論文 / 著書情報
Article / Book Information

論題(和文)	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
Title(English)	
著者(和文)	児玉 快, 横田 治夫
Authors(English)	Kai Kodama, Haruo Yokota
出典(和文)	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, , ,
Citation(English)	, , ,
発行日 / Pub. date	2015, 3

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

児玉　快†　横田　治夫†

† 東京工業大学情報理工学研究科計算工学専攻　〒 152-8552 東京都目黒区大岡山 2-12-1
E-mail: †kodama@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし　近年, 災害時の情報技術活用が注目され, 安否確認や情報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は, プライバシーやセキュリティに関して問題となり得る機微な情報が蓄積される可能性があり, 必要に応じて暗号化による保護が行われるべきである. しかしながら, 従来の暗号化手法では, データの保有者が全ての共有先の鍵情報を把握しなければならない上, 共有先ユーザが削除される度に暗号化し直す必要があり, 非効率的である. そこで本研究では, データやユーザに対してより効率的な追加や削除を行うために, 共有情報に対するアクセス制御としてロジカルなクラスを単位とした暗号化を行う手法を提案し, その評価を行う.

キーワード　プロキシ再暗号化, データ共有, リボケーション.

1. はじめに

近年, 災害時や災害後の復旧および復興を目的とした情報技術活用が注目され, 安否確認や情報共有を支援するためのシステムが利用されている. その代表として, 災害用ブロードバンド伝言サービスや, 地域に向けて提供されるより具体的な避難情報の配信を行うシステムが挙げられる [1]. このようなシステムに蓄積されるデータの中には, プライバシーやセキュリティに関して問題となる機微なものが含まれる可能性があり, そのようなデータには不正利用を防止するためのアクセス制御が行われるべきである. 加えて, システムの全体を信頼することが困難である場合などは, 暗号化によってデータを保護しなければならない. しかしながら, 従来の暗号化手法では, データの保有者が全ての共有先ユーザの鍵情報を把握する必要がある. また, 共有先のユーザのアクセス権限の変更や削除といったリボケーション操作がなされる度に, その影響を受けるデータに対して再度暗号化し直さなければならず, 非効率的である.

そこで本論文では, データやユーザの追加および削除をより効率的に行うために, 共有情報に対するアクセス制御としてロジカルなクラスを単位とした暗号化を行う手法を提案する. プロキシ再暗号化可能な暗号を利用することにより, アクセス制御の柔軟性を維持しながら, 従来の暗号化手法と比較して計算量を抑えられることを示す. また, RDF による情報表現が可能なデータベースとユーザの間で動作するプロキシとして提案手法を実装し, ベンチマークを用いて性能を評価する。

本論文は, 本章を含めて 6 章により構成される. 第 2 章では本論文に関して前提となる知識を述べる. 第 3 章では関連する既存研究を紹介する. 第 4 章では提案する手法について述べる. 第 5 章で評価実験の方法を述べ, その結果を考察する. 最後に, 第 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

情報工学においてアクセス制御とは, 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を制限することである [2]. この規則を形式的なモデルとして実装したものを, アクセス制御ポリシーと呼ぶ. システムは, ユーザによるアクセス要求に対して, 図 1 のようにアクセス制御ポリシーをもとに認可および認証を行う. これにより, 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.

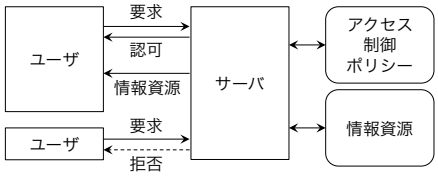

図 1　基本的なアクセス制御モデル.

しかしながら, アクセス制御システムの全体を信頼することは現実的ではない. 例えば, 情報資源はデータベースサーバのような蓄積装置に保管されるが, そのサーバの管理者を完全には信頼できない場合がある. このようなケースでは, サーバ管理者に許可なく情報を取得されることを防ぐために, アクセス制御に加えて暗号化によるデータ保護がなされる. 例として, 秘匿情報の蓄積や共有のために, セキュリティ上完全には信頼できないクラウドストレージサービスを利用する場合が挙げられる. データの保有者は, サービスの管理者による閲覧を防ぐために, 情報をあらかじめローカルシステム上で暗号化したもの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.

より利便性の高い手法のひとつに, Popa らによる研究 [3] がある. この手法では, CryptDB と呼ばれるアプリケーションがクライアントユーザとデータベースの間に配置され, プロキシやミドルレイヤのように働く. アプリケーションは, データを蓄積する際に強度の異なる暗号化を複数回行い, ユーザの問い合わせに対してはそのクエリの計算に必要なレベルまでデータベースを復号してから処理を行う. CryptDB を利用すると,

ユーザがデータベースを利用する際に, データベースの内容の露出を一部までに留めることができる. これにより, データベースの管理者を信頼することなく, 暗号化による保護を利用することが可能である. しかしながら, 代わりにプロキシサーバである CryptDB を運用する管理者を信頼しなければならない. そのため, データベースとプロキシサーバの双方の管理者ともを完全には信頼できない場合には, 他の手法の利用を検討する必要がある.

2.2 プロキシ再暗号化

再暗号化とは, あるユーザのために暗号化された暗号文を, 別のユーザが復号可能な暗号文に変換することである. 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の最終的な目標は, データの保有者と利用者を除く全ての第三者を信頼することなく, 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 つの手続きによって構成される. 鍵生成の手続きでは, セキュリティパラメータを入力として, 主体 A の公開鍵 pk_A および秘密鍵 sk_A の組 (pk_A, sk_A) を出力する. 暗号化の手続きでは, 主体 A の公開鍵 pk_A および平文 m を入力として, A の秘密鍵 sk_A により復号可能な暗号文 c_A を出力する. 復号の手続きでは, 主体 A の秘密鍵 sk_A およびそれによる復号が可能な暗号文 c_A を入力として, 平文 m を出力する. 再暗号化鍵生成の手続きでは, 再暗号化元の主体 A や再暗号化先の主体 B の公開鍵や秘密鍵を入力として, 対応する再暗号化鍵 $rk_{A\to B}$ を出力する. 再暗号化の手続きでは, 主体 A による復号が可能な暗号文を主体 B による復号が可能な暗号文に変換する再暗号化鍵 $rk_{A\to B}$ および暗号文 r_A を入力として, 暗号文 r_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 らにより考案された BBS 方式がある [5]. BBS によるアルゴリズムは次のように表現される.

鍵生成 $sk \in \mathbb{Z}^*_{p-1}, \quad pk = g^{sk}$
ただし g は安全素数 p を法とする原始根であるとする.
暗号化 $m \in \mathbb{G}, \ k \in \mathbb{Z}^*_{p-1}$
$c_1 = mg^k \mod p$
$c_2 = (g^{sk})^k \mod p$
復号 $m = c_1((c_2^{(sk^{-1})})^{-1}) \mod p$
再暗号化鍵生成 $rk_{A\to B} = sk_A^{-1} sk_B$
再暗号化 $c_2 \mapsto c_2^{rk}$

プロキシ再暗号化技術では, 再暗号化を行う際に全ての第三者の持つ機能を利用する必要のないことが望ましい. しかしながら, 第三者を一切信頼することなくプロキシ再暗号化やそれに利用するデータベース管理を行う手法は現在提案されていない. 代わりに, 多くの手法では信頼可能な第三者として秘密鍵生成局をモデルに含めたり, プロキシサーバそのものを信頼可能な第三者として利用することによって再暗号化を実現している.

2.3 RDF

RDF (Resource Description Framework) は, 情報を表現する方法のひとつである. 情報の単位としてトリプル (3 つ組) が用いられ, トリプルは 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) から構成される. 例えば, “リソース :Alice はリソース :Bob を知っている” という情報は, 図 2 に示すトリプルとして表現される.

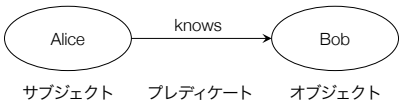

図 2 RDF トリプルによる情報の表現.

また, トリプルの集合は RDF グラフと呼ばれ, RDF グラフとして記述された情報への問い合わせ言語として SPARQL が標準的に用いられている [6]. SPARQL による問い合わせはグラフパターンのマッチングにより行われ, グラフパターンを構成する基本的な要素であるベーシックグラフパターンはトリプルパターンの集合として表現される. 例えば, “リソース :Alice が知っているリソース” を選択するための問い合わせは, SELECT ?X WHERE { :Alice :knows ?X } のような SPARQL クエリとして記述される. これに対して, 結果は変数と値のペアであるバインディングの集合であるソリューションの集合として表現される. 例えば, 先のクエリは図 2 に示すトリプルからなる RDF グラフ上で {{ ?X = :Bob }} という結果を返す.

RDF や SPARQL といった技術はセマンティックウェブにおける情報記述手法として事実上の標準である. 近年では特に相互にリンクするオープンデータの注目が高まり, そのための機械可読なフォーマットとして RDF による表現を利用することが増えている. 我が国においても, 政府や地方自治体による透明性や信頼性の向上を目的とした情報公開がオープンデータ形式で行われている [7].

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. また, 直感的かつ記述に容易な問い合わせ言語として SPARQL を利用することができる [8]. そのため, 本論文では RDF による情報表現と SPARQL による情報検索が利用可能なデータベースを想定し, 手法を提案する.

3. 関 連 研 究

3.1 RDF の部分的暗号化

RDF 項の暗号文は常に RDF リテラルとして扱われる. しかし, 現在の W3C 勧告 [9] においては, RDF リテラルはオブジェクトの位置にのみ記述可能とされている. そのため, サブジェクトやプレディケートの位置にある RDF 項を暗号化した値を, どのようなノードに格納するかが問題となる.

Giereth は, 暗号化前のグラフ構造を変化させることでこの問題を解決した. 例えば, 1 個のサブジェクトやプレディケートを暗号化する際には, グラフ構造を変化させるために空白ノードやトリプルを生成する. この手法に加えて, 復号の際にグラフ構造を戻すアルゴリズムや, 暗号化された RDF にハッシュダ

イジェストや公開鍵の情報などを組み込む方法を提案した [10]. 具体的には, RDF 項やトリプルのような平文をユーザに対して共有する際に, アルゴリズム 1 に示す手続きにより暗号化を行う.

アルゴリズム 1 RDF の部分的暗号化手法における情報蓄積

入力 平文集合 $M = \{m_1, m_2, \ldots, m_m\}$, 公開鍵集合 $P = \{pk_1, pk_2, \ldots, pk_n\}$, 共通鍵 k による対称関数 f_k, 公開鍵 pk による非対称関数 g_{pk}, ダイジェスト関数 h

出力 暗号化コンテナを含む RDF グラフ G

for all $m_i \in M$ **do**
  f のセッション鍵 k を生成する.
  $c_i \leftarrow f_k(m_i)$
  $P_i \leftarrow m_i$ の受信者が持つ公開鍵集合 ($P_i \subseteqq P$)
  for all $pk_j \in P_i$ **do**
    $c_{i,kj} \leftarrow g_{pk_j}(k)$
  end for
  $h_i \leftarrow h(m_i)$
  $c_i, c_{i,k1}, c_{i,k2}, \ldots, c_{i,kn}, h_i$ を暗号化コンテナ ec に格納する.
  ec をリテラルとして直列化し, G に挿入する.
end for

しかしながら, この手法では, 複数のユーザに対する共有を目的とする RDF グラフを暗号化する際に, 保有者は共有対象となる全てのユーザの鍵情報を保持しなければならない. また, 共有対象のユーザの追加や削除が行われる度に暗号化し直さなければならないという点でも非効率的である. 実際に, ユーザの追加や削除はアルゴリズム 2 やアルゴリズム 3 の示すように行われる.

アルゴリズム 2 RDF の部分的暗号化手法におけるユーザ追加

入力 平文集合 $M = \{m_1, m_2, \ldots, m_m\}$, 公開鍵集合 $P = \{pk_1, pk_2, \ldots, pk_n\}$, 共通鍵 k による対称関数 f_k, 公開鍵 pk による非対称関数 g_{pk}, ダイジェスト関数 h, 追加対象のユーザの公開鍵 $pk_u \in P$, 追加対象のユーザを受信者に含む平文集合 $M_u \subseteqq M$

出力 暗号化コンテナを含む RDF グラフ G

for all $m_i \in M_u$ **do**
  $k \leftarrow m_i$ に対応する f のセッション鍵
  $c_i \leftarrow f_k(m_i)$
  $c_{i,ku} \leftarrow g_{pk_u}(k)$
  $h_i \leftarrow h(m_i)$
  $ec \leftarrow h_i$ を含む暗号化コンテナ
  ec をリテラルとして直列化したものを, G から削除する.
  ec に $c_{i,ku}$ を格納する.
  ec をリテラルとして直列化し, G に挿入する.
end for

3.2 XML の構造化情報の階層関係に着目した暗号化

XML は構造化された情報を記述することができるが, その階層関係に着目した暗号化手法が提案されている [11]. 例えば, あるノードにアクセス可能なユーザは, その親ノードにもアクセス可能でなければならない. また, あるノードへのアクセスが許可されないユーザは, その子ノードへのアクセスも許可されることはない. このように, 階層関係を推論してアクセス制御を実装するという点で, 本論文の提案する手法に関連性がある.

アルゴリズム 3 RDF の部分的暗号化手法におけるユーザ削除

入力 平文集合 $M = \{m_1, m_2, \ldots, m_m\}$, 公開鍵集合 $P = \{pk_1, pk_2, \ldots, pk_n\}$, 共通鍵 k による対称関数 f_k, 公開鍵 pk による非対称関数 g_{pk}, ダイジェスト関数 h, 削除対象のユーザの公開鍵 $pk_u \in P$, 削除対象のユーザを受信者に含む平文集合 $M_u \subseteqq M$

出力 暗号化コンテナを含む RDF グラフ G

for all $m_i \in M_u$ **do**
  f のセッション鍵 k を生成する.
  $c_i \leftarrow f_k(m_i)$
  $P_i \leftarrow m_i$ の受信者が持つ公開鍵集合 ($P_i \subseteqq P$)
  for all $pk_j \in P_i - \{pk_u\}$ **do**
    $c_{i,kj} \leftarrow g_{pk_j}(k)$
  end for
  $h_i \leftarrow h(m_i)$
  h_i を含む暗号化コンテナを G から削除する.
  $c_i, c_{i,k1}, c_{i,k2}, \ldots, c_{i,kn}, h_i$ を暗号化コンテナ ec に格納する.
  ec をリテラルとして直列化し, G に挿入する.
end for

4. 提案手法

共有情報に対するアクセス制御としてロジカルなクラスを単位とした暗号化を行う手法を提案する. まず, 暗号化情報を蓄積するデータベースとユーザとの間にミドルレイヤアプリケーションを配置する. アプリケーションは, ユーザの認証を行う際にそのクラスを調べ, 対応するアクセスのレベルに紐付けられた再暗号化鍵を秘密鍵生成局から取得する. その後, ユーザの要求に基づいて, データベースに対する検索が可能な問い合わせを生成し, 得られた結果を再暗号化してユーザに送信する. 以下では, 手法に関するモデルやオントロジについて説明する.

4.1 モデル

本研究において想定するモデルを図 3 に示す. 本手法では, ユーザは一意に識別可能な ID を持つ. ユーザは, プロキシサーバに対して, 既に蓄積されている情報への問い合わせや, ユーザの保有する新たな情報の蓄積を要求する.

プロキシサーバは, ユーザの持つ ID をユーザのクラスに対応付ける写像を持つ. ユーザの ID は, 高々 1 個のクラスに対応付けられる. また, プロキシサーバは, ユーザのクラスをアクセスのレベルに対応付ける写像も持つ. ユーザのクラスは, 0 個以上のアクセスのレベルに対応付けられる.

また, 本研究で提案する手法は, 全ての情報が RDF トリプルにより記述されていることを前提とする. 情報はアクセス制御ポリシーに基づいて暗号化された状態で蓄積される. 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. ここで, L は RDF 項に対応付けられるアクセスのレベルを示し, C は 4.6 小節で後述する個人間関係を利用するアクセス制御ポリシーを示す.

アクセスのレベルは階層関係を持ち, あるレベル l やそれよ

り高いレベルに対応付けられるクラスのユーザは, l を対応付けられた RDF 項へのアクセスが許可される.

各アクセスレベルに対応する公開鍵秘密鍵ペアは, 信頼可能な秘密鍵生成局により生成されるが, それらはプロキシに送信されない. 代わりに, プロキシは生成局に対して再暗号化鍵の生成を要求し, それをもとにデータベースサーバから受信した問い合わせ結果やデータベースサーバ内の RDF グラフに対して再暗号化を行い, ユーザのために生成された復号鍵により復号可能な状態でユーザに送信する. ユーザは, 生成局に対して復号鍵の送信を要求し, それをもとにプロキシから受信した情報を復号し, 結果を得る.

また, ユーザからプロキシに送信される問い合わせ内容やデータベース操作について, 変数ノードを除いて, 含まれる全ての RDF 項はユーザの公開鍵で暗号化される. そのため, プロキシからデータベースに送信される問い合わせ内容やデータベース操作は, 暗号化された RDF 項の再暗号化により生成される.

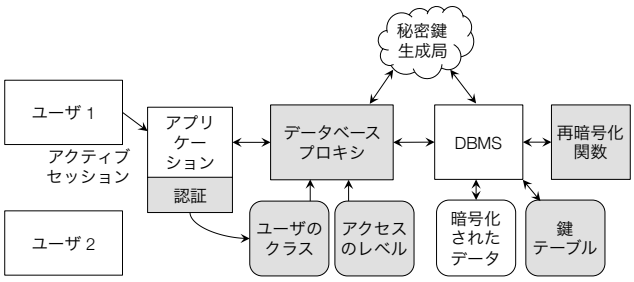

図 3　本論文の提案するアクセス制御モデル.

4.2　オントロジ

本手法では, 新たに 12 個の RDF プロパティを定義して利用する. 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される. 以下では, ドメインおよびレンジは W3C 勧告である RDF Schema 1.1 [12] における Domain および Range を指す. また, 関数的, 逆関数的, データプロパティ, オブジェクトプロパティとはそれぞれ, 同様に W3C 勧告である OWL 2 [13] における FunctionalProperty, InverseFunctionalProperty, DataProperty, ObjectProperty を指す.

:id　ドメインをユーザリソースとし, レンジをリテラルとする関数的かつ逆関数的なデータプロパティである. ユーザの持つ一意に識別可能な ID を示す.

:higherOf　ドメインおよびレンジをアクセスレベルリソースとするオブジェクトプロパティである. サブジェクトのレベルが, オブジェクトのレベルよりも高いことを示す.

:userClass　ドメインをユーザリソースとし, レンジをユーザクラスリソースとする関数的なオブジェクトプロパティである. アクセス制御ポリシー上のユーザが属するクラスを示す.

:mappingTo　ドメインをユーザクラスリソースとし, レンジをアクセスレベルリソースとする関数的なオブジェクトプロパティである. ユーザクラスからアクセスのレベルへの写像として記述される.

:higherOf　ドメインおよびレンジをアクセスレベルリソースとするオブジェクトプロパティである. そのサブジェクトのレベルが, そのオブジェクトのレベルよりも高いことを示す.

:value　ドメインを暗号化タームのリソースとし, レンジをリテラルとする関数的なデータプロパティである. タームの暗号化後の値を表現する.

:open　ドメインを暗号化タームのリソースとし, レンジをアクセスレベルリソースとする関数的なオブジェクトプロパティである. アクセス制御ポリシーに基づいてどのアクセスのレベルに対応付けられ暗号化されたかを示す.

:closed　ドメインを暗号化タームのリソースとし, レンジを個人間関係によるポリシーのリソースとするオブジェクトプロパティであり, 暗号化タームを任意個数のポリシーに紐付ける. 個人間関係によるポリシーは個人とユーザに関する関係として RDF トリプルのパスにより表現される. プロキシは, 暗号化されたアクセスのレベルに対応するユーザに加えて, 個人間関係によるポリシーにより許可されたユーザからのアクセスも可能にする.

:level　ドメインを個人間関係によるポリシーのリソースとし, レンジをアクセスレベルリソースとする関数的なオブジェクトプロパティである. ポリシーの指すレベルを示す.

:length　ドメインを個人間関係によるポリシーのリソースとし, レンジを非負整数リテラルとする関数的なオブジェクトプロパティである. ポリシーが許可するパスの最大の長さを示す.

:subject, :predicate, :object　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プロパティである. それぞれ, ステートメントのサブジェクト, プレディケート, オブジェクトの暗号化タームを示す.

これらのプロパティは, プロキシに保存されるユーザ情報や, 暗号化された値を含む RDF グラフの中で利用される. トリプルの暗号化は, タームの暗号化に加えて, 図 4 のように複数のステートメントを生成することにより行われる.

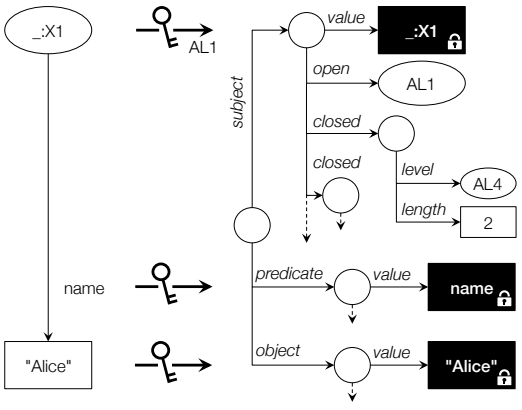

図 4　トリプルの暗号化.

4.3 データの追加と削除

アクセス制御ポリシーは, トリプルを入力として, 各タームに対応するアクセスのレベルと個人間関係によるポリシーを出力する. データの追加の際には, プロキシはポリシーの出力を入力とする規則 4 に基づいて複数のステートメントを生成し, データベースサーバに送信する. 生成されるステートメントには, 各タームの暗号文に加えて, 対応するアクセスのレベルや, 個人間関係によるポリシーが含まれる.

データの削除の際には, 対象を追加したポリシーが既知であるかどうかにより処理が異なる. ポリシーが既知である場合には, 暗号化により生成されるステートメントを計算し, 一致するものをデータベースサーバから全て削除する. ポリシーが既知でない場合には, すべてのアクセスのレベルによる暗号化を行い, それぞれの暗号化結果に一致するものをデータベースサーバから全て削除する.

規則 4 蓄積する情報の生成

入力 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

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

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

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

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

]

4.4 ユーザの追加と削除

ユーザの追加の際には, 対象のユーザの ID u_{id} に対応するクラス c_u について, $(?u, :\mathrm{id}, u_{\mathrm{id}}) \rightarrow (?u, :\mathrm{userClass}, c_u)$ として記述される規則によりステートメントを生成する. ステートメントはプロキシサーバ内または外部のユーザ情報グラフに蓄積され, ユーザの認証のために参照される. また, ユーザの削除の際には, ユーザ情報グラフから対象の ID を検索し, ID とクラスへの紐付けを記述するステートメントを削除する.

なお, ユーザ情報グラフの語彙上の制限として, :id は関数的かつ逆関数的なプロパティであり, かつ :userClass は関数的プロパティである必要がある. そのため, ユーザを追加する前に, 対象ユーザが記述されているかどうかを調べ, 既に記述されている場合には対象ユーザを一度削除する.

4.5 検　　索

問い合わせユーザの属するクラスやそれに対応付けられるアクセスのレベルといったオープンな情報による検索は, SPARQL クエリの変換により行われる. トリプルパターンの集合のみにより構成された 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 5 に示す.

4.6 個人間関係に基づくアクセス制御

より柔軟なアクセス制御を行うために, 情報が記述されている複数の個人の間の関係にも注目する. 具体的には, 個人 A を表すリソース r_A と個人 B を表すリソース r_B の間にプレディケートの有向パス $P : r_A \rightarrow \cdots \rightarrow r_B$ が存在し, パスの含む各トリプルがアクセスのレベル $l_{\max}$ 以下であり, かつ式 1 を満足するとき, A のユーザとしてのインスタンス u_A は, r_B をサブジェクトとして個人間関係によるポリシーが $C = (c_s, c_p, c_o)$ であるようなトリプルにアクセスすることができる.

$$\begin{aligned} &\min(c_{s_{\mathrm{level}}}, c_{p_{\mathrm{level}}}, c_{o_{\mathrm{level}}}) \leqq l_{\max} \quad \wedge \\ &\max(c_{s_{\mathrm{length}}}, c_{p_{\mathrm{length}}}, c_{o_{\mathrm{length}}}) \geqq length(P) \end{aligned} \tag{1}$$

ここで, r_A と r_B の間の有向パスはユーザ情報グラフに依存しないため, 暗号化された値を含むデータベースが変更されない限り, パスが追加されたり削除されることはない. そのため, インデックスとして事前に計算しておくことで, ユーザによる問い合わせの度にパスを評価する必要がなくなる.

アルゴリズム 5 データベースの検索が可能な問い合わせの生成

入力 ユーザ ID id_u, 再暗号化関数 r, 変数集合 V_{in}, トリプルパターン集合 T_{in}

出力 変数集合 V_{out}, トリプルパターン集合 T_{out}

$L \leftarrow \{l \mid (id_u, \hat{}:\mathrm{id}/:\mathrm{userClass}/:\mathrm{mappingTo}/:\mathrm{higherOf}^*, l)\}$
for all $l \in L$, $t_i = (s_i, p_i, o_i) \in T_{\mathrm{in}}$ **do**
　変数ノード v_i を作成する
　for all $(p, n) \in \{(:\mathrm{subject}, s_i), (:\mathrm{predicate}, p_i), (:\mathrm{object}, o_i)\}$ **do**
　　if n が変数ノードである **then**
　　　T_{out} に $(v_i, p, [:\mathrm{value}, n])$ を追加する
　　else
　　　T_{out} に $(v_i, p, [:\mathrm{value}, r_{id_u \rightarrow l}(n); :\mathrm{open}, l])$ を追加する
　　end if
　end for
end for
for all 変数ノード $v \in V_{\mathrm{in}}$ **do**
　V_{out} に $r_{l \rightarrow id_u}(v)$ を v として追加する
end for

5. 実　　験

提案手法と, 関連する既存手法である Giereth による部分的 RDF 暗号化手法 [10] の性能を比較する. 両手法の実装には, Apache により開発されている Java 向けセマンティックウェブフレームワークである Jena を利用した.

5.1 環境とデータセット

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
Java	1.7.0_65
JRE	OpenJDK IcedTea 2.5.3
Jena	2.12.1

実験データには, セマンティックウェブシステムのために考案されたベンチマークとして Lehigh University Benchmark (LUBM) [14] を利用した. LUBM は, テストデータとして, 大学や学部のような組織や, そこに所属する教員や学生に関する

情報を生成する．特に規模の大きいオントロジに対するテストのためのベンチマークとして，LUBM は事実上の標準になっている [15]．この実験では，生成されたトリプルに加えて，規則 6 に示す推論規則を用いて個人間関係を生成し，ポリシーの評価に利用した．

規則 6 個人間の情報の生成

($?x$, :worksFor/^:worksFor, $?y$)
→ ($?x$, :sameWorksFor, $?y$)
($?x$, :teacherOf/^:teacherOf, $?y$)
→ ($?x$, :sameTeacherOf, $?y$)
($?x$, :takesCourse/^:takesCourse, $?y$)
→ ($?x$, :sameTakesCourse, $?y$)

また，アクセス制御ポリシーは次のように人工的に決定した．

Employee　誰がどの講義を受けているかを参照可能．
Professor　Employee が参照可能な情報と，誰がどの組織の一員であり，どの講義を教えているかを参照可能．
Chair　Professor が参照可能な情報と，誰がどの組織に雇用されているかを参照可能．
上記を含む全てのユーザ　上記以外の情報を参照可能．

ユーザによる問い合わせの内容としては，LUBM の提供する 14 個のクエリのうち，University0 内の組織である Department0 に関するトリプルのみからなるデータセットに対して 1 個以上のソリューションを結果として返すことのできる 5 個 (#1, #3, #5, #6, #14) を選択し，ユーザによる入力とした．各クエリの内容は次のように記述される．

問い合わせ元のユーザは University0 内の組織である Department0 に所属する FullProfessor7 であるとした．このユーザに対応するユーザクラスは Chair であるため，アクセス制御ポリシーに基づいて，全てのデータにアクセスする権限を持つ．クエリの応答時間はアクセス範囲の広さ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により最悪の応答時間を評価することができる．

また，プロキシ再暗号化が可能な暗号化方式として，Blaze らによる BBS アルゴリズムを採用した．ただし，鍵サイズは 2048 ビットとし，平文と暗号化鍵から暗号文が一意に決定されるように発生させる乱数を固定した．

また，Giereth により提案されている部分的 RDF 暗号化アルゴリズムを比較対象として実装し，同様に評価した．実装の際に，共通鍵暗号方式と公開鍵暗号方式としてそれぞれ鍵サイズ 256 ビットの AES と鍵サイズ 2048 ビットの RSA を用いた．

5.2　データおよびユーザの追加と削除

図 5 は，1 学部に相当するユーザ情報と N ($1 \leqq N \leqq 8$) 学部に相当するデータが既に蓄積されている状態で，1 個のデータおよび 1 人のユーザの追加および削除に掛かる時間を計測した結果を示すものである．また，図 6 は，N ($1 \leqq N \leqq 8$) 学部に相当するユーザ情報と 1 学部に相当するデータが既に蓄積されている状態で，1 個のデータおよび 1 人のユーザの追加および削除に掛かる時間を計測した結果を示すものである．ただし，比較対象におけるユーザ削除時間は，より小さなデータセットにおいて計測した結果をもとに算出した．実験では，いずれの追加および削除操作についても，本手法は比較対象よりも短い時間で処理を完了した．

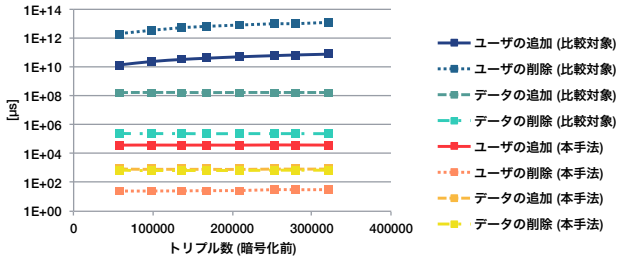

図 5　1 学部に相当するユーザ数における実行時間．

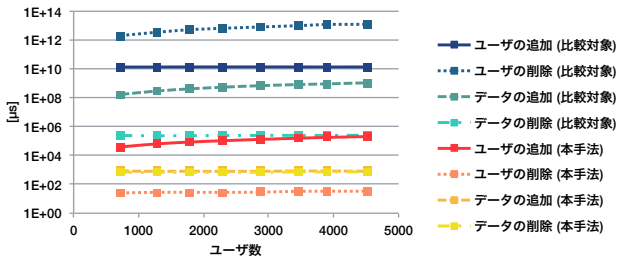

図 6　1 学部に相当するデータ数における実行時間．

蓄積されたデータ数を N，ユーザ数を U として，本手法に対して各操作の平均計算量を算出すると，ユーザの追加操作は U に比例する時間で完了し，ユーザの削除操作は定数時間で完了する．また，データの追加および削除の操作はユーザ側での 1 回の暗号化とサーバ側での 1 回の再暗号化により完了する．一方で，比較対象においては，計算量は各ユーザから参照可能な割合の平均値に大きく依存する．この値を α ($0 \leqq \alpha \leqq 1$) とすると，ユーザの追加および削除はそれぞれ αN，$\alpha^2 NU$ 回の暗号化を必要とする．また，データの追加は αU 回の暗号化により行われ，データの削除は αU 回の暗号文削除と 1 回の暗号化により行われる．

これらを実験結果と比較するために，図 5, 図 6 において，1 学部に相当するユーザやデータが既に蓄積されている状態の処理時間を 1 として，各計測時間をプロットしたものを図 7 および図 8 に示す．先に示した平均計算量に従うように，本手法におけるユーザ追加や，比較対象におけるデータ追加およびユーザ削除では，操作に掛かる時間は既存のユーザ数に比例している．また，比較対象におけるユーザの追加および削除の計算には，既存のデータ数に比例した時間が掛かることが確認できる．比較対象におけるデータ削除時間についてはユーザ数に対する比例関係が見られなかったが，これは暗号文生成のために行われる暗号化が暗号文削除の時間を無視できるほど長い時間を必要としたためである．

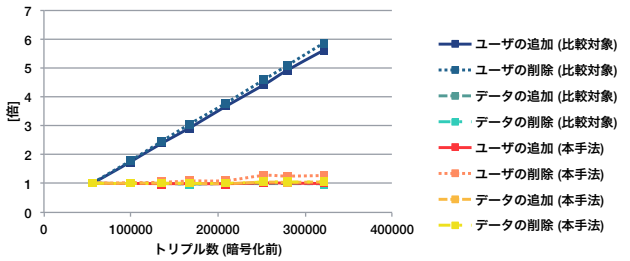

図 7　1 学部に相当するユーザ数における実行時間 (相対).

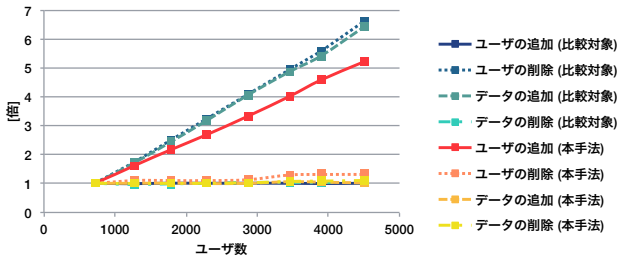

図 8　1 学部に相当するデータ数における実行時間 (相対).

5.3　検　　索

図 9 は, 1 大学 1 学部のみに対応するユーザおよびデータを蓄積した状態において, ユーザによる問い合わせに対して結果として全てのソリューション集合を返すまでの時間を計測した結果を示す. ただし, #1, #3, #5, #6, #14 のクエリが結果として返すソリューションの数はそれぞれ, 4 個, 6 個, 719 個, 678 個, 532 個である.

本論文の提案する手法においては, プロキシサーバはデータベースに対して問い合わせを行って結果を取得し, ユーザはその結果を復号する. 比較対象においては, サーバは RDF グラフからユーザが復号可能な部分のみを抽出し, ユーザはその部分グラフを復号してローカルシステム上で問い合わせを行う.

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, 中間生成されるインメモリデータが増加するため, SPARQL による問い合わせの計算に時間が掛かる [16]. 実際に, 複数のトリプルパターンにより構成される #1, #3 および #5 のクエリにおいて, 問い合わせを行う側では計算のためにより長い時間を必要とした. 実験結果からは比較対象におけるユーザ側の応答時間に変化はほとんど見られないが, これは全てのクエリに対してユーザがアクセス可能な部分全体を復号する必要があったためである. 一方, 本手法ではユーザは SPARQL の返すソリューション集合のみを復号すればよい. このため, ユーザ側の計算時間はソリューションの個数に比例している.

この実験では, 本論文の提案する手法と比較対象との差はクエリ#5 で最小となり, クエリ#14 で最大となった. 比較対象は, 本手法に対してクエリ#5 で 17 倍の時間を必要とし, クエリ#14 では 1195 倍の時間を必要とした.

5.4　個人間関係に基づくアクセス制御

図 10 は, 個人間関係に基づくアクセス制御や, その計算のために用いることの可能なインデックスの作成に要した時間を計

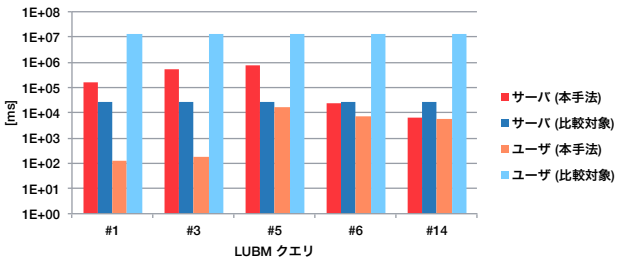

図 9　クエリの応答時間.

測した結果を示す. なお, これらの処理の計算量は, 問い合わせ元のユーザのクラスに依存しない.

インデックスは, 個人とプレディケートをそれぞれノードおよび辺とするグラフ上で幅優先探索を行うことにより作成することができる. 全ての個人に対して探索を行う必要があるため, その計算量は個人の数の 2 乗に比例する.

また, 個人間関係に基づくアクセス制御においては, 事前インデックスの有無により計算時間に差が生じる. 実験結果から, アクセス制御そのものに掛かる時間に対しては非常に小さい差であるものの, 個人の数に比例することが示されている. そのため, 可能であれば事前にインデックスを作成しておくことが望ましいといえる.

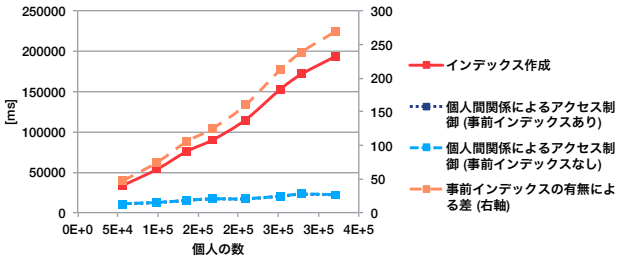

図 10　個人間関係に基づくアクセス制御の実行時間.

6.　おわりに

6.1　ま と め

本論文では, ユーザのクラスやデータにアクセスするためのレベルを利用し, プロキシ再暗号化による秘匿化を行うことで, データやユーザの追加や削除を効率良く行うことのできる秘匿情報アクセス手法を提案した.

また, 提案手法の性能を確かめるために, ミドルレイヤアプリケーションとして手法を実装し, LUBM によるデータセットを用いて, データやユーザの追加や削除に加えて, 検索や個人間関係に基づくアクセス制御に関して評価実験を行った. 実験の結果として, ユーザを単位とする既存の暗号化手法よりも, 効率良くユーザのアクセスを制御できることが示された. 特にユーザを削除する操作において, 提案手法は再度の暗号化を行う必要がないため, 10^{10} 倍以上の性能が得られた. また, ユーザによる検索問い合わせについては, 既存手法と比較して最小で 17 倍, 最大で 1195 倍の性能が得られた.

6.2 今後の課題

まず, 実用に向けて, より精密にセキュリティ評価を行う必要がある. 特に, プロキシ再暗号化に関する研究で一般に想定されている脅威に関して, それらへの耐性を評価しなければならない [17]. 例えば, 本手法では情報の保有者が自身の公開鍵で暗号化したままサーバに送信可能であるため “データ秘匿性” を持つことは明らかであるが, 権限のないユーザとプロキシが結託した場合にどれだけの情報が漏洩する可能性があるかに関する “耐結託性” について議論することは意義が大きい.

また, 本論文では, 単純な暗号化だけでなく, ユーザと個人の間の関係に注目したアクセス制御を提案した. そのため, 提案手法は, CryptDB [3] のようなミドルレイヤアプリケーションによるアプローチと同様であり, 完全なセキュリティを保証するものではない. しかしながら, 暗号化手法の適切な選択によって, 暗号化の安全性をより高めることはできると考えられる. 例えば, 検索可能暗号のようにトラップドアの利用をユーザやプロキシに対して公開すれば, 関連する情報の露出を低減し, 同時に秘密鍵生成局への依存を小さくすることもできるだろう.

実行速度に関して, 提案手法や評価実験では計算の並列化を行っていないが, 幾つかのアルゴリズムでは並列計算の適用も可能である. 例えば, 再暗号化対象の各ステートメントや, インデックス作成対象の各個人に対して, 並列に計算することでより高速な処理を期待できるだろう.

また, 評価実験において利用した LUBM は, 規模の大きいオントロジに対するテストのためのベンチマークとして事実上の標準であるが, ノード間の関連が疎であるため, 現実性を欠いているという批判がある [18]. 本論文でも, 個人間関係を利用するアクセス制御およびデータ検索のために, 幾つかの推論規則を人工的に導入した. より精密な評価を行うためには, 個人やユーザの間の関係を含むことを前提としたデータセットやベンチマークを利用すべきである. 例えば, SIBM [19] のように, 災害時や災害後の復旧および復興をケースとした, 避難所の位置や避難者の状況および避難者間の関係といった情報を含むデータセットを生成するベンチマークは, 本論文の提案する手法の評価に適していると考えられる.

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) (#25240014) の助成により行われた.

文　　献

[1] 総務省関東総合通信局防災対策推進室. 災害時に活用できる情報伝達手段. Retrieved December 31, 2014, from: http://www.soumu.go.jp/soutsu/kanto/saigai/osirase/saigaimanyuaru.pdf.

[2] 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.

[3] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In *23rd Symposium on Operating Systems Principles*, pp. 85–100. ACM, October 2011.

[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] 総務省情報流通行政局情報流通振興課. オープンデータ戦略の推進. Retrieved January 8, 2015, from: http://www.soumu.go.jp/menu_seisaku/ictseisaku/ictriyou/opendata/.

[8] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, Vol. 34, No. 3, pp. 16:1–16:45, September 2009.

[9] Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 concepts and abstract syntax. Recommendation, W3C, February 2014. Retrieved January 8, 2015, from: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[10] Mark Giereth. On partial encryption of RDF-graphs. In *Proceedings of the 4th International Conference on The Semantic Web*, pp. 308–322. Springer-Verlag, 2005.

[11] Naizhen Qi and Michiharu Kudo. XML access control with policy matching tree. In *European Symposium on Research in Computer Security*, Vol. 3679 in Lecture Notes in Computer Science, pp. 3–23. Springer Berlin Heidelberg, 2005.

[12] 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/.

[13] 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/.

[14] 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.

[15] Gang Wu, Juanzi Li, and Kehong Wang. System II: A hypergraph based native RDF repository. In *Proceedings of the 17th International Conference on World Wide Web*, pp. 1035–1036. ACM, 2008.

[16] Jiewen Huang, Daniel J. Abadi, and Kun Ren. Scalable SPARQL querying of large RDF graphs. In *Proceedings of the Very Large Database Endowment*, Vol. 4, pp. 1123–1134, 2011.

[17] Pei-Shan Chung, Chi-Wei Liu, and Min-Shiang Hwang. A study of attribute-based proxy re-encryption scheme in cloud environments. *International Journal of Network Security*, Vol. 16, No. 1, pp. 1–13, January 2014.

[18] Timo Weithöner, Thorsten Liebig, Marko Luther, Sebastian Böhm, Friedrich Von Henke, and Olaf Noppens. Real-world reasoning with owl. In *The Semantic Web: Research and Applications*, pp. 296–310. Springer, 2007.

[19] 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.