サーバ1台で構成可能な秘密分散法を用いた行列乗算の効率的でセキュアな委託計算

岩村 恵市[1,a)]　山根 将司[1,b)]

概要：行列乗算は深層学習を含む機械学習の分野で広く使用されている．しかし，行列のサイズが大きくなるにつれて計算量は膨大になっていく．個人が持つ携帯端末やPC等の限られた計算資源ではそのような大規模な計算を実行することは困難である．この問題を解決するために，個人がクラウド等の高性能サーバに入力や計算結果を秘密にしたまま安全に計算を委託できる委託計算手法が研究されている．本論文では，秘密分散法に基づき1台のサーバのみを使用して安全で効率的な行列乗算の委託計算を実現できる手法を提案する．1台のサーバによる秘密分散法を用いた委託計算は，1つの秘密情報の複数の分散値に異なる乱数を掛けることによって安全に実現される．提案手法は，少なくとも3台のサーバを必要とする秘密分散法に基づく手法の短所を解決し，さらに準同型暗号化に基づく手法よりも非常に少ない計算量で実行することを可能にする．提案手法は，受動的敵対者だけでなく能動的敵対者に対する安全性を持つ．さらに，提案手法は行列乗算に対する既存手法よりも効率的であることを示す．

キーワード：秘密分散法，秘匿計算，委託計算，行列乗算，クラウド

Secure Outsourcing Computation for Matrix Multiplication based on Secret Sharing Scheme using Only One Server

Keiichi Iwamura[1,a)]　Masashi Yamane[1,b)]

Abstract: Matrix multiplication has been widely used in the field of machine learning including deep learning. However, as the size of matrix is larger, the computational complexity becomes more enormous. Because of the limited performance or storage on the device, it is impossible for an individual to perform such large-scale computations. To tackle this problem, the technology which enables the consumer as a client to outsource computation securely for high-performance server in the cloud, namely, secure outsourcing computation is studied. In this paper, we propose secure and efficient outsourcing matrix multiplication based on secret sharing scheme and using only one server. Our method is realized by multiplying some shares of a secret by different random numbers. Proposed method allows to delete the demerit of secret sharing scheme which needs multiple servers, and moreover, to perform less computational complexity than that of the method based on homomorphic encryption. We prove the security of our method against not only passive adversaries but also active adversaries. In addition, we show that our method is more efficient than existing method of secure outsourcing computation.

Keywords: Secret sharing scheme, Secret computation, Outsourcing computation, Matrix multiplication, Cloud

1. はじめに

近年，モノのインターネット（Internet of Things）の発展などにより，多種多様かつ膨大なデータの利活用が検討

[1] 東京理科大学
Tokyo University of Science
[a)] iwamura@ee.kagu.tus.ac.jp
[b)] yamane_masashi@sec.ee.kagu.tus.ac.jp

© 2019 Information Processing Society of Japan

されている．[1] によると，利用可能なデータの量は 2013 年には 0.9 ゼタバイトだったのに対し，2020 年には 15 ゼタバイトを超えるという予測結果がある．これらの膨大なデータをデータマイニングなどして加工するには，かなりの計算コストが必要となることから，携帯端末や PC といった計算資源での処理には限界がある．この解決策として，より高スペックなクラウド上のサーバに計算を委託することが考えられる．しかし，委託する計算によっては，個人や企業の機密情報やプライバシ情報を含むことから，委託先のクラウドサーバからの情報漏洩が懸念事項となる．そのため，情報の秘匿性を保ったまま計算処理を実施できる委託計算手法の実現が重要である．

委託計算は，高スペックなクラウド側のサーバに，クライアントの持つ秘密情報を公開せずに計算を行う秘匿計算の 1 種である．大前提として，委託計算の際にクライアントに求められる計算量は，クライアント側で委託する計算を全て行ったときの計算量より少なくしなければならない．また，委託計算で実施される演算は，その目的から行列乗算 [2][3][4] や連立方程式 [5][6][7]，べき剰余 [8][9] など，必然的に計算量の多いものが対象となる．特に行列乗算は囲碁ソフト「AlphaGo」 [10] に代表されるディープラーニングをはじめとした機械学習の分野で多用されており，大規模な計算を行う際には，大量の計算リソースを消費することとなり，委託計算のニーズは大きい．

一方，委託計算を含む秘匿計算は大きく準同形暗号 [11] を用いる手法と秘密分散法 [12] を用いる手法に分類できる．準同形暗号を用いる手法は計算量が膨大になる場合が多いが，1 台のサーバ台数で秘匿計算することができる．それに対して，(k,n) 閾値秘密分散法を用いた秘匿計算では，必要なサーバ台数 n は $n \geq 2k-1$ という制限を持つ．よって，$k=2$ としても 3 台以上のサーバが必要であり，高速な計算を行うためにはそれらのサーバを高速な通信網で結ぶ必要がある．また，複数のサーバを同一組織が管理する場合，その組織への秘密情報漏洩を防ぐことができない．委託計算では，実用的な観点から 1 台のサーバで実行できることが望まれる．また，複数台のサーバを使用する場合には，サーバを同一の組織で管理しても情報が漏洩せず，かつ委託する計算が通信を必要としても，高速に秘匿計算できることが望まれる．

一般に委託計算の構成は，「事前処理」，「直前処理」，「委託処理」，「復元処理」の 4 つに分けられる．「事前処理」はクライアントの秘密情報を扱わない計算であり，前もって計算が可能である．これはクライアントが自身で計算することもできるし，信頼のできるサーバに任せることもできる．「直前処理」はクライアントの秘密情報を扱う計算である．よってクライアントのみが計算を行う．「委託処理」は委託先の高スペックなサーバで行う処理である．「復元処理」では委託先のサーバから送られてきた委託計算の結果を復元する．これもクライアントのみが計算を行う．委託計算の手順のうち，「直前処理」と「復元処理」はクライアントの低スペックな端末で行うので，計算量をできる限り少なくする必要がある．

委託計算のもう一つの問題点として，計算を委託されたサーバが正しい計算をせずに，本来の計算結果とは異なる値をクライアントに返す場合への対応がある．従って委託計算には秘密計算することのみならず，計算結果の検証ができることが必要不可欠である．よって，委託計算には上記に加えて「検証処理」が加えられる．[13] において，初めて秘密情報の秘匿と計算結果の検証の両方を解決する委託計算手法が提案された．ところが，大規模な演算に対しては実用的ではなく [14]，より計算量の少ない手法が望まれている．

このような委託計算に関するデメリットを解消するために，本稿では 行列乗算に対して 1 台のサーバで実行可能な秘密分散法を用いた委託計算法を提案する．これによって，サーバを同一の組織が管理しても情報漏洩がなく，かつ処理を 1 台のサーバ内で行うことから高速な通信網がなくても高速な秘匿計算が実現できる．そのために，秘密分散法を 1 台のサーバで実行できるように，1 つの秘密情報に対する n 個の分散値に異なる乱数をかけて，k 個以上の分散値が 1 台のサーバに集まっても復元できないようにする．また，異なる秘密情報に対する分散値には共通の乱数をかけることによって，秘密情報間の秘匿計算を可能とする．

また，委託計算には前述のように検証処理が必要になるが，提案手法は受動的敵対者に対する安全性だけでなく，能動的敵対者に対しても安全であることを証明する．ただし，提案手法の安全性は情報理論的安全性ではなく計算量的安全性となる．また，既存手法では検証処理が複雑になることが多いが，提案手法は検証処理も少ない計算量で実現でき効率的かつ安全であることを示す．

本稿の構成は，第 2 章において関連研究を紹介し，第 3 章において提案手法を示す．第 4 章で安全性及び処理量を評価する．最後に，第 5 章においてまとめを行う．

2. 関連研究

2.1 (k,n) 閾値秘密分散法

次の二つの条件を満たす秘密分散法を，(k,n) 閾値秘密分散法という．

(1) $k-1$ 個以下の分散値からは，秘密情報 s に関する情報は一切得ることはできない．

(2) 任意の k 個以上の分散値から，元の秘密情報 s を復元することができる．

代表的な (k,n) 閾値秘密分散法として Shamir によって提案された Shamir 法 [12]（以降 (k,n)Shamir 法）と，栗原らによって提案された XOR 法 [16] [17]，また加算のみ

で秘密分散が行える加法的秘密分散法 [15] などがある．

2.2 行列乗算の委託計算における既存手法

ここでは，行列乗算の委託計算における種々の既存手法について説明する．

(1) 準同型暗号を用いる方法 [2]：この手法は，あらかじめクライアントが委託する行列を準同型暗号で暗号化した暗号文を作っておき，委託時にかかる計算量を少なくしようという方法である．1 台のサーバで実現でき，行列の要素に 0 を含むことができる．ただし検証用として信頼できるサーバが必要となる．

(2) 乱数行列を用いる手法 [3]：この手法は準同型暗号も秘密分散法も使用せず，乱数行列のみを用いて計算を行う．サーバ 1 台で検証も含めて行うことができ計算量も少ないが，秘密情報に 0 を含む場合，秘匿した行列にも 0 が出現するため，情報が漏洩するという問題点を持つ．

(3) 秘密分散法を用いる方法 [4]：この手法は委託する行列を秘密分散し，サーバに計算を委託する．その際に，1 台のサーバで秘匿計算できるようクライアントがダミーの値を紛れ込ませることで，クラウド側のサーバに復元に必要な数の分散値が揃ってもクラウド側で正しく復元することができないようにする．提案方式も秘密分散法を用いるため，比較のため以下にこの手法の詳細を示す．

2.2.1 システムモデル

既存手法の登場人物は，クライアントとクラウド管理者の 2 人である．クライアントはクラウドにある 1 台のサーバに計算を委託する．クラウド管理者は悪意を持っており，計算に使用する情報を記録したり，計算をせず間違った値を計算結果として送信したりする．なので委託計算の入力及び出力は秘匿しなくてはならず，さらに計算結果を検証する必要がある．

2.2.2 委託計算の手順

クライアントは秘密情報 $\mathbf{X}$,$\mathbf{Y}$ の乗算結果である $\mathbf{Z}$ が知りたいとする．$\mathbf{X}$,$\mathbf{Y}$ はともに m 行 m 列である．

(1) 事前処理

クライアントは 2 セットの $2t+1$ 個の m 行 m 列の乱数行列 $\mathbf{A}_1,\cdots,\mathbf{A}_{2t+1}$ および $\mathbf{B}_1,\cdots,\mathbf{B}_{2t+1}$ を作成する．これらはダミーの役割を有する．さらに，乱数 $k_1,\cdots,k_{2t+1}$ を作成する．

(2) 直前処理

クライアントは $\mathbf{X}$,$\mathbf{Y}$ の各要素 x_{ij},y_{ij} について，$(t+1,2t+1)$Shamir 法を用いて分散した $P_{x_{ij}}(k_n),Q_{y_{ij}}(k_n)$ $(n=1,\cdots,2t+1)$ を i 行 j 列の要素とする行列 $\mathbf{P}_1,\cdots,\mathbf{P}_{2t+1}$ および $\mathbf{Q}_1,\cdots,\mathbf{Q}_{2t+1}$ を生成する．

$$P_{x_{ij}}^{(t)}(k_n)=\alpha_{ij_t}{k_n}^t+\cdots+\alpha_{ij_1}k_n+x_{ij} \tag{1}$$

$$Q_{y_{ij}}^{(t)}(k_n)=\alpha'_{ij_t}{k_n}^t+\cdots+\alpha'_{ij_1}k_n+y_{ij} \tag{2}$$

(3) 委託処理

(a) クライアントは作成した $(\mathbf{A}_n,\mathbf{B}_n),(\mathbf{P}_n,\mathbf{Q}_n)$ $(n=1,\cdots,2t+1)$ の計 $4t+2$ 個のペアを全てサーバに送信する．

(b) サーバは以下の計算を行い，計算結果をクライアントに送信する．

$$(\mathbf{A}_n\times\mathbf{B}_n)\ (n=1,\cdots,2t+1) \tag{3}$$

$$(\mathbf{P}_n\times\mathbf{Q}_n)\ (n=1,\cdots,2t+1) \tag{4}$$

(4) 復元処理

クライアントは送られてきた値から正しいペアを選び，$\mathbf{Z}$ の復元を行う．

(5) 検証処理

クライアントは (1) の処理において，$8t+2$ ペアの乱数を追加で生成し，(2) の処理において，$(k,n)=(t+1,6t+1)$ に変更する．これにより，復元処理時にエラー訂正を行うことができる．

2.3 提案手法

システムモデルは 2.2.1 と同様である．m 行 m 列の行列 $\mathbf{X},\mathbf{Y}$ の乗算結果である $\mathbf{Z}$ を委託計算する場合を示す．$\mathbf{X},\mathbf{Y},\mathbf{Z}$ の各要素を x_{ij},y_{ij},z_{ij} $(i,j=1,\cdots,m)$ で表す．ただし m は公知の値とする．

以下の手順での演算は全て法を p とし $GF(p)\ni(x_{ij}+c_1),(y_{ij}+c_1),(z_{ij}+c_1)$ であり，生成する乱数も $GF(p)$ の要素である．また，秘密情報 a に対する分散値を $\overline{[a]}_h$ で表す．また，1 台のサーバ内で秘密分散処理をする場合，n,k はできるだけ小さいほうが効率的であるので，$n=k=2$ とする．また, 加法的秘密分散法を用いることによって分散・復元処理を簡易化する．すなわち，$a=a_1+a_2$ となる乱数 a_1,a_2 を分散値とすれば 1 回の減算または加算で分散・復元ができる．

(1) 事前処理

(a) クライアントは 2 組の m^2 個の乱数 α_{ij},β_{ij} $(i,j=1,\cdots,m)$ を生成する．

(b) クライアントは乱数 δ_{ij} $(i,j=1,\cdots,m)$ と乱数 c_1 を生成し，$\delta_{ij}/\alpha_{ig}\beta_{gj},\delta_{ij}/\alpha_{ig},\delta_{ij}/\beta_{gj},\delta_{ij}$ $(i,j=1,\cdots,m)$ を秘密分散し，以下を得る $(h=0,1)$.

$$\overline{[\delta_{ij}/\alpha_{ig}\beta_{gj}]}_h,\overline{[\delta_{ij}c_1/\alpha_{ig}]}_h,\overline{[\delta_{ij}c_1/\beta_{gj}]}_h,\overline{[\delta_{ij}]}_h \tag{5}$$

(c) クライアントは乱数 τ を生成し以下を計算する．

$$\tau\overline{[\delta_{ij}/\alpha_{ig}\beta_{gj}]}_1=\tau\times\overline{[\delta_{ij}/\alpha_{ig}\beta_{gj}]}_1 \tag{6}$$

$$\tau\overline{[\delta_{ij}c_1/\alpha_{ig}]}_1=\tau\times\overline{[\delta_{ij}c_1/\alpha_{ig}]}_1 \tag{7}$$

$$\tau\overline{[\delta_{ij}c_1/\beta_{gj}]}_1 = \tau \times \overline{[\delta_{ij}c_1/\beta_{gj}]}_1 \quad (8)$$

$$\tau\overline{[\delta_{ij}]}_1 = \tau \times \overline{[\delta_{ij}]}_1 \quad (9)$$

(d) クライアントは以下を 1 台のサーバに送る．$\overline{[\delta_{ij}/\alpha_{ig}\beta_{gj}]}_0, \overline{[\delta_{ij}c_1/\alpha_{ig}]}_0, \overline{[\delta_{ij}c_1/\beta_{gj}]}_0, \overline{[\delta_{ij}]}_0$ $\tau\overline{[\delta_{ij}/\alpha_{ig}\beta_{gj}]}_1, \tau\overline{[\delta_{ij}c_1/\alpha_{ig}]}_1, \tau\overline{[\delta_{ij}c_1/\beta_{gj}]}_1, \tau\overline{[\delta_{ij}]}_1$

(e) クライアントは τ^{-1} を計算し，生成した乱数 c_1, τ^{-1} だけを復元および検証処理のために保存する．

(2) 直前処理

クライアントは $x'_{ij} = \alpha_{ij}(x_{ij}+c_1)$ と $y'_{ij} = \beta_{ij}(y_{ij}+c_1)$ を計算して，x'_{ij} を要素とする行列 $\mathbf{X}'$ と y'_{ij} を要素とする行列 $\mathbf{Y}'$ を 1 台のサーバに送る．

(3) 委託処理

サーバは，各 $i, j = 1, \cdots, m$ に対して以下を計算する．

$$\begin{aligned}\overline{[\delta_{ij}(z_{ij} - mc_1{}^2)]}_0 &= \overline{\left[\delta_{ij}\left(\sum_{g=1}^{m} x_{ig}y_{gj} - mc_1{}^2\right)\right]}_0 \\ &= \sum_{g=1}^{m}\left(x'_{ig} \times y'_{gj} \times \overline{\left[\frac{\delta_{ij}}{\alpha_{ig}\beta_{gj}}\right]}_0\right) \\ &\quad - \sum_{g=1}^{m}\left(x'_{ig} \times \overline{\left[\frac{\delta_{ij}c_1}{\alpha_{ig}}\right]}_0\right) \\ &\quad - \sum_{g=1}^{m}\left(y'_{gj} \times \overline{\left[\frac{\delta_{ij}c_1}{\beta_{gj}}\right]}_0\right) \quad (10)\end{aligned}$$

$$\begin{aligned}\tau\overline{[\delta_{ij}(z_{ij} - mc_1{}^2)]}_1 &= \tau\overline{\left[\delta_{ij}\left(\sum_{g=1}^{m} x_{ig}y_{gj} - mc_1{}^2\right)\right]}_1 \\ &= \sum_{g=1}^{m}\left(x'_{ig} \times y'_{gj} \times \tau\overline{\left[\frac{\delta_{ij}}{\alpha_{ig}\beta_{gj}}\right]}_1\right) \\ &\quad - \sum_{g=1}^{m}\left(x'_{ig} \times \tau\overline{\left[\frac{\delta_{ij}c_1}{\alpha_{ig}}\right]}_1\right) \\ &\quad - \sum_{g=1}^{m}\left(y'_{gj} \times \tau\overline{\left[\frac{\delta_{ij}c_1}{\beta_{gj}}\right]}_1\right) \quad (11)\end{aligned}$$

(4) 復元処理

(a) 復元者はサーバから $\overline{[\delta_{ij}(z_{ij} - mc_1{}^2)]}_0$, $\tau\overline{[\delta_{ij}(z_{ij} - mc_1{}^2)]}_1$, $\overline{[\delta_{ij}]}_0$, $\tau\overline{[\delta_{ij}]}_1$ を収集し，$\tau\overline{[\delta_{ij}(z_{ij} - mc_1{}^2)]}_1$, $\tau\overline{[\delta_{ij}]}_1$ に τ^{-1} をかけてもう 1 つの分散値も用いて $\delta_{ij}(z_{ij} - mc_1{}^2), \delta_{ij}$ を復元する．

(b) 復元者は以下を計算する．

$$z_{ij} - mc_1{}^2 = \delta_{ij}(z_{ij} - mc_1{}^2)/\delta_{ij} \quad (12)$$

(c) 復元者は乗算結果 $\mathbf{Z}$ の各要素を以下の式より復元する．

$$z_{ij} = (z_{ij} - mc_1{}^2) + (m \times c_1 \times c_1) \quad (13)$$

(5) 検証処理

(a) 事前処理で生成した乱数を全て異なるようにして，直前処理で新たな行列 $\mathbf{X}', \mathbf{Y}'$ を生成して，再びサーバに送り委託演算を行う．

(b) 復元処理において復元した $z_{ij} - mc_1{}^2$ と $z_{ij} - mc_2{}^2$ から以下を計算し，保存している c_1, c_2 を用いて $m(c_1{}^2 - c_2{}^2)$ を計算して比較し，一致すれば検証成功とする．ただし，c_2 は検証処理時の c_1 に相当する．

$$m(c_1{}^2 - c_2{}^2) = (z_{ij} - mc_2{}^2) - (z_{ij} - mc_1{}^2) \quad (14)$$

3. 評価

安全性については以下の 2 つの安全性要件を定め，機密性について入力と出力について評価する．

(1) 機密性：クライアントの秘密情報（入力）および委託計算の結果（出力）が攻撃者に漏洩しない．

(2) 完全性：クライアントは正しい委託計算結果を得る．

3.1 入力に対する機密性

まず、攻撃者はサーバに保存されている情報からクライアントの入力した秘密情報を知ろうとする．しかし，攻撃者は秘密情報である x_{ij}, y_{ij} をサーバに保存されている秘密情報を含んだ $x'_{ij} = \alpha_{ij}(x_{ij} + c_1)$ と $y'_{ij} = \beta_{ij}(y_{ij} + c_1)$ から知ることができない．すなわち，攻撃者は乱数 $\alpha_{ij}, \beta_{ij}, c_1$ をサーバに保存されている以下の情報から特定できない．$\overline{[\delta_{ij}/\alpha_{ig}\beta_{gj}]}_0, \overline{[\delta_{ij}c_1/\alpha_{ig}]}_0, \overline{[\delta_{ij}c_1/\beta_{gj}]}_0, \overline{[\delta_{ij}]}_0$ $\tau\overline{[\delta_{ij}/\alpha_{ig}\beta_{gj}]}_1, \tau\overline{[\delta_{ij}c_1/\alpha_{ig}]}_1, \tau\overline{[\delta_{ij}c_1/\beta_{gj}]}_1, \tau\overline{[\delta_{ij}]}_1$

情報理論的安全性とはある情報が分からなければ，無限の計算能力を持っていても解が特定できないことを指す．例えば，Shamir の (k, n) 閾値秘密分散法で $k = 2$ の場合，攻撃者が 1 つの分散値を知っても，もう 1 つの分散値が分からなければ，もう 1 つの分散値に全ての値を想定して解いても全てに解が存在するため，正しい解を特定できず情報理論的安全性を実現する．提案手法においても同様に考えられる．例えば，攻撃者が委託計算の結果を特定しようとして $\tau\overline{[\delta_{ij}]}_1$ に対して τ として全ての値 (τ' とする) で除算して $\overline{[\delta_{ij}]}_0$ と組み合わせて復元しようとしても，復元処理は全ての τ' に対して解を持つので，攻撃者は正しい解を特定できない．従って，δ_{ij} のエントロピーを $H(\delta_{ij})$ と表記すると，以下の式が成立し，提案手法は情報理論的安全性を実現することが言える．

$$H(\delta_{ij}) = H(\delta_{ij} \mid \overline{[\delta_{ij}]}_0, \tau\overline{[\delta_{ij}]}_1) \quad (15)$$

また，サーバに保存している分散値の組み合わせを考える．$\tau\overline{[\delta_{ij}/\alpha_{ig}\beta_{gj}]}_1, \tau\overline{[\delta_{ij}c_1/\alpha_{ig}]}_1, \tau\overline{[\delta_{ij}c_1/\beta_{gj}]}_1, \tau\overline{[\delta_{ij}]}_1$ に適当な τ' をかけて復元した結果を $(\delta_{ij}/\alpha_{ig}\beta_{gj})' = \delta_{ij}/\alpha_{ig}\beta_{gj} + r_1$, $(\delta_{ij}c_1/\alpha_{ig})' = \delta_{ij}c_1/\alpha_{ig} + r_2$, $(\delta_{ij}c_1/\beta_{gj})' = \delta_{ij}c_1/\beta_{gj} + r_3$, $\delta'_{ij} = \delta_{ij} + r_4$ とする．（正しい τ' が選ばれれば $r_1 = r_2 =$

$r_3 = r_4 = 0$)．正しい τ' が選択されている場合，以下の式より ${c_1}^2$ を得て，$\delta_{ij}, \delta_{ij}c_1/\alpha_{ig}$ より α_{ig}，$\delta_{ij}c_1/\beta_{gj}$ より β_{gj} を得ることができ，入力が漏洩する．

$$\frac{(\delta_{ij}c_1/\alpha_{ig})(\delta_{ij}c_1/\beta_{gj})}{(\delta_{ij}/\alpha_{ig}\beta_{gj})\delta_{ij}} = {c_1}^2 \tag{16}$$

しかし，正しくない τ' を用いた $(\delta_{ij}/\alpha_{ig}\beta_{gj})', (\delta_{ij}c_1/\alpha_{ig})', (\delta_{ij}c_1/\beta_{gj})', \delta'_{ij}$ からも同様に ${c_1}^{2'}$ を得ることができる．攻撃者は ${c_1}^2$ を知らないため正しい解を特定することができない．よって，1 組の分散値（上記において i, j, g が固定）の組み合わせに対しては情報理論的安全性を実現できる．

しかし，複数組の分散値の組み合わせを考えると情報理論的安全性は成立しない．例えば，$g = 1, 2$（i, j は任意）に対して以下が成り立つため，${c_1}^{2'} = {c_1}^{2''}$ となった τ' が正しい τ である可能性を持つ（偶然一致する場合もある）．これを複数の g または i, j に対して行えば，攻撃者は正しい τ を知ることができる．

$$\frac{(\delta_{ij}c_1/\alpha_{i1})(\delta_{ij}c_1/\beta_{1j})}{(\delta_{ij}/\alpha_{i1}\beta_{1j})\delta_{ij}} = {c_1}^{2'} \tag{17}$$

$$\frac{(\delta_{ij}c_1/\alpha_{i2})(\delta_{ij}c_1/\beta_{2j})}{(\delta_{ij}/\alpha_{i2}\beta_{2j})\delta_{ij}} = {c_1}^{2''} \tag{18}$$

ただし，法とする p を例えば 128 ビット程度の大きさとすると τ の候補は 2^{128} となり、十分な計算量的安全性を実現することができる．よって，入力に対しては $2^{|p|}$ 程度の機密性を実現する（$|p|$ は p のビット長を表す）．

3.2 出力および検証に対する機密性

攻撃者はサーバに保存されている情報から委託計算の結果を知ろうとする．しかし，攻撃者は計算結果である z_{ij} を入力に対する機密性と同様にサーバに保存されている情報 $\overline{[\delta_{ij}(z_{ij} - mc_1{}^2)]_0}, \tau\overline{[\delta_{ij}(z_{ij} - mc_1{}^2)]_1}$ から知ることができない．なぜならば，τ を知らなければ全ての解の中から正しい解を特定できないためである．

ただし，検証処理を行う場合，復元した $\left(z_{ij} - mc_1{}^2\right)'$, $(z_{ij} - mc_2{}^2)'$ と ${c_1}^{2'}, {c_2}^{2'}$ から下記演算を行い，0 になれば z_{ij} が漏洩する可能性がある．

$$\{(z_{ij} - mc_1{}^2)' - mc_1{}^{2'}\} - \{(z_{ij} - mc_2{}^2)' - mc_2{}^{2'}\} \tag{19}$$

入力に対する機密性の場合を含め，秘匿した乱数または秘密情報が漏洩するのは同じ値（前記の場合 c_1，検証では z_{ij}）を含む分散値（片方は τ がかかる）があり，それが τ' の選択により復元できると，その値の一致不一致によって乱数または秘密情報が漏洩する．よって，この場合の安全性も法とする p のサイズによる計算量的な安全性となる．

3.3 完全性

攻撃者は，サーバ内の情報を改ざんするなどして，クライアントに偽物の委託計算結果を復元させることができる．ここで検証処理において $r_{ij}(z_{ij} + mc_1{}^2), q_{ij}(z_{ij} + mc_2{}^2)$ が復元されたとする．r_{ij}, q_{ij} は正しい $(z_{ij} + mc_1{}^2), (z_{ij} + mc_2{}^2)$ との比である．このとき，$q_{ij}(z_{ij} - mc_2{}^2) - r_{ij}(z_{ij} - mc_1{}^2) = m({c_1}^2 - {c_2}^2)$ となるためには，$q_{ij}(z_{ij} - mc_2{}^2) = m({c_1}^2 - {c_2}^2) + r_{ij}(z_{ij} + mc_1{}^2)$ となるように r_{ij}, q_{ij} を調整する必要がある．しかし，攻撃者は機密性より $z_{ij}, {c_1}^2, {c_2}^2$ を知ることができないため調整できない．すなわち，上式を書き直すと以下になる．

$$(r_{ij} - q_{ij})z_{ij} + (r_{ij} - 1)mc_1{}^2 + (1 - q_{ij})mc_2{}^2 = 0 \tag{20}$$

よって，攻撃者が $z_{ij}, {c_1}^2, {c_2}^2$ を知らずに上式を成立させるには $r_{ij} = q_{ij} = 1$ 以外にないことが言える．

ただし，$\left(z_{ij} - mc_1{}^2\right) = 0$ となる場合，攻撃者は $z_{ij}, {c_1}^2, {c_2}^2$ を知らないので，$m({c_1}^2 - {c_2}^2) = r_{ij}(z_{ij} + mc_1{}^2)$ となるように r_{ij} を調整できない．$\left(z_{ij} - mc_2{}^2\right) = 0$ の場合も同様である．

よって，提案手法は委託計算結果に対して完全性を有する．

4. 処理量評価

本章では，既存手法と提案手法との計算量に関する比較を行う．ただし計算量については，直前処理，復元処理および検証処理においてクライアントが実行した計算量で評価する．既存手法における安全性は，$2^{-(t+1)}$ を無視できるほど小さくする t によって規定されているため，$t = 127$ とする．また，既存手法では法とする p は定められていないが，演算結果を有限体上の値ではなく整数上の演算結果としたい場合，p は演算結果である $z_{ij} = \sum_{g=1}^{m} x_{ig}y_{gj}$ より大きい必要があり，委託計算の性質上，x_{ig}, y_{gj}, m はある程度大きな値と考えられる．よって，既存手法においても p は 128 ビット程度の値とすることは妥当である．よって，既存手法及び提案手法において $|p| = 128$ とする．ただし，秘密分散法に用いる閾値 k は，既存手法では $k = t - 1$，提案手法では $k = 2$ である．以下に使われる記号の定義を示す．

- M：乗算・除算を 1 回行うのに必要な計算量
- A：加算・減算を 1 回行うのに必要な計算量
- m：行列乗算に用いる行列の行および列のサイズ
- S：Shamir 法による 1 つの分散値および復元に必要な計算量
- E：エラー訂正に必要な計算量

以下に，計算量を評価する．(k, n)Shamir 法において 1 つの分散値計算に必要な計算量を S とすると，既存手法における直前処理は $2(2t + 1)m^2$S となり，復元処理は m^2S となる．厳密には分散と復元に関する計算量は異なるが，どちらも多項式計算（復元ではラグランジュ補間公式）であるので，簡単のため復元の計算量も S で表す．また，検

証処理は $(2t+1)$ を $(6t+1)$ とするため秘密分散の計算量は $2(6t+1)m^2\mathrm{S}$ となり，エラー訂正は各要素毎に行うので $m^2\mathrm{E}$ となる．ただし，ダミーの乱数に関する処理は含めない．**表 1** に $t=127$ とした場合の結果を示す．

表 1 行列乗算における各手法の計算量
Table 1 Comparison of Computation Complexity.

	既存手法 [4]	提案手法
直前処理	$510m^2\mathrm{S}$	$2m^2\mathrm{A}+2m^2\mathrm{M}$
復元処理	$m^2\mathrm{S}$	$3m^2\mathrm{A}+5m^2\mathrm{M}$
検証処理	$1526m^2\mathrm{S}+m^2\mathrm{E}$	$7m^2\mathrm{A}+10m^2\mathrm{M}$
合計	$2037m^2\mathrm{S}+m^2\mathrm{E}$	$12m^2\mathrm{A}+18m^2\mathrm{M}$

一方，提案手法の直前計算における秘密情報の秘匿処理は 1 回の加算と乗算で行えるので，$2m^2(\mathrm{A}+\mathrm{M})$ である．また，復元処理は加法的秘密分散法を用いるため，2 つの秘密情報の復元に必要な計算量は 2A となる．また，それ以外に 5 回の乗除算と 1 回の加算を必要とする．よって，計 $(3\mathrm{A}+5\mathrm{M})m^2$ となる．また，検証処理は上記処理を繰返し，比較のために 3 回の乗算と 2 回の減算が加わる．よって，$7m^2\mathrm{A}+10m^2\mathrm{M}$ となる（比較や乱数生成の計算量は含めない）．

$t=127$ の場合，$\mathrm{S},\mathrm{E} \gg \mathrm{A},\mathrm{M}$ より圧倒的に提案手法の計算量が少ないことが分かる．

最後に，記憶容量に関しては提案手法では検証処理を含むと $\tau^{-1},\tau'^{-1},c_1,c_2$ の 4 つを記憶する（τ' は検証処理における τ,m は公知）．それに対して既存手法はランダムに送られたペアの中から正しいペアを選ぶための情報を保存する必要がある．よって，その数は少なくとも $2t+1$ 以上（検証処理を含むと $6t+1$）と考えられる．ただし，t は 127 程度の数値であるので，全体としては 100 ビットを超えない．よって，記憶容量は既存手法の方が少ない．

5. まとめ

本稿では，サーバ 1 台で構成可能な秘密分散法を用いた行列乗算のセキュアな委託計算を提案した．サーバが 2 台使える場合，τ をかける必要はなく，秘密分散法が本来もつ情報理論的な安全性を実現できる．また，提案手法によって秘密分散法を用いた委託計算が 1 台のサーバのみで効率的に実現できるようになるが，提案手法を 2 台のサーバにそのまま適用した場合，2 台のサーバを同一組織が管理していても安全な委託計算が実現できる（処理速度はサーバ 1 台の場合の倍になる）．今後の課題としては，べき乗剰余や連立方程式の効率的な委託計算への検討対応と，複数の参加者による秘匿計算への拡張が挙げられる．

参考文献

[1] Adshead, A.: *Data set to grow 10-fold by 2020 as internet of things takes off*, 入手先 ⟨https://www.computerweekly.com/news/2240217788/Data-set-to-grow-10-fold-by-2020-as-internet-of-things-takes-off⟩ (2019.0727).

[2] Zhang, S., Li, H., Dai, Y., Li, J., He, M. and Lu, R.: *Verifiable Outsourcing Computation for Matrix Multiplication with Improved Efficiency and ApplicabilityData set to grow 10-fold by 2020 as internet of things takes off*, IEEE Internet of Things Journal, pp. 5076-5088 (2018).

[3] Lei, X., Liao, X., Huang, T. and Heriniaina, F.: *Achieving security, robust cheating resistance, and highefficiency for outsourcing large matrix multiplication computation to a malicious cloud*, Information Sciences, Vol. 280, pp. 205-217 (2014).

[4] Atallah, M. J. and Frikken, K. B.: *Securely outsourcing linear algebra computations*, Proc. 5th ACM Symp. Information, Comput. Commun. Secur., p. 48 (2010).

[5] Wang, C., Ren, K., Wang, J. and Wang, Q.: *Harnessing the cloud for securely outsourcing large-scale systems of linear equations*, IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 6, pp. 1172-1181(2013).

[6] Chen, X., Huang, X., Li, J., Ma, J., Lou, W. and Wong, D. S.: *New Algorithms for Secure Outsourcing of LargeScale Systems of Linear Equations*, IEEE Transactions on Information Forensics and Security, Vol. 10, No. 1, pp. 69-78 (2015).

[7] Yunpeng, Y., Yuchuan, L., Dongsheng, W., Shaojing, F. and Ming, X.: *Efficient , Secure and Non-iterative Outsourcing of Large-Scale Systems of Linear Equations*, Proc. IEEE International Conference on Communications (ICC), pp. 1-6 (2016).

[8] Chen, X., Li, J., Ma, J., Tang, Q. and Lou, W.: *New algorithms for secure outsourcing of modular exponentiations*, IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 9, pp. 238-2396 (2014).

[9] Ding, Y., Xu, Z., Ye, J. and Choo, K. K. R.: *Secure outsourcing of modular exponentiations under single untrusted programme model*, Journal of Computer and System Sciences, Vol. 90, pp. 1-13 (2017).

[10] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. and Hassabis, D.: *Mastering the game of Go with deep neural networks and tree search*, Nature, Vol. 529, pp.484-489 (2009).

[11] Gentry, C.: *A Fully Homomorphic Encryption Scheme*, PhD Thesis, Stanford University (2009).

[12] Shamir, A.: *How to share a secret*, Communications of the ACM, Vol. 22, No. 11, pp. 612-613 (1979).

[13] Gennaro, R., Gentry, C. and Parno, B.: *Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers*, CRYPTO, pp. 465-482 (2010).

[14] Shan, Z., Ren, K. and Blanton, M.: *Practical Secure Computation Outsourcing: A Survey*, ACM Computing Surveys, Vol. 51 (2018).

[15] 尾形わかは，黒沢馨：秘密分散法とその応用，電子情報通信学会誌，Vol. 82, pp. 1228-1236 (1999).

[16] Kurihara, J., Kiyomoto, S., Fukushima, K. and Tanaka, T.: *On a Fast(k,n)-Threshold Secret Sharing Scheme*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E91-A, No. 9, pp. 2365-2378 (2008).

[17] Kurihara, J., Kiyomoto, S., Fukushima, K. and Tanaka, T.: *A New (k,n)-Threshold Secret Sharing Scheme and*

Its Extension, Proc. ISC ’ 08, pp. 455-470 (2008).
[18] 神宮武志，青井健，ムハンマド カマルアフマド アクマルアミヌディン，岩村惠市：秘密分散法を用いた次数変化のない秘匿計算手法，情報処理学会論文誌，Vol. 59, No. 3, pp. 1038-1049 (2018).
[19] ムハンマド カマルアフマド アクマルアミヌディン，岩村惠市：秘密分散法を用いた四則演算の組み合わせに対して安全な次数変化のない秘匿計算，情報処理学会論文誌，Vol. 59, No. 9, pp. 1581-1595 (2018).
[20] 鴇田恭平，岩村惠市：多値化法による秘匿四則演算及びその秘匿部分一致検索への応用，電子情報通信学会技術研究報告，Vol. 117, No. 55, pp. 107-114 (2017).
[21] Sergio, S., Changqing, L., Xuhui, C. and Pan, L.: *Efficient Secure Outsourcing of Large-Scale Linear Systems of Equations*, Proc. IEEE Conference on Computer Communications (INFOCOM), pp. 1035-1043 (2015).