**SCIS 2025** 2025 Symposium on
Cryptography and Information Security
Kokura, Japan, Jan. 28 – 31, 2025
The Institute of Electronics,
Information and Communication Engineers

# 準同型暗号 CKKS 方式における暗号文のランダム逆行列の考察
# Computing the random inversion matrix using homomorphic encryption CKKS scheme

磯川 楽丸 * 若杉 飛鳥 † 服部 大地 † 小寺 雄太 *
Tanoshimaru Isokawa Asuka Wakasugi Daichi Hattori Yuta Kodera
野上 保之 *
Yasuyuki Nogami

**あらまし** データの漏洩リスクを低減し，プライバシーを確保しつつ機密情報を解析する方法の 1 つとして，準同型暗号が注目されている．準同型暗号では，いくつかの方式が実現されており，その内の 1 つに CKKS 方式が知られている．いくつかの機械学習アルゴリズムでは，与えられた正定値対称行列に対して，逆行列を求めるサブルーチンが用いられる．cleartext 状態で逆行列を求めるアルゴリズムとして，ガウス消去法が知られているが，加算と乗算のみを扱える準同型暗号文状態では，同様のアルゴリズムを実現することは難しい．本稿では，ガウス消去法ではなくニュートン法をベースとして，準同型暗号 CKKS 方式における暗号文の逆行列の算出アルゴリズムを提案する．これらの実装を行なったのち，単位行列およびランダム正定値対称行列の逆行列を計算した際の実行時間を計測する．また，cleartext 状態で算出した逆行列との誤差評価を行なった結果として，誤差が大きい場合がみられた．よって，実際には，ランダム正定値対称行列にはいくつかの制約が存在すると考えられるため，本稿では，それらの議論を展開する．

**キーワード** 準同型暗号，CKKS，逆行列

## 1 はじめに

昨今，様々なデータの活用が急速に進むなか，自身の会社や組織内のデータ解析を外部へ委託することで，知見を得られる場合がある．しかし，医療や金融などの分野では，機密性の高いデータも含まれるため，そのまま外部へ共有することが容易ではない．そこで，データの漏洩リスクを低減し，プライバシーを確保しつつ機密情報を解析する方法が求められている．その 1 つとして注目されている準同型暗号は，暗号化されたデータに対して直接計算を施す事ができ，復号せずに演算結果を得られる特性を持つ．これにより，外部組織へ機密データを直接共有することなくデータ分析が可能となり，外部からの知見を得ることができる．

### 1.1 準同型暗号

準同型暗号とは，その暗号文を復号することなく，加算や乗算などの演算を実行できる暗号方式である．準同型暗号における暗号文空間内で定まる演算は，平文に対して定まる演算と同一である．例えば，暗号文に対して加法と乗法が定まっている準同型暗号であれば，平文でもそれぞれの演算を実行することが可能である．つまり，このような準同型暗号に対して，暗号化関数を Enc とすると，任意の平文 $A, B$ に対して以下の式が成り立つ:

$$\mathrm{Enc}(A) + \mathrm{Enc}(B) = \mathrm{Enc}(A + B)$$
$$\mathrm{Enc}(A) \times \mathrm{Enc}(B) = \mathrm{Enc}(A \times B)$$

格子暗号方式とは，LWE 問題 [12] と呼ばれる計算量困難性を安全性の基盤としている暗号方式であり，多くの準同型暗号方式は，格子暗号をベースとしている．例えば，2009 年に Craig Gentry らが提案したイデアル格子暗号を用いた方式 [7]，2011〜12 年に構成された BGV 方式 [3] や BFV 方式 [6]，2016 年に発表された CKKS 方式 [4] などが存在する．

* 岡山大学大学院 環境生命自然科学研究科, 〒 700-8530 岡山県岡山市北区津島中 3-1-1. Graduate School of Environmental, Life, Natural Science and Technology, Okayama University, 3-1-1 Tshushima-naka, Kita-ku, Okayama-shi, Okayama, 700-8530, Japan.
† EAGLYS 株式会社, 〒 151-0051 東京都渋谷区千駄ヶ谷 5 丁目 27-3 やまとビル 7F. EAGLYS Inc., 7F Yamato building, 5-27-3 Sendagaya, Shibuya-ku, Tokyo, 151-0051, Japan.

### 1.2 先行研究

逆行列の算出にはガウス消去法が用いられることがあり，そのサブルーチンでは，比較演算が用いられる．一方で，準同型暗号で暗号化された空間上では，加算や乗算などの基本的な演算のみを仮定することがある．よって，このような状況下では，準同型暗号を用いてガウス消去法を実現することは困難である．そのため，逆行列の算出にはニュートン法 [2] を用いたアルゴリズム [5, 8, 10, 11] を用いることが一般的である．

### 1.3 構成

本稿は次のように構成される．まず，第 1 章では，準同型暗号について概説し，ニュートン法を用いた逆行列の算出法に関する先行研究を紹介した．第 2 章では，準同型暗号 CKKS 方式の詳細について述べる．第 3 章では，ニュートン法を用いた逆行列算出アルゴリズムをいくつか紹介し，近似解を得るまでの反復回数について議論する．本稿で，どの逆行列算出アルゴリズムを用いるかについても説明する．第 4 章では，そのアルゴリズムを CKKS 暗号文上で実現するためのアルゴリズムを与え，その詳細を述べる．第 5 章では，提案手法を用いて，暗号文の逆行列を算出し，誤差および実行時間の計測を行い，結果を考察する．最後の第 6 章で，結論を述べる．

### 1.4 貢献

本稿では，ニュートン法をベースとして，cleartext 及び準同型暗号 CKKS 方式における暗号文の逆行列の算出アルゴリズムを提案する．cleartext で既存手法と提案手法の実装を行ない，逆行列に収束するまでの反復回数を比較し，提案アルゴリズムがより高速に収束することを示す．また，提案アルゴリズムを暗号文上で実現させ，これを用いて暗号文状態の単位行列およびランダム正定値対称行列の逆行列を算出し，その実行時間と誤差を評価する．その結果，提案アルゴリズムに入力されるランダム正定値対称行列にはいくつかの制約が存在することが示唆された．

## 2 準備

本章では，準同型暗号 CKKS 方式についての基礎知識と，ニュートン法について述べる．

### 2.1 CKKS 方式

本節では，CKKS 方式の概要を説明する．分析対象データを cleartext という．また，暗号化対象データを plaintext（平文）と呼び，cleartext から plaintext への変換アルゴリズムを encode, その逆変換アルゴリズムを decode という．CKKS 方式で用いられる encode 方式には, Coefficients（以下 Coeff）方式と Slot 形式の 2 種類が知られている．Coeff 方式とは，Cheon らによる encode 方式 [4] である．本稿では，Coeff 形式の encode/decode アルゴリズムを $\mathrm{Encode}_{\mathrm{Coeff}}, \mathrm{Decode}_{\mathrm{Coeff}}$ とし，その平文を $p_{\mathrm{Coeff}}$ と表す．Slot 方式とは，cleartext に対して，フーリエ変換アルゴリズムを適応させてから，Coeff 方式を用いる encode 方式である．本稿では，Slot 形式の encode/decode アルゴリズムを $\mathrm{Encode}_{\mathrm{Slot}}, \mathrm{Decode}_{\mathrm{Slot}}$ とし，その平文を $p_{\mathrm{Slot}}$ と表す．平文空間は，整数係数多項式環の円分多項式による剰余環によって定められ，本稿では，その多項式の長さを $N$ とする．cleartext が 1 次元配列のデータ型であるとき，Coeff 方式及び Slot 方式いずれの場合でも，平文は長さ $N$ の多項式で表現される．本稿では，cleartext の配列の長さを $n$ で表し，$n \leq N$ とする．

#### 2.1.1 Coeff 形式による暗号文に対するアルゴリズム

秘密鍵を sk, 公開鍵を pk とし，$\mathrm{Encrypt}_{\mathrm{pk}}(p_{\mathrm{Coeff}})$ で定まる暗号文を $c_{\mathrm{Coeff}}$ とする．$c_{\mathrm{Coeff}}$ は長さ $N$ の整数係数多項式である．このとき，sk による復号関数を用いると，$\mathrm{Decrypt}_{\mathrm{sk}}(c_{\mathrm{Coeff}}) = p_{\mathrm{Coeff}}$ が成り立つ．$c_{\mathrm{Coeff}}$ に対して，次のアルゴリズムの組 Inner_ Product_Coeff が定義される．以下では，$c_{\mathrm{Coeff}}$ が定まる暗号文空間内での演算を考える．

- $\mathrm{Inner\_Product\_Coeff}(c_{\mathrm{Coeff}}, c_{\mathrm{Coeff}}) \mapsto c_{\mathrm{Coeff}}$
  Coeff 形式による暗号文を多項式乗算する．以降では，この演算を IPC と略記する．

#### 2.1.2 Slot 形式による暗号文に対するアルゴリズム

秘密鍵を sk, 公開鍵を pk とし，$\mathrm{Encrypt}_{\mathrm{pk}}(p_{\mathrm{Slot}})$ で定まる暗号文を $c_{\mathrm{Slot}}$ とする．$c_{\mathrm{Slot}}$ は長さ $N$ の整数係数多項式である．このとき，sk による復号関数を用いると，$\mathrm{Decrypt}_{\mathrm{sk}}(c_{\mathrm{Slot}}) = p_{\mathrm{Slot}}$ が成り立つ．$c_{\mathrm{Slot}}$ に対して，次の 4 つのアルゴリズムの組 (Add, Sub, Mul, Inner_Product_Slot) が定義される．以下では，$c_{\mathrm{Slot}}$ が定まる暗号文空間内での演算を考える．Add/Sub/Mul では，それぞれ，Slot 形式による暗号文を多項式加算/減算/乗算する．

- $\mathrm{Inner\_Product\_Slot}(c_{\mathrm{Slot}}, c_{\mathrm{Slot}}, n) \mapsto c_{\mathrm{Slot}}$
  本アルゴリズムは Alice と Bob による通信を要する．Slot 形式による暗号文の組 $(c_{1,\mathrm{Slot}}, c_{2,\mathrm{Slot}})$ を Bob が所有し，これらの暗号文に対応する秘密鍵 sk を Alice が所有しているものとする．また，正整数 $n$ に関する情報は Alice, Bob どちらにも共有されているものとする．このとき，本アルゴリズムは次の 11 ステップからなる:

  1. Bob は $(c_{1,\mathrm{Slot}}, c_{2,\mathrm{Slot}})$ を Alice へ送信する．

2. Alice は $(c_{1,\mathrm{Slot}}, c_{2,\mathrm{Slot}})$ に対して，復号及び $\mathrm{Decode}_{\mathrm{Slot}}$ を行い，cleartext の組 $(r_1, r_2)$ を得る．
3. Alice は $r_2$ に対して，その配列を逆順にした $r_2'$ を得る．
4. Alice は $(r_1, r_2')$ に対して，$\mathrm{Encode}_{\mathrm{Coeff}}$ 及び暗号化を行い，暗号文の組 $(c_{1,\mathrm{Coeff}}, c'_{2,\mathrm{Coeff}})$ を得る．
5. Alice は $(c_{1,\mathrm{Coeff}}, c'_{2,\mathrm{Coeff}})$ を Bob へ送信する．
6. Bob は $(c_{1,\mathrm{Coeff}}, c'_{2,\mathrm{Coeff}})$ に対して，IPC を実行し，暗号文 $c_{\mathrm{Coeff}}$ を得る．
7. Bob は $c_{\mathrm{Coeff}}$ を Alice へ送信する．
8. Alice は $c_{\mathrm{Coeff}}$ に対して，復号及び $\mathrm{Decode}_{\mathrm{Coeff}}$ を行い，cleartext $r$ を得る．
9. Alice は $r$ の $n-1, 2n-1, \ldots, r'n-1$ 番目の要素の組 $(r_{n-1}, r_{2n-1}, \ldots, r_{r'n-1})$ を取得する．ここで $r' = \lfloor |r|/n \rfloor$ である．
10. Alice は組 $(r_{n-1}, r_{2n-1}, \ldots, r_{r'n-1})$ に対して，各要素へ $\mathrm{Encode}_{\mathrm{Slot}}$ 及び暗号化を行い，暗号文の組 $(c_{\mathrm{Slot},0}, \ldots, c_{\mathrm{Slot},r'-1})$ を得る．
11. Alice は $(c_{\mathrm{Slot},0}, \ldots, c_{\mathrm{Slot},r'-1})$ を Bob へ送信する．

なお，実際には，Bob から Alice へ暗号文を送信する前に乱数多項式を生成して，暗号文へ加える．これは，Alice が復号処理を行なっても，元の cleartext の値を特定できないようにするための処理である．また，以降では，この演算を IPS と略記する．

### 2.2 CKKS 方式を用いた行列の暗号化

本節では cleartext が $m \times n$ 行列 $A$ の場合を説明する．その準備として，以降では，長さ $n$ のベクトル $a$ が暗号化された $c_{\mathrm{Slot}}$ を $c$ と記載する．また，秘密鍵 sk, 公開鍵 pk, $c$ に対して， $\mathrm{Tile\_Encrypt}_{\mathrm{Slot,pk}}(a)$, $\mathrm{Detile\_Decrypt}_{\mathrm{Slot,sk}}(c)$ を与える．

- $\mathrm{Tile\_Encrypt}_{\mathrm{Slot,pk}}(a) \mapsto c$
  $a$ を $[a_0, \ldots a_{m-1}, a_0, \ldots a_{m-1}, \ldots]$ のようにサイズ $N$ のベクトルに拡張し，$\mathrm{Encrypt}_{\mathrm{Slot,pk}}$ を適応する．

- $\mathrm{Detile\_Decrypt}_{\mathrm{Slot,sk}}(c) \mapsto a$
  $c$ に対して，$\mathrm{Decrypt}_{\mathrm{Slot,sk}}$ を適応し，0 番目の要素から $n-1$ 番目の要素を出力する．

表 1: 各演算の計算量

| 演算 | (Mat, Vec) | (Mat, Mat) |
|---|---|---|
| Add/Sub | $O(mN)$ | $O(mN)$ |
| Mul | $O(mN^2)$ | $O(mN^2)$ |
| IPS | $O(mN^2)$ | $O(m^2N^2)$ |

また，cleartext が $m \times n$ 行列 $A$ の場合は，暗号化された行列を $C$ とすると以下のように表される．ここで，$C$ は 各要素に $c$ を持つ配列である．

- $\mathrm{Tile\_Encrypt}_{\mathrm{Slot,pk}}(A) \mapsto C$
  $A$ の各行ベクトル $a_i$ に対して，
  $\mathrm{Tile\_Encrypt}_{\mathrm{Slot,pk}}(a_i)$ を作用させ，これらを要素とする配列 $C$ を出力する．

- $\mathrm{Detile\_Decrypt}_{\mathrm{Slot,sk}}(C) \mapsto A$
  $C$ の各要素 $c_i$ に対して，$\mathrm{Detile\_Decrypt}_{\mathrm{Slot,sk}}(c_i)$ を適応し，これらを要素とする配列 $A$ を出力する．

以降では，$\mathrm{Tile\_Encrypt}_{\mathrm{Slot,pk}}$ を単に $\mathrm{Encrypt}_{\mathrm{pk}}$ と扱い，$\mathrm{Detile\_Decrypt}_{\mathrm{Slot,sk}}$ を単に $\mathrm{Decrypt}_{\mathrm{sk}}$ と扱う．

### 2.3 暗号化された行列に定まる演算

暗号化された行列 $C$ に対する演算を以下に示す．また，広く統計処理に用いられる数値解析ライブラリでの Add, Sub, Mul では，行列とベクトルの計算もサポートされており，以下のようになっている．また，サブルーチンとして，2.1.2 節で述べた演算を使用する．以下では暗号化されたベクトルを $c$ とし，Ope を Add/Sub/Mul のいずれかとする．$\mathrm{Ope}(C_1, c)$ とは，$C_1$ の各要素 $c_{1,i}$ と $c$ に対して，$\mathrm{Ope}(c_{1,i}, c)$ を適用し，暗号化された行列を得るアルゴリズムである．また，$\mathrm{Ope}(C_1, C_2)$ とは，$C_1, C_2$ の各要素 $c_{1,i}, c_{2,i}$ に対して，$\mathrm{Ope}(c_{1,i}, c_{2,i})$ を適用し，暗号化された行列を得るアルゴリズムである．

- $\mathrm{Inner\_Product\_Slot}(C, c, n) \mapsto c$
  暗号化された行列 $C$ の各要素 $c_i$ と暗号化されたベクトル $c$ に対して，$\mathrm{IPS}(c_i, c, n)$ を適用し，その結果を足し合わせて，暗号化された行列を得る．以降では，この演算を IPS と記載する．

- $\mathrm{Inner\_Product\_Slot}(C_1, C_2, n) \mapsto C$
  暗号化された行列 $C_1, C_2$ と，$C_2$ の各要素 $c_{2,i}$ に対して，$\mathrm{IPS}(C_1, c_{2,i}, n)$ を適用し，暗号化された行列を得る．以降では，この演算を IPS と記載する．

以上の演算は， $A$ の列数 $n$ が $n \leq N$ を満たす場合にのみ適応可能である．**表 1** は以上の演算の計算量をまとめた表であり，Vec/Mat は，それぞれ，暗号化されたベクトル/行列を表す．

**Algorithm 1** Ahn らのアルゴリズム

**Input:** $A \in \mathbb{R}^{n\times n}, r$

**Output:** $X_r$

$\lambda \leftarrow \mathrm{tr}(A)$
$\alpha \leftarrow 1/\lambda$
$X_0 \leftarrow \alpha I_n$
$Y_0 \leftarrow I_n - \alpha A$
**for** $i = 1$ **to** $r$ **do**
  $X_i \leftarrow X_{i-1}(I_n + Y_{i-1})$
  $Y_i \leftarrow Y_{i-1}^2$
**return** $X_r$

**Algorithm 2** 提案アルゴリズム 1

**Input:** $A \in \mathbb{R}^{n\times n}, r$

**Output:** $X_r$

$\lambda \leftarrow \mathrm{tr}(A)$
$\alpha' \leftarrow 2/\lambda$
$\tilde{\tilde{A}} \leftarrow I_n - \alpha' A$
$X_0 \leftarrow \alpha' I_n$
**for** $i = 1$ **to** $r$ **do**
  $X_i \leftarrow X_{i-1}\tilde{\tilde{A}} + X_0$
**return** $X_r$

## 3 ニュートン法による逆行列算出アルゴリズム

ニュートン法とは，非線型方程式 $f(X) = 0$ に対して，$X$ に関する漸化式を繰り返し用いることで，その解 $X$ を近似的に得る方法である．また，$n \times n$ 正方行列 $A$ が正定値対称行列であるとは，$A$ は対称行列であり，さらにそれらの固有値が全て正であることをいう．本章では，正定値対称行列に対して，ニュートン法を用いた逆行列算出アルゴリズムを考察し，その近似解を得るまでの反復回数について議論する．

### 3.1 先行研究の手法

Ahn ら [1] によるアルゴリズムは**アルゴリズム 1** のように表される．正定値対称行列 $A$ と正整数 $r$ に対して，$n \times n$ 行列 $X_r$ を出力するアルゴリズムである．ここで，$r \geq 1$ のとき，$\tilde{A} = I_n - \alpha A$ とすると，

$$X_r = \alpha\left(\tilde{A}^{2^i-1} + \cdots + \tilde{A} + I_n\right) \tag{1}$$

と表せる．

### 3.2 提案アルゴリズム

本節では，正定値対称行列に対して，ニュートン法を用いた逆行列算出アルゴリズムを提案する．**アルゴリズム 2** と**アルゴリズム 3** のようにかけ，これらも正定値対称行列 $A$ と正整数 $r$ に対して，$n \times n$ 行列 $X_r$ を出力するアルゴリズムである．$\tilde{\tilde{A}} = I_n - \alpha' A$ とすると，$r \geq 1$ において，**アルゴリズム 2** では，

$$X_r = \alpha'\left(\tilde{\tilde{A}}^i + \cdots + \tilde{\tilde{A}} + I_n\right) \tag{2}$$

と表せ，**アルゴリズム 3** では，

$$X_r = \alpha'\left(\tilde{\tilde{A}}^{2^{i+1}-1} + \cdots + \tilde{\tilde{A}} + I_n\right) \tag{3}$$

となる．以下では，式 2 及び式 3 が $r \to \infty$ で $A^{-1}$ に収束することを証明する．ここで，一般に次の定理が成り立つことが知られている：

**定理 1.** 対角化可能な $n \times n$ 正方行列 $A$ に対して，以下は同値：
(i) $A$ のスペクトル半径，つまり，$A$ の固有値の絶対値の最大値を $\rho(A)$ とするとき，$\rho(A) < 1$ が成り立つ．
(ii) Neumann 級数 $\sum_{n=0}^{\infty} A^n$ は $(I - A)^{-1}$ に収束する．

よって，$\tilde{\tilde{A}} = I - \alpha' A$ が対角化可能であり，(i) を満たすなら，上記の定理より，**アルゴリズム 2** と**アルゴリズム 3** において，$r$ を適切な値とすることで，その出力は $A^{-1}$ に十分近づく．式 2 及び式 3 は $r \to \infty$ で $\alpha'(I - \tilde{\tilde{A}})^{-1}$ に収束する．

以下では，$\tilde{\tilde{A}} = I - \alpha' A$ が対角化可能であることと (i) を満たすことを示す．まず，アルゴリズムの入力である $A$ は $n \times n$ 実対称行列であるため，$n$ 個の固有値 $\lambda_1, \ldots, \lambda_n$ を持ち，さらにそれらは実数である．よって，ある直行行列 $P$ が存在して，$A = P\mathrm{diag}(\lambda_1, \ldots, \lambda_n)P^{-1}$ のように $A$ は対角化できる．ここで，$\mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ とは，$n \times n$ 対角行列であり，その成分が左上から $\lambda_1, \lambda_2, \ldots$ で与えられる行列である．よって，

$$\tilde{\tilde{A}} = I - \alpha' A = P\mathrm{diag}(1 - \alpha'\lambda_1, \ldots, 1 - \alpha'\lambda_n)P^{-1}$$

となるので，$\tilde{\tilde{A}}$ は対角化可能であることが示された．

**Algorithm 3** 提案アルゴリズム 2

**Input:** $A \in \mathbb{R}^{n\times n}, r$

**Output:** $X_r$

$\lambda \leftarrow \mathrm{tr}(A)$
$\alpha' \leftarrow 2/\lambda$
$\tilde{\tilde{A}} \leftarrow I_n - \alpha' A$
$Y_0 \leftarrow \tilde{\tilde{A}}$
$X_0 \leftarrow \alpha'(\tilde{\tilde{A}} + I_n)$
**for** $i = 1$ **to** $r$ **do**
  $Y_i \leftarrow Y_{i-1}^2$
  $X_i \leftarrow X_{i-1}Y_i + X_{i-1}$
**return** $X_r$

また，$A$ は正定値行列ゆえ，任意の $1 \leq i \leq n$ に対して，$\lambda_i > 0$ である．$\alpha' = 2/(\lambda_1 + \cdots + \lambda_n)$ ゆえ，$0 < \alpha'\lambda_i < 1$ から $0 < 1 - \alpha'\lambda_i < 1$ となる．以上より，$\rho(\tilde{A}) < 1$ ゆえ，(i) も示された．

### 3.3 アルゴリズムの比較

前節までの議論から，任意の $i \in \{2, 3, 4\}$ と $\varepsilon$ に対して，ある $R_2, R_3, R_4$ が存在し，任意の $r \geq R_i$ とアルゴリズム $i$ の出力 $X_r$ と逆行列 $A^{-1}$ に対して，$|X_r - A^{-1}| < \varepsilon$ が成り立つ．ここで，$|X_r - A^{-1}| < \varepsilon$ とは，行列 $X_r - A^{-1}$ の各要素の絶対誤差が $\varepsilon$ 未満であることを表す．

本節では，$R_2, R_3, R_4$ の大小関係について議論する．式 1 の右辺は $\tilde{A}$ に関して，長さが $2^i - 1$ の多項式である．また，式 2，3 の右辺は $\tilde{\tilde{A}}$ に関して，長さがそれぞれ $i, 2^{i+1} - 1$ の多項式である．よって，$R_4 < R_2 < R_3$ であることが予想される．$R_4 < R_3$ または $R_2 < R_3$ であることは明らかなので，以下では，ランダム正定値対称行列に対して，$R_4 < R_3$ が成り立つことを保証する実験を行う．

**アルゴリズム 1** 及び**アルゴリズム 3** を実装し，ランダム正定値対称行列を入力したとき，近似逆行列に収束するまでに必要な反復回数を行列サイズ毎に計測した．収束条件は，$|X_r - A^{-1}| < \varepsilon$ とした．$\varepsilon$ は，$0 \leq s, t \leq n-1$ に対して，$\varepsilon = \text{atol} + \text{rtol} * \text{abs}(X_r(s,t) - A^{-1}(s,t))$ とし，$\text{atol} = 1.0 \times 10^{-8}, \text{rtol} = 1.0 \times 10^{-5}$ とした．$\text{abs}(X_{r(s,t)} - A^{-1}_{(s,t)})$ は，$X_r$ の要素 $X_{r(s,t)}$ と $A^{-1}$ の要素 $A^{-1}_{(s,t)}$ の絶対誤差を表す．

各行列サイズに対して，100 回の試行を行い，収束条件を満たした際の反復回数の平均値を求めた．その結果を**図 1** に示す．**図 1** より，**アルゴリズム 3** は，**アルゴリズム 1** に比べて，収束までに必要な平均反復回数が少ないことが確認できた．以上より，**アルゴリズム 3** は，ランダム正定値対称行列に対して，$R_4 < R_2 < R_3$ が成り立つことが実験的に示された．

また，アルゴリズムの出力が逆行列に収束しない場合についても考察する．反復回数 $r = 100$ までに一度も収束条件を満たさなかった行列の数を計測した．その結果を**図 2** に示す．**図 2** より，両アルゴリズムとも出力が逆行列に収束しない場合が存在することが確認できた．しかし，その中でも，**アルゴリズム 3** は，**アルゴリズム 1** に比べて，正しい逆行列を得られる確率が高いことが確認できた．以上より，**アルゴリズム 3** は，ランダム正定値対称行列に対して，**アルゴリズム 1** よりも高い収束性能を持つことが実験的に示された．

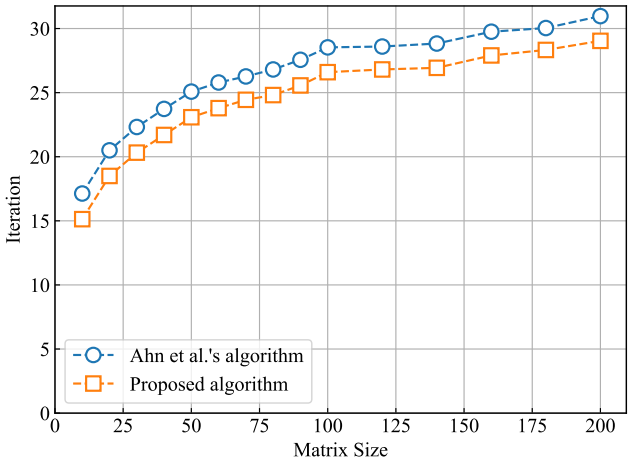


図 1: 収束条件を満たす平均反復回数の比較

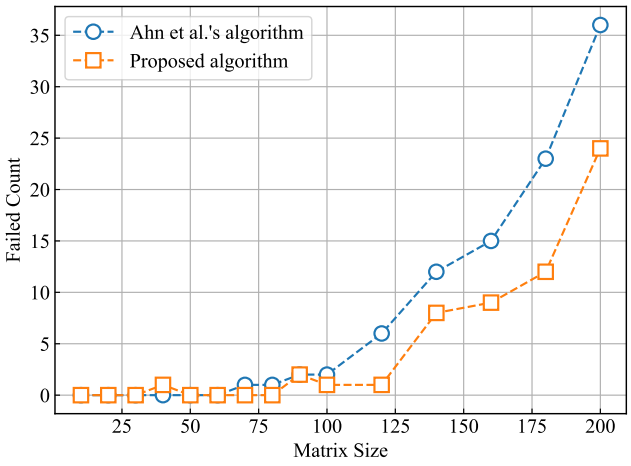


図 2: $r = 100$ までに一度も収束条件を満たさなかった行列数の比較

## 4 実装対象アルゴリズム

本章では，前章で提案した**アルゴリズム 3** を CKKS 暗号文上で実現するためのアルゴリズムを与える．**アルゴ**

**Algorithm 4** 暗号文状態での逆行列算出アルゴリズム

**Input:** $C_A, n, r$

**Output:** $C_{X_r}$

1: $A \leftarrow \mathrm{Decrypt}_{\mathrm{sk}}(C_A)$
2: $\lambda \leftarrow \mathrm{tr}(A_{\mathrm{dec}})$
3: $\alpha \leftarrow 2/\lambda$
4: $C_{\alpha U_n} \leftarrow \mathrm{Encrypt}_{\mathrm{pk}}(\alpha U_n)$
5: $C_{I_n} \leftarrow \mathrm{Encrypt}_{\mathrm{pk}}(I_n)$
6: $C_{\tilde{A}} \leftarrow \mathrm{Sub}(C_{I_n}, \mathrm{Mul}(C_\alpha, C_A))$
7: $C_{Y_0} \leftarrow C_{\tilde{A}}$
8: $C_{X_0} \leftarrow \mathrm{Mul}(C_{\alpha U_n}, \mathrm{Add}(C_{\tilde{A}}, C_{I_n}))$
9: **for** $i = 1$ **to** $r$ **do**
10: $\quad C_{Y_i} \leftarrow \mathrm{IPS}(C_{Y_{i-1}}, C_{Y_{i-1}}, n)$
11: $\quad C_{X_i} \leftarrow \mathrm{Add}(\mathrm{IPS}(C_{X_{i-1}}, C_{Y_i}, n), C_{X_i})$
12: **return** $C_{X_r}$

表 2: 実験環境

| | |
|---|---|
| OS | Ubuntu 22.04.3 LTS |
| CPU | Intel(R) Xeon(R) Silver 4314 |
| クロック周波数 | 2.40 GHz |
| RAM | 64 GB |
| SEAL | 4.0.0 |
| Python | 3.12.3 |
| NumPy | 1.26.4 |

**リズム 4** は，暗号化した正定値対称行列 $C_A$ と cleartext の長さ $n$, 正整数 $r$ に対して，近似逆行列 $C_{X_r}$ を出力するアルゴリズムである．**アルゴリズム 4** の 1–10 行目は**アルゴリズム 3** の 1–5 行目に対応し，変数の初期化を行う．11–13 行目は 6–8 行目に対応し，反復を行いながら出力を逆行列に近似させる．**表 1** に示すように $\mathrm{IPS}(C, C, n)$ の計算量は $O(n^2N^2)$ であるため，**アルゴリズム 4** の計算量はこれに反復回数 $r$ を掛けた $O(rn^2N^2)$ である．

## 5 実験および結果

本章では，前章で提案した**アルゴリズム 4** を実装し，その実行時間および誤差を計測し，その結果について考察する．ここで誤差は，cleatext で NumPy[9] モジュールに実装されている関数 numpy.linalg.inv を用いて算出した逆行列との絶対誤差及び相対誤差を計測する．本研究における実験環境を表 2 に示す．

### 5.1 実行時間

暗号文状態の $n$ 次単位行列に対して，反復回数 $r = 40$ で**アルゴリズム 4** を用いて逆行列を算出したときの実行時間を計測し，漸近的な計算量を確認した．なお，行列サイズは $n = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200$ とした．結果を**表 3**，**図 3** に示す． 4 章で述べた計算量 $O(rn^2N^2)$ に従い，実行時間が増加することが確認できた．

表 3: 行列サイズ毎の実行時間の抜粋

| 行列サイズ $n$ | 40 | 80 | 120 | 160 | 200 |
|---|---|---|---|---|---|
| 実行時間 [sec] | 408 | 1474 | 3224 | 5645 | 8691 |

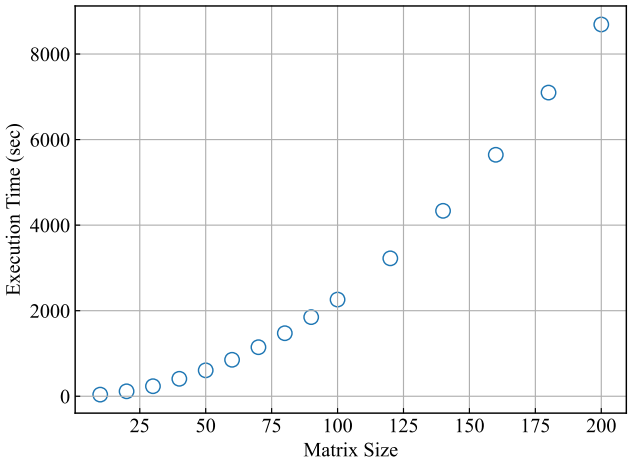


図 3: 行列サイズ毎の実行時間

### 5.2 入力が単位行列の場合の誤差

**アルゴリズム 4** を用いて，暗号文状態での逆行列を算出したときの絶対誤差を計測し，提案アルゴリズムの正しさを確認する．本節では，逆行列の算出に用いる行列 $A$ を，$n \times n$ 単位行列とする．行列サイズ $n$，反復回数 $r$ は前節と同じ条件とし，各行列サイズにおいて 10 回の試行を行い，各試行における絶対誤差の最大値および最小値を求めた．ここでの絶対誤差の最大値および最小値とは，暗号文状態で逆行列を算出し復号した行列と，cleartext で numpy.linalg.inv 関数を用いて算出した逆行列に対して，各要素の絶対誤差の最大値および最小値を示す．実験結果を**図 4** に示す．絶対誤差の最大値は，行列サイズが大きくなるにつれて増加しているが，$10^{-5}$ に収まっている．よって，提案アルゴリズムは単位行列の逆行列を算出できている．

### 5.3 入力がランダム正定値対称行列の場合の誤差

本節では，逆行列の算出に用いる行列 $A$ を，$n \times n$ のランダム正定値対称行列とする．そのため，各要素が 0 以上 1 未満の一様乱数である $n$ 次正方行列 $A'$ を生成し，それの転置行列との行列積 $A'' = A'^{\top}A'$ を求める．一般に，$A''$ の各要素は，平均 $n$，分散 $2n$ のカイ二乗分布に従うことが知られている．よって，$A$ は $A''$ の各要素を $n$ で割った行列 $A = A''/n$ とする．行列サイズ $n$，反復回数 $r$ は前節と同じ条件とし，各行列サイズにおいて 10

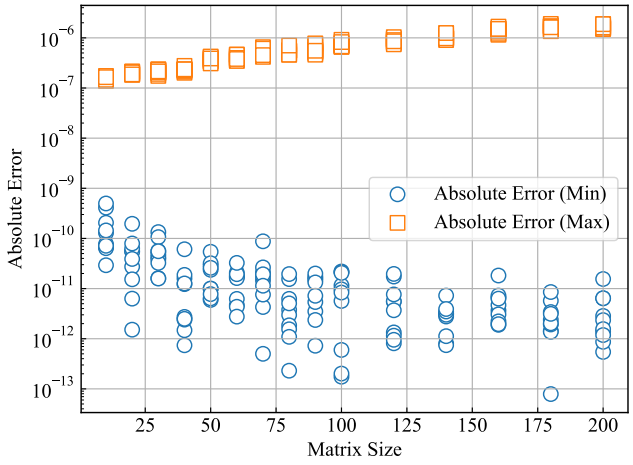


図 4: 入力が単位行列の場合の逆行列の絶対誤差

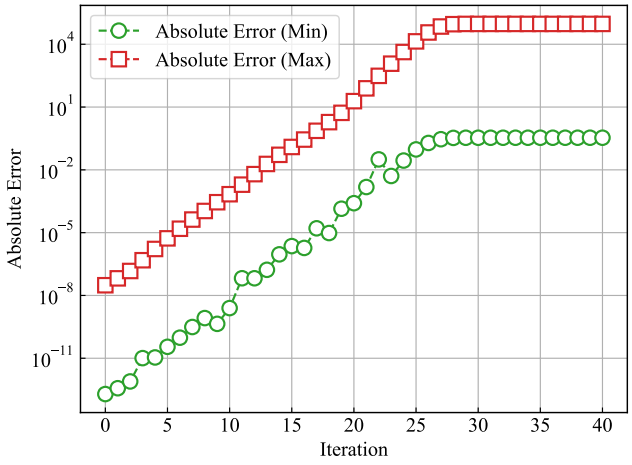


図 5: $n = 100$ における各反復の出力 $C_{X_i}$ と $X_i$ との絶対誤差

回の試行を行い，各試行における絶対誤差の最大値および最小値を求めた．実験結果を**図 6** に示す．絶対誤差，相対誤差の最大値は，行列サイズが大きくなるにつれて増加しており，最大で絶対誤差が $10^8$，相対誤差が $10^5$ 程度になっている．よって，ランダム正定値対称行列の逆行列を暗号文状態で算出する際には，誤差が大きくなる可能性があることが確認できた．

ここで，行列がランダム正定値対称行列の場合に誤差が大きくなる原因を考察する．考えられる原因の 1 つとして，暗号化・復号のプロセスにおいて加わるノイズによる誤差が考えられる．もう 1 つの原因として，計算途中の値が，暗号化前の行列に正しく復号できる定義域を超え，復号に失敗することで発生する誤差が考えられる．

まず，前者の原因について考察する．$n = 100, r = 40$ において，暗号文状態の**アルゴリズム 4** の各反復の出力 $C_{X_i}$ を復号した行列と，cleartext の**アルゴリズム 3** の各反復の出力 $X_i$ との各要素の絶対誤差の最大値および最小値を計測した．結果を**図 5** に示す．**図 5** の縦軸は対数表記のため，各反復の出力 $C_{X_i}$ と $X_i$ との絶対誤差は，反復を重ねるごとに指数関数的に増加していることが確認できる．よって，暗号化・復号のプロセスにおいて加わるノイズ，例えば，暗号化アルゴリズムに用いられる乱数の影響による誤差が蓄積されていると考えられる．

次に，後者の原因について考察する．今回の場合，暗号化アルゴリズムの定義域に含まれるためには，cleartext での行列の各要素の絶対値が $2^{19}$ 未満である必要がある．そこで，行列 $A$ に対して，$r = 40$ としたときの**アルゴリズム 3** の各反復において，行列 $X_i$ の各要素の絶対値が $2^{13}$ 未満になるように制約を加えた．実験結果を**図 7** に示す．絶対誤差の最大値は，$10^4$ に収まっており，制約を加える前と比べて大幅に減少している．よって，行列 $A$ の各要素の値は，アルゴリズムの途中で暗号化または復号可能な定義域に収まるように制約を加える必要があることが確認できる．

## 6 まとめ

本稿では，ニュートン法をベースとして，cleartext 及び準同型暗号 CKKS 方式における暗号文の逆行列の算出アルゴリズムを提案した．まず，cleartext で既存手法と，提案手法の実装を行ない，逆行列に収束するまでの反復回数を比較した結果，既存手法よりも高速に収束することを示した．また，暗号文状態の単位行列やランダム正定値対称行列の逆行列を算出し，実行時間の測定および cleartext 状態の逆行列との誤差評価を行なった．実行時間は，行列サイズが大きくなるにつれて，行列のサイズの 2 乗に従って増加することを確認した．単位行列の逆行列を計算した際の絶対誤差の最大値は，$10^{-5}$ に収まり，提案アルゴリズムによって正しく逆行列を算出できていると言える結果が得られた．一方で，ランダム正定値対称行列の逆行列を計算した際の誤差の最大値は，極めて大きくなる場合が見られた．本稿では，その原因の 2 つの可能性を考え，1 つは，暗号化・復号のプロセスにおいて加わるノイズによって蓄積された誤差であることを確認した．もう 1 つの原因は，計算途中の値が，暗号化前の行列に正しく復号できる定義域を超え，復号に失敗することで発生する誤差であることを確認した．そのため，行列の各要素の値がアルゴリズムの途中でこのような定義域に収まるように，入力される行列に制約を加える必要があることを確認した．しかし，上記のような制約をかけていても，行列サイズによっては誤差が大きくなる場合があるため，必要な制約は他にもあると考えられる．よって，今後の課題として，暗号文状態で正しく逆行列を算出できる，ランダム行列に必要な制約を更に明らかにしたい．

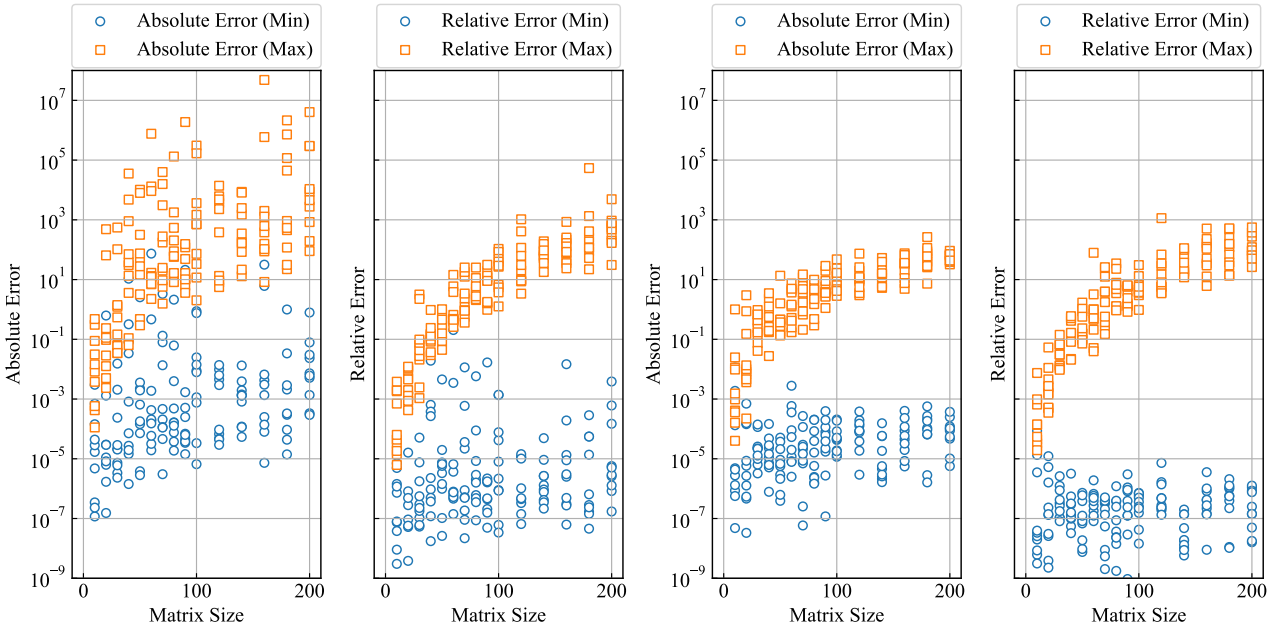


図 6: 入力がランダム正定値対称行列の場合の逆行列の絶対誤差・相対誤差

図 7: 入力が制約付きランダム正定値対称行列の場合の逆行列の絶対誤差・相対誤差

## 参考文献


[1] Tae Min Ahn et al. “Cheap and Fast Iterative Matrix Inverse in Encrypted Domain”. In: *Computer Security – ESORICS 2023*. Ed. by Gene Tsudik et al. Cham: Springer Nature Switzerland, 2024, pp. 334–352. ISBN: 978-3-031-50594-2.

[2] Alen Alexanderian. *A basic note on iterative matrix inversion*. July 2013. URL: https://aalexan3.math.ncsu.edu/articles/mat-inv-rep.pdf.

[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) Fully Homomorphic Encryption without Bootstrapping”. In: *ACM Trans. Comput. Theory* 6.3 (July 2014). ISSN: 1942-3454. DOI: 10.1145/2633600. URL: https://doi.org/10.1145/2633600.

[4] Jung Cheon et al. “Homomorphic Encryption for Arithmetic of Approximate Numbers”. In: Nov. 2017, pp. 409–437. ISBN: 978-3-319-70693-1. DOI: 10.1007/978-3-319-70694-8_15.

[5] Martine de Cock et al. “Fast, Privacy Preserving Linear Regression over Distributed Datasets based on Pre-Distributed Data”. In: *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*. AISec ’15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 3–14. ISBN: 9781450338264. DOI: 10.1145/2808769.2808774. URL: https://doi.org/10.1145/2808769.2808774.

[6] Junfeng Fan and Frederik Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2012/144. 2012. URL: https://eprint.iacr.org/2012/144.

[7] Craig Gentry. “Fully homomorphic encryption using ideal lattices”. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC ’09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178. ISBN: 9781605585062. DOI: 10.1145/1536414.1536440. URL: https://doi.org/10.1145/1536414.1536440.

[8] Rob Hall, Stephen E. Fienberg, and Yuval Nardi. “Secure multiple linear regression based on homomorphic encryption”. In: *Journal of Official Statistics* 27 (2011), pp. 669–691. URL: https://api.semanticscholar.org/CorpusID:14752788.

[9] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[10] Wen-jie Lu, Shohei Kawasaki, and Jun Sakuma. “Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data”. In: Jan. 2017. DOI: 10.14722/ndss.2017.23119.

[11] Nitish Mital, Cong Ling, and Deniz Gunduz. *Secure Distributed Matrix Computation with Discrete Fourier Transform*. 2021. arXiv: 2007.03972 [cs.IT]. URL: https://arxiv.org/abs/2007.03972.

[12] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *J. ACM* 56.6 (Sept. 2009). ISSN: 0004-5411. DOI: 10.1145/1568318.1568324. URL: https://doi.org/10.1145/1568318.1568324.