RL-004

Proxy re-encryption を用いたマルチユーザ向け完全準同型暗号の提案

Fully Homomrphic Encryption for Multi-user system using Proxy re-encryption

柴田崇夫 [1] 松澤智史 [2] 武田正之 [2]
Takao Shibata Tomofumi Matsuzawa Masayuki Takeda

株式会社電通国際情報サービス [1]
Information Services International-Dentsu, Ltd.

東京理科大学 理工学部 情報科学科 [2]
Tokyo University of Science

1 はじめに

近年，企業によるパブリッククラウドの利用が盛んに行われている．パブリッククラウドを利用することで，企業は必要に応じて効率良く計算資源を利用できる．しかし，顧客情報を始めとする企業の機密情報を処理する際は，安全性の点からパブリッククラウドを利用できない．また，暗号化することで機密情報の安全性を高めることも可能であるが，暗号化した機密情報に対しては検索や統計処理などを行えない．

完全準同型暗号は，暗号化した状態で計算が可能な暗号系である．これにより，暗号化した機密情報をパブリッククラウド上で安全に処理できる．しかし，既存の完全準同型暗号の多くは単一の鍵ペアを想定しており，複数のユーザによる利用を考慮していない．

また，企業による利用を考えた場合，社員の所属や役職に応じた復号権限の制御が求められる．例えば，複数のプロジェクト間で機密情報を共有する場合がある．各プロジェクトは異なる権限を持ち，それぞれ異なる鍵ペアを持つものとする．この場合，共有する機密情報を各プロジェクトが復号・閲覧するために，共有する機密情報は各プロジェクトの秘密鍵で復号できる必要がある．また，各プロジェクト固有の機密情報と，共有する機密情報を一緒に計算する場合，これら 2 種類の機密情報は異なる鍵で暗号化されるため，鍵の異なる暗号文同士の計算が実行できる必要がある．

そこで本稿では，Proxy re-encryption を用いた 復号権限の委任 に基づく，マルチユーザ向け完全準同型暗号を提案する．具体的には，次の 3 つの性質を持つ完全準同型暗号を提案する．第一に，全てのユーザは鍵ペアを 1 つのみ持つ．第二に，ある暗号文を複数の秘密鍵で復号できる．第三に，鍵の異なる暗号文同士の計算が可能である．

2 完全準同型暗号とは

定義 2.1 (完全準同型暗号) 鍵ペア (sk, pk) ，平文 m_i ，暗号文 $c_i \leftarrow Encrypt(pk, m_i)$ とする．任意の回路 f に対し

$$\begin{aligned} & c' \leftarrow Evaluate(pk, f, \{c_1, ..., c_t\}) \\ \Rightarrow \quad & Decrypt(sk, c') = f(m_1, ..., m_t) \end{aligned} \tag{1}$$

が成り立つような処理 $Evaluate$ が存在するとき，この暗号系は完全準同型暗号である．

完全準同型暗号（Fully Homomorphic Encryption）は $KeyGen, Encrypt, Decrypt, Evaluate$ の 4 つのアルゴリズムから構成される．

完全準同型暗号

$KeyGen(1^\lambda)$ 鍵ペア (sk, pk) を生成する．

$Encrypt(pk, m)$ 平文 m を公開鍵 pk で暗号化し，暗号文を返す．

$Decrypt(sk, c)$ 暗号文 c を秘密鍵 sk で復号し，平文を返す．

$Evaluate(pk, f, \{c_1, ..., c_t\})$ 公開鍵 pk ， t 個の入力を取る回路 f ，暗号文 $\{c_1, ..., c_t\}$ を取る．式 (1) を満たす暗号文 c' を返す．

完全準同型暗号は暗号化した状態で復号回路を含む任意の回路を評価できる．この性質を利用して，暗号化した状態で復号回路を実行することで，平分や秘密鍵を秘匿したまま暗号文を復号・再暗号化できる．この処理を $Recrypt$ と呼ぶ．$Recrypt$ は計算によって蓄積される暗号文のノイズを削減するために用いられる．

Recrypt

$Recrypt(pk_2, sk'_1, c)$ 鍵ペア $(sk_1, pk_1), (sk_2, pk_2)$ とする．$pk_2, sk'_1 = Encrypt(pk_2, sk_1)$ $c = Encrypt(pk_1, m)$ を受け取る．このとき $c' = Encrypt(pk_2, c)$ かつ D を復号回路とし

$$\begin{aligned} & Evaluate(pk_2, D, \{sk'_1, c'\}) \\ = \quad & Encrypt(pk_2, m) \end{aligned} \tag{2}$$

を返す．

3 既存研究

3.1 マルチユーザ向け暗号

属性ベース暗号 [1] は，ユーザの持つ属性に応じて復号の可否を制御可能な暗号系である．属性ベース暗号を用いることで，社員の部署や役職などの属性に応じて，社員の閲覧できる情報を柔軟に制御できる．

属性ベース暗号は，一つの暗号文を複数の復号鍵で復号できるという特徴をもつ．しかしこれらの暗号系では，暗号文同士の計算はできない．

3.2 単一ユーザ向け完全準同型暗号

2009 年に Gentry がイデアル格子を用いて初めて完全準同型暗号を実装した [2]．それ以降，整数上の実装 [3][4] や，Ring-LWE に基づく実装 [5][6] などが登場した．しかし，既存の完全準同型暗号の多くは単一鍵ペアのみを想定しており，マルチユーザ向けではない．

3.3 マルチユーザ向け完全準同型暗号

従来の完全準同型暗号では単一の鍵ペアによる処理しか考慮しておらず，鍵の異なる暗号文同士は計算できなかった．これに対し，複数の鍵ペアが登場するマルチユーザ向けの完全準同型暗号として，Xiao らの暗号系 [7] と Lopez-ALt らの暗号系 [8] がある．

Xiao ら [7] は各ユーザが異なる鍵を持つようなサーバクライアントモデルのシステムを構築した．各ユーザは自身の鍵 k_i を持ち，暗号化や復号には k_i を用いる．一方で，サーバに格納された暗号文は全てマスタ鍵 k で暗号化される．ユーザとサーバが暗号文を送受信する際には，暗号文の鍵を k_i から k へ（あるいはその逆）変換する．これにより，各ユーザは個別の鍵 k_i で暗号化・復号でき，また一方で全ての暗号文同士をマスタ鍵 k の下で計算できる．しかし Xiao らのシステムでは全ユーザの復号権限は平等であり，暗号系による復号権限の制御はできない．

Lopez-Alt ら [8] は，鍵の異なる暗号文同士の演算が可能な完全準同型暗号を提案した．各ユーザ u_i の鍵ペアを (sk_i, pk_i)，平文を m_i とし，暗号文 $c_i \leftarrow Encrypt(pk_i, m_i)$ とするとき，暗号文 $\{c_1, ..., c_t\}$ と任意の回路 f に対して $Evaluate$ を実行できる．この実行結果を復号するには，関係するすべての秘密鍵が必要となる（すなわち先述の $Evaluate$ の実行結果を c' としたとき $Decrypt(\{sk_1, ..., sk_t\}, c') = f(m_1, ..., m_t)$ ）．この暗号系では，各ユーザは異なる鍵ペアによる異なる復号権限を持つ．

Lopez-Alt の暗号系を用いた場合，ユーザが別のユーザへ自身の秘密鍵を送ることで復号権限を委任できる．しかし，復号権限構造が複雑になった場合に各ユーザが複数の秘密鍵を管理する必要があり，鍵管理が煩雑になる．また，復号権限の委任を取り消すには，委任元の鍵ペアの再生成と，関連する全ての暗号文の再暗号化を伴う．

4 プロキシ暗号

プロキシ暗号 [9] は，ユーザ u_i の公開鍵で暗号化された暗号文を，プロキシを経由することで u_i にかわって暗号文を復号する暗号方式である．また proxy re-encryption(PRE)[10] は，プロキシが平文の情報を得ずにユーザ u_i の暗号文をユーザ u_j の暗号文に変換できる暗号方式である．

既存の PRE はペアリングなどを用いて実装されているが，完全準同型暗号でも前述の $Recrypt$ アルゴリズムを用いることで PRE を実現できる [2].

Recrypt を用いた PRE

$ReKeyGen(sk_1, pk_2)$ 再暗号化鍵 $\pi_{1\to2} \leftarrow Encrypt(pk_2, sk_1)$ を返す.

$ReEncrypt(\pi_{1\to2}, c_1)$ $c_1 = Encrypt(pk_1, m)$ と再暗号化鍵 $\pi_{1\to2}$ を入力とする．暗号文 $c_2 \leftarrow Recrypt(pk_2, \pi_{1\to2}, c_1) = Encrypt(pk_2, m)$ を返す.

また Brakerski ら [5] の完全準同型暗号では $Recrypt$ ではなく $SwitchKey$ アルゴリズムにより PRE を実現できる．以降，完全準同型暗号を用いた PRE を FHE-PRE とし，中でも $Recrypt$ を用いた PRE を FHE-PRE$_{\mathrm{Recrypt}}$ とする．

Ateniese[11] は PRE の満たすべきいくつかの性質を定義した．そのうち Collusion-safe,Non-transitive,Non-transferable について，FHE-PRE$_{\mathrm{Recrypt}}$ がこれらの性質を満たすかどうか検討する．Collusion-safe とは，プロキシと復号権限の委任先のユーザが結託しても委任元の秘密鍵が漏洩しない性質を指す．Non-transitive とは，プロキシ単独で再暗号化鍵 $\pi_{i\to j}, \pi_{j\to k}$ から $\pi_{i\to k}$ を生成できない性質を指す．Non-transferable とは，復号権限の委任元のユーザが復号権限を委任していないユーザに対して，プロキシと復号権限の委任先のユーザが結託しても，復号権限を与えられない性質を指す．

結果，FHE-PRE$_{\mathrm{Recrypt}}$ は先述の 3 つの性質を満たさない．

補題 4.1 *FHE-PRE$_{Recrypt}$ は Collusion-safe を満たさない.*

証明 4.1 再暗号化鍵 $\pi_{i\to j}$ と sk_j から $sk_i \leftarrow Decrypt(sk_j, \pi_{i\to j})$ が求まる.

補題 4.2 *FHE-PRE$_{Recrypt}$ は Non-transitive を満たさない.*

証明 4.2 再暗号化鍵 $\pi_{i\to j}, \pi_{j\to k}$ から

$$\begin{aligned} \pi_{i\to k} &\leftarrow Recrypt(pk_k, \pi_{j\to k}, Encrypt(pk_k, \pi_{i\to j})) \\ &= Encrypt(pk_k, sk_i) \end{aligned} \quad (3)$$

が求まる.

補題 4.3 *FHE-PRE$_{Recrypt}$ は Non-transferable を満たさない.*

証明 4.3 証明 *4.1* より求めた秘密鍵 sk_i から $\pi_{i\to k} \leftarrow Encrypt(pk_k, sk_i)$ が求まる.

5 提案手法

5.1 節で提案手法の満たすべき性質について述べる．その後 5.2 節と 5.3 節で提案手法の 2 つのシステム構成について述べる.

5.3 節のシステム構成は，5.2 節を変更し，擬似的な Non-transitive を満たすようにしたものである．反面，信頼できる第三者機関を設ける必要があり，またクラウドが管理する再暗号化鍵が多くなる.

5.1 性質

本稿では，復号権限の委任に基づくマルチユーザ向け完全準同型暗号を提案する．ユーザ u_i の鍵ペア (sk_i, pk_i) ，鍵 pk_i による暗号文の集合 $C_i = \{c | c \leftarrow Encrypt(pk_i, m)\}$ とし，ユーザ u_i から u_j への復号権限の委任を $u_i \rightarrow u_j$ と表記する．このとき，提案手法は次の性質を満たす．

- 各ユーザ u_i は復号鍵 sk_i を 1 つだけ持つ．
- $u_i \rightarrow u_j$ のとき，暗号文 $c \in C_i$ はユーザ u_i, u_j のどちらでも復号できる．
- $u_i \rightarrow u_j$ のとき，暗号文 $c_i \in C_i, c_j \in C_j$ と任意の回路 f に対して $Evaluate$ の結果 $c' \in C_j$ を計算できる，

復号権限の委任は 4 章で述べた FHE-PRE により実現できる．

補題 5.1 *FHE-PRE* により上記の性質を満たす暗号系を実現できる．

証明 5.1 各ユーザ u_i は復号鍵 sk_i を 1 つだけ持つものとする．このとき以下が成り立つ．

- $u_i \rightarrow u_j$ のとき $\pi_{i \rightarrow j}$ が存在し，$c_i \in C_i$ に対して

$$\begin{aligned} m &= Decrypt(sk_i, c_i) \\ &= Decrypt(sk_j, ReEncrypt(\pi_{i \rightarrow j}, c_i)) \end{aligned} \quad (4)$$

が成り立つ．

- $u_i \rightarrow u_j$ のとき $\pi_{i \rightarrow j}$ が存在し，$c_i \in C_i, c_j \in C_j$ と任意の回路 f に対して

$$\begin{aligned} & c'_i \leftarrow ReEncrypt(\pi_{i \rightarrow j}, c_i) \\ \Rightarrow \quad & Evaluate(pk_j, f, \{c'_i, c_j\}) \quad \in C_j \end{aligned} \quad (5)$$

が計算できる．

FHE-PRE は， $Recrypt$ を実行可能な任意の完全準同型暗号や，Brakerski ら [5] の完全準同型暗号を用いて実現できる．よって，提案手法は特定の完全準同型暗号に依存せず実現できる．

5.2 システム構成

$Cloud$ を非信頼なパブリッククラウドとする．$Cloud$ は膨大なストレージや高い計算能力を持つが，機密情報を盗み見する可能性のある非信頼ノードである．また u_i をユーザとし， $Cloud$ とユーザ u_i は結託しないものとする．

提案手法のアルゴリズムと区別するため，2 章で述べた完全準同型暗号のアルゴリズムを FHE として記す．

$KeyGen(1^\lambda)$ 各ユーザ u_i は自身の鍵ペア $(sk_i, pk_i) \leftarrow FHE.KeyGen(1^\lambda)$ を生成する．

$Encrypt(pk_i, m)$ 平文 m ，ユーザ u_i の公開鍵 pk_i を入力とする．ユーザは暗号文 $c \leftarrow FHE.Encrypt(pk_i, m)$ を計算し， $Cloud$ に格納する．

$Evaluate(pk_{sup}, f, \{c_1, ..., c_t\})$ 暗号文 $\{c_1, ..., c_t\}$ ，回路 f を入力とし，以下の処理を行う．

1. $c_i = FHE.Encrypt(pk_i, m_i)$ とする．$Cloud$ は定義された委任関係から，全ての暗号文 c_i に対して再暗号化鍵 $\pi_{i \rightarrow sup}$ が存在するような公開鍵 pk_{sup} が存在するか確認する．存在しない場合は失敗を返す．
2. $Cloud$ は定義された委任関係から，各 c_i に対して $c'_i \leftarrow ReEncrypt(\pi_{i \rightarrow sup}, c_i) = FHE.Encrypt(pk_{sup}, m_i) \in C_{sup}$ を計算する．
3. $Cloud$ は $c'' \leftarrow FHE.Evaluate(pk_{sup}, f, \{c'_1, ..., c'_t\}) \in C_{sup}$ を計算し，結果を格納する．

$Decrypt(sk_i, c)$ 暗号文 c ，ユーザ u_i の秘密鍵 sk_i を入力とし，以下の処理を行う．

1. $c = Encrypt(m, sk_j)$ とする．$Cloud$ は定義された委任関係から， $\pi_{j \rightarrow i}$ が存在するか確認する．存在しない場合は失敗を返す．
2. $Cloud$ は $c' \leftarrow ReEncrypt(\pi_{j \rightarrow i}, c) = FHE.Encrypt(pk_i, m) \in C_i$ を計算し，結果をユーザ u_i に返す．
3. ユーザ u_i は受信した c' と自身の秘密鍵 sk_i から $m \leftarrow FHE.Decrypt(sk_i, c')$ を得る．

$AddAttorney(pk_j, sk_i)$ 委任関係 $u_i \rightarrow u_j$ を定義する．ユーザ u_i は再暗号化鍵 $\pi_{i \rightarrow j} \leftarrow ReKeyGen(pk_j, sk_i)$ を作成し $Cloud$ に格納する．

$RevokeAttorney(\pi_{i \rightarrow j})$ 委任関係 $u_i \rightarrow u_j$ を失効する．$Cloud$ に再暗号化鍵 $\pi_{i \rightarrow j}$ が格納されているか確認し，存在するならば $Cloud$ 上から削除する．

5.3 擬似的な Non-transitive を満たすシステム構成

5.2 節のシステムを変更し，Non-transitive を擬似的に満たすシステムを以下に示す．このシステムでは新たに，信頼できる第三者機関である鍵管理局（以降 KM ）を設ける必要がある．

$KeyGen(1^\lambda)$ ユーザ u_i の鍵ペアを生成する．

1. KM は各ユーザ u_i の鍵ペア $(sk_i, pk_i), (sk'_i, pk'_i) \leftarrow FHE.KeyGen(1^\lambda)$ を生成する．
2. KM は (sk_i, pk_i) と pk'_i をユーザに渡し， sk'_i を KM で管理する．
3. KM は $\pi_{i' \rightarrow i} \leftarrow ReKeyGen(pk_i, sk'_i)$ を生成し， $Cloud$ に格納する．

$Encrypt(pk'_i, m)$ 平文 m ，ユーザ u_i の公開鍵 pk'_i を入力とする．ユーザは暗号文 $c \leftarrow FHE.Encrypt(pk'_i, m)$ を計算し， $Cloud$ に格納する．

$Evaluate(pk_{sup}, f, \{c_1, ..., c_t\})$ 5.2 節と同様．

表 1　実行環境

OS	Ubuntu 10.04.4 LTS
kernel	2.6.32-41-server
CPU	Intel(R) Xeon(R) CPU E5420 @ 2.50GHz
Memory	15GB
コンパイラ・ライブラリ	gcc4.4.3, NTL6.0.0, HElib

表 2　パラメータ

m, p, r (Native plaintext space is $\mathbf{Z}[x]/(\Phi_m(\mathbf{X}), p^r)$)	15709,257,1
d (Plaintext space $\mathbf{F}_{p^d}$)	0
Levels	16
Number of columns in key switching matrix	3
Hamming weight of a secret key	64
Security	128

$Decrypt(sk_i, c)$ 5.2 節と同様.

$AddAttorney(pk'_j, sk_i)$ 委任関係 $u_i \to u_j$ を定義する. KM は $\pi_{i'\to j} \leftarrow ReKeyGen(pk_j, sk'_i)$ を作成し $Cloud$ に格納する.

$RevokeAttorney(\pi_{i'\to j})$ 委任関係 $u_i \to u_j$ を失効する. $Cloud$ に委任状 $\pi_{i'\to j}$ が格納されているか確認し, 存在するならば $Cloud$ 上から削除する.

5.3.1　擬似的な Non-transitive

このシステムでは $u_{i\to j}$ かつ $u_{j\to k}$ のとき, クラウドに格納された再暗号化鍵 $\pi_{i'\to i}, \pi_{j'\to j}, \pi_{k'\to k}, \pi_{i'\to j}, \pi_{j'\to k}$ から $\pi_{i'\to k}$ を求めることはできない.

6　実験

実験環境を表 1 に示す. 実装には IBM が公開している完全準同型暗号ライブラリである HElib[12] を用いた. これは Brakerski ら [5] の完全準同型暗号を実装したものである（よって $ReEncrypt$ には $Recrypt$ ではなく $SwitchKey$ を用いる）. HElib で使用したパラメータを表 2 に示す. 実験は全て同一計算機上で行った. よってユーザ・クラウド間の通信などのオーバヘッドは考慮していない.

6.1 節, 6.2 節の実験は 5.2 節のシステム構成にて, 6.3 節の実験は 5.3 節のシステム構成にて行った.

6.1　鍵サイズと計算時間の計測

6.1.1　実験内容

異なる鍵ペア (sk_i, pk_i) を持つ複数のユーザ $u_i (i = 0, ..., N)$ について, 委任関係 $u_i \to u_0 (i = 1, ..., N)$ を定義する. 各ユーザ u_i が暗号化した暗号文 $c_i = Encrypt(pk_i, m_i)$ に対し, ユーザ u_0 が $\sum_{i=1}^{N} c_i$ と $\prod_{i=1}^{N} c_i$ を計算・復号する. ユーザ数 N を変化させ, ユーザ u_0 とクラウドが管理する鍵のデータサイズと, 計算にかかる時間を計測する.

また, 比較対象として, 各ユーザ $u_i (i = 1, ..., N)$ の秘密鍵 sk_i を u_0 に直接譲渡した場合にユーザ u_0 が管理する秘密鍵のデータサイズと, 全てのユーザ u_i が同じ鍵ペア (sk, pk) を使用した場合（すなわち $c_i = Encrypt(pk, m_i)$ ）の計算時間を測定した.

6.1.2　実験結果

ユーザ u_0 とクラウドが管理する鍵のデータサイズを図 1 に示す. 秘密鍵 sk_i を u_0 に直接譲渡する場合, ユーザ u_0 は複数の秘密鍵 $\{sk_i\} (i = 0, ..., N)$ を管理しなければならないのに対し, 提案手法ではユーザ u_0 は自身の秘密鍵 sk_0 のみを管理すれば良い. よってユーザ u_0 の鍵管理コストが軽減される. 一方, 提案手法ではクラウドは再暗号化鍵 $\pi_{i\to 0} (i = 1, ..., N)$ を管理する必要がある. よって提案手法では, ユーザ u_0 の負荷が軽減したのに対し, クラウドの負荷が増加した.

しかし, クラウドはユーザと比べて十分な計算能力とストレージを有している. また, ユーザ u_0 から秘密鍵 sk_i が漏洩した場合は秘密情報 m_i の漏洩につながるが, 万が一クラウドから再暗号化鍵 $\pi_{i\to 0}$ が漏洩しても秘密情報 m_i は漏洩しない. よって, クラウドの負荷の増大よりもユーザの負荷軽減の方が利点が多い.

$\sum_i c_i$ と $\prod_i c_i$ の計算にかかる時間を図 2 に示す. 同じ鍵ペアを使用した場合と比べ, 提案手法の方が加算・乗算にかかる時間が長くなる. これは, 通常の暗号文同士の加算・乗算に加え, $ReEncrypt$ の処理を行うためである.

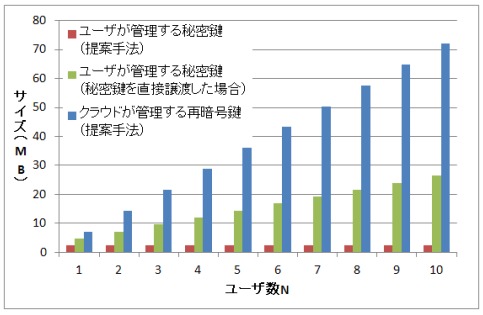

図 1　ユーザとクラウドが管理する鍵のサイズ

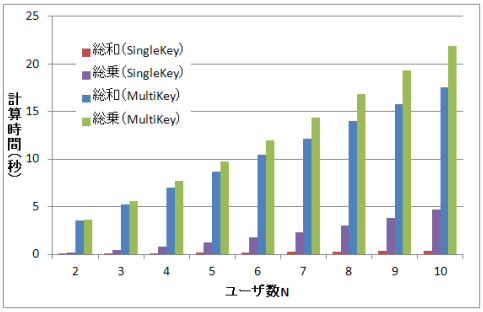

図 2　単一/複数の鍵の計算時間の比較

6.2　多段階の $ReEncrypt$ について

6.2.1　実験内容

Non-transitive を満たす必要のないシステムを仮定する. 例として役職による復号権限の委任がある. 部下は上司へ復号権限を委任するものとする. 主任 u_j は部下

である社員 u_i から復号権限を委任され，上司である課長 u_k へ復号権限を委任する．このとき，社員は課長の部下であるから，課長は社員から復号権限を委任される．よって，再暗号化鍵 $\pi_{i\to j}, \pi_{j\to k}$ から $\pi_{i\to k}$ が生成されることは問題ない．

上司 u_k が社員 u_i の暗号文を復号する方法に，再暗号化鍵 $\pi_{i\to j}, \pi_{j\to k}$ に対して 2 回 *ReEncrypt* を行う方法と，予め $\pi_{i\to k}$ を作成し 1 回のみ *ReEncrypt* を行う方法（以降，ショートカットと呼ぶ）がある（図 3）．

多段階の委任関係 $u_1 \to u_2, u_2 \to u_3, ..., u_{N-1} \to u_N$ について，$\pi_{1\to 2}, \pi_{2\to 3}, ..., \pi_{N-1\to N}$ を用いて複数回の *ReEncrypt* を行う場合と，全ての委任関係に関して予めショートカット $\pi_{1\to 3}, ..., \pi_{1\to N}, \pi_{2\to 4}, ..., \pi_{N-2\to N}$ を作成する場合について，クラウドが管理する再暗号化鍵のデータサイズと，復号にかかる時間を計測した．

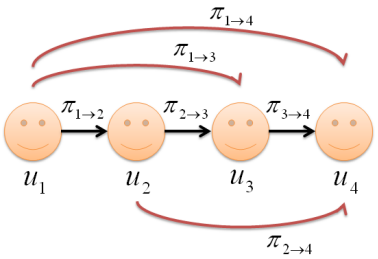

図 3　ショートカットの例

6.2.2　実験結果

クラウドが管理する再暗号化鍵のサイズを図 4 に，復号処理にかかる時間を図 5 に示す．ショートカット無しの場合，権限委任の段数に比例して *ReEncrypt* の回数が増加し，復号にかかる時間が長くなる．一方，ショートカットを作成した場合，クラウドが管理する再暗号化鍵の個数は増えるが，*ReEncrypt* の回数は 1 回で済むため，復号にかかる時間は常に均一になる．

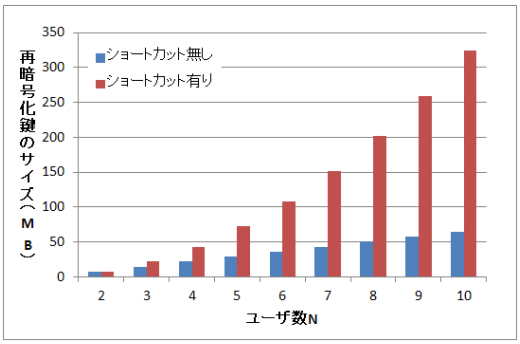

図 4　クラウドが管理する再暗号化鍵のサイズ

6.3　拡張したシステム構成について

6.3.1　実験内容

5.3 節のシステム構成に対して 6.1 節と同様の実験を行った．このとき，5.2 節と 5.3 節のシステムについて，クラウドが管理する再暗号化鍵のデータサイズと，計算にかかる時間を比較した．

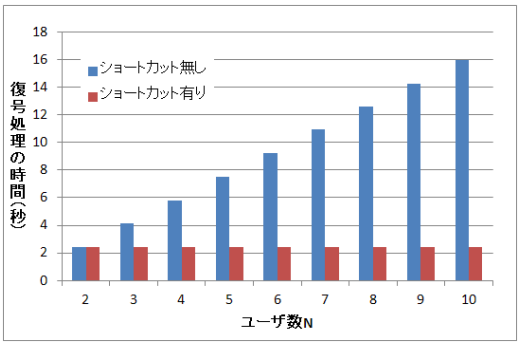

図 5　復号処理にかかる時間

6.3.2　実験結果

クラウドが管理する再暗号化鍵のデータサイズを図 6 に示す．5.3 節のシステムの方が 5.2 節のシステムと比べて *Cloud* の管理する再暗号化鍵のデータサイズが大きい．

また $\sum_{i=1}^{N} c_i$ と $\prod_{i=1}^{N} c_i$ の計算にかかる時間を図 7 に示す．結果，5.3 節と 5.2 節のシステムで，計算時間に大きな差は生じなかった．

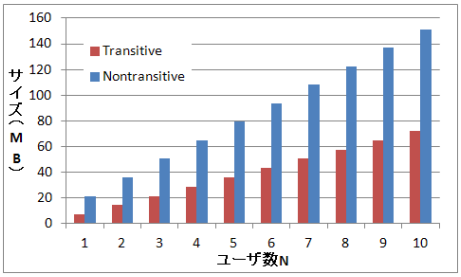

図 6　クラウドが管理する再暗号化鍵のサイズ

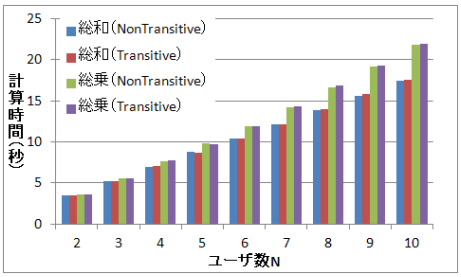

図 7　計算時間の比較

7　評価と考察

7.1　セキュリティ

7.1.1　提案手法の安全性

5.1 節で述べたとおり，提案手法は特定の暗号系に依存しない．また，提案手法は既存の完全準同型暗号を応用したシステムの提案であり，使用する暗号系に対して変更は加えない．よって，提案手法の安全性は実際に使用する完全準同型暗号の安全性に準拠する．

7.1.2　委任関係の循環定義

定義 7.1 *circular security* ある暗号系において，鍵ペア $\{(sk_1, pk_1), ..., (sk_n, pk_n)\}$ と定数 S に対し，$Enc(pk_2, sk_1), ..., Enc(pk_n, sk_{n-1}), Enc(pk_1, sk_n)$ と

$Enc(pk, S)$ を区別できないとき，この暗号系は *circular security [13]* を満たす．

$u_1 \to u_2 \to ... \to u_n \to u_1$ のように委任関係を循環して定義した場合を考える．このとき $\pi_{1\to2}, \pi_{2\to3}, ..., \pi_{n\to1}$ が存在する．FHE-PRE$_{\mathrm{Recrypt}}$ の場合，定義より $\pi_{i\to j} = Encrypt(pk_j, sk_i)$ なので，実際に使用する完全準同型暗号が circular security を満たさない場合，攻撃者は平文や秘密鍵に関する何らかの (部分) 情報を知ることが出来る．

そのため，実装に circular security を満たさない暗号系を用いる場合，循環する委任関係を定義すると，システムの安全性が低下する．

7.2 復号権限の委任の取り消し

既存の完全準同型暗号でも，ユーザが別のユーザへ秘密鍵を送ることで復号権限を委任できる．しかし 3.3 節で述べたとおり，復号権限の委任を取り消すには，委任元の鍵ペアの再生成と，関連する全ての暗号文の再暗号化を伴う．特に，クラウド上で膨大なサイズの暗号文を扱う場合，それらの暗号文全てに対して再暗号化を行う必要がある．

一方，提案手法ではクラウド上の再暗号化鍵を削除することで，復号権限の委任を容易に取り消すことができる．このとき，委任元の鍵ペアの再生成や，関連する暗号文の再暗号化は必要ない．

8 今後の課題

5.3 節の擬似的な Non-transitive を満たすシステム構成では，クラウドが管理する再暗号化鍵のサイズが大きくなった．今後はより効率的に Non-transitive を満たす手法を検討する．

また，提案手法は Collusion safe と Non transferable を満たさない．今後はこれらを満たす手法を検討する．

また，今回の実験は全て同一計算機上で行ったため，ユーザ・クラウド間の通信などは考慮していない．今後はより実際の利用環境に近い状態で実験を行う．

9 おわりに

PRE を用いた権限委任に基づくマルチユーザ向け完全準同型暗号を構築した．既存研究と比べて提案手法はユーザの管理する鍵のデータサイズが小さく，また柔軟な復号権限の委任関係の設定が可能である．

参考文献

[1] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer Berlin Heidelberg, 2005.

[2] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA, 2009.

[3] Marten Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer Berlin Heidelberg, 2010.

[4] David Naccache, Jean-Sebastien Coron, Avradip Mandal, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference*, volume 6841 of *Lecture Notes in Computer Science*, page 483. Springer, 2011.

[5] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer Berlin Heidelberg, 2011.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011.

[7] Liangliang Xiao, Osbert Bastani, and I-Ling Yen. An efficient homomorphic encryption protocol for multi-user systems. *IACR Cryptology ePrint Archive*, 2012:193, 2012.

[8] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the 44th Symposium on Theory of Computing*, STOC '12, pages 1219–1234, New York, NY, USA, 2012.

[9] Masahiro MAMBO and Eiji OKAMOTO. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts (special section on cryptography and information security). *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 80(1):54–63, jan 1997.

[10] M. BLAZE. Divertible protocols and atomic proxy cryptography. *Proc. EUROCRYPT'98*, pages 127–144, 1998.

[11] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, February 2006.

[12] Helib - github. https://github.com/shaih/HElib. Accessed: 2013-12-02.

[13] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, pages 93–118, London, UK, 2001.