改竄への耐性と人のつながりを表すグラフ更新の反映を可能とするアクセス権制御手法

宮沢　駿輔† 　平田　拓三† 　Hieu Hanh Le† 　横田　治夫†

† 東京工業大学　〒 152–8550 東京都目黒区大岡山 2–12–1

E-mail: †{miyazawa,hirata,hanhlh}@de.cs.titech.ac.jp, †yokota@cs.titech.ac.jp

あらまし　家族や友人といった情報提供者と繋がりがある人と暗号文を共有したい場合が存在する。人と人との繋がりを利用したアクセスを実現するために、関係をグラフ構造である RDF で表現した先行研究では改竄したデータの挿入によりアクセス権が改変される課題や、グラフの更新に伴う繋がりの変化がアクセス権に反映されない課題が解決されていない。本研究は人と人との繋がりを利用したアクセス制御の実現における上記の脆弱性の課題を解決する事を目的とする。そのためノード間を繋ぐ関係と当該ノードに対する署名で改竄を検知する手法と、人と人との繋がりの変更に対する到達可能性をアクセス制御に反映する手法を提案し、実験により評価した。

キーワード　RDF, 暗号データベース, デジタル署名, グラフ, プロキシ再暗号化

1. はじめに

近年、災害時の復旧・復興を目的とした情報技術の活用が注目されており、情報共有を行うためのシステムが利用されている。その代表として、災害時伝言サービスや、避難情報配信サービスなどが挙げられる [1]。そのようなシステムではプライバシーやセキュリティに関わる機微な情報が蓄積される場合もあり、情報の機密性を保つ事が必要になる。従って、情報を暗号化してデータベースに保存することで機密性を保ち、暗号化された情報を適切なユーザーにのみ公開するアクセス制御が重要となる。

アクセス制御の 1 つにユーザー毎にアクセスレベルを与え、情報に設定されたアクセスレベルよりも上位の権限を持つユーザーがアクセス可能となる階層型アクセス制御が存在する。もう 1 つの例として、階層型のアクセス制御では実現が難しい、友人や家族といった情報提供者と関係を持つユーザーにアクセスを許可する、人と人との繋がりを利用したアクセス制御が存在する。災害時の復旧・復興においては、医療従事者や政府関係者が正しく被災者の状況を把握する場合には民間人よりも上位の権限を持たせるために階層型アクセス制御が必要となりる。また被災者と関係を持つものが安否確認を行うケースでは被災者と関係を持つ人物が個人の状況を確認するために、人と人との繋がりを利用したアクセス制御が必要となる。このように、2 種類のアクセス制御が重要となってくる。

これらの 2 種類のアクセス制御を取り入れた暗号データベースの先行研究には児玉らの研究 [2] が存在する。しかしながら先行研究では、改竄したデータの挿入によりアクセス権が改変される課題や、グラフの更新に伴う繋がりの変化がアクセス権に反映されない課題が解決されておらず、人と人との繋がりを利用したアクセス制御が実現できていない。

そこで本研究は、人と人との繋がりを利用したアクセス制御実現における上記の脆弱性の課題を解決する事を目的とする。そのため、ノード間を繋ぐ関係と該当ノードに対する署名で改竄を検知する手法と、人と人との繋がりの変更に対する到達可能性をアクセス制御に反映する手法を提案する。災害時の状況を再現できるベンチマークツールを用いて既存手法との比較を行い、署名導入においては約 20%程度の遅延でデータを挿入できることを示した。また到達可能テーブル導入によって既存手法の約 124 倍の速度で動的な更新が可能になることを示した。

2. 背景知識

本章では背景知識となる、アクセス制御手法、RDF (Resource Description Framework)、プロキシ再暗号化について説明する。

2.1 アクセス制御手法

アクセス制御とは、ある情報に対し許可されたユーザーのみがアクセスできるように制御する事である [3]。またアクセス制御ポリシーとは、どのユーザーがどの情報にアクセスできるようにするか判断する基準である。しかしながら、情報を盗み見るためのシステムへの攻撃の存在や、サービス運用者が信用できない等、アクセスポリシーを保つためにはアクセス制御だけでは不十分な場合が存在する。そのため情報に暗号化を施す事で、情報の機密性を保つ事が重要となる。

さらにアクセス制御ポリシーの具体的な実現も、システムの実装における重要な課題の 1 つとなる。1. 章で述べたように、アクセス制御には階層型のアクセス制御や、人と人との繋がりを利用したアクセス制御が考えられる。また人と人との繋がりを利用したアクセス制御では、直接関係を持つユーザー同士が情報を共有するだけでなく、“友人の友人” や “親の兄弟” といった、グラフ上で隣接していないが間接的な繋がりを持つユーザー同士でも、情報を共有できる柔軟なアクセス制御も考えられる。本研究では、間接的な繋がりを持つユーザー同士でも情報を共有する事ができる様な、人と人との繋がりを利用したアクセス制御に着目しており、より柔軟なアクセス制御の実

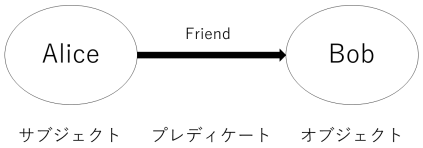

図 1　RDF トリプルによる情報の表現

現を目指す。

2.2　RDF

2.2.1　グラフ構造の表現

RDF は、情報をグラフ構造で表現する方式である。情報の単位としてトリプルが用いられており、2 つノード (サブジェクトとオブジェクト) と当該ノードを繋ぐ有向辺 (プレディケート) の 3 つ組みから成り立つ。例として図 1 では “Alice は Bob の友達である” を表現している。RDF を用いたグラフ構造のデータ構造をとる事で、複雑な人と人との繋がりもグラフを辿る事で認識できる。そのため “友達の友達” など、直接 RDF トリプルで繋がれていないユーザー同士の関係を把握することにも利用する事ができ、人と人との繋がりを利用したアクセス制御に適したデータ構造と言える。本研究においても RDF を用いたグラフ構造を利用する事で、人と人との繋がりを表現する。

2.2.2　暗号化コンテナ

次に暗号化 RDF を格納する構造である暗号化コンテナについて説明する。RDF ノードの暗号文は常に RDF リテラルとして扱われる。しかし、W3C 勧告 [4] においては、RDF リテラルはオブジェクトの位置のみに記述可能とされており、サブジェクトの位置には URI リソースまたは空白ノードのみが、プレディケートの位置には URI リソースのみがそれぞれ許されている。そのため暗号化を行ったオブジェクト、プレディケート、オブジェクトの値をどのようなノードに格納するかが課題になる。

Giereth らの研究 [5] では、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。変化したグラフ構造を暗号化コンテナと定義しており、リテラルである暗号文やハッシュ値を RDF リソースとして扱うことを可能にする。図 2 では、RDF トリプルを暗号化コンテナに変形する様子を示している。本研究においても RDF を暗号データベース上で扱う上で、暗号化コンテナを取り入れている。オブジェクトの変換は、サブジェクトの場合と同様である。

2.3　プロキシ再暗号化

プロキシ再暗号化 [6] とは暗号化された情報を平文に戻すことなく、あるユーザーの鍵による暗号文を別のユーザーの鍵で暗号化された状態に変換する技術である。プロキシ再暗号化を用いる事で、通信路やサーバー上では暗号文の状態のままデータのやり取りが行われるため、情報提供者と情報利用者の 2 者以外に平文を公開する事なく、安全に情報を共有する事ができる。プロキシ再暗号化では再暗号化鍵と呼ばれる、暗号文の変換を行うための鍵が必要であり、プロキシサーバーなどのデータベースサーバーとクライアントの中間層でプロキシ再暗号化を行うことで、クライアントに負荷をかける事なく情報を共有

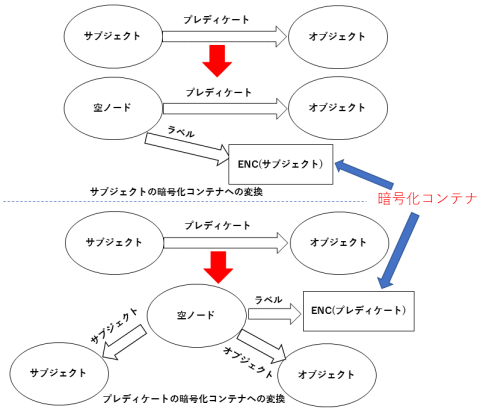

図 2　RDF トリプルの暗号化コンテナへの変換

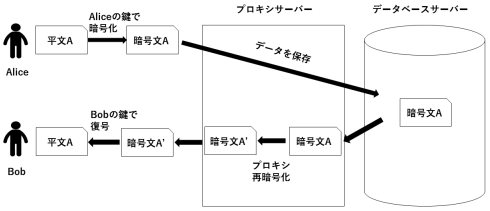

図 3　プロキシ再暗号化を用いた情報共有

することができる。図 3 では Alice が保存した暗号文を Bob 用に変換し、Bob がアクセスできるようになる例である。再暗号化鍵から平文や、情報提供者や情報利用者の秘密鍵を導かれることはなく、あらかじめ信頼できるサーバーで再暗号化鍵を生成しておく事で、信頼できないサーバーにおいてもプロキシ再暗号化を施すことができる。本研究では暗号化された情報を安全に共有するために、プロキシ再暗号化を用いる。

3.　関 連 研 究

本章では、本研究の先行研究、関連研究について説明する。

3.1　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

人と人との繋がりを利用したアクセス制御手法を取り入れた暗号データベースを提案した、児玉らの先行研究 [2] が存在する。アクセスレベルによる階層型のアクセス制御と、人と人との繋がりを利用したアクセス制御の 2 種類のアクセス制御を取り入れており、柔軟なアクセス制御手法を提案している。また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、ユーザークラスと呼ばれるロジカルなクラスを単位とした暗号化を行う手法をとっており、目的に合わせたアクセス制御ポリシーの実現が容易である。図 4 は児玉らの研究で提案された手法のモデル図である。プロキシサーバー、データベースサーバー、秘密鍵生成局からシステムが構成されている。秘密鍵生成局が生成した再暗号化鍵を用いて暗号文を変換する事により、プロキシサーバーやデータベースサーバーが信頼できないサーバーであっても、安全にユーザー同士で情報を共有することができる。

しかし先行研究では、人と人との繋がりを利用したアクセス制御において、3.1.2.a)、b) 節で述べる 2 つ課題が残されて

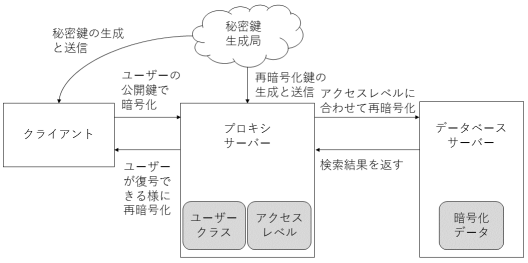

図 4　児玉らのシステムモデル

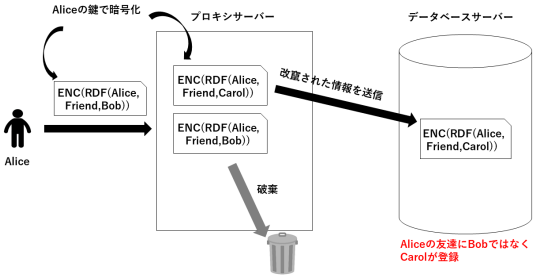

図 5　プロキシサーバーによる改竄の例

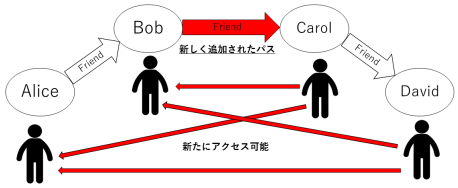

図 6　グラフの更新に伴うアクセス権の更新の例

いる。これらの課題はアクセス制御ポリシーを満たすために特に重要な課題であり、人と人との繋がりを利用したアクセス制御実現のために解決が求められる。

3.1.1　人と人との繋がりを利用したアクセス制御

先行研究では人と人との繋がりを利用したアクセス制御を実現するために、ユーザーが持つアクセスレベルとは独立してプレディケートに関係の深さを表すアクセスレベルを設定している。ユーザーは保存した各データにアクセスを許可するプレディケートのレベルを設定し、許可された以上のレベルの繋がりを持つユーザーのみがアクセスできるようになる。さらに、どの程度の間接的な繋がりを持つユーザーのアクセスを許可するのかを示す最大距離を設定し、アクセスできるユーザーの距離を制限している。例として、あるデータが友人のアクセスを許可しており、間接アクセスできる最大距離が 2 の場合、“友人の友人” までアクセスが可能となる。児玉らの研究では、繋がりを利用したアクセス制御をクローズドアクセスと称し、次のように定義している。

ユーザー A を表すリソース r_A とユーザー 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 $P : r_A \to \cdots \to r_B$ 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クローズドなアクセス制御ポリシーが $C = (c_s, c_p, c_o)$ であるようなトリプルにアクセスすることができる。ここでクローズドなアクセス制御ポリシーの要素 c_s, c_p, c_o はそれぞれ、アクセスを許可するプレディケート最小レベルの繋がりを表す c_{level}、アクセスを許可する最大の距離 c_{length} が設定されている。

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

本研究においても、人と人との繋がりを利用したアクセスを実現する上で、上記の条件を満たしたユーザーにのみ対象データを公開するアクセス制御の実現を目指す。

3.1.2　先行研究が抱える課題

a)　改竄データ挿入によるアクセス権改変の課題

1 つ目の課題は改竄されたデータの挿入による、アクセス権の改変の課題である。公開鍵を用いたシステムでは、公開鍵は一般に公開されているため、悪意のあるユーザーやサーバーも各ユーザーの公開鍵を入手する事ができる。秘密鍵は知りえないため暗号文を復号できないが、公開鍵を用いて暗号文を作成する事は可能である。データベースサーバーに保存された RDF データ構造を用いてアクセス制御を行う場合、改竄された暗号文の挿入によってアクセス権が変更されてしまう。図 5 では悪意を持ったプロキシサーバーによって、Alice の友達に Bob ではなく Carol と改竄されてしまい、Carol が不正に Alice のデータにアクセスできる事を示す。このようにデータベースサーバーに改竄された情報が挿入されることにより、ユーザー同士の繋がりの正確性が担保できず、アクセス制御ポリシーを満たすことができない。攻撃者にはプロキシサーバーの他に、各ユーザー、データベースサーバー、悪意のある第 3 者が想定される。

b)　グラフ更新に伴うアクセス権変更の非反映の課題

2 つ目の課題として、グラフの更新に伴うアクセス権の変更が反映されない課題がある。人と人との繋がりを利用したアクセス制御では、グラフに新しくユーザー同士の繋がりが生じる度に、ユーザーがアクセスできるデータにも変化が生じる。図 6 の例では、友達のアクセスが許可されている場合に、新たに Bob と Carol との間に友達の関係が追加される事で、Carol と David が新たに Alice と Bob のデータにアクセスできる事を示している。

先行研究ではクローズドインデックスと称する、人と人との繋がりの情報を表すインデックスを作成しアクセス制御に取り入れている。しかしながら、更新前は直接つながりのないユーザー間で人と人とのつながりが生じる問題が動的な更新を難しくしている。そのため先行研究では新しく人と人との繋がりが生じた時に、動的にクローズドインデックスを更新しておらず、クローズドインデックスを 1 から作成し直す必要がある。本研究では新しく人と人とのつながりが生じた時に、既に作成されている人と人との繋がりの情報を元に、動的に人の繋がりを示す情報を更新する手法を提案する。

3.2　SIBM

SIBM (Shelter Information Benchmark) は Nguyen らの研究 [7] で提案された RDF データベース用のベンチマークツールである。東日本大震災時の避難情報を基にデータセットを生

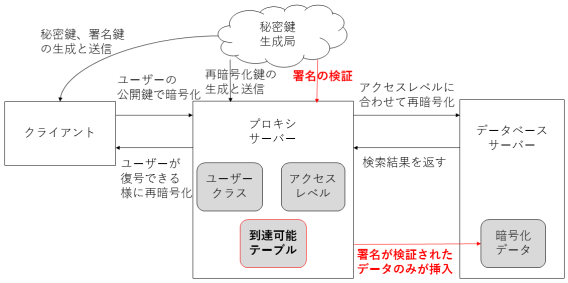

図 7　システムモデル

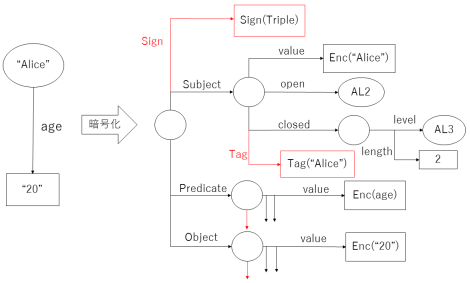

図 8　トリプルの暗号化とデータ構造

成する事ができ、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、災害時の復旧・復興の際に利用されると思われるクエリセットが用意されており、RDF データベースのベンチマークとして利用できる。SIBM には人と人との繋がりの情報がデータセットに含まれており、繋がりを利用したアクセス制御を取り入れたデータベースの評価に適していると考えられる。そこで本研究では SIBM ベンチマークツールを利用して、既存手法と提案手法の比較を行う。

3.3　検索タグ

暗号には大きく分けて、確定的暗号と確率的暗号の 2 種類が存在する。確定的暗号は同じ平文、同じ暗号化鍵ならば、毎回同じ暗号文が生成される方式である。しかし毎回同じ暗号文が生成される確定的暗号は安全性の観点で不十分である。一方、確率的暗号では、同じ平文、同じ暗号化鍵でも毎回異なる暗号文が生成される。確定的暗号に比べて安全性の観点で優れてはいるが、データベース上で暗号文の検索が行えなくなる。

そこで、確率的暗号でも検索を行えるように、暗号文と共に検索用タグをおく手法 [8] が提案されている。検索タグには完全一致で検索を行うもの [8] や、比較を行えるもの [9,10] 等が存在する。3.1 節で紹介した児玉らの研究では、検索を可能とするために、Blase らが提案した暗号方式である BBS 暗号 [11] の乱数を定数に固定した確定的暗号を利用しており、安全性の観点で好ましくない。本研究では、検索用タグを用いる事で確率的暗号を利用できるようにする。

4.　提案手法

はじめに提案手法の概要について説明し、次に 2 つの提案手法について説明する。

4.1　提案するシステムの概要

本研究で提案するシステムは児玉らが提案したシステムを基に実装がなされている。提案システムモデルは、児玉らの研究と同様に、クライアント、プロキシサーバー、データベースサーバー、秘密鍵生成局から成り立つ。図 7 に本研究のモデル図を掲載する。また各サーバーが果たす役割は 4.1.1 節にて説明を行う。

4.1.1　システム構成要素の役割

本研究で提案するシステムにおける、各構成要素の役割について述べる。

クライアント　クライアント上の各ユーザーは固有の ID とアクセスレベルを持ち、サーバーにデータを保存、または SPARQL クエリによる検索を行う事ができる。データを保存する場合、保存したデータにアクセス可能な範囲を設定する事ができる。これにより他のユーザーと情報を共有および制限を行う事ができる。

プロキシサーバー　プロキシサーバーでは秘密鍵生成局から入手した再暗号化鍵を用いて、クエリの結果として出力される暗号文を、検索を行ったユーザーが復号できるよう変換し、クライアントに送信する。本研究では 4.3 節で説明するユーザーがアクセス可能なデータの範囲をテーブルとして持ち、グラフの到達可能性を利用したアクセス制御を行う。

データベースサーバー　プロキシサーバーにて変換されたクエリを SPARQL にて受理し、検索を行う。検索結果は暗号化された状態であり、そのためプロキシサーバーに送信し、クライアントで復号できるようにプロキシ再暗号化を施してもらう。また、プロキシサーバーを介して送られてくる暗号化 RDF コンテナをデータベースに保存する。

秘密鍵生成局　秘密鍵生成局は秘密鍵、公開鍵のペアの生成や再暗号化鍵の生成、署名鍵の生成や署名の検証を行う。またデータベース上のグラフがアクセス制御ポリシーを満たしている場合にのみ、再暗号化鍵をサーバーに送信する。

4.1.2　データ構造

RDF による暗号化の表現として、児玉らが提案したシステムで定義したプロパティに加え、新たに Sign プロパティと Tag プロパティを定義する。データ構造を図 8 に示す。

プロキシサーバーでは、クライアントから受け付けた SPARQL クエリを、上記のデータ構造で検索を行えるように、SPARQL クエリの変換を行う。

Sign プロパティは RDF トリプル全体に対して情報提供者が署名を行った値を格納する。秘密鍵生成局はこの署名を確認し、挿入されたデータが正しい場合にのみ再暗号化鍵を発行する。詳しくは 4.2 節で説明する。次に Tag プロパティでは確率的暗号を利用した際に、毎度生成される暗号文が異なる場合でも、値を検索できるように検索用のタグを設置する。この際、従来と同様に value プロパティに暗号文を格納しておく。また、検索タグが完全一致だけでなく、パターン一致や順序比較に対応している場合には、対応できるクエリの幅を広げる事ができる。さらに暗号化コンテナに格納する暗号文に制限はないため、任

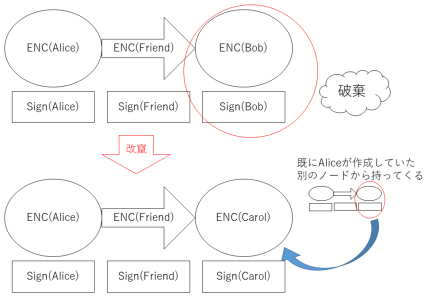

図 9　要素署名の場合の改竄例

意のプロキシ再暗号化可能な暗号方式を適用できる。そのため暗号方式が準同型暗号の場合には集約演算を可能とし、比較可能な暗号文の場合には検索タグを必要とせずに、大小比較を行うクエリを受理できる。

4.2　改竄データ挿入によるアクセス権改変の課題の解決手法

3.1.2.a) 節で説明した 1 つ目の課題である、改竄されたデータによるアクセス権の改変を解決する手法について説明する。本研究ではこの課題に対して、RDF トリプル全体に対してデータを保存する情報提供者が署名を行い、秘密鍵生成局が署名を検証する事で解決を図る。

まず署名を RDF トリプル全体に対して行う理由について説明する。情報提供者がサブジェクト、プレディケート、オブジェクトの 3 つに対して個別に署名を行った場合には、それぞれのノードの正しさを確認する事はできるが、サブジェクト、プレディケート、オブジェクトの関係が正しいものであるか確認できない。例えば、図 9 で示すように、攻撃者はオブジェクトのみを別の値に改竄する事ができる。これは、オブジェクト自体は正しいノードであるため署名の検証では True が出力されるためである。この攻撃方式はオブジェクトだけではなく、サブジェクトやプレディケートにも適応でき得る。

次にアルゴリズム 1 を用いて提案手法の説明を行う。(STEP1) クライアントにて情報提供者の署名鍵を使用し、暗号化されたオブジェクト、プレディケイト、サブジェクトの内容を連結させた値を対象に署名を作成し、署名および暗号化データをプロキシサーバーに送信する。(STEP2) プロキシサーバーは秘密鍵生成局に挿入するデータの検証依頼をする。(STEP3) データ挿入時に秘密鍵生成局が検証鍵を用いて署名を検証し、True ならばデータベース上で有効なグラフとして認める。False の場合にはデータベースサーバー上に対象トリプルが格納されている場合でも、秘密鍵生成局は無効なグラフとして扱い、アクセス制御ポリシーを満たさない再暗号化鍵を生成しないようにする。(STEP4) データベースサーバーにて生成された署名は 4.1.2 節で説明した Sign プロパティに格納される。

従って本研究では RDF トリプルを関係を含め全体に対する署名を行う事を提案する。RDF トリプルの一部または全部が改変される事があっても、情報提供者の署名鍵を知りえない攻撃者は署名を偽造する事ができず、改竄を行う事ができない。

アルゴリズム 1 署名の検証と暗号化データの格納

Input: ユーザ ID id_u, id_u の署名鍵 sk　　暗号化 RDF トリプル $\{subject, predicate, object\}$,

(クライアント) //STEP1
$target \leftarrow subject||predicate||object$
$signature \leftarrow Sign(sk, id_u, target)$
$\{id_u$, {subject, predicate, object}, signature$\}$ をプロキシサーバーに送信

(プロキシサーバー)
$\{id_u$, {subject, predicate, object}, signature$\}$ を受信
秘密鍵生成局に検証要求　//STEP2
if 検証結果が $true$ **then**
　$\{id_u$, {subject, predicate, object}, signature$\}$ をデータベースサーバーに挿入要求 //STEP4
end if

(秘密鍵生成局) //STEP3
検証要求を受信
$vk \leftarrow getVerifyKey(id_u)$
$target \leftarrow subject||predicate||object$
if $Verify(vk, id_u, target, signature) = true$ **then**
　{subject, predicate, object} を有効 RDF として保存
end if
検証結果をプロキシサーバーに送信
if 新しくユーザー間に繋がりが生じた **then**
　再暗号化鍵を生成し、プロキシサーバーに送信
end if

4.3　グラフ更新に伴うアクセス権変更の非反映の課題の解決手法

3.1.2.b) 節で説明した 2 つ目の課題である、グラフの更新に伴うアクセス権の変更が反映されない課題を解決する手法について説明する。この課題を解決するために、グラフの到達可能性を利用したアクセス制御を提案する。

3.1.1 節で説明したアクセス制御ポリシーを満たすためには、あるデータが格納されているノードと、そのデータにアクセスを行うユーザーのノード間には少なくともグラフ上にパスが存在していなければならない。そのため、更新が行われたサブジェクトノード及びオブジェクトノードを始点にグラフ上の探索を行う事で、アクセス制御ポリシーを満たす可能性のある、新しい人と人との繋がりを検出する事ができる。サブジェクトノード側からの探索は有効辺とは逆向きの探索となり、アクセス制御ポリシーの性質 1 が成り立つため、サブジェクトがアクセス可能なノードのみを確かめれば良い。一方、オブジェクト側はオブジェクトから有効辺で繋がる全ノードを確かめる必要がある。

性質 1. あるノード A からノード B までのパス P_1が存在し、ノード B からノード C までのパス P_2が存在する時、
「P_1がアクセスポリシー C_Aを満たさない」
⇒「連結パス P_1P_2はアクセスポリシー C_Aを満たさない」

Proof.　P_1がアクセスポリシー C_Aを満たさないならば、パ

ス P_1中に存在するあるノード n のレベルが $C_{a_{\text{level}}}$よりも小さい、または $C_{a_{\text{length}}} > length(P_1)$ が成り立つ。前者の場合、連結パス P_1P_2はノード n を通るためパス中の最小アクセスレベルが許可されたレベルよりも大きくなる。後者の場合は連結パス P_1P_2がパス P_1よりも長いため、アクセスを許可する最大距離の条件に当てはまらない。 □

本提案手法では、グラフの到達可能性を利用する事でアクセス制御ポリシーを満たすノードの組を検出し、ノード間のパスが含んでいるアクセス制御に関わる情報を、プロキシサーバーが持つテーブルに格納する。本研究ではこのテーブルを “**到達可能テーブル**” と称する。そして検索時には保存されたノード間のパス長及び最小アクセスレベルを確認し、アクセス制御ポリシーを満たす場合のみアクセスを許可する。アルゴリズム 2 にグラフの到達可能性を利用する事でアクセス制御ポリシーを満たすノードの組を検出し、パス長、最小アクセスレベルを到達可能テーブルに保存するまでのアルゴリズムを示す。そしてアルゴリズム 3 では、到達可能テーブルに保存された情報を利用し、人と人との繋がりを利用したアクセスを行うアルゴリズムを示す。

アルゴリズム 2 グラフの到達可能性を利用したアクセス制御

Input: 暗号化 RDF トリプル $\{subject, predicate, object\}$
Output: 更新された到達可能テーブル
$PersonNodes\_S = \{nodes \mid subjects$ がアクセス可能な $node\}$
$PersonNodes\_O = \{nodes \mid object$ からのパスが存在する $node\}$
$level\_P \leftarrow predicate$ に設定されているレベル
for all $node\_S \in PersonNodes\_S$ **do**
  $length_{\text{node\_S} \rightarrow \text{subject}} \leftarrow length(node\_S, subject)$
  $level\_S \leftarrow$ パス $node\_S$ と $subject$ 中の最小レベル
  for all $node\_O \in PersonNodes\_O$ **do**
    $length_{\text{object} \rightarrow \text{node\_O}} \leftarrow length(object, node\_O)$
    $level\_O \leftarrow$ パス $object$ と $node\_O$ 中の最小レベル
    $length_{\text{node\_S} \rightarrow \text{node\_O}} \leftarrow length\_S + length\_O + 1$
    $level_{\min} \leftarrow MIN(level\_S, level\_P, level\_O)$
    if レベルが $level_{\min}$である $node\_S$ から $node\_O$ へのパスが到達可能テーブルに保存されていない ∨ ((保存されている距離 $< length_{\text{node\_S} \rightarrow \text{node\_O}}$) ∧ (保存されている最小レベル $> level_{\min}$)) **then**
      $\{node\_S, node\_O, length_{\text{node\_S} \rightarrow \text{node\_O}}, level_{\min}\}$ を到達可能テーブルに保存
    end if
  end for
end for

5. 実　　験

提案手法導入によるオーバーヘッドを調べるために、既存手法との比較実験を行う。

5.1 実 験 環 境

実験には表 1 の環境を用いた。また回線速度は 100Mbps である。なお実装の都合上、本実験ではプロキシサーバーとデータベースサーバーを同一のサーバー上に実装した。データベースサーバーは Apache Jena 2.12.1 を用いて構築している。

アルゴリズム 3 RDF トリプルへの繋がりを利用したアクセス

Input: ユーザ ID id_u, 再暗号化鍵 RK,
  対象 RDF のアクセス制御ポリシー $C_R = (C_S, C_P, C_O)$,
  対象 RDF トリプル $\{subject, predicate, object\}$
Output: $\{subject', predicate', object'\}$

$length_{id_u \rightarrow \text{R}} \leftarrow getLengthFromTable(id_u, R)$
$level\_min \leftarrow getMinLevelFromTable(id_u, R)$
if $(length_{id_u \rightarrow \text{R}} \geqq MAX(C_{S_{\text{level}}}, C_{P_{\text{level}}}, C_{O_{\text{level}}})) \wedge (level\_min \leqq MIN(C_{S_{\text{length}}}, C_{P_{\text{length}}}, C_{O_{\text{length}}}))$ **then**
  $subject' = RK(subject)$
  $predicate' = RK(prediacte)$
  $object' = RK(object)$
end if

表 1　実 験 環 境

サーバー	プロキシサーバー, データベースサーバー	秘密鍵生成局	クライアント
CPU	AMD Opteron 6174 (12 コア/2.2GHz/L3 12MB) x 4	AMD Opteron 4184 (6 コア/2.8GHz/) x 2	Intel Core i7-2600 3.4GHz
RAM	32GB	32GB	4GB
OS	Ubuntu 16.04.3	Ubuntu 17.04	Windows7 Professional
Java	1.8.1_151	1.8.0_151	1.7.0_80

表 2　データセットの情報

対象情報	含まれている情報
個人情報	氏名、生年月日、年齢、性別、住所、所属、携帯番号、メールアドレス、家族関係、友人関係、病気、避難先
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、備蓄情報

5.2 データセット

本実験では、3.2 節で紹介した SIBM ベンチマークツールからデータセットを生成した。本実験では、情報を生成する対象地点を被災地である福島県とし、実際の災害時の復旧・復興を目的としたシステムの利用を再現する。データセットには表 2 の情報が含まれている。

5.3 比 較 対 象

提案手法の評価を行うために、次の 4 つの手法で比較実験を行う。

- 既存手法: 児玉らの手法。検索タグも付与せず、署名の格納と検証を行わない手法。暗号方式は BBS 暗号を利用した。
- 検索タグ付与: 新しく設けた Tag プロパティを用いて、検索は暗号文の代わりにタグを用いて行う。タグに SHA256 を利用。
- 署名導入: 既存手法に 4.2 節で提案した署名を導入した手法。署名には DSA 署名を利用し、データセット挿入時に RDF トリプルの署名検証が行われる。

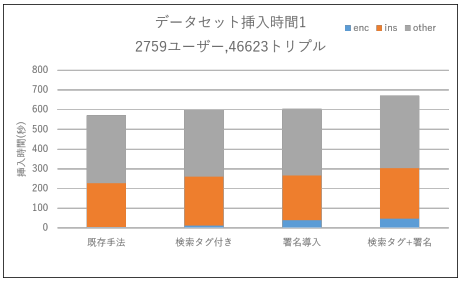

図 10　データセット挿入時間 1

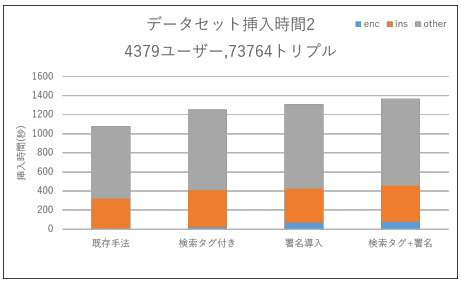

図 11　データセット挿入時間 2

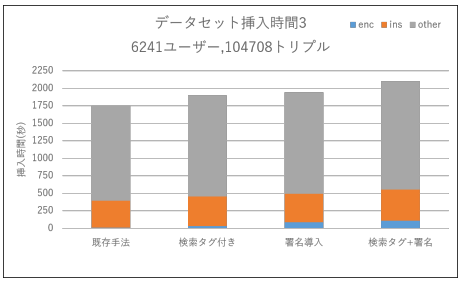

図 12　データセット挿入時間 3

- 検索タグ+署名: 既存手法に検索タグと署名を導入した手法。検索タグ付きと署名導入の組み合わせである。

5.4　実験結果

5.4.1　実験 1:署名導入によるオーバーヘッドの測定

始めに、署名を導入する事で生じるオーバーヘッドの測定を行う。署名はデータ挿入時に利用されるため、SIBM ベンチマークで生成したデータセットの挿入時間を既存手法と比較を行う。

図 10、11、12 はそれぞれ、(2,759 ユーザー 46,623 トリプル),(4,379 ユーザー 73,764 トリプル),(6,241 ユーザー 104,708 トリプル) のデータセットを挿入した時の挿入時間を表している。enc は平文に暗号化を施す時間と署名や検索タグ作成の時間の合計である。ins はデータベースサーバーが挿入リクエストを受け取ってから挿入し終えるまでの時間の総和である。

5.4.2　実験 2:到達可能テーブルの更新時間の測定

次に、到達可能テーブルの更新にかかる時間を計測した。計測方法として、ランダムなユーザー間にパスを作成する事で人と人との繋がりを生成し、その際に到達可能テーブルの更新にかかる時間を測定する。本実験ではランダムなユーザーの組み合わせ数は重複しない 100,000 通りであり、更新に関わるユーザー数毎に平均値を集計した。ここでの更新に関わるユーザー数とはアルゴリズム 2 に出現するテーブルサイズの積、「$PersonNodes\_S$ のサイズ ×$PersonNodes\_O$ のサイズ」と定める。

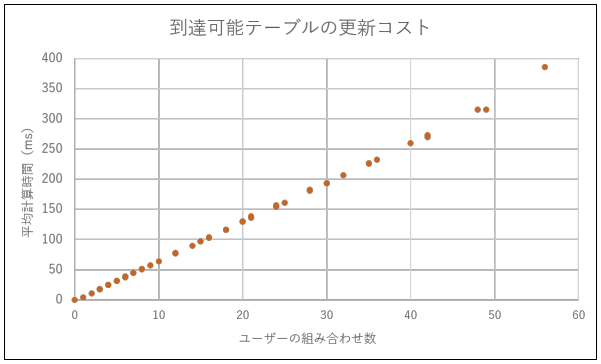

図 13　到達可能テーブル更新時間

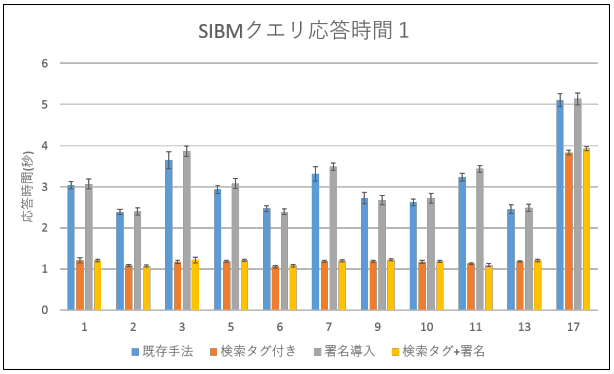

図 14　SIBM クエリ応答時間 1

到達可能テーブルの更新では、更新に関わるユーザー数が最大である 56 の場合で、385.9 ミリ秒の時間がかかる事が実験結果から伺えた。また、到達可能テーブルの更新に関わるユーザーが、パスを生成されたユーザー二人のみの場合、最小値である 3.73 ミリ秒を計測した。さらに、到達可能テーブルの更新時間は、更新に関わるユーザー数に対して線形の関係がある事がグラフから推測できる。

また、この到達可能テーブルのテーブルサイズは、2,759 ユーザー、46,623 トリプルのデータセットに対して 5.30MB であり、6,241 ユーザー、104,708 トリプルのデータセットに対して 10.7MB、9,835 ユーザー、164,576 トリプルのデータセットに対しては 15.6MB であった。

5.4.3　実験 3:クエリ応答時間の比較

提案手法の性能を評価するために、4 つの手法における SIBM ベンチマークツールのクエリ応答時間の計測を行った。SIBM クエリ 18 種類それぞれ 10 回の平均を集計した。クエリによって応答時間が大きな差が生じていたため、その結果を異なるスコープを持つ図 14 と図 15 に分けて示す。

6.　考　　察

実験 1 の結果、既存手法と検索タグ付きを比較する事で図 11 から既存手法 1,080 秒、検索タグ付き 1,255 秒より最大 16.3%遅延が生じる事が分かった。一方、既存手法と署名導入を比較す

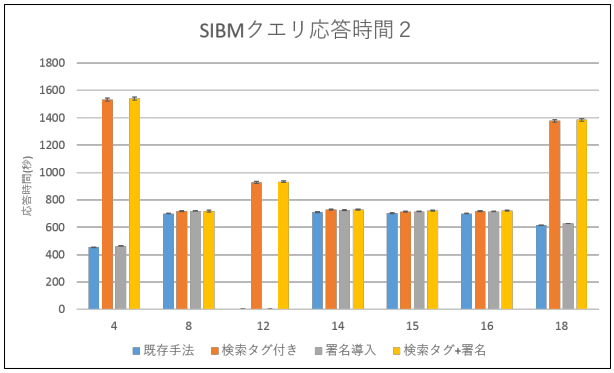

図 15　SIBM クエリ応答時間 2

る事で署名の導入により、図 11 から既存手法 1,080 秒、署名導入 1,310 秒より最大 21.3%の遅延が生じる事が分かった。データの挿入が実用可能な速度で行われている。

実験 2 より、到達可能テーブルの更新は 0.4 秒未満で可能な事が分かった。初回のテーブル作成時間 117 秒であるが、更新時間は児玉らの手法でクローズドインデックスを再作成する時間である 49 秒に比べて約 124 倍の速さである。システムを運用する上で実用的な時間だと考えられる。一方、テーブルのサイズは実験結果よりデータセットのサイズに比例すると考えられる。システムを県で運用する場合、1 つの県の人口を 100 万人と仮定すると、約 14GB の容量が必要となる見積もりである。これは SIBM は地域ごとに閉じたデータセットを作成するため、最大でも更新に関係するユーザー数が 1 地域で 56 人であることが大きく関与している。

実験 3 ではクエリの応答時間を計測した。実験結果より、署名導入による応答時間の増加は平均 1.66%と極めて小さい事が分かった。一方、検索タグを付与した場合には図 14 のクエリに関しては応答時間が減少したが、クエリ 4、12、18 の応答時間は大きく増加した。クエリの細分化やクエリ条件やデータ構造の変更を通した検証実験を行った結果、結合を必要とするクエリはでハッシュ関数で実装された検索タグを利用するに伴い応答時間が増加する事が分かった。その詳細については、いまだに解明されていない。

7. おわりに

7.1 まとめ

災害時の復旧・復興を目的とした情報共有システムでは、階層型のアクセス制御と人と人との繋がりを利用したアクセス制御の 2 種類のアクセス制御が必要となる。本研究では、人と人との繋がりを利用したアクセス制御実現における先行研究で解決されていない 2 つの課題の解決を図った。1 つ目の課題は、改竄されたデータの挿入によるアクセス権の改変の課題であり、本研究では RDF トリプル全体に対する署名を導入し検証を行う手法を提案した。2 つ目の課題は、グラフの更新に伴うアクセス権の変更が反映されない課題であり、グラフの到達可能性を利用し更新を行う到達可能テーブルを利用しアクセス制御を行う手法を提案した。署名導入は既存手法と比べて、挿入時間では最大 21.3%、クエリ応答時間では平均 1.66%の遅延が生じることがわかった.。一方、到達可能テーブル導入により、2 回目以降の更新では既存手法と比べて 124 倍の速度で動的に更新可能であることを実験により示した。

7.2 今後の課題

7.2.1 タグを導入したことによるクエリの変化の原因究明

本研究で行った実験では、実験 3 におけるタグを導入した場合にクエリ応答時間が既存手法に比べて大きく増加してしまった原因を特定できなかった。確率的暗号を利用するにあたって、タグを用いる事が必要になるため、原因の究明が求められる。

7.2.2 データ構造に適応したデータベース問い合わせ設定

現在のシステムでは複数の RDF トリプルを用いて 1 つの論理的な暗号化 RDF トリプルを実装している。データ構造に適応したデータベース問い合わせ言語を実装し、1 つの論理的な RDF トリプルを 1 度の探索で評価できれば、クエリ応答時間の短縮が見込める。

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 22, 2017, from:http://www.soumu.go.jp/main_content/000497711.pdf

[2] 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.

[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 22, 2017, from:http://www.ieice-hbkb.org/files/03/03gun_07hen_02.pdf.

[4] Dan Brickley and R.V. Guha. RDF schema 1.1. Recommendation, W3C,February 2014. Retrieved December 22, 2017, from: http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[5] Mark Giereth. On partial encryption of RDF-graphs. In Proc. the 4^{th} Int. Conf. The Semantic Web, pp. 308-322. Springer-Verlag, 2005.

[6] 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.

[7] 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.

[8] Boneh D., Di Crescenzo G., Ostrovsky R., Persiano G. (2004) Public Key Encryption with Keyword Search. In: Cachin C., Camenisch J.L. (eds) Advances in Cryptology - EUROCRYPT 2004. EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027. Springer, Berlin, Heidelberg.

[9] Kevin Lewi and David J. Wu. 2016. Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). ACM, New York, NY, USA, 1167-1178.

[10] 平田拓三, 宮沢駿輔,HieuHanhLe. 横田治夫, プロキシ再暗号化と検索タグを用いた範囲クエリ可能な秘匿情報の耐結託共有手法. 第 10 回データ工学と情報マネジメントに関するフォーラム, 2018 年 3 月.

[11] 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.