Vol.2015-CSEC-70 No.8
Vol.2015-SPT-14 No.8
2015/7/2

除算を含む四則演算に適用可能な秘密分散法を用いた秘匿計算手法の提案

神宮 武志[†1]　　岩村 惠市[†1]

クラウドサービスが広がるなかで，個人情報などの秘密情報の保護が大きな課題となっている．そのため，入力データを秘匿化したまま計算結果を出力する秘匿計算が注目されている．特に，秘密分散を用いた秘匿計算は情報保護と欠損耐性に優れ，なおかつ高速に計算が可能な手法としていくつかの手法が提案されている．本稿では，秘密分散法を用いてサーバ台数を変化させることなく除算を含む四則演算に対応できる秘匿計算法を提案する．特に乗算に関して本手法は，多値化方式を用いて(k,n)=(2,2)であればサーバ毎に 2 回の乗算で秘匿乗算を実行できる．よって，全体でも 4 回の乗算で高速な秘匿乗算を実現する．

Secure Computation using Secret Sharing Scheme that can be applied to Four Arithmetic Operations

TAKESHI SHINGU[†1]　　KEIICHI IWAMURA[†1]

It becomes important to protect information such as private information while Cloud service spread. Therefore there is a method called Secure multi-party computation (SMC) that is able to perform calculations of encrypted data without decrypting them. In particular, the SMC based on Secret sharing scheme is robust over an information leakage and loss. Several SMC method have been proposed. In this paper, we propose high-speed SMC based on Secret sharing scheme that can be applied to Four Arithmetic Operations.

1. はじめに

近年，クラウドコンピューティング [1]の利用やアプリの開発が盛んに行われている．クラウドは従来ユーザが自前の装置などに管理していたデータを，オンライン上のサーバに保有，管理する．そのため，ユーザは自身のデータを持ち歩くことなしに，ネットワークを介していつでも必要なデータにアクセスできるようになり，クラウド上に大量のデータを保存しておくこともできる．さらに，短期間での導入が可能というメリットもある．しかし，クラウドを使用する際には，基本的にすべてのデータがクラウド上に集約されるため，安全性に関して考慮しなければならない問題点がある．その一つは，サーバやネットワークの障害などによって，クラウドサービスが利用できなくなってしまう場合である．さらに集中的なデータの管理は攻撃の標的となりやすく，情報流出の危険性が高まる．特に，企業の機密情報などが流出した場合には，致命的な被害を生じさせる場合もある．また，情報を守るために通常暗号化を行うが，秘匿計算まで考慮に入れた場合，一般に準同型性を持つ暗号化は処理が重く，秘匿計算しようとした際に時間がかかってしまう．以上の問題を解決するため，クラウドシステムに「秘密分散法」を適用することが考えられている．

秘密分散法（secret sharing scheme） [2]-[6]は，一つの情報を異なる複数の情報（分散情報）に変換し，その分散情報のうち一定数以上が集まれば元の情報の復元が可能だが，一定数未満では元の情報は復元されないという手法である．これによって，サーバやネットワークの障害などによりデータの一部が使えなくても，それが一定数以下ならば元の情報が復元でき，さらに一定数以上の情報が漏洩しない限り情報漏洩は起こらないという安全な情報管理システムが実現できる．(k,n) しきい値秘密分散法は，秘密情報 s から生成された n 個の分散情報のうち，(1) 任意の k 個から秘密情報sを復元することができる，(2) $k-1$個以下の分散情報からは，秘密情報 s に関する情報は一切得ることができない，という特徴を持つ．

データを秘匿化したまま演算を行う技術として秘匿計算がある．特に，Shamir の (k,n) しきい値秘密分散法 [2]を用いた秘匿計算は，加減算は簡単に実現できるが乗算に関しては乗算結果の復元に必要なサーバ台数がk台ではなく $2k-1$ 台に変化してしまうという問題がある．除算に関しては演算自体が困難である．そこで，M.Ben-Or らによる方式 [7]，千田らによる方式 [8]，渡辺らによる方式 [9]などで上記の問題を解決する秘匿計算手法が提案され

†1 東京理科大学
Tokyo University of Science

Vol.2015-CSEC-70 No.8
Vol.2015-SPT-14 No.8
2015/7/2

ている．

提案方式では，秘密情報に乱数をかけて秘匿化した後秘密分散し，乗算や除算の際はその秘匿化された秘密情報を一時的にスカラー量として復元できるようにすることで，必要なサーバ台数を変化させずに乗算や除算を行うことを実現した．

本論文の構成を以下に示す．2 章において秘密分散法とその 1 つである Shamir らによる秘密分散法，千田らによる秘密分散法を用いた秘匿計算手法を紹介する．3 章では提案方式について説明し，4 章では従来方式との比較を行う．5 章では提案方式の安全性について検討し，6 章ではまとめを行う．

2. 従来方式

2.1 高橋らの多値化方式

この手法は高速な秘密分散法として栗原らが提案した XOR を用いた秘密分散 [10]においてビット列として扱っている秘密情報を数値として扱えるように多値化を行い，秘匿計算に対応可能な秘密分散手法 [11]である．この手法は加法準同型性を持つため秘匿計算への適用が可能である．

[分散]

(1) 情報提供者 A は秘密情報 S を $n-1$ 個の部分秘密情報に分割し，$S_0 \in \{0\}^d$ を生成する．

$$S = S_1 \| S_2 \| \cdots \| S_{n-1}$$

(2) A は d ビットの乱数 r_β^α を全て独立に $(k-1)n-1$ 個生成する．

$$r_0^0, r_1^0, \cdots, r_{n-2}^0, r_0^1, \cdots, r_{n-2}^1, r_{n-2}^1, \cdots, r_0^{k-2}, \cdots, r_{n-1}^{k-2}$$

(3) A は部分分散情報 $W_{(i,j)}$ を以下の式により $0 \leqq i \leqq n-1, 0 \leqq j \leqq n-2$ においてそれぞれ生成する．

$$\mathrm{W}_{(i,j)} = S_{j-1} + \left\{ \sum_{h=0}^{k-2} r_{h \cdot i+j}^h \right\}$$

$$(0 \leqq i \leqq n-1, 0 \leqq j \leqq n-2)$$

次のとき S_{j-i} の符号を反転する．

$$i = 1 \cap j = 2,3$$
$$i \geq 2 \cap j = 1$$

(4) A は $0 \leq i \leq n-1$ において各部分分散情報 $W_{(i,0)}, W_{(i,1)}, \cdots, W_{(i,n-2)}$ を連結し，分散情報 W_i を生成し各サーバに配布する．

$$W_i = W_{(i,0)} \| W_{(i,1)} \| \cdots \| W_{(i,n-2)}$$

[復元]

(1) 復元に用いる分散情報を $W_{t_0}, \ldots, W_{t_{k-1}}$ とする $(0 \leq t_0 \leq \cdots \leq t_k \leq n-1)$． k 個の分散情報を部分分散情報に分割する．

$$\mathrm{W}_{t_0} \rightarrow W_{(t_0,0)}, W_{(t_0,1)}, \cdots, W_{(t_0,n-2)}$$
$$\vdots$$
$$W_{t_{k-1}} \rightarrow W_{(t_{k-1},0)}, W_{(t_{k-1},1)}, \cdots, W_{(t_{k-1},n-2)}$$

(2) 分割した部分分散情報を以下のように表し 2 進数ベクトル $V_{(t_i,j)}$ を生成する．

部分分散情報 $W_{(t_i,j)}$ の場合

$$W_{(t_i,j)} = V_{(t_i,j)} \cdot R_{(k,n)}$$

$$R_{(k,n)} = \left(S_1, \cdots, S_{n-2}, r_0^0, \cdots, r_{n-2}^0, r_0^1, \cdots, r_{n-1}^1, \cdots, r_0^{k-2}, \cdots, r_{n-1}^{k-2}\right)^T$$

(3) (2)で集まった $V_{(t_0,0)}, \cdots, V_{(t_{k-1},n-2)}$ の $k(n-1)$ 個のベクトルから以下の 2 進数の $\{k(n-1) \times (kn-2)\}$ の行列 $M_{(t_0,\cdots,t_{k-1})}^{(k,n)}$ を生成する．

$$M_{(t_0,\cdots,t_{k-1})}^{(k,n)} = V_{(t_0,0)}, \cdots, V_{(t_0,n-2)}, \cdots, V_{(t_{k-1},0)}, \cdots, V_{(t_{k-1},n-2)}$$

(4) 集まった全ての部分分散情報を $k(n-1)$ 元のベクトル $W_{(t_0,\cdots,t_{k-1})}$ と表す

$$W_{(t_0,\cdots,t_{k-1})} = \left(W_{(t_0,0)}, \cdots, W_{(t_0,n-2)}, \cdots, W_{(t_{k-1},0)}, \cdots, W_{(t_{k-1},n-2)}\right)^T$$

$$W_{(t_0,\cdots,t_{k-1})} = M_{(t_0,\cdots,t_{k-1})}^{(k,n)} \cdot R_{(k,n)}$$

ここで，行列 $M_{(t_0,\cdots,t_{k-1})}^{(k,n)}$ を Guass-Jordan の消去法（掃き出し法）を用いて対角化処理を行う．これによって，全ての部分分散情報に該当する部分を求める．

(5) 全ての部分分散情報を連結して秘密情報を復元する．

$$S = S_1 \| S_2 \| \cdots \| S_{n-1}$$

2.2 千田らの方式

この手法は $(k,n) = (2,3)$ のシステムに特化した秘密分散，秘匿計算手法である．秘密分散手法として，乱数を用いて各サーバの分散情報を定め，それらを加算することで元の秘密情報を復元する加算的秘密分散法を用いている．

[分散]

入力：$a \in \boldsymbol{Z}/p\boldsymbol{Z}$

出力：$[a]_i$

(1) 情報提供者 A は乱数 $a_0, a_1 \in \boldsymbol{Z}/p\boldsymbol{Z}$ を生成する．

(2) A は $a_2 \coloneqq a - a_0 - a_1$ を計算する．

(3) A はサーバ $P_i (i = 0,1,2)$ に，分散情報として $[a]_i \coloneqq (a_i, a_{i+1})$ を送信する．

[復元]

入力：$[a]_i$

出力：$a \in \boldsymbol{Z}/p\boldsymbol{Z}$

(1) 復元者は任意の 2 台のサーバから α_0, α_1, α_2 を収集する．

(2) $a = \alpha_0 + \alpha_1 + \alpha_2$ を計算する．

[加減算]

入力：$[a]_i, [b]_i$

出力：$[a+b]_i$

(1) サーバ P_i は $[a]_i, [b]_i$ より，$[a+b]_i = (a_i + b_i, a_{i+1} + b_{i+1})$ を計算する．

[乗算]

入力：$[a]_i, [b]_i$

出力：$[ab]_i$

(1) P_0 は乱数 r_1, r_2, $c_0 \in \boldsymbol{Z}/p\boldsymbol{Z}$ を生成し，$c_1 \coloneqq (a_0 + a_1)(b_0 + b_1) - r_1 - r_2 - c_0$ を計算する．P_1, P_2 にそれぞれ $(r_1, c_1), (r_2, c_0)$ を送信し，$[ab]_0 \coloneqq (c_0, c_1)$ を分散情報とする．

(2) P_1, P_2 はそれぞれ $y \coloneqq a_1 b_2 + a_2 b_1 + r_1, z \coloneqq a_2 b_0 + a_0 b_2 + r_2$ を計算し P_2, P_1 に送信する．

(3) P_1, P_2 は $c_2 \coloneqq y + z + a_2 b_2$ を計算してそれぞれ $[ab]_1 \coloneqq (c_1, c_2), [ab]_2 \coloneqq (c_2, c_0)$ を分散情報とする．

3. 提案方式

提案方式は，秘密情報に乱数をかけて秘匿化（以降，秘匿化秘密情報と呼ぶ）した後秘密分散し，乗算や除算の際は秘匿化秘密情報を一時的に復元してスカラー量とし，それをもう一方の分散情報にかけ合わせることにより，復元に必要なサーバ台数を変化させずに乗算や除算を行うことを実現する．特に，$(k, n) = (2,2)$ のとき，秘密分散法として多値化方式 [11] を用いることで乗算回数を減らし高速な秘匿乗算手法を実現する．本章では，秘密分散・復元，乗算，加減算，除算の順に説明する．なお，情報提供者 A,B は秘密情報 $a, b \in \boldsymbol{Z}/p\boldsymbol{Z}$（$p$ は素数）を持つ．また，計算は基本的に $\boldsymbol{Z}/q\boldsymbol{Z}$（$q$ は $q > p^2$ を満たす素数）上で行われるものとし，秘密情報及び，生成する乱数は 0 を含まないとする．また，除算では必ず割り切れる（解が整数になる）数を扱うとする．以下では $(k, n) = (2,2)$ の場合を例として，秘密情報 a を提案方式の秘密分散をしたときのサーバ P_i の分散情報を $[a]_i$ と表す．一般的な場合を付録に示す．

図 1 に提案方式の全体図を示す．

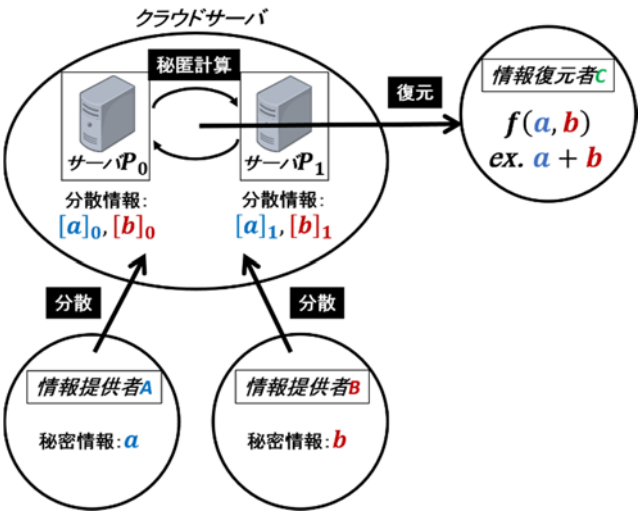

図 1　提案方式の全体図

3.1 秘密分散

入力：a

出力：$[a]_i \coloneqq (W'_{ai}, W_{a_0 i}, W_{a_1 i})$ $(i = 0,1)$

(1) 情報提供者 A は乱数 $\alpha_0, \alpha_1, r_a, r_{a0}, r_{a1} \in \boldsymbol{Z}/p\boldsymbol{Z}$ を生成し，$\alpha = \alpha_0 \alpha_1$ を計算する．

(2) A は以下を計算する．

$$W'_{a0} = \alpha a + r_a$$
$$W_{a_0 0} = \alpha_0 + r_{a0}$$
$$W_{a_1 0} = \alpha_1 + r_{a1}$$
$$W'_{a1} = r_a$$
$$W_{a_0 1} = r_{a0}$$
$$W_{a_1 1} = r_{a1}$$

(3) A はサーバ P_0，P_1 にそれぞれ，$[a]_0 \coloneqq (W'_{a0}, W_{a_0 0}, W_{a_1 0})$，$[a]_1 \coloneqq (W'_{a1}, W_{a_0 1}, W_{a_1 1})$ を送信する．

3.2 復元

入力：$[a]_i = W'_{ai}, W_{a_0 i}, W_{a_1 i}$ $(i = 0,1)$

出力：a

(1) 復元者は P_0, P_1 からそれぞれ $[a]_0, [a]_1$ を収集する．

(2) 復元者は以下のようにして秘密情報 a を復元する．なお，α_0^{-1} は α_0 の逆元を示す．

$$\alpha a = W'_{a0} - W'_{a1}$$
$$\alpha_0 = W_{a_0 0} - W_{a_0 1}$$
$$\alpha_1 = W_{a_1 0} - W_{a_1 1}$$
$$a = \alpha a \times \alpha_0^{-1} \times \alpha_1^{-1}$$

3.3 乗算

前提として情報提供者 A, B は秘密情報 $a, b \in \boldsymbol{Z}/p\boldsymbol{Z}$ を持ち，サーバ P_0, P_1 は秘密情報 a, b の分散情報を保持しているとする．なお秘密情報 b は 3.1 節で a に対して示した秘密分散と同様に以下のように分散されているとする．

$$W'_{b0} = \beta b + r_b$$
$$W_{b_0 0} = \beta_0 + r_{b0}$$
$$W_{b_1 0} = \beta_1 + r_{b1}$$
$$W'_{a1} = r_b$$
$$W_{b_0 1} = r_{b0}$$
$$W_{b_1 1} = r_{b1}$$
$$[b]_0 \coloneqq (W'_{b0}, W_{b_0 0}, W_{b_1 0})$$
$$[b]_1 \coloneqq (W'_{b1}, W_{b_0 1}, W_{b_1 1})$$

乗算手順，すなわち a, b の分散情報から $c = ab$ の分散情報を生成する手順を以下に示す．

入力：$[a]_i, [b]_i$ $(i = 0,1)$

出力：$[c]_j \coloneqq [ab]_i$

(1) P_0, P_1 はそれぞれ $(W'_{a0}, W_{a_1 0}, W_{b_1 0}), (W'_{a1}, W_{a_0 1}, W_{b_0 1})$ を P_1, P_0 に送信する．

(2) P_0, P_1 は以下より αa を計算する．

$$\alpha a = W'_{a0} - W'_{a1}$$

(3) P_0 は以下のようにして W'_{c0}, $(\alpha_0, \beta_0), (\alpha_1, \beta_1)$を計算する.

$$W'_{c0} = W'_{b0} \times \alpha a = \alpha a(\beta b + r_b)$$
$$\alpha_0 = W_{a_0 0} - W_{a_0 1}$$
$$\beta_0 = W_{b_0 0} - W_{b_0 1}$$
$$\alpha_1 = W_{a_1 0} - W_{a_1 1}$$
$$\beta_1 = W_{b_1 0} - W_{b_1 1}$$

(4) P_1 は以下のようにして W'_{c1}, $(\alpha_0, \beta_0), (\alpha_1, \beta_1)$を計算する.

$$W'_{c1} = W'_{b1} \times \alpha a = \alpha a r_b$$
$$\alpha_0 = W_{a_0 0} - W_{a_0 1}$$
$$\beta_0 = W_{b_0 0} - W_{b_0 1}$$
$$\alpha_1 = W_{a_1 0} - W_{a_1 1}$$
$$\beta_1 = W_{b_1 0} - W_{b_1 1}$$

(5) P_0は $\alpha_0\beta_0$ を計算し以下のように秘密分散する（r_{c0} は乱数，P_1 に$W_{c_0 1}$を送信する）.

$$W_{c_0 0} = \alpha_0\beta_0 + r_{c0}$$
$$W_{c_0 1} = r_{c0}$$

(6) P_1は $\alpha_1\beta_1$ を計算し以下のように秘密分散する（r_{c1} は乱数，P_0 に$W_{c_1 0}$を送信する）.

$$W_{c_1 0} = \alpha_1\beta_1 + r_{c1}$$
$$W_{c_1 1} = r_{c1}$$

(7) P_0, P_1 はそれぞれ
$[c]_0 \coloneqq (W'_{c0}, W_{c_0 0}, W_{c_1 0}), [c]_1 \coloneqq (W'_{c1}, W_{c_0 1}, W_{c_1 1})$ を
$c = ab$ の分散情報とする.

3.4 加減算

a, bの分散情報から$d = a \pm b$の分散情報を生成する手順を以下に示す.

入力：$[a]_i, [b]_i\ (i = 0,1)$
出力：$[d]_j \coloneqq [a \pm b]_i$

(1) P_0, P_1 はそれぞれ$(W_{a_1 0}, W_{b_1 0}), (W_{a_0 1}, W_{b_0 1})$ をP_1，P_0 に送信する.

(2) P_0 は以下のようにして(α_0, β_0)を復元し，$\alpha_0\beta_0^{-1}$ を計算しP_1 に送信する.

$$\alpha_0 = W_{a_0 0} - W_{a_0 1}$$
$$\beta_0 = W_{b_0 0} - W_{b_0 1}$$

(3) P_1 は以下のようにして(α_1, β_1)を復元し，$\alpha_1\beta_1^{-1}$ を計算しP_0 に送信する.

$$\alpha_1 = W_{a_1 0} - W_{a_1 1}$$
$$\beta_1 = W_{b_1 0} - W_{b_1 1}$$

(4) P_0, P_1 は以下より $\alpha\beta^{-1}$ を復元する

$$\alpha\beta^{-1} = \alpha_0\beta_0^{-1} \times \alpha_1\beta_1^{-1}$$

(5) P_0 は以下より W'_{d0} を計算する.

$$W'_{d0} = W'_{a0} \pm \alpha\beta^{-1}W'_{b0} = \alpha(a \pm b) + r_a + \alpha\beta^{-1}r_b$$

(6) P_1 は以下より W'_{d1} を計算する.

$$W'_{d1} = W'_{a1} \pm \alpha\beta^{-1}W'_{b1} = r_a + \alpha\beta^{-1}r_b$$

(7) P_0, P_1 はそれぞれ
$[d]_0 \coloneqq (W'_{d0}, W_{a_0 0}, W_{a_1 0}), [d]_1 \coloneqq (W'_{d1}, W_{a_0 0}, W_{a_1 0})$ を
$d = a \pm b$ の分散情報とする.

3.5 除算

a, b の分散情報から $e = b/a$ の分散情報を生成する手順を以下に示す.

入力：$[a]_i, [b]_i\ (i = 0,1)$
出力：$[e]_j \coloneqq [ba^{-1}]_i$

(1) P_0, P_1 はそれぞれ $(W'_{a0}, W_{a_1 0}, W_{b_1 0}), (W'_{a1}, W_{a_0 1}, W_{b_0 1})$ をP_1, P_0 に送信する.

(2) P_0, P_1 は以下より αa を計算する.

$$\alpha a = W'_{a0} - W'_{a1}$$

(3) P_0 は以下より W'_{e0} を計算する.

$$W'_{e0} = (\alpha a)^{-1}W'_{b0} = (\alpha a)^{-1}(\beta b + r_b)$$

(4) P_1 は以下より W'_{e1} を計算する.

$$W'_{e1} = (\alpha a)^{-1}W'_{b1} = (\alpha a)^{-1}r_b$$

(5) P_0 は (α_0, β_0) を復元し，$\beta_0\alpha_0^{-1}$を計算し，以下のように秘密分散する（r_{e0} は乱数）.

$$W_{e_0 0} = \beta_0\alpha_0^{-1} + r_{e0}$$
$$W_{e_0 1} = r_{e0}$$

(6) P_1 は (α_1, β_1) を復元し，$\beta_1\alpha_1^{-1}$を計算し，以下のように秘密分散する（r_{e1} は乱数）.

$$W_{e_1 0} = \beta_1\alpha_1^{-1} + r_{e1}$$
$$W_{e_1 1} = r_{e1}$$

(7) P_0, P_1 はそれぞれ
$[e]_0 \coloneqq (W'_{e0}, W_{e_0 0}, W_{e_1 0})$，$[e]_1 \coloneqq (W'_{e1}, W_{e_0 1}, W_{e_1 1})$ を
$e = b/a$ の分散情報とする.

3.6 定数乗算

a の分散情報と定数 C から $f = Ca$ の分散情報を生成する手順を以下に示す.

入力：$[a]_i, C$
出力：$[f]_j \coloneqq [Ca]_i$

(1) P_0 は以下のように W'_{f0} を計算する.

$$W'_{f0} = C(\alpha a + r_a)$$

(2) P_1 は以下のように W'_{f1} を計算する.

$$W'_{f1} = Cr_a$$

(3) P_0，P_1 はそれぞれ
$[f]_0 \coloneqq (W'_{f0}, W_{a_0 0}, W_{a_1 0})$，$[f]_1 \coloneqq (W'_{f1}, W_{a_0 0}, W_{a_1 0})$ を
$f = Ca$ の分散情報とする.

3.7 秘匿計算の組み合わせ

本提案方式では，各秘匿計算アルゴリズムにおいて各演算結果の分散情報を生成しているため，アルゴリズムを組

み合わせた連続での演算が可能である．例えば，$a \times b + c$ を出力したいときは，$\mathrm{Mul}([a]_i, [b]_i) = [ab]_i$ を計算した後 $\mathrm{Add}([ab]_i, [c]_i) = [ab + c]_i$ を計算することで演算結果 $a \times b + c$ の出力が可能である．(Mul，Add は提案方式の秘匿乗算，秘匿加算を示す)

4. 評価

以下では，従来方式（千田らの方式）と提案方式の比較を行う．なお，従来方式は秘密情報を 3 台のサーバに秘密分散し，3 台のサーバを用いて秘匿計算し，任意の 2 台のサーバより演算結果の復元を行うのに対し，提案方式は秘密情報を 2 台のサーバに秘密分散し，2 台のサーバを用いて秘匿計算し，2 台のサーバより演算結果の復元を行うという点で異なる．ただし，従来方式では 3 台のサーバ中 1 台でも稼動していない場合乗算できず，2 台のサーバが攻撃されると秘密が漏洩するので，欠損耐性と情報漏えい耐性いう観点からは同等と考えられる．ただし，従来方式は秘匿除算を実現しないので，比較は省略する．

4.1 計算量

提案方式，従来方式において最も計算量を必要とする演算は乗算・除算である．そこで，以下では各アルゴリズムにおいて全サーバが実行する乗算・除算の回数の合計回数を計算量として比較を行う．

秘匿乗算：従来方式では，秘密分散において乗算・除算 0 回，秘匿乗算において乗算 6 回，復元において乗算・除算 0 回となっている．提案方式では，秘密分散において乗算 2 回，秘匿乗算において乗算 4 回，復元において乗算 1 回，除算 1 回となっている．上記より，分散・秘匿乗算・復元の一連の流れにおける計算量は従来方式の方が少ないが，クラウド内で繰り返し秘匿乗算を行う場合，秘匿乗算における計算量が重要となってくる．すなわち，従来方式は乗算 6 回必要なのに対し，提案方式では 4 回の乗算で秘匿乗算が可能であり，連続の演算により分散・復元の計算量が無視出来るような場合，提案方式は$2/3$ の計算量で秘匿乗算が可能である．

秘匿加減算：提案方式では乗算 5 回，除算 2 回を必要とするのに対し，従来方式では加減算のみで計算が可能である．

4.2 記憶容量

1 つの秘密情報に対するサーバ 1 台あたりの分散情報の記憶容量の比較を行う．なお，各情報のサイズは同じとする．従来方式では，サーバ P_0 が保存する分散情報は $[a]_0 \coloneqq (a_0, a_1)$ であるのに対し，提案方式では，サーバ P_0 が保存する分散情報は $[a]_0 \coloneqq (W'_{a0}, W_{a_0 0}, W_{a_1 0})$ である．サーバ 1 台あたりの記憶容量は従来方式と比べ $3/2$ 倍必要となる．

4.3 通信量

各情報のサイズは同じとし，通信を行う回数の全体の合計回数を通信量として比較を行う．

秘密分散：従来方式では，情報提供者 A が秘密情報から生成した 2 個の分散情報 $[a]_i = (a_i, a_{i+1})$ を 3 台のサーバに送信するので通信回数は 6 回である．提案方式では，情報提供者 A が秘密情報から生成した 3 個の分散情報 $[a]_i = (W'_{ai}, W_{a_0 i}, W_{a_1 i})$ を 2 台のサーバに送信するので通信回数は 6 回である．よって通信量等しい．

秘匿乗算:従来方式では 6 回, 提案方式では 8 回である．よって，従来方式の $4/3$ 倍となる..

秘匿加減算：従来方式では 0 回，提案方式では 4 回である．

復元：従来方式では 4 回，提案方式では 6 回である．よって，従来方式の $3/2$ 倍となる．

5. 提案方式の安全性

5.1 秘密分散の安全性

提案方式では以下の表 1 のようにして 2 台のサーバに秘密分散する． a は秘密情報，α_0，α_1，r_a，r_{a0}，r_{a1} は乱数である．

表 1　各サーバが持つ分散情報

P_0 が持つ分散情報	$W'_{a0} = \alpha a + r_a$ $W_{a_0 0} = \alpha_0 + r_{a0}$ $W_{a_1 0} = \alpha_1 + r_{a1}$
P_1 が持つ分散情報	$W'_{a1} = r_a$ $W_{a_0 1} = r_{a0}$ $W_{a_1 1} = r_{a1}$

例えば，αa の秘密分散は，αa に乱数 r_a を加算しその加算結果と r_a をそれぞれ 2 台のサーバに配布することで分散を行う．ここで秘密情報も乱数であると仮定すれば，乱数 r_a を加算した結果である $W'_{a0} (= \alpha a + r_a)$ と $W'_{a1} (= r_a)$ は互いに独立で一様ランダムである．すなわち以下のことが成り立つ．

$$H(\alpha a) = H(\alpha a | W'_{a0}) = H(\alpha a | W'_{a1})$$
$$H(\alpha_0) = H(\alpha_0 | W_{a_0 0}) = H(\alpha_0 | W_{a_0 1})$$
$$H(\alpha_1) = H(\alpha_1 | W_{a_1 0}) = H(\alpha_1 | W_{a_1 1})$$

秘密情報 a の復元は，$\alpha a, \alpha_0, \alpha_1$ を全て復元することによってはじめて復元することができる. よって以下が成り立つ.

$$H(a) = H(a | [a]_0) = H(a | [a]_1)$$

すなわちしきい値個以下の分散情報 $[a]_i$ から秘密情報が漏洩することはない.

5.2 秘匿乗算の安全性

秘匿乗算においてサーバで一度 αa を復元するが，αa か

ら秘密情報 a が漏洩することはないことを以下に示す. αa は秘密情報 a に乱数 α をかけることによって秘匿化を行っている．ここで例として，法を 7 としたとき秘匿化秘密情報$\alpha a = 5$ のときの，乗算による秘匿化（$\alpha \times a$）の考えられる秘密情報 a と秘匿化鍵 α の組み合わせを表 2 に示す.

表 2　乗算による情報の秘匿化の安全性

	秘密情報 a	秘匿化鍵 α	秘匿化秘密情報αa
乗算	6	2	5
	5	1	
	4	3	
	3	4	
	2	6	
	1	5	

表 2 より，秘匿化秘密情報$\alpha a = 5$ のとき，秘密情報 a は全ての値をとることが分かる．よって以下の式が成り立つ.

$$H(a) = H(a|\alpha a)$$

5.3 秘匿乗算と秘匿加算を組み合わせたときの安全性

提案方式における秘匿乗算では，秘匿化秘密情報 αa をサーバで一時的に復元することにより高速な乗算を実現しているが，秘匿加算において復元者は加算結果の復元時に α を得るため，αa から秘密情報 a を知り得てしまう．よって秘密情報の安全性を一定に保つために，定期的または演算後などにおいて分散情報を更新する必要がある.

6. まとめ

本稿では，$(k, n) = (2,2)$ において，従来よりも高速に秘匿乗算を実行できる秘匿計算手法を提案した．従来方式に比べ，サーバの通信量，記憶容量は多くなるが，乗算に特化した秘密分散を用いることにより，計算量の少ない秘匿乗算を実現した．また，本提案方式は多値化方式を用いているが，秘密分散法は任意の秘密分散法でよく，記憶容量に優れた高橋らの秘密分散法 [12]やランプ型秘密分散法 [4]など既存の手法の適用が可能である.

参考文献

[1] P. Mell and T. Grance, The NIST Definition of Cloud Computing. National Institute of Standards and Technology (2011)

[2] A. Shamir, How to share a secret. Communications of the ACM, 22, (11), pp. 612-613 (1979)

[3] G. R. Blakley, Security of ramp schemes. CRYPTO ’84, pp. 242-268 (1984)

[4] 山本博資, (k, L, n) しきい値秘密分散システム, 電子通信学会論文誌 vol.J68-A, no.9, pp.945-952 (1985)

[5] H. Krawczyk, Secret Sharing Made Short. CRYPTO ’93, pp. 136-146 (1994)

[6] P.Feldman, A practical scheme for non-interactive verifiable secret sharing. 28th IEEE Symposium on the Foundations of Computer Science, pp. 427-438 (1987)

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson, Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation, STOC ’88, pp.1-10, ACM Press (1988)

[8] 千田浩司, 五十嵐大, 濱田浩気, 高橋克巳, エラー検出可能な軽量 3 パーティ秘匿関数計算の提案と実装評価, 情報処理学会論文誌, Vol.52, No.9, pp.2674-2685 (2011)

[9] 渡辺泰平, 岩村惠市, 秘密分散法を用いたサーバ台数変化がない乗算手法, 第 63 回 CSEC 研究会 (2013)

[10] 栗原淳, 清本晋作, 福島和英, 田中俊昭, 排他的論理輪を用いた高速な(4,n)閾値秘密分散法と(k,n)閾値法への拡張, ISEC2007-4, pp.23-30 (2007)

[11] 高橋加寿子, 須賀祐治, 岩村惠市, XOR を用いる秘密分散法の多値化とそれを用いた秘匿計算法, 第 65 回 CSEC 研究会 (2014)

[12] 高橋慧, 小林士郎, 岩村惠市, 記憶容量削減と計算量的安全性及び復元の独立性を実現するクラウドに適した秘密分散法, 情報処理学会論文誌, Vol.54, No.9, pp.2146-2155 (2013)

付録

本稿では $(k, n) = (2,2)$ の場合を示したが，提案方式は任意の (k, n) でも実行が可能である．すなわち，秘密情報を n 台のサーバに秘密分散し， k 台のサーバを用いて秘匿計算し，その k 台のサーバより演算結果の復元を行う．また，提案方式で用いる任意の (k, n) しきい値秘密分散により生成される分散情報を $\overline{[a]}_i$ と表す.

A　提案方式（(k, n)の場合）

A.1　秘密分散

入力：a

出力：$[a]_i \coloneqq \left(\overline{[\alpha a]}_i, \overline{[\alpha_1]}_i, \cdots, \overline{[\alpha_k]}_i\right)(i = 1,2, \cdots, n)$

(1) 情報提供者 A は k 個の乱数 $\alpha_1, \cdots, a_k \in \boldsymbol{Z}/p\boldsymbol{Z}$を生成し，$\alpha = \prod_{i=1}^{k} \alpha_i$ を計算する.

(2) A は αa を計算し，$\alpha a, \alpha_1, \cdots, \alpha_k$ を任意の(k, n)しきい値秘密分散を用いて分散情報$\left(\overline{[\alpha a]}_i, \overline{[\alpha_1]}_i, \cdots, \overline{[\alpha_k]}_i\right)$ を生成する.

(3) A はサーバ P_i に$[a]_i \coloneqq \left(\overline{[\alpha a]}_i, \overline{[\alpha_1]}_i, \cdots, \overline{[\alpha_k]}_i\right)$を送信する.

A.2　復元

入力：$[a]_j \coloneqq \left(\overline{[\alpha a]}_j, \overline{[\alpha_1]}_j, \cdots, \overline{[\alpha_k]}_j\right)(j = 1,2, \cdots, k)$

Vol.2015-CSEC-70 No.8
Vol.2015-SPT-14 No.8
2015/7/2

出力：a

(1) 復元者は P_j からそれぞれ $[a]_j$ を収集する．

復元者は $\alpha a, \alpha_1, \cdots, \alpha_k$ を復元し，以下より a を復元する．

$$\left(\overline{[\alpha a]}_1, \cdots, \overline{[\alpha a]}_k\right) \rightarrow \alpha a$$

$$\alpha = \prod_{i=1}^{k} \alpha_i$$

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

A.3 乗算

前提として情報提供者 A, B は秘密情報 $a, b \in \mathbf{Z}/p\mathbf{Z}$ を持ち，サーバ P_0, P_1 は秘密情報 a, b の分散情報を保持しているとする．

a, b の分散情報から $c = ab$ の分散情報を生成する手順を以下に示す．

入力：$[a]_j, [b]_j \ (j = 1,2,\cdots,k)$

出力：$[c]_j = [ab]_j$

(1) P_j は P_1 に $\overline{[\alpha a]}_j$ を送信する．

(2) P_1 は αa を復元し演算に参加する $P_1, \cdots, P_k$ に送信する．

$$\left(\overline{[\alpha a]}_1, \cdots, \overline{[\alpha a]}_k\right) \rightarrow \alpha a$$

(3) P_j は A.1(2)の秘密分散に対応する秘匿定数乗算を用いて，$\overline{[\beta b]}_j, \alpha a$ から $\overline{[\alpha\beta ab]}_j$ を計算する．

(4) P_j は $\overline{[\alpha_j]}_1, \cdots, \overline{[\alpha_j]}_k$，$\overline{[\beta_j]}_1, \cdots, \overline{[\beta_j]}_k$ を収集し α_j, β_j を復元する．P_j は $\alpha_j\beta_j$ を計算し，任意の (k, n)しきい値秘密分散を用いてサーバ $P_1, \cdots, P_k$ へ $\alpha_j\beta_j$ を秘密分散する．

$$\overline{[\alpha_j\beta_j]}_1, \cdots, \overline{[\alpha_j\beta_j]}_k \ (j = 1,2,\cdots,k)$$

(5) P_j は $[c]_j \coloneqq \left(\overline{[\alpha\beta ab]}_j, \overline{[\alpha_1\beta_1]}_j, \cdots, \overline{[\alpha_k\beta_k]}_j\right)$ を $c = ab$ の分散情報とする．

A.4 加減算

a, bの分散情報から$d = a \pm b$の分散情報を生成する手順を以下に示す．

入力：$[a]_j, [b]_j \ (j = 1,2,\cdots,k)$

出力：$[d]_j \coloneqq [a \pm b]_j$

(1) P_j は $\overline{[\alpha_j]}_1, \cdots, \overline{[\alpha_j]}_k$，$\overline{[\beta_j]}_1, \cdots, \overline{[\beta_j]}_k$ を収集し α_j, β_j を復元する．

(2) P_j は $\alpha_j\beta_j^{-1}$ を計算し，A.1(2)の秘密分散を用いて k 台のサーバへ秘密分散する[

$$\overline{[\alpha_1\beta_1^{-1}]}_j, \cdots, \overline{[\alpha_k\beta_k^{-1}]}_j \ (j = 1,2,\cdots,k)$$

(3) P_j は $\alpha_j\beta_j^{-1}$ を P_1 に送信する．

(4) P_1 は以下より $\alpha\beta^{-1}$ を復元し $P_1, \cdots, P_k$ に送信する．

$$\alpha\beta^{-1} = \prod_{j=1}^{k} \alpha_j\beta_j^{-1}$$

(5) P_j はそれぞれ以下のようにして $\overline{[\alpha(a+b)\,]}_j$ を計算する．

$$\overline{[\alpha(a+b)\,]}_j = \overline{[\alpha a]}_j + \alpha\beta^{-1}\overline{[\beta b]}_j$$

(6) P_j は $[d]_j \coloneqq \left(\overline{[\alpha(a+b)\,]}_j, \overline{[\alpha_1]}_j, \cdots, \overline{[\alpha_k]}_j\right)$ を $d = ab$ の分散情報とする．

A.5 除算

a, b の分散情報から $e = b/a$ の分散情報を生成する手順を以下に示す．

入力：$[a]_j, [b]_j \ (j = 1,2,\cdots,k)$

出力：$[e]_j \coloneqq [ba^{-1}]_j$

(1) P_j は P_1 に $\overline{[\alpha a]}_j$ を送信する．

(2) P_1 は k 個の$\overline{[\alpha a]}_j$から αa を復元し，$P_1, \cdots, P_k$ に送信する．

(3) P_j はそれぞれ以下のようにして $\overline{[(\alpha a)^{-1}\beta b]}_j$ を計算する．

$$\overline{[(\alpha a)^{-1}\beta b\,]}_j = \overline{[\beta b]}_j \times (\alpha a)^{-1}$$

(4) P_j はそれぞれ $\overline{[\alpha_j]}_1, \cdots, \overline{[\alpha_j]}_k$，$\overline{[\beta_j]}_1, \cdots, \overline{[\beta_j]}_k$ を収集し α_j, β_j を復元する．

(5) P_j は $\beta_j\alpha_j^{-1}$ を計算し，A.1(2)の秘密分散を用いてサーバ$P_1, \cdots, P_k$ へ秘密分散する．

$$\overline{[\beta_j\alpha_j^{-1}]}_1, \cdots, \overline{[\beta_j\alpha_j^{-1}]}_k \ (j = 1,2,\cdots,k)$$

(6) P_j は$[e]_j \coloneqq \left(\overline{[(\alpha a)^{-1}\beta b]}_j, \overline{[\beta_1\alpha_1^{-1}]}_j, \cdots, \overline{[\beta_k\alpha_k^{-1}]}_j\right)$ を $e = ba^{-1}$ の分散情報とする．

A.6 定数乗算

a の分散情報と定数 C から $f = Ca$ の分散情報を生成する手順を以下に示す．

入力：$[a]_j, C \ (j = 1,2,\cdots,k)$

出力：$[f]_j \coloneqq [Ca]_j$

(1) P_j は A.1(2)の秘密分散に対応する秘匿定数乗算を実行し，$C, \overline{[\alpha a]}_j$ から $\overline{[C\alpha a]}_j$を計算する．

$$\overline{[C\alpha a]}_j = \overline{[\alpha a]}_j \times C$$

(2) P_j はそれぞれ $[f]_j \coloneqq \left(\overline{[C\alpha a]}_j, \overline{[\alpha_1]}_j, \cdots, \overline{[\alpha_k]}_j\right)$ を $f = Ca$ の分散情報とする．