Vol.2018-CSEC-82 No.60
Vol.2018-SPT-29 No.60
2018/7/26

１台のサーバで実行可能な秘密分散法を用いた秘匿計算法

山根将司[†1]　岩村惠市[†1]　安田裕之[†2]

概要：本稿では，1 台のサーバで実行可能な秘密分散法を用いた秘匿計算法を提案する．秘密分散法は秘匿計算技術の１つである．一般に，秘密分散法は少なくとも 2 台のサーバを必要とし，秘匿計算を行う場合は 3 台以上のサーバを必要とする．しかし，そのサーバ群を１つの組織が運用する場合，その組織は任意のサーバの情報を取得可能なので，秘密分散された秘密情報を復元できてしまう．また，秘匿計算を高速処理するためには十分な計算能力をもつ高性能なサーバを複数準備する必要がある．そのため複数のサーバ中 1 台のサーバの処理能力が低ければ，それがボトルネックとなり全体のスループットが低下する．提案手法を用いることで，秘匿計算用のサーバ群が 1 つの組織のみによって管理される場合でもその組織への情報漏洩のリスクを回避でき，サーバ管理の負担や必要な計算資源が軽減される．さらに，1 台の高性能サーバのみで安全かつ高速な処理が実現できるようになる．提案手法の詳細を述べた後，本手法が passive な攻撃者に対して情報理論的な安全性を持つことを示す．

キーワード：秘密分散法，秘匿計算

A Secure Computation Method via Secret Sharing Scheme using Only One Server

MASASHI YAMANE[†1]　KEIICHI IWAMURA[†1]
HIROYUKI YASUDA[†2]

Abstract: In this paper, we propose a secret sharing scheme using only one server for the secure computation. Secret sharing scheme is one candidate for the secure computation. This method needs two or more servers; three servers are required generally. When a group operates the method, the group can collect the shares and recover secret. Whereas, high-speed computation requires multiple high-performance servers. If one of the servers composing the secure computation system has a low throughput, the server decreases the throughput of overall the system. In our proposed method, a group operating the method cannot recover the secret even if the group collects shares. Also, the group can reduce the energy of server management and the computational resource required. In addition, high-speed and secure computation can be achieved by a high-performance server. We show that our method is information-theoretically secure for passive adversary.

Keywords: Secret Sharing Scheme, Secure Computation

1. はじめに

近年，ビッグデータおよびモノのインターネット（IoT）環境の発展に伴い，個人に関する情報を利用したサービスが急速に広まっている．例えば，顔写真から美顔度を判定する *Deeplooks* [1]というディープニューラルネットワークを用いたサービスがある．しかし，このようなサービスでは顔写真は秘匿されることなく用いられているため，ユーザは少なからず自身のプライバシ情報が漏洩するかも知れないという不安を抱くことになる．このような背景から，ビッグデータ技術への応用を想定したプライバシ情報を保護する技術が盛んに研究されている．

プライバシ情報の保護技術の１つに，秘匿計算というデータの秘匿性を保ちながら演算を行う技術がある．秘匿計算を実現する手法は，鍵を用いてデータを秘匿する準同型暗号 [2]を用いる手法と，鍵を用いずにデータを秘匿する秘密分散法 [3]を用いる手法 [4]の 2 つに大別される．準同型暗号は一般的に計算量が多く，演算の処理に多大な時間がかかるという問題がある [5]. そのため，実利用に向けては計算量の多い準同型暗号よりも，計算量が軽い秘密分散法を用いるというアプローチが検討されている．秘密分散法は，ユーザが持っている秘密情報を複数の異なる値（以降，分散値）に変換し，分散する手法である．秘密分散法の 1 つである Shamir の(k, n)閾値秘密分散法 [3]は，1 つの秘密情報をn個の分散値に変換し，n台のサーバに分散する．この方式の特徴は，分散したn個の分散値からk個の分散値を集めれば，元の秘密情報を復元することができるが，k個未満の情報からは，秘密情報に関する情報を一切得ることができないということである．

通常，秘密分散法を用いた秘匿計算 [4]は 3 台以上のサーバを必要とするため，ディープニューラルネットワークなどの大規模な計算を行なう際には高性能なサーバが 3 台以上必要になる．さらに，サーバ管理者側への情報漏洩のリスクを回避するためには，これらのサーバは独立した組

†1 東京理科大学大学院　工学研究科
Tokyo University of Science, Graduate School of Engineering,

†2 東京大学　生産技術研究所
The University of Tokyo, Institute of Industrial Science

Vol.2018-CSEC-82 No.60
Vol.2018-SPT-29 No.60
2018/7/26

織で安全に管理される必要がある. 加えて, 3 台のうち 1 台のサーバの処理能力が低ければ，それがボトルネックとなり全体のスループットは低下する.

そこで本論文では，1 台のサーバで計算可能な秘密分散法による秘匿計算手法を提案する．本手法では，生成した分散値に異なる乱数を乗じ, 新たな分散値系列を生成する．このとき，第三者が生成された複数の分散値を揃えても，乱数値を推定することができないため元の分散値を復元されない．1 台のサーバのみで計算を実施することにより，計算資源の節約が可能となり，サーバ管理の負担を軽減しつつ複数サーバの独立的な管理に対する必要性を解決できる．また，この秘匿計算は 1 台のサーバで秘匿計算を行うが，情報理論的な安全性を実現する．ただし，本稿では passive な攻撃者を想定する.

2. 準備

2.1 Shamir の(k,n)しきい値秘密分散法[3]

Shamir の(k,n)しきい値秘密分散法はある秘密情報sをn個の分散情報に分割し，n台のサーバに分散する手法である．特徴として，k個未満の分散情報からはsについての情報は一切漏えいしない．以下に Shamir の(k,n)しきい値秘密分散法の分散処理と復元処理を示す.

「分散処理」

1. ユーザは$s<p$かつ$n<p$の条件を満たす任意の素数pを選択する.
2. ユーザは$\mathrm{GF}(p)$の元から，n個の$x_i\ (i=0,1,2,\dots,n-1)$を選び，サーバ ID とする.
3. ユーザは$\mathrm{GF}(p)$の元から，$k-1$個の乱数$a_l\ (l=1,2,\dots,k-1)$を選び，以下の式(1)を生成する.

$$W_i = s + a_1 x_i + a_2 {x_i}^2 + \cdots + a_{k-1} {x_i}^{k-1} \pmod p \qquad (1)$$

4. ユーザは上記式のx_iに各サーバ ID を代入し，分散値W_iを計算して各サーバS_iに送信する.

「復元処理」

1. 復元に用いる分散情報を$W_i\ \ (i=0,1,2,\dots,k-1)$として, その分散情報に対応するサーバ ID を$x_i$とする.
2. 分散式にx_iとW_iを代入し，k個の連立方程式を解いて，元の秘密情報sを復元する.

2.2 TUS2 法 [6]

Shamir の(k,n)しきい値秘密分散法では秘匿乗算を行う場合，復元に必要な分散情報の数が増加するという問題点がある．そこで TUS2 法では，秘匿乗算を行う際に一時的に一方の秘密情報を復元してスカラー量とし，もう一方の秘密情報の多項式との乗算を行う．これにより，秘匿乗算を行っても復元に必要な分散情報の数は変化せず，即ち最低 2 台のサーバがあればシステムを構築できる．本方式では秘匿加減算・乗除算ができることが知られている．ただし以下に記す 3 つの条件が前提となる.

1. 秘匿乗算において秘密情報と乱数に 0 を含まない.
2. 攻撃者に知られない乱数を用いた 1 に対する分散値集合がある.
3. 演算の連続において各サーバが扱う分散値集合内の分散値の位置は固定される.

ここで，以下に記号を定義する.

- $\overline{[a]}_i$：値aに対するサーバS_iが保持する分散値.
- $[a]_i$：値aに関連するサーバS_iに保持する分散値集合.

前提条件 2 の攻撃者に知られない乱数を用いた 1 に対する分散値集合とは以下のようなものである．ここで，$\delta_j, \eta_j\ (j=0,1,\dots,n-1)$及び$\delta,\eta$は攻撃者が知らない乱数で有り，以下のように生成される．1 に対する分散値集合の生成者は応用に応じて設定できる（入力者が生成し，互いに異なる入力者の秘密情報に適用するなど）が，ここでは信頼できる第三者（サーバ）などによって生成されるとする.

$$[1]^{(\delta)}{}_i = \left(\overline{[\delta]}_i, \overline{[\delta_0]}_i, \dots, \overline{[\delta_{k-1}]}_i\right) \qquad (2)$$

「1 に対する分散値の生成」

1. k個の乱数$\delta_0,\delta_1,\dots,\delta_{k-1}$生成し，乱数$\delta=\prod_{j=0}^{k-1}\delta_j$を計算する.
2. $\delta,\delta_0,\delta_1,\dots,\delta_{k-1}$を Shamir の$(k,n)$-閾値秘密分散法で$n$台のサーバ$S_i\ (i=0,1,2,\dots,n-1)$に分散する.
3. サーバ$S_i\ (i=0,1,2,\dots,n-1)$は乱数 1 に関する分散情報として$[1]^{(\delta)}{}_i = \left(\overline{[\delta]}_i, \overline{[\delta_0]}_i, \dots, \overline{[\delta_{k-1}]}_i\right)$を保持する.

前提条件 3 は秘匿演算を繰り返すときに必要な条件であり，例えば 1 つの演算でα_j,β_jを扱ったサーバS_jは次の演算においてもα_j,β_jを扱うことを意味する.

以下に、分散処理，復元処理，および秘匿積和演算の詳細な手順を記す。本アルゴリズムにおいて，秘密情報a,b,cは$GF(p)$であり，分散処理および秘匿演算で生成する乱数も$GF(p)$である（ただし，秘匿乗算において秘密情報a,bは 0 を含まず，秘匿加減算及び秘匿除算において秘密情報に 0 を含んでも良い．また，いずれの場合でも乱数は 0 を含まない）．秘密分散の処理も含めてすべての秘匿演算はpを法として行われ，エンティティは各々の秘密情報を秘密分散して入力する入力者と，その分散値を用いて秘匿計算を行う n 台のサーバと，演算結果の分散値から復元を行う復元者からなる.

「分散処理」

1. ユーザAはk個の乱数$\alpha_0,\alpha_1,\dots,\alpha_{k-1}$を生成し，乱数$\alpha=\prod_{j=0}^{k-1}\alpha_j$を計算する.
2. 計算された乱数αと秘密情報aをかけて，αaを計算し，$\alpha a,\alpha_0,\alpha_1,\dots,\alpha_{k-1}$を Shamir の$(k,n)$しきい値秘密分散法で$n$台のサーバ$S_i\ (i=0,1,2,\dots,n-1)$に分散する.
3. サーバ$S_i\ (i=0,1,2,\dots,n-1)$はaに関する分散情報として$[a]_i = \left(\overline{[\alpha a]}_i, \overline{[\alpha_0]}_i, \dots, \overline{[\alpha_{k-1}]}_i\right)$を保持する.

「復元処理」

1. 復元者はk台のサーバよりk個の分散情報$[a]_j\ (j=0,1,\dots,k-1)$を収集する.

Vol.2018-CSEC-82 No.60
Vol.2018-SPT-29 No.60
2018/7/26

2. 収集した分散情報の$\overline{[\alpha a]}_j, \overline{[\alpha_0]}_j, \dots, \overline{[\alpha_{k-1}]}_j$から$\alpha a, \alpha_0, \dots, \alpha_{k-1}$を復元し，乱数$\alpha = \prod_{j=0}^{k-1} \alpha_j$を計算する．
3. 復元した秘匿した秘密情報αaと乱数αを用いて，以下の式(3)より秘密情報aを復元する．

$$\alpha a \times \alpha^{-1} = a \tag{3}$$

「秘匿積和演算」

入力：$[a]_j = \left(\overline{[\alpha a]}_j, \overline{[\alpha_0]}_j, \dots, \overline{[\alpha_{k-1}]}_j\right) (j = 0, 1, \dots, k-1)$
$[b]_j = \left(\overline{[\beta b]}_j, \overline{[\beta_0]}_j, \dots, \overline{[\beta_{k-1}]}_j\right) (j = 0, 1, \dots, k-1)$
$[c]_i = \left(\overline{[\lambda c]}_i, \overline{[\lambda_0]}_i, \dots, \overline{[\lambda_{k-1}]}_i\right) (j = 0, 1, \dots, k-1)$

出力：$[d]_i = [ab + c]_i =$
$\left(\overline{[\gamma'(ab + c)]}_i, \overline{[\gamma'_0]}_i, \dots, \overline{[\gamma'_{k-1}]}_i\right) (i = 0, 1, \dots, n-1)$

1. サーバS_0はk台のサーバより$\overline{[\alpha a]}_j$を収集し，一時的にαaをスカラー量として復元する．
2. 全サーバS_iにαaを送信する．
3. 全サーバS_iは以下の式を用いて，αaと$\overline{[\beta b]}_i$の乗算を行い，$\overline{[\alpha\beta ab]}_i$を計算する．

$$\overline{[\alpha\beta ab]}_i = \alpha a \times \overline{[\beta b]}_i \tag{4}$$

4. サーバS_0はk台のサーバより$\overline{[\alpha\beta ab]}_j, \overline{[\lambda c]}_j$を収集し，一時的に$\alpha\beta ab, \lambda c$のスカラー量を復元する．
5. 全サーバS_iに$\alpha\beta ab, \lambda c$を送信する．
6. 全サーバS_iは以下の式を用いて，$\alpha\beta ab$と$[1]^{(1)}{}_i$中の$\overline{[\delta]}_i$との乗算を行い，$\overline{[\alpha\beta\delta ab]}_i$を計算する．

$$\overline{[\alpha\beta\delta ab]}_i = \alpha\beta ab \times \overline{[\delta]}_i \tag{5}$$

7. 全サーバS_iは以下の式を用いて，λcと$[1]^{(2)}{}_i$中の$\overline{[\eta]}_i$との乗算を行い，$\overline{[\lambda\eta c]}_i$を計算する．

$$\overline{[\lambda\eta c]}_i = \lambda c \times \overline{[\eta]}_i \tag{6}$$

8. k台のサーバは各々$\overline{[\alpha_j]}_{l'}, \overline{[\beta_j]}_{l'}, \overline{[\lambda_j]}_{l'}, \overline{[\delta_j]}_{l'}, \overline{[\eta_j]}_l$ $(l = 0, \dots, k-1)$を収集し，$\alpha_j, \beta_j, \lambda_j, \delta_j, \eta_j$を復元し，乱数$\gamma_j$を生成する．
9. k台のサーバS_jは各々$\gamma_j/\alpha_j\beta_j\delta_j, \gamma_j/\lambda_j\eta_j$を計算し，サーバ$S_0$に送信する．
10. サーバS_0は$\gamma_j/\alpha_j\beta_j\delta_j, \gamma_j/\lambda_j\eta_j$を用いて，以下の式より$\gamma/\alpha\beta\delta, \gamma/\lambda\eta$を計算し，全サーバ$S_i$に送信する．

$$\frac{\gamma}{\alpha\beta\delta} = \prod_{j=0}^{k-1} \frac{\gamma_j}{\alpha_j\beta_j\delta_j} \tag{7}$$

$$\frac{\gamma}{\lambda\eta} = \prod_{j=0}^{k-1} \frac{\gamma_j}{\lambda_j\eta_j} \tag{8}$$

11. 全サーバS_iは以下の式を用いて，$\overline{[\gamma(ab + c)]}_i$を計算する．

$$\overline{[\gamma(ab + c)]}_i = \frac{\gamma}{\alpha\beta\delta}\left(\overline{[\alpha\beta\delta ab]}_i\right) + \frac{\gamma}{\lambda\eta}\left(\overline{[\lambda\eta c]}_i\right) \tag{9}$$

12. k台のサーバS_jは，秘匿乗算abのみであれば，$\gamma_j = \alpha_j\beta_j$とし，秘匿加算を含む場合は$\gamma_j = \gamma_j$として$\gamma_j$を Shamir の$(k, n)$-閾値秘密分散法で全サーバ$S_i$に分散する．
13. サーバS_i $(i = 0, 1, 2, \dots, n-1)$は秘密情報$ab + c$に関する分散情報として$[ab + c]_i =$
$\left(\overline{[\gamma(ab + c)]}_i, \overline{[\gamma_0]}_i, \dots, \overline{[\gamma_{k-1}]}_i\right)$を保持する．

上記秘匿積和演算$ab + c$において$c = 0$とすれば秘匿乗算となり，$a = 1$とすれば秘匿加算となる．また，式(3)における乗算をαaによる除算とし，12.の$\gamma_j = \alpha_j\beta_j$を$\gamma_j = \beta_j/\alpha_j$とすれば秘匿除算となる．また，加算を減算とすれば秘匿減算を実現できるので，上記アルゴリズムによって秘匿除算と秘匿減算を含む全ての四則演算と積和演算が実現できる．一般に，秘匿除算において除数が 0 の場合，解が定義できないため，除数が 0 である場合は排除する必要がある．TUS2 法では除数が 0 であったとしても，1.において$\alpha a = 0$が検出されるため，そこで処理を止めることにより 0 による除数を排除できる．

3. 提案手法 1（入力者＝復元者の場合）

大規模なニューラルネットワークなどを用いて膨大な量の計算を行なう際，クライアントが高性能な計算機を持たない場合にはクラウドへ計算を委託することが多い．このような環境では情報の入力者と演算結果の利用者が同一であり，まずは，このような環境への適用を想定した提案手法の詳細について述べる．即ち，クライアントは自身が入力した情報とその出力結果をクラウドにも知られたく，秘密情報の入力者と秘匿計算結果の復元者が同一である場合を考える．

提案手法はサーバが最低 2 台必要であった TUS2 法を基に 1 台のサーバのみで秘匿計算を可能にする．TUS2 法で生成した 2 つの分散値にかかる乱数を異なるものとすることで，2 つの分散値を 1 つのサーバに集めても異なる乱数を知らなければ正しく復元できない．この手法には栗原らによって提案された XOR 法 [7]を多値化した多値化法 [8]を用いることができる． 多値化法は分散値に対して、加減算と定数乗算を実現でき，TUS2 法は加減算と定数乗算によって構成されるため，多値化法で対応できる．多値化法は分散・復元処理を加減算によって実現できるため非常に効率的である．以下に分散・復元処理および秘匿積和演算$d = ab + c$の流れを示す．この手法の前提条件として，秘密情報および乱数は正の整数に限定される．

「分散処理」

1. 入力者は乱数α, β, γを生成し，式(10)を計算し，その結果をサーバに送信する．

$$\begin{aligned} \alpha a &= \alpha \times a \\ \beta b &= \beta \times b \\ \gamma c &= \gamma \times c \end{aligned} \tag{10}$$

2. 入力者は乱数δ_0, δ_1を生成し，$\frac{\delta_0}{\alpha\beta}, \frac{\delta_0}{\gamma}$を秘密分散し，
$\left[\frac{\delta_0}{\alpha\beta}\right]_0, \left[\frac{\delta_0}{\gamma}\right]_0, \left[\frac{\delta_0}{\alpha\beta}\right]_1, \left[\frac{\delta_0}{\gamma}\right]_1$を得る．

Vol.2018-CSEC-82 No.60
Vol.2018-SPT-29 No.60
2018/7/26

3. 入力者は式(11)を計算する．($\delta = \delta_0\delta_1$)

$$\left[\frac{\delta}{\alpha\beta}\right]_1 = \delta_1\left[\frac{\delta_0}{\alpha\beta}\right]_1 \qquad \left[\frac{\delta}{\gamma}\right]_1 = \delta_1\left[\frac{\delta_0}{\gamma}\right]_1 \tag{11}$$

4. 入力者は$\left[\frac{\delta_0}{\alpha\beta}\right]_0, \left[\frac{\delta_0}{\gamma}\right]_0$と$\left[\frac{\delta}{\alpha\beta}\right]_1, \left[\frac{\delta}{\gamma}\right]_1$をサーバに送信する．

「秘匿積和計算」

サーバは以下の式(12),(13)を計算する．

$$[\delta_0(ab+c)]_0 = \alpha a \times \beta b \times \left[\frac{\delta_0}{\alpha\beta}\right]_0 + \gamma c \times \left[\frac{\delta_0}{\gamma}\right]_0 \tag{12}$$

$$[\delta(ab+c)]_1 = \alpha a \times \beta b \times \left[\frac{\delta}{\alpha\beta}\right]_1 + \gamma c \times \left[\frac{\delta}{\gamma}\right]_1 \tag{13}$$

「復元処理」

1. 入力者は$[\delta_0(ab+c)]_0$と$[\delta(ab+c)]_1$を集めて$[\delta(ab+c)]_0 = \delta_1[\delta_0(ab+c)]_0$を計算し，$\delta(ab+c)$を復元する．
2. 入力者は$\delta(ab+c)$をδで割ることにより，解$ab+c$を得る．

上記秘匿積和演算$ab+c$において$c=0$とすれば秘匿乗算となり，$a=1$とすれば秘匿加算となる．また，演算を継続する場合，復元処理 1 で復元した$\delta(ab+c)$をサーバに送り，新たなαaまたは$\beta b, \gamma c$として分散処理の 2,3 を実行する．また，提案手法 1 は全乱数を入力者が知るため，分散処理 2,3 の処理を予め実行して配布しておくことができる．

3.1 提案手法 1 における安全性

提案手法 1 において想定される攻撃として，悪意のある攻撃者がサーバの中身を覗き，秘密情報を得ようとする場合を考える．

サーバにある一方の分散値は乱数δ_1がかかっているので，攻撃者はもう一方の分散値に適当な乱数δ_1'をかけ，秘密情報の復元を試みる．ここで，$\left[\frac{\delta_0}{\alpha\beta}\right]_0, \left[\frac{\delta_0}{\gamma}\right]_0, [\delta_0(ab+c)]_0$に$\delta_1'$をかけて復元した解を$\left(\frac{\delta}{\alpha\beta}\right)' = \frac{\delta}{\alpha\beta} + r_1$，$\left(\frac{\delta}{\gamma}\right)' = \frac{\delta}{\gamma} + r_2$，$\delta_0(ab+c)' = \delta_0(ab+c) + r_3$とする．($\delta_1 = \delta_1'$であれば$r_1 = r_2 = r_3 = 0$となるが，そうでないなら$r_1, r_2$は用いた分散式に応じて異なる．) ここで,どのようなδ_1'をかけても$\alpha a \times \beta b \times \left(\frac{\delta}{\alpha\beta}\right)' + \gamma c \times \left(\frac{\delta}{\gamma}\right)'$と$\delta_0(ab+c) + r_3$は一致するため，攻撃者はどれが正しい$\delta_1$か判定できない．よって，提案手法 1 は情報理論的安全性が保証される．

4. 提案手法 2（入力者≠復元者の場合）

例えば秘匿ニューラルネットワークを用いて，ユーザの持つプライバシ情報を企業の持つ学習器に学習させるシステムを構成することを考える．この場合，秘密情報はユーザのプライバシ情報であり，秘匿計算結果は修正された学習器のパラメータとなる．よって，提案手法 2 では，入力者と復元者が異なる場合について考える．

以下に分散・復元処理および秘匿積和演算$d = ab + c$の流れを示す．提案手法 1 と同様に，秘密情報および乱数は正の整数に限定される．

「前提条件」

信頼できる演算支援装置があるとする．演算支援装置における分散・復元は k=2 に限定するため，TUS2 法のように乱数等は分散せず，直接演算支援装置及びサーバに配布する．よって，式(17)(18)を計算するサーバに比べて，演算支援装置は式(16)を計算するだけなので処理が軽い．

「事前処理」

演算支援装置は予め乱数ε, μを生成し，秘密分散を行い$[\varepsilon]_0, [\varepsilon]_1, [\mu]_0, [\mu]_1$を作成し，$[\varepsilon]_0, [\mu]_0$をサーバに送信する．この処理は TUS2 法における，「1 の分散値の生成」に相当する．

「分散処理」

1. 入力者 A は秘密情報aに対して乱数α_0, α_1を生成して$\alpha = \alpha_0\alpha_1$を計算して，$\alpha_0$を演算支援装置に，$\alpha a, \alpha_1$をサーバに送る．
2. 入力者 B は秘密情報bに対して乱数β_0, β_1を生成して$\beta = \beta_0\beta_1$を計算して，β_0を演算支援装置に，$\beta b, \beta_1$をサーバに送る．
3. 入力者 C は秘密情報cに対して乱数γ_0, γ_1を生成して$\gamma = \gamma_0\gamma_1$を計算して，$\gamma_0$を演算支援装置に，$\gamma c, \gamma_1$をサーバに送る．

「秘匿積和計算」

1. サーバは乱数$\delta_{0,1}$を生成し，$\frac{\delta_{0,1}}{\alpha_1\beta_1}, \frac{\delta_{0,1}}{\gamma_1}$を演算支援装置に送る．($\delta_0 = \delta_{0,0}\delta_{0,1}$)
2. 演算支援装置は乱数$\delta_{0,0}, \delta_1$を生成し，以下の計算を行う．

$$\frac{\delta_0}{\alpha\beta\varepsilon} = \frac{\delta_{0,0}}{\alpha_0\beta_0} \times \frac{\delta_{0,1}}{\alpha_1\beta_1} \times \frac{1}{\varepsilon} \tag{14}$$

$$\frac{\delta_0}{\gamma\mu} = \frac{\delta_{0,0}}{\gamma_0} \times \frac{\delta_{0,1}}{\gamma_1} \times \frac{1}{\mu} \tag{15}$$

3. 演算支援装置は式(16)を計算する．($\delta = \delta_0\delta_1$)

$$[\delta_1\varepsilon]_1 = \delta_1[\varepsilon]_1 \qquad [\delta_1\mu]_1 = \delta_1[\mu]_1 \tag{16}$$

4. 演算支援装置は$\left(\frac{\delta_0}{\alpha\beta\varepsilon}\right), \left(\frac{\delta_0}{\gamma\mu}\right), [\delta_1\varepsilon]_1, [\delta_1\mu]_1$をサーバに送る．
5. サーバは以下の式(17),(18)を計算する．

$$[\delta_0(ab+c)]_0 = \alpha a \times \beta b \times \left(\frac{\delta_0}{\alpha\beta\varepsilon}\right) \times [\varepsilon]_0 + \gamma c \times \left(\frac{\delta_0}{\gamma\mu}\right) \times [\mu]_0 \tag{17}$$

Vol.2018-CSEC-82 No.60
Vol.2018-SPT-29 No.60
2018/7/26

$$[\delta(ab+c)]_1 = \alpha a \times \beta b \times \left(\frac{\delta_0}{\alpha\beta\varepsilon}\right) \times [\delta_1\varepsilon]_1 + \gamma c \times \left(\frac{\delta_0}{\gamma\mu}\right) \times [\delta_1\mu]_1 \quad (18)$$

「復元処理」

1. 演算支援装置は$[\delta_0\{(ab+c)\}]_0, [\delta\{(ab+c)\}]_1$を集めて式(19)を計算する．
$$[\delta\{(ab+c)\}]_0 = \delta_1[\delta_0\{(ab+c)\}]_0 \quad (19)$$
2. 演算支援装置は$[\delta\{(ab+c)\}]_0$と$[\delta\{(ab+c)\}]_1$から$\delta\{(ab+c)\}$を復元する．
3. 復元者はサーバから$\delta_{0,1}$を，演算支援装置から$\delta_{0,0}\delta_1, \delta\{(ab+c)\}$を受け取り，$\delta = \delta_{0,1} \times \delta_{0,0}\delta_1$を計算して秘匿計算結果$d = ab + c$を求める．演算を継続する場合は，$\delta\{(ab+c)\}$の復元を演算支援装置が行った後，それを新たな$\alpha a$としてサーバに送り公開すれば$\alpha_0, \alpha_1$に相当する$\delta_{0,0}, \delta_{0,1}$は演算支援装置とサーバが各々保持しているため，復元者が関与することなく演算を継続できる．

$ab + c$において$b = 1$とすれば加算$a + c$， $c = 0$とすれば乗算abを計算できる．

4.1 提案手法 2 における安全性

提案手法 2 は TUS2 法において k=2 として行っていた処理を 1 台のサーバに集め，安全性のために必要な最低限の処理を演算支援装置に行わせたものと考えることができる．サーバと演算支援装置内の情報を両方知られると秘密情報は漏洩してしまう．しかし，演算支援装置は簡単な処理のみを行うので，耐タンパ性を持たせた IC チップなどで構成することにより，演算支援装置の中を見ることをできなくすることができる．また，演算支援装置を耐タンパ化できなくても，秘密分散と同様に演算支援装置とサーバの情報は同時に知られないとすれば以下のように安全にできる．

4.1.1 外部攻撃者に対する安全性

演算支援装置は一部の乱数を知るだけで$\alpha a, \beta b, \gamma c$は知らないので，秘密情報$a, b, c$は漏洩しない．また，演算を継続する場合，演算支援装置は$\delta\{(ab+c)\}$を復元するが，δは復元しないので，出力結果も漏洩しない．

次に，攻撃者がサーバの中身を覗き，秘密情報を得ようとする場合を考える．この場合，3.1 に示した安全性と同様に説明できるので，情報理論的安全性が保証される．

4.1.2 内部攻撃者に対する安全性

まず，入力者 A,B,C のうち 1 人がサーバもしくは演算支援装置の中身を覗き，他の入力者の秘密情報を得ようとすることを想定する．仮に攻撃者が A だとすれば，A の持つ情報は，重複を削除するとサーバの場合は式(20)，演算支援装置の場合は式(21)の通りである．

$$\alpha_0, \alpha_1, a, \beta b, \beta_1, \gamma c, \gamma_1, \delta_{0,1}, \left(\frac{\delta_0}{\alpha\beta\varepsilon}\right), \left(\frac{\delta_0}{\gamma\mu}\right) \quad (20)$$

$$, [\delta_0(ab+c)]_0, [\delta(ab+c)]_1, [\varepsilon]_0, [\mu]_0, [\delta_1\varepsilon]_1, [\delta_1\mu]_1$$
$$\varepsilon, \mu, \alpha_0, \alpha_1, a, \beta_0, \gamma_0, \frac{\delta_{0,1}}{\alpha_1\beta_1}, \frac{\delta_{0,1}}{\gamma_1}, \delta_{0,0}, \delta_1, \delta\{(ab+c)\} \quad (21)$$

攻撃者はこれらの情報から，これ以上の秘密情報に関する情報を知ることはできない．B,C が攻撃者である場合についても同様である．よって，外部攻撃者に対する安全性と同等になり，情報理論的安全性が保証される．

次に，A,B,C のうち 2 人が同様の攻撃をしてくることを想定する．仮に攻撃者が A,B だとすれば，A,B で共有する情報は，重複を削除するとサーバの場合は式(22)，演算支援装置の場合は式(23)の通りである．

$$\alpha_0, \alpha_1, a, \beta_0, \beta_1, b, \gamma c, \gamma_1, \delta_{0,1}, \left(\frac{\delta_0}{\alpha\beta\varepsilon}\right), \left(\frac{\delta_0}{\gamma\mu}\right), [\delta_0(ab+c)]_0, [\delta(ab+c)]_1, [\varepsilon]_0, [\mu]_0, [\delta_1\varepsilon]_1, [\delta_1\mu]_1 \quad (22)$$

$$\varepsilon, \mu, \alpha_0, \alpha_1, a, \beta_0, \beta_1, b, \gamma_0, \frac{\delta_{0,1}}{\alpha_1\beta_1}, \frac{\delta_{0,1}}{\gamma_1}, \delta_{0,0}, \delta_1, \delta\{(ab+c)\} \quad (23)$$

攻撃者はこれらの情報から，これ以上の秘密情報に関する情報を知ることはできない．A,C と B,C が攻撃者である場合についても同様である．よって，外部攻撃者に対する安全性と同等になり，情報理論的安全性が保証される．

最後に，A,B,C のうち 1 人と復元者が攻撃者である場合を想定する．仮に攻撃者が A と復元者だとすれば，共有する情報は，重複を削除するとサーバの場合は式(24)，演算支援装置の場合は式(25)の通りである．

$$\alpha_0, \alpha_1, a, \beta b, \beta_1, \gamma c, \gamma_1, \delta_{0,1}, \delta_{0,0}\delta_1, \left(\frac{\delta_0}{\alpha\beta\varepsilon}\right), \left(\frac{\delta_0}{\gamma\mu}\right), [\delta_0(ab+c)]_0, [\delta(ab+c)]_1, ab+c, [\varepsilon]_0, [\mu]_0, [\delta_1\varepsilon]_1, [\delta_1\mu]_1 \quad (24)$$

$$\varepsilon, \mu, \alpha_0, \alpha_1, a, \beta_0, \beta_1, b, \gamma_0, \frac{\delta_{0,1}}{\alpha_1\beta_1}, \frac{\delta_{0,1}}{\gamma_1}, \delta_{0,0}, \delta_{0,1}, \delta_1, \delta\{(ab+c)\}, ab+c \quad (25)$$

攻撃者はこれらの情報から，これ以上の秘密情報に関する情報を知ることはできない．B,C が攻撃者である場合についても同様である．よって，外部攻撃者に対する安全性と同等になり，情報理論的安全性が保証される．

4.2 提案手法のメリット

従来手法と比べて提案手法には大きなメリットが 2 つある．1 つ目はサーバが 1 台のみで運用可能な点である．今までの秘密分散法での問題点であった複数サーバの独立的な管理に対する必要性を解決できることから，実運用上の情報漏洩のリスクを減らすことができ,サーバ管理の負担も軽減される. 演算支援装置は簡単な処理のみを行うので，耐タンパ性を持たせた IC チップなどで構成できれば，演算支援装置の中を見ることはできず，従来手法より安全であると言える. 2 つ目は計算資源の節約である. 一般に 3 台

Vol.2018-CSEC-82 No.60
Vol.2018-SPT-29 No.60
2018/7/26

以上のサーバが必要な秘密分散法を用いた秘匿計算に比べ，提案手法は 1 台のみで構成できる．ディープニューラルネットワークなどの大規模な演算を要する手法には高性能な計算処理能力を持つサーバが必要であるが，これが 1 台で済むのでボトルネックを減らすことになり，計算資源も節約できる．

表 1　提案手法と従来手法の比較

	提案方式 1	提案方式 2	TUS2 法
最低限必要なサーバ数	1	1	2
想定する攻撃者	Passive	Passive	Passive
安全性	情報理論的安全性	情報理論的安全性	情報理論的安全性
分散に必要な計算量	$(2C_2)$	0	$3C_0$
分散に必要な通信量	$7d_1$	$9d_1$	$6d_1$
分散に必要なラウンド数	1	1	1
復元に必要な計算量	C_3	C_3	$3C_1$
復元に必要な通信量	$2d_1$	$5d_1$	$6d_1$
復元に必要なラウンド数	1	2	1
積和計算に必要な計算量	0	0	$2C_0 + 13C_1$
積和計算に必要な通信量	0	$6d_1$	$44d_1$
積和計算に必要なラウンド数	0	2	8
計算量の合計	$(2C_2) + C_3$	C_3	$5C_0 + 16C_1$
通信量の合計	$9d_1$	$20d_1$	$56d_1$
ラウンド数の合計	2	5	10

4.3 従来手法との比較

ここでは提案手法 1 および 2 と従来手法との計算量および通信量の比較を行う．以下に使われる記号の定義を示す．

- d_1：扱うデータのサイズ
- C_0：秘密分散(Shamir 法・分散)に関する計算量
- C_1：秘密分散(Shamir 法・復元)に関する計算量
- C_2：秘密分散(多値化法・分散)に関する計算量
- C_3：秘密分散(多値化法・復元)に関する計算量

ただし，多値化法に関する計算量C_2, C_3は Shamir 法に関する計算量C_0, C_1よりも小さい．比較対象として，著者らのグループによる TUS2 法を選んだ．また，TUS2 法においては 2 台で構成した場合の計算量および通信量を導出している．

表 1 より，提案手法 1 および 2 は従来手法と比べ，計算量，通信量，ラウンド数ともに優れている．提案手法 1 は，入力者＝復元者という制約はあるが，入力者が前もって秘密分散($2C_2$)を行えることから，計算量において特に優れていることがわかる．

5. まとめ

本稿では，1 台のサーバで構成できる秘密分散法を用いた秘匿計算を提案した．複数台のサーバが必要であった既存手法と比較して，1 台のサーバで構成可能にすることより計算資源の節約が可能となり，そのうえ，サーバ管理の負担を軽減しつつ複数サーバの独立的な管理に対する必要性を解決できる．

今後，計算機への実装により本手法の性能評価を実施する必要がある．さらに，近年ビッグデータの分野で注目されているニューラルネットワークへの応用を想定した手法の改良についても検討を行っていく．

参考文献

[1] “Deeplooks - a technology that visualizes the invisible” https://deeplooks.com/. (閲覧日: 2018 年 6 月 20 日)
[2] Craig Gentry. A Fully Homomorphic Encryption Scheme. Ph.D Thesis, Stanford University, 2009.
[3] Adi Shamir. How to share a secret. Communications of the ACM, 1979, 22, (11), pp.612-613.
[4] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Compleateness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. STOC, 1988, pp.1-10.
[5] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) Fully Homomorphic Encryption without Bootstrapping. ITCS, 2012.
[6] Ahmad Akmal Aminuddin Mohd Kamal and Keiichi Iwamura. Conditionally Secure Secrecy Computation using Secret Sharing Scheme for $n < 2k - 1$. Cryptology ePrint Archive, 2017, Report 2017/718.
[7] Jun Kurihara, Shinsaku Kiyomoto, Kazuhide Fukushima, and Toshiaki Tanaka. On a Fast (k, n)-Threshold Secret Sharing Scheme. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2008, vol. E91-A, No. 9, pp. 2365-2378.
[8] 鵜田恭平, 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. 信学技報, 2017, 117(55), pp.107-114.
[9] Takeshi Shingu, Keiichi Iwamura, and Kitahiro Kaneda. Secrecy Computation without Changing Polynomial Degree in Shamir’s (K, N) Secret Sharing Scheme. In Proceedings of the 13th International Joint Conference on e-Business and Telecommunications, 2016, Volume 1: DCNET, (ICETE 2016), pp.89-94.