行っていた部分を疑似確率的暗号化することで, 実行時間の増加を抑えながら秘匿性を高めるとともに, 避難場所情報管理システムを評価するための避難場所ベンチマーク SIBM(Shelter Information Benchmark) [2] を利用した暗号化システムの評価を行う.

本論文の構成は以下の通りである.2 章では本研究に関して前提となる知識を述べる.3 章で関連する先行研究を紹介する.4 章では本論文において提案する手法について述べる.5 章で実験及び評価を行い, 最後に 6 章で本論文のまとめと今後の課題を述べる.

2. 前提知識

2.1 アクセス制御手法

アクセス制御とは, 情報資源の不正利用を防止するために, 決められた規則に従って資源へのアクセス要求に対し, その可否を制御することで情報およびシステムを守ることが目的である. [3]. アクセスを許可する条件を定義した規則の集まりをアクセス制御ポリシーと呼ぶ. このポリシーにより, システムのセキュリティにおける情報資源の秘匿性, システムの状態が正しいと保証する完全性, そしてアクセス権を持つユーザが実際にシステムを利用できることを保証する可用性が守られているとし, アクセス制御システムを経由して実行されたすべてのアクセスが正当なものであることを保証する.

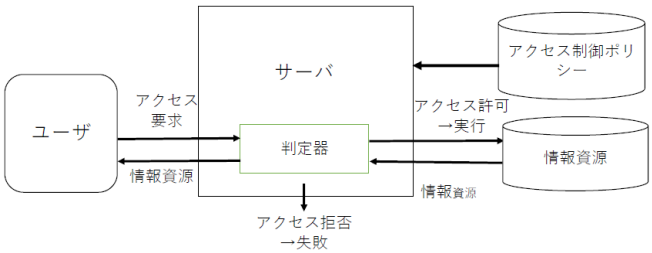


図 1　基本的なアクセス制御モデル.

しかしながら, アクセス制御システムの全体を完全に信頼することは現実的ではない. 例えばシステム管理者などの内部関係者による悪意のあるデータの取得や, 外部からの攻撃によるデータの漏洩が起きる場合がある. このような場合に備え, 不適切なアクセスによってデータが取得されるのを防ぐアクセス制御だけでなく, 暗号化による保管データの保護が必要がである. 近年, 急激に普及しているクラウドストレージサービスは企業により情報資産を管理する手段として利用されるようになり,

秘匿情報の効率的な蓄積や共有ができる. データ保有者はローカルシステム上で, 暗号化したデータをアップロードするデータの利用者は, 暗号化された情報をダウンロードした後にローカルシステム上で復号して使用する. これにより暗号化鍵を適切に管理する必要はあるものの, クラウドサーバがたとえ攻撃された場合でも被害を最小限に抑えることができる.

2.2 プロキシ再暗号化

再暗号化とは, 暗号化されたデータを復号化することなく再び暗号化することであり, 自分にしか復号化できないデータを自分以外の他者が復号化できるように変換することである. この操作は代理人暗号, 代理人再暗号化とも言われるが特に, 第三者であるプロキシによって再暗号化が行われる暗号体系は, プロキシ再暗号化と呼ばれる. プロキシ再暗号化技術によりプロキシは平文の情報を得ずに再暗号化ができるので, 従来の他のユーザが持つ情報を得る際の復号化方法を, 再暗号化処理と受信するユーザによる復号処理に分けることができ, プロキシに再暗号化処理のみを任せることで, データをプロキシ内で復号化せず, 暗号文のまま受信するユーザへ渡すことができる. このようにプロキシ再暗号化技術の最終的な目標は, 信頼されていないサーバを通しながらも, データの再暗号化に成功することである [4].

一般に, 公開鍵暗号方式をベースとするプロキシ再暗号化アルゴリズムは, 5 つの処理によって構成される.

鍵生成 $KeyGen(1^k) \rightarrow (pk_A, sk_A)$
鍵生成アルゴリズム KeyGen は, セキュリティパラメータ 1^k の入力を元に主体 A の公開鍵と秘密鍵の組 (pk_A, sk_A) を出力する.

暗号化 $Enc(pk_A, m) \rightarrow \quad C_A$
暗号化アルゴリズム Enc は, 主体 A の公開鍵 pk_A と平文 m を入力として,A の秘密鍵 sk_A で復号可能な暗号文 C_A を出力する.

復号 $Dec(sk_A, C_A) \rightarrow \quad m$
復号アルゴリズムは Dec, 主体 A の秘密鍵 pk_A とそれによる復号が可能な C_A を入力として, 平文 m を出力する.

再暗号化鍵生成 $ReKeyGen(pk_A, sk_A, pk_B, sk_B) \rightarrow rk_{A\rightarrow B}$
再暗号化アルゴリズム ReKeyGen は, 主体 A 及び主体 B の公開鍵と秘密鍵の組を入力として, 再暗号化鍵 $rk_{A\rightarrow B}$ を出力する

再暗号化 $ReEnc(rk_{A\rightarrow B}, C_A) \rightarrow C_B$
再暗号化アルゴリズム ReEnc は, 再暗号化鍵 $rk_{A\rightarrow B}$ と, 暗号文 C_A を入力として,sk_B による復号可能な暗号文 C_B を出力する

プロキシ再暗号化方式のひとつに, ElGamal 暗号をベースとして Blaze ら [5] により考案された BBS 方式がある. BBS によるアルゴリズムは次のように表現される.

鍵生成 $sk_a \in \mathbb{Z}^*_{2q}, \quad pk = g^{sk}$
g は安全素数 $p = 2q + 1$ を法とする原始根である.

暗号化 $m \in \mathbb{G},\ k \in \mathbb{Z}^*_{2q}$
$c_1 = mg^k \mod p$
$c_2 = (g^{sk})^k \mod p$

復号 $m = c_1((c_2^{(sk^{-1})})^{-1}) \mod p$

再暗号化鍵生成 $rk_{A\rightarrow B} = sk_A^{-1} sk_B$

再暗号化 $c_2 \mapsto c_2^{rk}$

プロキシ再暗号化技術により, 再暗号化を行う際には注意が必要であり, いかにユーザの秘密鍵を漏洩させずに再暗号化鍵を生成するかが問題となる. しかし現在のところ第三者を一切信頼することなく鍵生成を行う手法は提案されていない. また, 再暗号化鍵ではデータの復号化はできないことから多くの手法では信頼可能な秘密鍵生成局などの機関に預託するなどの第三者の存在を利用することで再暗号化は実現されている.

2.3 RDF について

RDF (Resource Description Framework) は, Web 上に存在する資源に関する情報 (メタデータ) を記述するための枠組みである. RDF の基本構造は, 図 2 のように, 2 個のノード (サブジェクトおよびオブジェクト) とそれらを結ぶ有向辺 (プレディケート) から成るトリプル (3 つ組) でリソースに対する関係情報を表現する. サブジェクトはウェブ識別子 (Uniform Resource Identifer(URI)) によって確実に識別できるリソースのみであり, サブジェクトとオブジェクトの関係を表すプレディケートはメタデータの種類を表す語彙により表現する.

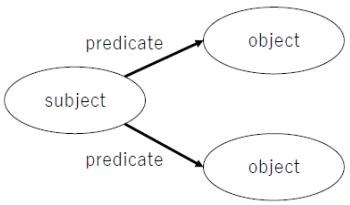


図 2　RDF トリプルによる情報の表現.

トリプルの集合は RDF グラフと呼ばれグラフ構造をもつ. その場合, 特定のグラフパターンに当てはまるトリプルを得られるようにするために標準の問い合わせクエリ言語である SPARQL [6] を用いてグラフパターン検索を提供することが多い. SPARQL による問い合わせは任意のグラフパターンのマッチングをすることができ, 小さなパターンを組み合わせることで, より複雑なグラフパターンを作成することができる. また,SPARQL の機能として, 一度のクエリで複数の複数の SPARQL エンドポイントに問い合わせ, 結果を統合する事ができる.RDF による表現で記述されたデータベースを利用すると, 同様に機械可読なフォーマットで記述された外部のデータベースと柔軟に連携することが可能になる. これはデータベース間にリンクがある場合に仮想的に統合したグラフによる処理が行えるからである.

クエリは条件に合った部分グラフを探すため解の組み合わせが増大する. またクエリ内の処理において処理順序によりステップ数が変わることからクエリを構成するアルゴリズムが重要となる.

2.4 確定的暗号と確率的暗号

同じ平文をを暗号化した際に出力される暗号文は一意に決まる方式を確定的暗号という. これに対し出力される暗号文が複数存在する場合の方式を確率的暗号という. 確定的暗号におけるデータ検索においては暗号文のままでの一致比較をすること

ができるといった利便性も存在する. データサイズが n とするとき, 確率的暗号での全数探索を行う必要があり,$O(n)$ の検索時間がかかるとされ, 確定的暗号ではその必要がなく,$O(\log n)$ ですむと伊藤ら [7] に指摘されている. しかしながら確定的暗号を利用した場合, 暗号化キーワードの頻度からキーワードを推測する「頻度分析攻撃」が挙げられている. 例えば名前や, 都道府県などデータベース内に格納されたデータの種類を知られると, その出現頻度からデータの内容を調べられる場合がある.

3. 関 連 研 究

児玉らの研究 [1] では共有情報に対し, ロジカルなクラスを単位とした暗号化を行うことでクラス単位でのアクセス制御を可能にする手法を提案し, 暗号化されたデータベースとユーザ間にプロキシを配置した情報共有管理システムを作成した.

この手法ではユーザの認証として, ユーザは一意に識別可能な ID を持ち,ID はユーザは高々一つのクラスに対応付けられる. また, このクラスは 0 個以上の階層関係を持ったアクセスレベルに対応付けられる.

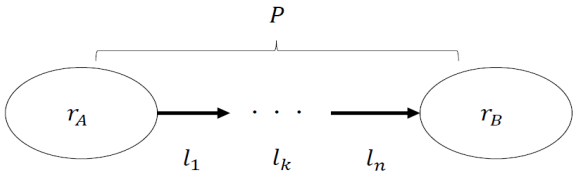
蓄積される情報は全て RDF 形式での記述を前提とし, ユーザから送られる全ての RDF トリプルは, プロキシサーバにより, アクセス制御ポリシーに基づいた暗号化が行われる. アクセス制御ポリシーは,RDF トリプル (s, p, o) を入力として, 各 RDF 項へのアクセス制御に関する情報 $(L, C) = ((l_s, l_p, l_o), (c_s, c_p, c_o))$ を出力する. L は RDF 項に対応づけられるアクセスレベルを示し,C は後述する個人間関係の情報を用いたアクセス制御ポリシーである. また情報は暗号化された RDF と共にアクセス制御ポリシーが保存される.

このシステムでは RDF 項に対応付けられたアクセスレベルにより暗号化されることから, 各アクセスレベルは公開鍵秘密鍵ペア持ち, その鍵は第三者である信頼可能な秘密鍵生成局により生成される. この鍵はプロキシには送られず, 代わりにプロキシは, 秘密鍵生成局に対し再暗号化鍵の生成を要求する. これにより, プロキシがユーザからプロキシに送信される情報や問い合わせをユーザの公開鍵で暗号化した暗号文をアクセスレベルに対応した鍵による暗号文に再暗号化する. 同様に, プロキシ内でデータベースから受信した RDF グラフをユーザが復号可能な暗号文に再暗号化し, これをユーザへ送信する. ユーザは生成局に対し復号化鍵の送信を要求し, プロキシから送信された情報を復号化する. このようにユーザとデータベースの間に配置されるが, 例えプロキシが攻撃されようとユーザやアクセスレベルの鍵が漏洩されることはない.

ユーザによるアクセスは, プロキシによりその要求に基づいた暗号化されたデータベースを検索可能な問い合わせに変換される. 先行研究ではユーザによるデータベースへの問い合わせの手法として 2 つの手法が提案されている.

一つはオープンな情報をもとにしたアクセス制御情報による制御手法である. ユーザの属するクラスやそれに対応するアクセスのレベルといったオープンな情報をもとにしたアクセス制御情報をもとに,SPARQL クエリを変換し問い合わせを行う. この SPARQL クエリにより, あるアクセスレベル l 以上の高いアクセスレベルに対応づけられたクラスに属するユーザは,l を対応付けられた RDF 項へアクセスすることができ, また l 以下の低いレベルが対応付けられた RDF 項にもアクセスが可能である.

もう一方は, クローズドな情報をもとにしたアクセス制御情報による制御手法である. より柔軟なアクセス制御を行うために, 個人関係が記述された情報を用いたアクセス制御である. これは個人間の間柄が記述された情報を用いた制御ポリシーを生成し, オープンな情報のアクセス制御によってアクセスできない情報でも, この制御ポリシーの条件を満たせば, データを受信できる方法である. 図 3 のように人を表すリソース間に有向パス $P : r_A \rightarrow r_B$ が存在する場合, かつ下式の条件を満たす時,B に関する closed ポリシーを持った情報への参照を A に許可するものである. このポリシーによりアクセスレベルの低いユーザでも, 高いアクセスレベルに対応付けられた特定の情報に対してもアクセス可能である.



$$n \leq \max(c_{slength}, c_{plength}, c_{olength}) \quad \wedge$$
$$\forall l \,.\, l \geq \min(c_{slevel}, c_{plevel}, c_{olevel})$$

図 3　個人間関係によるアクセス制御

児玉らが提案した情報共有管理システムではプロキシ再暗号化方式として,Blaze らによる BBS アルゴリズムを採用し, 鍵サイズを 2048 ビットとした. また BBS 暗号は確率的暗号であるため, 暗号文による一致検索ができるよう確定的暗号にするため, 発生され乱数を固定した.

プロキシ再暗号化は暗号化の利便性を下げることなく, クラウドサービス上などでより安全に通信ができる技術としても注目がされている. 現在においても, 他の暗号化方式として, ペアリング技術を用いた Identity Based encryption(IBE) [8] が提案され, 実用に向け様々なプロキシ再暗号化方式が提案されている.

それと同時に, 既存研究における暗号化手法に対しての, 安全性における脆弱性, 及び攻撃方法も発見されている.Ateniese ら [4] は BBS 方式は推移性があることから再暗号化鍵 $rk_{B\rightarrow A} = a/b$ と $rk_{C\rightarrow B} = b/c$ から Alice から Carol への再暗号化鍵 $rk_{C\rightarrow A} = c/a$ をプロキシが生成できること, またシステムに対する攻撃例として, プロキシとユーザ Bob が結託し, プロキシが受信した再暗号化鍵 (a/b) と Bob により提供された秘密鍵 b を用いることでユーザ Alice の秘密鍵 a が特定できると指摘した. また同著者 [9] は, より簡単な攻撃方法として再暗号化鍵 $rk = b/a$ と仮定であれば, 公開鍵の組 (g^a, g^b) を用いて $(g^a)^{rk} = g^b$ である rk かどうかを攻撃者は行うことができると指摘した.

3.1 SIBM(ベンチマークツール)

NGUTYEN らの研究 [2] では,RDF 形式で記述した災害・避難

情報のデータセット SIBM(Shelter Information Benchmark) を提案した. SIBM では, 避難場所情報を現実に近い情報源を生成することを目的とし, 指定された避難場所数によるデータを生成可能であり, 生成した情報が実際の避難場所情報や被災者間の関係などを再現できる特徴をもつ. また生成された情報に対するクエリセットが用意され,RDF データ構造に対するクエリや, 実際の情報を問い合わせするケースに対するクエリが生成されている.

一方で RDF/OWL データベンチマークに関して Guo ら [10] により LUBM(Lehigh University Benchmark) が開発された.LUBM はテストデータとして, 大学や学部の情報やその関係者 (教員や学生など) に関する情報を生成する. 児玉らによる先行研究において提案された情報共有管理システムの評価は LUBM ベンチマークツールが用いられた.

表 1 は SIBM と LUBM の対象とするデータモデルの違いである.LUBM により生成されるデータは学生間の関係性を表す情報が少ないため, 被災地における情報共有システムの評価において, 被災者間の関係などを再現しない. また, 個人間関係を考慮しながら人や場所を検索など避難する際にある問い合わせを再現しない. そのため, 避難場所に関する情報や避難者間の情報を生成する SIBM は, 先行研究における情報共有管理システムを用いた本研究の提案する手法を評価するのに適していると考えられる.

対象情報	SIBM	LUBM
個人情報	氏名、生年月日、年齢、性別、住所、所属、携帯番号、メール	氏名、学歴、メール、携帯番号、授業、専門
対象者間の関係	家族関係、友人関係	指導教員関係
災害による状況	病気・怪我情報、避難先	対象外
対象とする場所	避難場所: 住所、避難場所名、管理番号、災害種別、収容量、位置情報物品、備蓄情報	大学:大学名、学科、学部、授業

表 1　SIBM と LUBM の対象情報.

4. 提案手法

4.1 暗号化

頻度分析が用いられる要因として, データベースに保存されている暗号文が同一であるものが多く存在することが原因である. この攻撃に対する対策として暗号化を確率暗号を利用することが望ましいが, 暗号化されたデータベースを検索の応答時間が大きく増加する恐れがある. 情報へのアクセスを管理する上で, データベースへのアクセスの応答時間が遅くなることは大きな問題となる.

そこで, データベースに保存される情報に対し, 従来手法を用いた確定的暗号化の後,XOR 演算 (排他的論理和) により乱数を加算する手法を用いる. 乱数が加算されることで同じ内容の情報が送られた場合でも, 生成される暗号文は同一にならないため, データベースへの頻度分析攻撃への耐性を高めることができると考えられる. 保存される RDF に対しての乱数の加算は,RDF 項にそれぞれ乱数を加算する.

$$rdf(s,p,o) \oplus rand == rdf(s \oplus rand, p \oplus rand, o \oplus rand)$$
$$rdf(s \oplus rand, s \oplus rand, o \oplus rand) \oplus rand == rdf(s,p,o)$$

XOR 演算の利点として, 一度加えた値を再度加えることで元の値になることから,XOR による乱数の除去の処理が容易であることからデータベースへのアクセス時に, 一度 XOR 演算による復号化を行うことで, 従来のアクセス制御を適用できると考えられる.

4.2 情報の蓄積

RDF による暗号文と暗号化コンテナの表現として, 児玉らが提案するシステムで定義したプロパティに加え, 新たに key プロパティを定義する. これらのプロパティは, プロキシによって意味解釈され, アクセス制御に利用する. ドメインとレンジは W3C 勧告である RDF Schema1.1 [11] における rdf:domain および rdf:range を指し, 関数的とは同様に W3C 勧告である OWL [12] における FunctionalProperty である.

:key ドメインをステートメントリソースとし, レンジを暗号化タームのリソースとする関数的なオブジェクトのプロパティである. ステートメント毎のトリプルに付与される値のタームを指す.

プロキシにおいて, 図 4 のように, 一つのトリプルを入力とし, 複数のステートメントを生成し, データサーバに送信する. 生成されるステートメントには, 各タームの暗号文とそれに対応するアクセスのレベルや, 個人間関係によるアクセス制御に用いるクローズドなアクセス制御ポリシーも組まれる. これに加え, 本研究では, これらのステートメントに加え, 各トリプル毎にデータを一意に識別できる *key* を対応させるためのステートメントを生成する. この一意の値である *key* をデータベースに蓄積するために行うステートメントの暗号化の際に, プロキシは *key* の生成を秘密鍵生成局に要求する. この要求に対し, 秘密鍵生成局は *key* およびハッシュ関数 *h* により, ハッシュ値 *h*(*key*) が生成され, これらをプロキシへ返す. 蓄積される情報はユーザの公開鍵で暗号化されているため, プロキシ内でアクセスレベルに対応した公開鍵による暗号文に再暗号化された後, 生成局により生成された *h*(*key*) 値との排他的論理和 (XOR) 演算が行われる. 生成されるデータは以下の規則 1 により生成される.

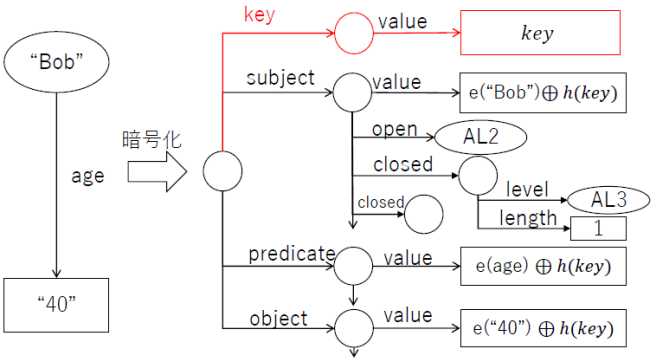


図 4　トリプルの暗号化.

規則 1 蓄積する情報の生成

Require: 再暗号化関数 r, アクセス制御ポリシーによる出力 $(L = (l_s, l_p, l_o), C = (c_s, c_p, c_o))$, 追加元ユーザ ID id_u, ハッシュ関数 h

$(?s, ?p, ?o) \rightarrow$ [

:subject, [:value, $r_{id_u \rightarrow l_s}(?s) \oplus h(key)$; :open, l_s;
:closed$_i$, [:level, $c_{s_{i_{\text{level}}}}$; :length, $c_{s_{i_{\text{length}}}}$]]

:predicate, [:value, $r_{id_u \rightarrow l_p}(?p) \oplus h(key)$; :open, l_s;
:closed$_i$, [:level, $c_{p_{i_{\text{level}}}}$; :length, $c_{p_{i_{\text{length}}}}$]]

:object, [:value, $r_{id_u \rightarrow l_o}(?o) \oplus h(key)$; :open, l_o;
:closed$_i$, [:level, $c_{o_{i_{\text{level}}}}$; :length, $c_{o_{i_{\text{length}}}}$]]

:key, [:value, key]

]

4.3 検　　索

問い合わせユーザの属するクラスやそれに対応づけられるアクセスレベルといったオープンな情報による検索は SPARQL クエリの変換によって行われる. しかしながら, 暗号化されたデータは key による乱数が加算されているため, この乱数を考慮した問い合わせ方式に変更しなければならない.　本研究では提案手法により暗号化されたデータベースへの問い合わせ方式として 2 つの手法を提案する.

（1） D-xor: アクセスしたデータの暗号化されたトリプル要素に $h(key)$ を再度トリプルに XOR 演算を行うことで確定的暗号化された RDF トリプルに変換し, この RDF トリプルに対し従来手法により変換されたクエリによる用いた問い合わせを行う手法 (アルゴリズム 2)

アルゴリズム 2 問い合わせの生成　D_xor

Require: ユーザ ID id_u, 再暗号化関数 r, ハッシュ関数 h, 変数集合 V_{in}, トリプルパターン集合 T_{in}, 変数集合 W_{n}

Ensure: 変数集合 V_{out}, トリプルパターン集合 T_{out}

L:問い合わせユーザが属するクラスに対応したアクセスレベル以下のレベル集合

for all $l \in L,\ t_i = (s_i, p_i, o_i) \in T_{\text{in}}$ **do**
　変数ノード v_i を作成する
　T_out に $(v_i, :key, [:value, rand_i])$ を追加する
　for all $(p, n) \in \{(:\text{subject}, s_i), (:\text{predicate}, p_i), (:\text{object}, o_i)\}$ **do**
　　T_{out}に新しい変数ノード v_{node}を定義し $(v_i, p, [:\text{value}, v_{\text{node}}])$ を追加する
　　if n が変数ノードである **then**
　　　if 変数ノード n が T_{out}に存在しないとき **then**
　　　　$BIND(n\ AS\ v_{\text{node}} \oplus h(rand_i))$ を追加する
　　　else
　　　　$FILTER(n = v_{\text{node}} \oplus h(rand_i))$ を追加する
　　　end if
　　else
　　　T_{out} に $(v_i, p, [:\text{value}, W_{\text{i}} \oplus h(rand_i); :\text{open}, l])$ を追加する
　　　$FILTER(v_{\text{node}} = r_{id_u \rightarrow l}(n))$ を追加する
　　end if
　end for
end for
for all 変数ノード $v \in V_{\text{in}}$ **do**
　V_{out} に $r_{l \rightarrow id_u}(v)$ を v として追加する
end for

（2） Q-xor: 従来の手法で変換したクエリに対し, データ毎に加えられている $h(key)$ を XOR 演算によって加算することで, 乱数が加算されたクエリによるデータベースへの問い合わせを行う手法 (アルゴリズム 3)

アルゴリズム 3 問い合わせの生成 2 Q_xor

Require: ユーザ ID id_u, 再暗号化関数 r, ハッシュ関数 h, 変数集合 V_{in}, トリプルパターン集合 T_{in},

Ensure: 変数集合 V_{out}, トリプルパターン集合 T_{out}

L:問い合わせユーザが属するクラスに対応したアクセスレベル以下のレベル集合

for all $l \in L,\ t_i = (s_i, p_i, o_i) \in T_{\text{in}}$ **do**
　変数ノード v_i を作成する
　T_{out}に $(v_i, :key, [:value, rand_i])$ を追加する
　for all $(p, n) \in \{(:\text{subject}, s_i), (:\text{predicate}, p_i), (:\text{object}, o_i)\}$ **do**
　　if n が変数ノードである **then**
　　　T_{out}に新しい変数ノード v_{node}を作成し, $(v_i, p, [:\text{value}, v_{\text{node}}])$ を追加する
　　　if n が T_{out}に存在しないとき **then**
　　　　$BIND(n\ AS\ v_{\text{node}} \oplus h(rand_i))$ を追加する
　　　else
　　　　$FILTER(n = v_{\text{node}} \oplus h(rand_i))$ を追加する
　　　end if
　　else
　　　T_{out} に $(v_i, p, [:\text{value}, r_{id_u \rightarrow l}(n) \oplus h(rand_i); :\text{open}, l])$ を追加する
　　end if
　end for
end for
for all 変数ノード $v \in V_{\text{in}}$ **do**
　V_{out} に $r_{l \rightarrow id_u}(v)$ を v として追加する
end for

2 つのアルゴリズムでは SPARQL 内の WHERE クローズからなる SPARQL 選択クエリを変換する手続きをアルゴリズム,2,3 のどちらも v_{node} というアクセスしたデータを示す変数を用意し, この変数に対し XOR 演算の処理を追加している.

5. 実　　験

提案手法の実装には RDF データを処理する Java アプリケーションを構築するためのライブラリ,Apache jena を利用した.

本実験の目的として提案手法が実現可能な時間で実行できるかを評価を行う. 比較対象として児玉らによる確定的暗号化の手法を用い, システムの各操作の実行時間を計測することで, 提案手法との実行時間差を計測する.

5.1 実 験 環 境

実験には, 以下の環境を用いた.

CPU	Intel Xeon Processor E5620
RAM	PC3-10600 DDR3 SDRAM 24 GiB
OS	Ubuntu 11.10 64-bit
Apache Jena	2.13.0

5.2 データセット

SIBM は指定した被災地の中心から半径 $d$km 内に存在する避難所情報と避難者に関する情報を生成する. 本実験には, 避難場所情報ベンチマークとして SIBM を利用し, 生成場所の指定は被災地として福島県郡山市とした.

データ・ユーザの追加・削除の実験 5.3 では d= {2.0,3.0,4.0,4.5,4.7,5.1}km と生成範囲を変更し,SIBM により生成された 6 つのデータセットを用いた.

オープンな情報を用いた検索の実験 5.4 では d=2.0km を指定し,SIBM により生成されたデータセットを用いた.

インデックス作成の実験 5.5.1 では d= {1.3,1.5,2.0,2.1,2.9}km と生成範囲を変更し,SIBM により生成された 5 つのデータセットを用いた.

個人間関係を用いた検索の実験 5.5.2 では 5.4 と同様の d=2.0km のデータセットを用いた.

5.3 データ・ユーザの追加・削除

6 つのデータセットにおいて, 図 5 では 1 つのデータの追加・削除,1 人のユーザの追加・削除の実行に要する実行時間を測った.

ユーザの追加, 及び削除は暗号化されていないユーザクラスデータベースへの追加と削除である. ユーザクラスデータベースにはユーザの一意な ID とユーザの属するクラス情報のみが蓄積されているが, これらの RDF データは暗号化されていないため, 問い合わせクエリの暗号化を行う必要がない. そのため本手法と比較対象では, 同数のデータが蓄積されたユーザクラスデータベースに対する問い合わせでありユーザの追加・削除では, 実行時間の差がほとんど見られない. またユーザの追加がユーザの削除では, ユーザの追加では自身の ID が重複しないようユーザクラスデータベースにて自身の ID を探索することからユーザの追加と削除にかかる時間が近しい値となった.

データの追加での実行時間は, 本手法では比較対象に対し, 追加対象の RDF トリプルに対応した *key* の生成, 及び RDF トリプル項への *key* ハッシュ値の加算を行うことから, 計算時間がかかると考えられる. また本手法でははは一つの情報を暗号化する場合, 暗号化されたトリプル項に *key* を対応づける RDF 情報を追加するため, 比較対象に比べ RDF トリプル数が 2 つ多くなる. そのため, データベースに蓄積する暗号化前のデータセット内のデータ数が増加するにつれ, 比較対象に比べ本手法はより実行時間が必要となる. 本実験では確定的暗号化の比較対象に対し, 最大で 2.1 倍の実行時間差となった.

データの削除は削除対象のデータをデータベースから検索し, データが存在した場合, そのデータを削除する. 実験結果として Q_xor が D_xor に対し, 最大で 1.7 倍の性能となった. 検索に用いる問い合わせクエリの構成は XOR 演算によって乱数をクエリ, データのどちらに加算をするかの違いである. 従って Q_xor と D_xor の性能差はデータの削除による実行時間差に表れていると考えられる. 本実験では確定的暗号化の比較対象に対し,D_xor では 2.0 倍,Q_xor では 1.2 倍の実行時間差となった.

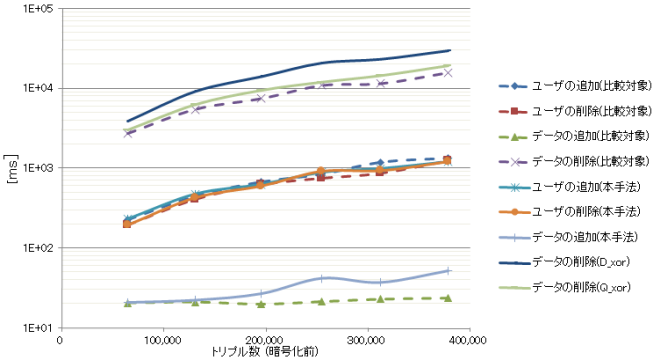


図 5　データ・ユーザの追加・削除

5.4 オープンな情報を用いた検索

以下のクエリを Medical Person クラスに属するユーザからの問い合わせクエリに対し, 問い合わせにかかる時間を測る.Medical Person はアクセス制御に基づき最も多くデータにアクセス可能である. またクエリの応答時間は探索範囲に対して単調に増加するため, このユーザからの問い合わせが最悪の応答時間となる.

#1　避難場所 P20_100319 に避難している避難者を検索
#2　ボランティアクラスに属するユーザを検索
#3　あるユーザの子供がどこの避難場所に避難しているかを検索
#4　避難者であり, 且つ怪我 injured2 の状態の人を検索
#5　クエリ 4 のトリプルパターンの順序を変えたクエリ.

図 6,7,8 は, 福島県の中心地から周囲 2km に存在する避難場所に避難しているユーザおよびデータを蓄積した状態において, ユーザの問い合わせに対して全ての結果を返すまでの時間を計測した結果を示す. クエリ#1,#2,#3,#4,#5 の問い合わせクエリが返す結果数は 292 個,427 個,4 個,212 個,212 個である. #1,#2 は一つのトリプルパターンからなるクエリであり,#3,#4,#5 は複数のトリプルパターンからなる.　プロキシサーバはユーザからの暗号化されたデータベースに問い合わせ可能なクエリに変換しデータベースに対し, その問い合わせクエリによる結果を取得した後, ユーザが復号可能な暗号文に再暗号化を行い, 結果をユーザへ送る. この実行時間をサーバ側の応答時間とする. ユーザ側の処理として, 受信した結果を全て復号化するのにかかる実行時間をユーザ側の応答時間とする.

実験結果からはユーザ側の処理時間の差はほとんどみられない. これはどの手法においても, プロキシサーバから送られる結果はユーザの秘密鍵により復号可能な状態であり, 送られるデータ数が同数であれば, どの手法においても結果は同じである. よってユーザ側の処理時間は結果数に比例すると考えられる.

一般に, グラフパターンを構成するトリプルパターンの個数が多いほど, 中間生成される結果が増加するため,SPARQL による問い合わせの計算に時間がかかる. このため,#4 と#5 ではたとえクエリの内容は同じであり出力される結果は同じであって

も, 初めのトリプルパターンの解決により中間生成される結果数はそれぞれ 3275 個,212 個であり, トリプルパターンの解決順序によりサーバ側での応答時間が比較対象では約 12 倍,D_xor では約 14.7 倍,Q_xor では約 15.0 倍の差となった.

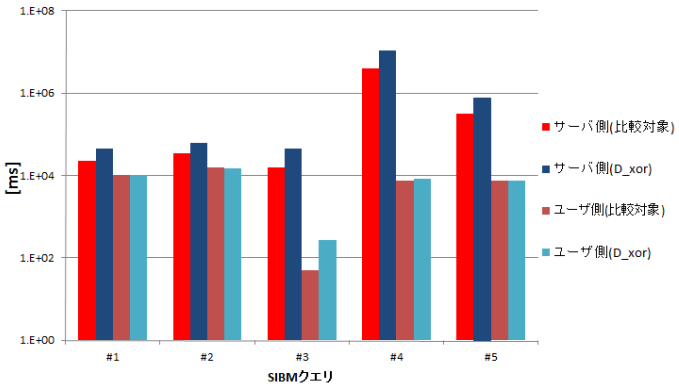


図 6　比較対象と D_xor におけるクエリの応答時間

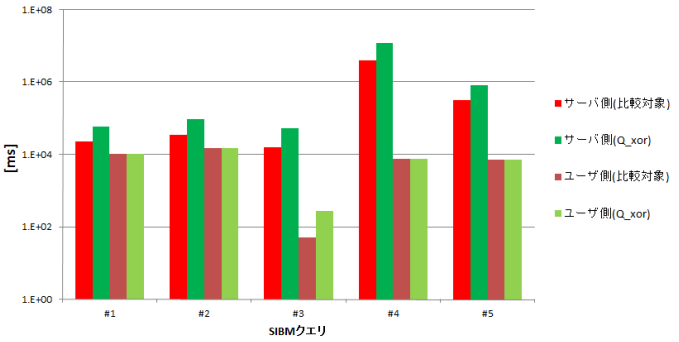


図 7　比較対象と Q_xor におけるクエリの応答時間

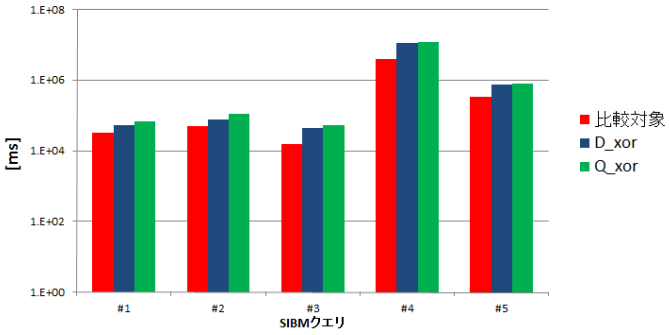


図 8　D_xor と Q_xor におけるクエリの応答時間

5.5　クローズドな情報を用いたアクセス制御

5.5.1　インデックスの作成

図 9 では 4 つのデータセットにおける, インデックスの作成にかかる実行時間を表したグラフである.

インデックスは, 暗号化されたデータベースから個人を表すノードとプレディケートを辺としたグラフ上で幅優先検索を行うことにより作成される. インデックスの作成には SIBM データセットによって作成された親族関係を対象とした個人間関係の情報を利用し,closed ポリシーの付いたデータへのアクセス制御を比較対象と本手法の評価を行った.

インデックスの作成ではユーザ間の個人間関係を表す情報の問い合わせを行うが, この問い合わせクエリではデータの一致検索を用いず, 暗号化されたデータの対応付けられた制御ポリシーを条件とする一致検索を行う. これはユーザを表すノードは必ず決まったアクセスレベルで暗号化されているため, 暗号化されていないポリシーによる比較により同じ問い合わせ結果を得ることができるためである. また幅優先検索では, データは確定的暗号化で暗号化された状態で行われるので, 本手法の問い合わせによる結果には XOR 加算による乱数の除去が行われる. このため, 本手法では乱数の除去が行われることから, 比較対象である従来手法に対して, どのデータセットにおいても約 1.06 倍の実行時間を要している.

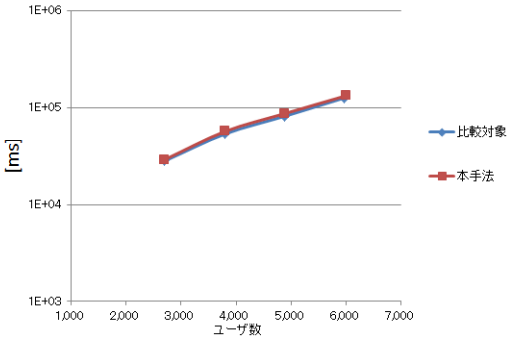


図 9　インデックス作成にかかる実行時間

5.5.2　個人間関係を利用した検索

図 10 では, 個人間関係を用いた検索により 5 人のユーザによる参照可能なデータの作成にかかる実行時間を表した実験結果である.

このアクセス制御では closed ポリシーが対応づけられた情報を問い合わせ, その情報を参照可能かをインデックスを用いて図 3 のように, 個人間関係の距離, またアクセスレベルを条件とした判定を行う. この判定とともに, 参照できる RDF トリプルから乱数を除去する必要があるため実験は D_xor のみとし, 本手法を D_xor とする. 本手法は比較対象である従来手法に対して, どのユーザにおいても約 2.6 倍の実行時間を要した.

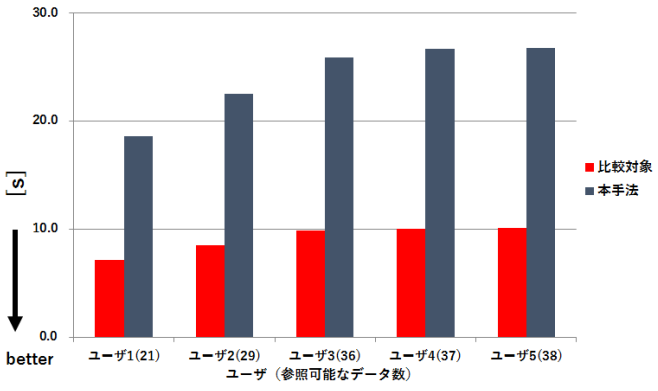


図 10　参照可能なデータの検索応答時間

6. おわりに

6.1 まとめ

本論文では先行研究における暗号手法を XOR 演算を利用した疑似確率的暗号化することで, 秘匿性を高める暗号化手法を提案した. 頻度分析攻撃への耐性を持つ本手法は, 確率的暗号を用いた場合に対して, より少ない実現可能な実行時間での問い合わせが可能と考えられる.

提案手法の評価には, 災害時における情報管理システムの評価により適した SIBM によるデータセットを用いることで, 先行研究において児玉らが提案したシステム, またそれにランダム性を追加した本手法の評価を実験としてデータ・ユーザの追加・削除, 暗号化されたデータベースの検索時間の計測, クローズドな情報におけるアクセス制御についての評価実験を行った. 実験の結果として提案手法は, 従来手法である確率的暗号と比較して, より計算量が多く, 先行研究における手法と比較して問い合わせによるデータ検索では最大で 3.0 倍, データ・ユーザの追加・削除では最大で 2.1 倍, 個人間関係を用いた検索では 2.6 倍の実行時間を要した.

6.2 今後の課題

データベースにおける頻度分析攻撃への耐性を持たせることで情報の秘匿性を高める手法を提案した. しかしながら頻度分析攻撃としてクエリに対する頻度分析が想定される. これはシステム内で利用されるクエリの頻度を分析する手法も考えられるが, 本手法ではクエリの内での結果は乱数が除かれた確定的暗号文になるため, クエリを用いた頻度分析攻撃が考えられるため, 完全な確率的暗号化を行う必要がある. また安全性の評価として, 関連研究で述べたように”耐結託性”, 暗号化の”推移性”についての評価を行わなければならない.

暗号化の安全性を維持しながら, システムの性能を向上される手法として, キーワード検索可能暗号の実装が考えられる. 平文及びデータの検索キーワードをランダム性を持った暗号化を行い, 検索を行う場合は, キーワードの一致をトラップドアを用いることで可能であるとされている. また従来手法と本手法では一致検索しか行えず, キーワード検索による部分一致も行えることが期待される.

謝　辞

本研究の一部は, 日本学術振興会科学研究費補助金基盤研究 (A) の助成によりおこなわれた.

文　献

[1] 児玉快, 横田治夫. データやユーザの効率的な追加・削除が可能な秘匿情報アクセス手法. 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論文集, 2015.

[2] Nguyen Hoai Nam, Yoshitaka Arahori, and Haruo Yokota. SIBM: 避難場所情報に対する RDF データセットベンチマークツール. 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, 2015.

[3] 電気情報通信学会. 知識ベース 知識の森 3 群 7 編 2 章. Retrieved December 31, 2014, from: http://www.ieice-hbkb.org/files/03/03gun_07hen_02.pdf.

[4] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, Vol. 9, No. 1, pp. 1–30, February 2006.

[5] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, Vol. 1403, pp. 127–144. Springer, 1998.

[6] Florian Haag, Steffen Lohmann, and Thomas Ertl. Sparqlfilterflow: SPARQL query composition for everyone. *The Semantic Web: European Semantic Web Conference*, pp. 362–366, 2014.

[7] 伊藤隆, 服部充洋, 松田規, 坂井祐介, 太田和夫. 頻度分析耐性を持つ高速秘匿検索方式 (情報通信基礎サブソサイエティ合同研究会). 電子情報通信学会技術研究報告. WBS, ワイドバンドシステム, Vol. 110, No. 444, pp. 1–6, 2011.

[8] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. Advances in Cryptology—CRYPTO 2001, pp. 213–229. Springer, 2001.

[9] Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-private proxy re-encryption. In *Topics in Cryptology–CT-RSA 2009*, pp. 279–294. Springer, 2009.

[10] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 3, No. 2-3, pp. 158–182, October 2005.

[11] Dan Brickley and R.V. Guha. RDF schema 1.1. Recommendation, W3C, February 2014. Retrieved January 26, 2015, from: http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[12] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 web ontology language structural specification and functional-style syntax (second edition). Recommendation, W3C, December 2012. Retrieved January 26, 2015, from: http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.