# 完全準同型暗号を用いたクラウド上の演算機能の実装

2011SE111 神邊 智貴　2011SE234 佐脇 大貴　2011SE272 坪井 将洋
指導教員:　河野 浩之

## 1　はじめに

クラウドコンピューティング (以下クラウド) は様々な用途で用いられている．その例として，企業が提供するクラウドサービスに対して，利用者はデータのバックアップやストレージの利用，計算の依頼など，ネットワークを介して様々なサービスを受けることができる．クラウド利用の問題点として，(1) クラウドに預けられた平文のデータはクラウド提供者側からは閲覧可能であるというプライバシーの問題や，(2) ハッキングや社内スパイによってデータが漏れてしまうセキュリティの問題などがある．これらを解決する従来の暗号方式 [4] に，RSA 暗号や ElGamal 暗号，Paillier 暗号 [1] などが広く用いられているが，これらの暗号方式でデータを暗号化することによって，二つの暗号文の演算が困難になる問題もある．

そこで，本研究ではクラウドにデータを送信する前に，完全準同型暗号 (Fully Homomorphic Encryption) [5] と呼ばれる公開鍵暗号方式を用いて暗号化することで (1)(2) の問題点を改善し，より安全なデータの受け渡しを可能にするシステムを実装する．完全準同型暗号はデータを暗号化したままでも加算や乗算などの演算が可能な暗号方式であるため，データをクラウドに預けても従来と同じように計算などの依頼が可能である．また先行研究 [2][3] を参考に実装し評価を行う．本論文は 6 章で構成され，2 章では準同型暗号を用いた先行研究について述べる．3 章では完全準同型暗号を用いたシステムの概要について述べる．4 章では医療データを用いた完全準同型暗号の実装について述べる．5 章では本研究で実装したシステムの評価を述べる．6 章ではまとめを述べる．

## 2　準同型暗号を用いた先行研究の比較

本章では，2.1 節で準同型暗号の概要を，2.2 節で加法準同型暗号を用いたクラウドへの書き込みと索引付けの評価を，2.3 節で準同型暗号を用いたデータの受け渡しを述べる．

### 2.1　準同型暗号の説明

本節では，暗号の種類と特性を比較し，本研究で扱う完全準同型暗号を用いる利点を述べる．

表 1　現代暗号の種類

| 分類 | 例 | 性質 |
|---|---|---|
| 共通鍵暗号 | DES(ブロック暗号) | |
| | RC4(ストリーム暗号) | |
| 公開鍵暗号 | RSA 暗号 | 乗法準同型性 |
| | ElGamal 暗号 | 乗法準同型性 |
| | Paillier 暗号 | 加法準同型性 |
| | 完全準同型暗号 | 加法準同型性<br>乗法準同型性 |

暗号を用いる利点として，データをクラウドに送信する前に平文データを暗号化処理してからクラウドへと預けることで，クラウド管理者などの第三者によって内容を読まれるのを防ぐことが可能になる．そのためプライバシー問題やセキュリティ問題を大きく改善する．また，公開鍵暗号は暗号化と復号に異なる鍵を使用するため，鍵自体の受け渡し過程で鍵自体の情報が漏れるリスクを解決する．表 1 の公開鍵暗号うち，RSA, ElGamal, Paillier 暗号が現在用いられている暗号方式として広く知られているが，これらは暗号化した状態では乗法または加法のどちらか一方しか演算できないとされる．本研究で扱う完全準同型暗号は暗号化した状態のまま加算や乗算の両方の演算が可能であるため，データを暗号化してクラウドへ預けても，クラウドのサービスは従来とほぼ同じように受けられる利点がある．ただし平文や従来の暗号方式での計算よりも計算処理時間が長くなる問題がある．

### 2.2　クラウドへの書き込みと索引付けの評価 [1]

2.1 節で述べた準同型暗号のうち，加法準同型暗号の暗号方式を用いて，クラウドへのデータ書き込みと索引付けの評価を行う文献 [1] の概要を述べる．加法準同型性を持つ Paillier 暗号によって私的データを暗号化する．私的データには様々な言語で書かれた Wiki の平文データを用いる．また暗号化する際，平文の Wiki のキーワードに索引付けされているのと同様に，暗号化状態でもキーワードが対応するように索引付けして暗号化する．暗号化したデータはクラウドへ預け，検索する際に入力したキーワードを暗号化して検索することで，クラウド上の暗号データから暗号状態のまま検索することを可能にする．検索結果は各 PC で復号して平文の Wiki データを閲覧できる．暗号化した Wiki データから検索するのにかかった時間は 1 秒を切っている．

### 2.3　クラウドを介したデータの受け渡し [2]

本節では，公開鍵を用いた加法準同型暗号の暗号方式による医療センターから各クライアントへの医療データの受け渡しを効率的かつ安全に行う [2] の方式を示す．この研究の概要として，まず各クライアント $(a, b)$ に対応する (公開鍵，秘密鍵) のペア $(pka, pra)$ を作成しておく．次に，(1) 医療センターは公開鍵を用いてデータを暗号化することで，暗号化したデータをクラウドに保存する．また，(2) 医療センターはあらかじめ作成していた秘密鍵をクライアント a，b へ受け渡す．(3) 各クライアントは復号許可印 $(T_a, T_b)$ がクラウドに認知されることで，暗号文のデータをクラウドから受け取ることができ，(4) 医療センターから受け取っていた秘密鍵 $pra$，$prb$ を用いることで暗号化されたデータを復号できる．

## 3　完全準同型暗号を用いた医療クラウドの提案

本章では，3.1 節で本研究の提案内容を述べ，3.2 節でデータの暗号化プロセスについて述べ，3.3 節で暗号化したデータを復号するプロセスを述べる．

### 3.1　本研究の提案内容

重要なデータをクラウドに預けるにあたって，セキュリティの安全性への心配が重要な課題である．そのためにクライアント各自で暗号化してクラウドへデータを預けることが最も安全な方法であるが，従来の RSA などの暗号方式で暗号化してしまうと暗号状態でのデータの計算ができない問題があり，クラウドサービスの利便性が損なわれてしまう．そこで本研究では，完全準同型暗号による暗号方式を用いることで，医療クラウドを介した安全で効率的な医療データの受け渡しシステムを構築し，暗号状態でもクラウド上で演算可能なプログラムを実装する．

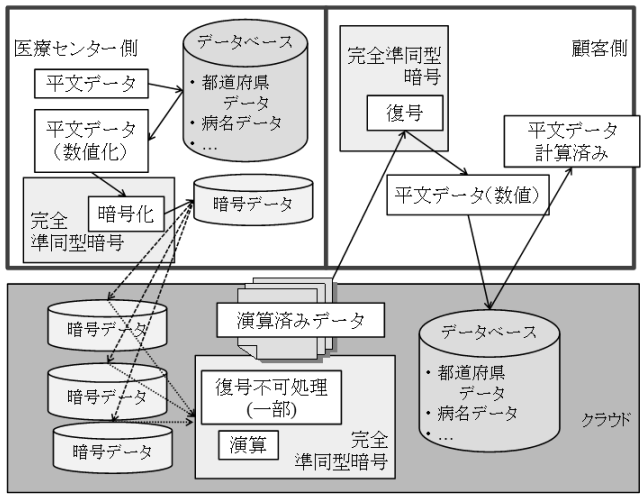


図 1　完全準同型暗号を用いた医療クラウドのアーキテクチャ

図 1 は，実装するシステムのアーキテクチャである．まず，医療側でデータを暗号化するために，漢字仮名交じりの平文データを全て数値化する．数値化された平文データは医療センター側で完全準同型暗号を用いて暗号化する．暗号化されたデータはクラウドへアップロードし格納する．完全準同型暗号によって暗号化されたデータは暗号状態でも加算や乗算が可能であるため，復号する前にデータを演算することで，様々な復号結果を得ることができる．顧客側では，クラウドから演算済みの暗号データをダウンロードすることができ，手元の PC で復号可能となる．復号したデータは数値化されているため，データベースから各数値に対応する文字データと置き換えることで，元の平文データによる復号結果が得られるシステムを作成する．

### 3.2　データの暗号化と鍵の受け渡しのプロセス

クライアントそれぞれに対応する公開鍵 $pk$，秘密鍵 $pr$ のペアを作る．クライアント a の場合，(公開鍵，秘密鍵) $= (pka, pra)$ となる．同様にしてクライアント b，c の鍵 $(pkb, prb)$，$(pkc, prc)$ も生成する．次にそれぞれの公開鍵で暗号化関数 $E_{pk}$ を使いデータ $d$ を暗号化し，暗号化済みのデータ $E_{pk}(d)$ をクラウドに保存する．データ $d$ には n 個の属性があるとし，それを $d = (d_1, d_2, \cdots, d_n)$ と表す．クライアント a のデータを $E_{pk}(d)$ によって暗号化するための関数を式 (1) のように定義する．

$$E_{pka}(d) = E_{pka}(d_1,\ d_2,\ \cdots, d_n) \tag{1}$$

図 2 のように，医療センターが持っている医療データを鍵 $pk$ による暗号化関数 $E_{pk}(d)$ によって暗号化し，種類の異なる秘密鍵 $pra$，$prb$，$prc$ をそれぞれのクライアントに預ける．

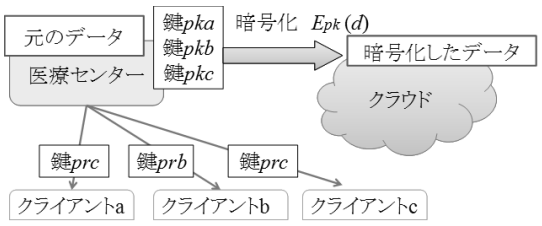


図 2　暗号化のプロセス

### 3.3　復号化のプロセス

本節では，3.2 節で暗号化済みのクラウド上のデータから，秘密鍵を用いて復号化するプログラムを記述する．クライアントはクラウドから暗号化されたデータをダウンロードし，自らが持つ秘密鍵を用いて復号化関数 $D_{pr}$ によってデータを復号化する．また，クライアント a のデータを鍵 $pra$ を用いて復号するための関数を式 (2) のように定義する．

$$\begin{aligned} D_{pra}\{E_{pka}(d)\} &= D_{pra}\{E_{pka}(d_1,\ d_2,\ \cdots, d_n)\} \\ &= (d_1,\ d_2,\ \cdots, d_n) \end{aligned} \tag{2}$$

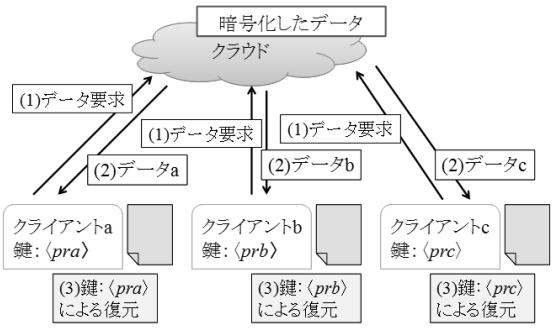


図 3　復号化のプロセス

図 3 は復号化のプロセスである．まず，(1) 各クライアントは必要な医療データをクラウドに要求する．次に，(2)

クラウド側はそれぞれの要求に応じて，暗号化された医療データを各クライアントへ送信する．そして，(3) クライアント側ではあらかじめ医療センターから受け取っていた秘密鍵を用いることで，復号化関数 $D_{pr}$ により必要なデータを復号できる．

## 4 完全準同型暗号による実装

本章では，3 章で示した提案を実装するためのプロセスを述べる．4.1 節で暗号化プログラム実装の手順について述べ，4.2 節で HElib を用いたアルゴリズムについて，4.3 節で暗号化について，4.4 節で暗号文での演算について，4.5 節で復号結果について述べる．

### 4.1 暗号化プログラムの実装の手順

表 2 実験に用いた医療データ

| 患者名 | 年齢 | ID | 住所 | 病名 | 血糖値 | 血圧 |
|---|---|---|---|---|---|---|
| 神邊 | 22 | 110 | 愛知 | 骨折 | 123 | 120/80 |
| 本田 | 10 | 432 | 北海道 | 喘息 | 100 | 128/87 |
| 香川 | 40 | 921 | 香川 | 糖尿 | 205 | 145/92 |
| 内田 | 65 | 923 | 東京 | ガン | 91 | 138/94 |
| 大久保 | 29 | 716 | 東京 | 骨折 | 105 | 124/84 |
| 岡崎 | 80 | 012 | 愛知 | 骨折 | 100 | 170/73 |
| 吉田 | 75 | 384 | 東京 | ガン | 228 | 169/100 |
| 中田 | 59 | 721 | 東京 | ガン | 130 | 139/89 |
| 稲本 | 54 | 024 | 愛知 | 骨折 | 83 | 122/86 |

表 2 は実装に用いるデータである．まず，医療センター側で表 2 のデータを暗号化するために MySQL からデータを読み込み，漢字仮名交じりの平文データを全て数値化する．元の平文と数値との置き換えは，データベースにあらかじめ仮名表記部分の平文データを格納しておき，数値と対応付けすることで元の平文と数値を置き換える作業を行う．数値化された平文データは医療センター側で HElib を用いて暗号化する．暗号化されたデータは Ubuntu 環境によって構築したクラウドへアップロードし格納する．さらに，暗号文どうしの計算をするプログラムをクラウド上で動かすことで演算を行う．顧客側では，クラウドから演算済みの暗号データをダウンロードし手元の PC で復号できる．復号したデータは数値化されているため，データベースから各数値に対応する文字列データと置き換えることで，元の平文データによる復号結果が得られるシステムを作成する．各顧客のアカウント管理は WordPress で行う．

### 4.2 HElib

今回，完全準同型暗号を用いたプログラムを実装するにあたって，IBM 社の提供する HElib(Homomorphic Encryption library) を用いる．HElib は IBM が開発したオープンソースで，完全準同型暗号を C++で実装している．HElib は IBM 社の研究者である Craig Gentry 氏が 2009 年に発表した論文 [5] に基づき実装されている．HElib は https://github.com/shaih/HElib からダウンロードできる．図 4 は HElib 内のプログラムの一部である．L の値を変えることで暗号のセキュリティレベルを変更することが出来る．秘密鍵と公開鍵は 11 から 17 行目で生成され，p，r の値を変更することで鍵の種類を変更することができる．

```
 1:  long m=0, p=2, r=16; // Native plaintext space
                          // Computations will be 'modulo p'
 2:  long L=16;           // Levels
 3:  long c=3;            // Columns in key switching matrix
 4:  long w=64;           // Hamming weight of secret key
 5:  long d=0;
 6:  long security = 128;
 7:  ZZX G;
 8:  m = FindM(security,L,c,p, d, 0, 0);
 9:  FHEcontext context(m, p, r);
10:  buildModChain(context, L, c);
11:  FHESecKey secretKey(context);
12:  const FHEPubKey& publicKey = secretKey;
13:  G = context.alMod.getFactorsOverZZ()[0];
14:  secretKey.GenSecKey(w);
15:  EncryptedArray ea(context, G);
16:  long nslots = ea.size();
```

図 4 HElib に実装されているプログラム

### 4.3 データの暗号化

データを暗号化するプログラムを図 5 に示す．1 から 8 行目でファイルから読み込んだデータを v1 に格納し，9 行目の ea.encrypt 関数で鍵 publicKey によって平文 v1 を ct1 に暗号化している．10 から 16 行目でファイル名を指定してファイルに出力する．

```
 1:  ifstream fin(input_file);
 2:if(!fin){
 3:    cout << "入力ファイルをオープンできません" << endl;
 4:    return 1;}
 5:while(fin.getline(plain,sizeof(plain))){
 6:  int t=atoi(plain);
 7:      v1.push_back(t);}
 8:  Ctxt ct1(publicKey);
 9:  ea.encrypt(ct1, publicKey, v1);
10:  char cryfile[100];
11:  cout << "暗号化ファイルの名前を入力してください" << endl;
12:  cin >> cryfile;
13:  ofstream fout(cryfile);
14:  if(!fout){
15:    cout << "出力ファイルをオープンできません" << endl;
16:    return 1;}
17:  fout << ct1 << endl;
18:  fout.close();
19:  cout << "Complete" << endl;
20:  return 0;
21:}
```

図 5 暗号化プログラム

### 4.4 暗号文での演算

本節では暗号状態で演算するためのプログラムを記述する．まず，健康管理人がデータを復号する際に，データ成分の一部を復号できないようにするプログラムである．表 2 のうち，データ一列分の成分を $D$ とし $D = \{d_1,\ d_2,\ \ldots\ , d_7\}$ とする．データ $D$ を暗号化したものを $E = \{d'_1,\ d'_2,\ \ldots\ , d'_7\}$ とする．健康管理人の場合，ID に値する暗号データ $d'_3$ に，暗号化した“ 0 ”を掛け，$d'_3$ 以外の属性には“ 1 ”を暗号化したもの掛けることで，$d'_3$ の部分のみが復号不可となる．ID の部分のみを伏せるための平文デー

タ $D_0 = \{1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1\}$ とし，$D_0$ の暗号化データを $E_0 = \{1',1',0',1',1',1',1'\}$ とする．ここで暗号状態での掛け算 $E \cdot E_0 = \{d'_1 \times 1',\ d'_2 \times 1',\ d'_3 \times 0',\ d'_4 \times 1',\ d'_5 \times 1',\ d'_6 \times 1',\ d'_7 \times 1'\} = \{d'_1,\ d'_2,\ 0',\ d'_4,\ d'_5,\ d'_6,\ d'_7\}$ を行う．$E \cdot E_0$ を復号すると $\{d_1,\ d_2,\ 0,\ d_4,\ d_5,\ d_6,\ d_7\}$ という結果が得られる．

次に各属性の合計を計算する．データの取り出し方は上の方法と同様の演算を行い，必要な部分のみに暗号状態の“ 1 ”を掛け，それ以外には“ 0 ”を掛ける．例えば表 2 のうち，血糖値の合計を計算するプログラムの場合，各患者のデータの暗号文を $E_1,\ E_2,\ E_3,\ \ldots,\ E_n$ とし，$E_0 = \{0',\ 0',\ 0',\ 0',\ 0',\ 1',\ 0'\}$ をそれぞれに掛けた値を $E_1 \cdot E_0,\ E_2 \cdot E_0,\ \ldots,\ E_n \cdot E_0$ とする．ここで $E_1 \cdot E_0$ から $E_n \cdot E_0$ をすべて足すと $\{0',\ 0',\ 0',\ 0',\ 0',\ d'_{16} + d'_{26} + d'_{36} + \ldots + d'_{n6},\ 0'\}$ となり，これを復号すれば血糖値の部分のみ合計値を得る．

### 4.5 復号結果

表 3 はユーザである患者がデータを復号した結果である．これにより，クラウドに預けた暗号化したデータから復号したデータを得られ，患者のプライバシーを保護しながらクラウドを介したデータ共有が可能となる．

表 3 患者による復号

| 患者名 | 年齢 | ID | 住所 | 病名 | 血糖値 | 血圧 |
|---|---|---|---|---|---|---|
| 吉田 | 75 | 384 | 東京 | ガン | 228 | 169/100 |

表 4 は健康管理人がデータを復号した結果である．患者のプライバシーを守るために，患者の ID を復号できなくする処理をクラウド上で行うことにより，データを健康管理人が復号しても ID の部分は表示されない．

表 4 健康管理人による復号

| 患者名 | 年齢 | ID | 住所 | 病名 | 血糖値 | 血圧 |
|---|---|---|---|---|---|---|
| 神邊 | 22 | - | 愛知 | 骨折 | 123 | 120/80 |
| 香川 | 40 | - | 香川 | 糖尿 | 205 | 145/92 |
| 吉田 | 75 | - | 東京 | ガン | 228 | 169/100 |

表 5 は役所が復号した結果である．役所は自分の地域の病気傾向を把握する事が目的である．そのため，役所などの組織が医療データを要求する場合，不特定多数の人がデータを閲覧する可能性がある．よって必要な部分の計算はクラウドに依頼し，計算結果のみ復号できる処理を行う．

表 5 役所による復号

| 病名 | 合計値 |
|---|---|
| 血糖値 | 1165 |
| 血圧最大値 | 1225 |
| 血圧最小値 | 785 |

## 5 実装したプログラムの性能と評価

本章では，実装したシステムの結果を評価する．今回，完全準同型暗号によるデータ保護を HElib によって実装した結果，属性が 8 つの場合，暗号化したデータの容量は平均約 2.3Mbyte で平文データの約 1454 倍となった．また暗号化にかかった時間はセキュリティレベルが 16 の場合平均 44.06 秒で，暗号データをクラウドへアップロードするのにかかる時間は約 0.30 秒であった．

厚生労働省の調査によると平成 25 年中における全国の病院の 1 日平均在院患者数は 1,275,347 人である．そのことを考慮すると，データ容量は 2.8TByte 以上となることが予想される．また暗号化にかかる平均処理時間は，1 日に約 15587 時間以上かかり，クラウドへアップロード又はクラウドからダウンロードするのにかかる時間は約 106 時間以上である．

## 6 まとめ

本研究では，完全準同型暗号を用いることで，暗号状態での演算をクラウド上で行うシステムを HElib，MySQL，WordPress を使用して実装した．医療データの暗号化，クラウド上で暗号データの演算，ユーザ側での復号を正常に動作することを確認した．性能測定結果として，HElib によって暗号化した暗号データの容量は平文データのの約 1454 倍，クラウドとの平均通信時間は約 0.30 秒，暗号状態での平均計算時間は約 44 秒という結果を得た．

外来患者数の一日平均が約 3000 人を越える東京大学医学部附属病院の場合，一日に訪れる外来患者のデータを全て暗号化するのに約 36.7 時間以上かかる．今日実験に用いた市販 pc のスペックで実装したシステムを実行させると，業務面で支障をきたすことが予想される．5 から 10 年後の性能が向上した市販 pc では，実行時間に関して支障をきたすことなく実行できるシステムといえる．

## 参考文献


[1] Bermardo Ferreira, Henrique Domingos, “ Searching Private Data in a Cloud Encrypted Domain, ”Proc. of the 10th Conference on OAIR, pp.165-172, 2013.

[2] Bharath K. Samanthula, Gerry Howser, Yousef Elmehdwi, and Sanjay Madria,“ An Efficient and Secure Data Sharing Framework using Homomorphic Encryption in the Cloud, ”in Cloud 1st conference by ACM, ISSN: 978-1-4503, DOI: 1596-8/12/08, 2012.

[3] Tatiana Emakova, Benjamin Fabian, “ Secret Sharing for Health Data in Multi-provider Clouds, ”Proc. of IEEE International Conference on Business Informatics, pp.93-100, 2013.

[4] 一色寿幸, 尾花賢, 森健吾, “ 準同型暗号を用いたプライバシー保護統計演算のソフトウエア実装報告, ”電子情報通信学会技術研究報告. ISEC, 情報セキュリティ Vol.111(455), pp.135-140, 2012-02-23.

[5] Craig Gentry, “ A fully homomorphic encryption scheme, ”PhD thesis, Stanford University, 2009.