ランプ型秘密分散法を用いた秘匿乗算手法の提案

神宮 武志† 岩村 惠市†

†東京理科大学工学部第一部電気工学科
125-8585 東京都葛飾区新宿 6-3-1
shingu@sec.ee.kagu.tus.ac.jp

あらまし 著者らは7月のCSEC研究会において，秘密分散を用いてサーバ台数を限定することなく秘匿除算にも適用可能な秘匿乗算法を提案した．この手法は千田らの手法より計算量を少なくできるが，記憶容量や通信量が増加する．本論文では，ランプ型秘密分散法を用いて記憶容量や通信量も削減する手法を提案する．特に，n=k=2の場合を例に，千田らの手法より計算量，記憶容量，通信量ともに効率的な手法であることを示す.

Secure Multiplication using Ramp Secret Sharing Scheme

Takeshi Shingu† Keiichi Iwamura†

†Tokyo University of Science
6-3-1 Niijyuku, Katsushika-ku, Tokyo 125-8585, Japan
shingu@sec.ee.kagu.tus.ac.jp

Abstract It becomes important to protect information such as private information. Therefore there is a method called Secure multi-party computation that can 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. However, Secure Division using Secret sharing scheme is not proposed as far as authors know it. Furthermore, efficient Secure Multiplocation method have been proposed by Chida. In this paper, we propose Secure Multiplication and Division based on Secret sharing scheme that is not limited the number of servers.

1 はじめに

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方式である. 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.

一方，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．そこで，千田らによる方式 [8]などが

上記の問題を解決する秘匿計算手法が提案されている．一般に，除算に関しては多項式を用いているため，演算自体が困難である．

著者らは 7 月の CSEC 研究会において，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手法を提案した．この手法は千田らの手法より計算量を少なくできるが，記憶容量や通信量は増加する．そこで，本論文ではランプ型秘密分散法を用いて記憶容量や通信量も削減する手法を提案する．

本論文の構成を以下に示す．2 章において適用する秘密分散法と千田らによる秘匿計算手法を紹介する．3 章では 7 月の CSEC において提案した方式について説明し，4 章ではランプ型秘密分散法への応用を行う．5 章では提案方式と従来方式との比較を具体例によって行う．6 章では提案方式の安全性について検討し，最後にまとめを行う．

2 従来方式

2.1 高橋らの多値化方式

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]があるが，この手法は秘密情報をビット列として扱っているため準同型性を持たない．そこで，秘密情報を数値として扱えるように多値化を行い，XOR の代わりに加減算のみを用いた秘密分散手法 [11]が高橋らによって提案された．この手法は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 $(0 \leq t_0 \cdots t_k \leq n-1)$．$k$ 個の分散情報を部分分散情報に分割する．

$$\mathrm{W}_{\mathrm{t}_0} \rightarrow W_{(t_0,0)}, W_{(t_0,1)}, \cdots, W_{(t_0,n-2)}$$

$$\vdots$$

$$W_{t_{k-1}} \rightarrow W_{(t_{k-1},0)}, W_{(t_{k-1},1)}, \cdots, W_{(t_{k-1},n-2)}$$

(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

$W_{(t_0,\cdots,t_{k-1})}$

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

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

ここで, 行列$M^{(k,n)}_{(t_0,\cdots,t_{k-1})}$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．

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

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，それらを全て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．また，$(k,n) = (2,3)$ としているが，乗算実行時には 3 台全てのサーバによる協調演算が行われるため，乗算に関しては欠損耐性を持たない．

[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．

[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．

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 基本方式

以下に，7 月の CSEC で提案した方式を説明する．この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することにより，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．本章では，秘密分散・復元，乗算，除算の順に説明する．なお，秘密情報を $a, b \in \boldsymbol{Z}/p\boldsymbol{Z}$（$p$ は素数）とする．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，除算も同様に有限体上の除算となる．提案方式では，任意の (k,n) しきい値秘密分散法を用いることができるが，分散情報に対する加減算と定数乗算が実行可能な手法であるとする．

3.1 表記規則

- p, q：
  素数
- $\overline{[a]_i}$：
  秘密情報 a をある秘密分散法で分散したときサーバ P_i が持つ分散情報．
- $[a]_i$：
  提案方式によってサーバ P_i が持つ分散情報の集まり．
- $SHR(\alpha a) = \left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{n-1}\right)$：
  ある秘密分散法を用いてαa から分散情報 $\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{n-1}$ を生成する．
- $REC\left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}\right) = \alpha a$：
  SHR に対応する復元法を用いて，$\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}$ からαa を復元する．

3.2 秘密分散

入力：a

出力：$[a]_i \coloneqq \left(\overline{[\alpha a]}_i, \overline{[\alpha_0]}_i, \cdots, \overline{[\alpha_{k-1}]}_i\right)\ (i = 0,1,\cdots,n-1)$

(1) 情報提供者 A は k 個の乱数 $\alpha_0, \cdots, a_{k-1} \in \boldsymbol{Z}/p\boldsymbol{Z}$ を生成し，$\alpha = \prod_{j=0}^{k-1} \alpha_j$ を計算する．

(2) A は αa を計算し，ある (k, n) しきい値秘密分散法を用いて $\alpha a, \alpha_0, \cdots, \alpha_{k-1}$ から各々に対応する分散情報 $\left(\overline{[\alpha a]}_i, \overline{[\alpha_0]}_i, \cdots, \overline{[\alpha_{k-1}]}_i\right)$ を生成する．

$$SHR(\alpha a) = \left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}\right)$$
$$SHR(\alpha_0) = \left(\overline{[\alpha_0]}_0, \cdots, \overline{[\alpha_0]}_{k-1}\right)$$
$$\vdots$$
$$SHR(\alpha_{k-1}) = \left(\overline{[\alpha_{k-1}]}_0, \cdots, \overline{[\alpha_{k-1}]}_{k-1}\right)$$

(3) A は n 台のサーバ P_i に $[a]_i \coloneqq \left(\overline{[\alpha a]}_i, \overline{[\alpha_0]}_i, \cdots, \overline{[\alpha_{k-1}]}_i\right)$ を各々送信する．

3.3 復元

入力：$[a]_j \coloneqq \left(\overline{[\alpha a]}_j, \overline{[\alpha_0]}_j, \cdots, \overline{[\alpha_{k-1}]}_j\right)\ (j = 0,1,\cdots,k-1)$

出力：a

(1) 復元者は k 台のサーバ P_j から $[a]_j$ を収集する．

(2) 復元者は $\overline{[\alpha a]}_j, \overline{[\alpha_0]}_j, \cdots, \overline{[\alpha_{k-1}]}_j$ から $\alpha a, \alpha_0, \cdots, \alpha_{k-1}$ を復元し，以下より a を復元する．

$$REC\left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}\right) = \alpha a$$
$$REC\left(\overline{[\alpha_0]}_0, \cdots, \overline{[\alpha_0]}_{k-1}\right) = \alpha_0$$
$$\vdots$$
$$REC\left(\overline{[\alpha_{k-1}]}_0, \cdots, \overline{[\alpha_{k-1}]}_{k-1}\right) = \alpha_{k-1}$$
$$\alpha = \prod_{j=0}^{k-1} \alpha_j$$
$$\alpha a \times \alpha^{-1} = a$$

3.4 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，n 台のサーバ P_i 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：$[a]_j, [b]_j\ (j = 0,1,\cdots,k-1)$

出力：$[c]_i \coloneqq [ab]_i\ (i = 0,1,\cdots,n-1)$

(1) P_0 は $k-1$ 台のサーバ P_j から $\overline{[\alpha a]}_j, \overline{[\beta b]}_j$ を収集する．

(2) P_0 は $\alpha a, \beta b$ を復元し $\alpha\beta ab$ を計算する．

$$REC\left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}\right) = \alpha a$$
$$REC\left(\overline{[\beta b]}_0, \cdots, \overline{[\beta b]}_{k-1}\right) = \beta b$$

(3) P_0 は $SHR(\alpha\beta ab)$ を実行し，$P_0, \cdots, P_{n-1}$ へ秘密分散する．その後，$\alpha\beta ab$ を破棄する．

(4) P_j は $\overline{[\alpha_j]}_0, \cdots, \overline{[\alpha_j]}_{k-1}$，$\overline{[\beta_j]}_0, \cdots, \overline{[\beta_j]}_{k-1}$ を $k-1$ 台のサーバから収集し α_j, β_j を復元する．P_j は $\alpha_j\beta_j$ を計算し，$SHR(\alpha_j\beta_j)$ を実行し，$P_0, \cdots, P_{n-1}$ へ秘密分散する．

(5) P_i は $[c]_i \coloneqq \left(\overline{[\alpha\beta ab]}_i, \overline{[\alpha_0\beta_0]}_i, \cdots, \overline{[\alpha_{k-1}\beta_{k-1}]}_i\right)$ を $c = ab$ の分散情報とする．

4 ランプ型秘密分散法の適用

基本方式では 1 つの秘密情報 a について $k+1$ 個の分散情報 $[a]_i \coloneqq \left(\overline{[\alpha a]}_i, \overline{[\alpha_0]}_i, \cdots, \overline{[\alpha_{k-1}]}_i\right)$ を保存・通信する必要がある．そこで，分散情報のサイズを最大 $1/k$ にすることのできるランプ型秘密分散 [4]を 3.2 節(2)において提案方式に適用することにより，分散情報のサイズを $1/k$ にすることを考える．以下に，具体例として $n = k = 2$ として 2.1 に示す高橋方式にランプ型秘密分散を適用した場合を示す．

[表記規則]

- $(a)^0, (a)^1$：
  a を半分のビットに分割したときの値
- $a = (a)^0 || (a)^1$：
  $(a)^0, (a)^1$ を結合し a を復元

[分散]

入力：a

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

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

(2) A は $\alpha a, \alpha_0, \alpha_1$ を以下のようにビット分割

する.

$$\alpha a = (\alpha a)^0 || (\alpha a)^1$$
$$\alpha_0 = (\alpha_0)^0 || (\alpha_0)^1$$
$$\alpha_1 = (\alpha_1)^0 || (\alpha_1)^1$$

(3) A は以下を計算する.

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

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

[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 W'_{a0}, W'_{a1} から, $(\alpha a)^0, (\alpha a)^1$ を復元する. $(\alpha a)^0$ と $(\alpha a)^1$ を結合して αa を復元する. 同様にして α_0, α_1 を復元する.

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

(3) 復元者は以下のようにして秘密情報 a を復元する.

$$a = \alpha a \times (\alpha_0 \times \alpha_1)^{-1}$$

[乗算]

入力：$[a]_i, [b]_i\ (i = 0,1)$

出力：$[c]_i \coloneqq [ab]_i$

(1) P_0, P_1 はそれぞれ $(W_{a_1 0}, W_{b_1 0}), (W'_{a1}, W'_{b0}, W_{a_0 1}, W_{b_0 1})$ を P_1, P_0 に送信する.

(2) P_0 は以下より $\alpha a, \beta b$ を復元する.

$$(\alpha a)^0 = W'_{a0} - W'_{a1}$$
$$(\alpha a)^1 = W'_{a1}$$
$$\alpha a = (\alpha a)^0 || (\alpha a)^1$$
$$(\beta b)^0 = W'_{b0} - W'_{b1}$$
$$(\beta b)^1 = W'_{b1}$$
$$\beta b = (\beta b)^0 || (\beta b)^1$$

(3) P_0 は $\alpha\beta ab$ を計算し以下のように秘密分散する（P_1 に$W'_{c_0 1}$を送信する）.

$$W'_{c0} = (\alpha\beta ab)^0 + (\alpha\beta ab)^0$$
$$W'_{c1} = (\alpha\beta ab)^0$$

(4) P_0 は(α_0, β_0)を復元する.

$$(\alpha_0)^0 = W_{a_0 0} - W_{a_0 1}$$
$$(\alpha_0)^1 = W_{a_0 1}$$
$$\alpha_0 = (\alpha_0)^0 || (\alpha_0)^1$$
$$(\beta_0)^0 = W_{b_0 0} - W_{b_0 1}$$
$$(\beta_0)^1 = W_{b_0 1}$$
$$\beta_0 = (\beta_0)^0 || (\beta_0)^1$$

(5) P_1 は(α_1, β_1)を復元する.

$$(\alpha_1)^0 = W_{a_1 0} - W_{a_1 1}$$
$$(\alpha_1)^1 = W_{a_1 1}$$
$$\alpha_1 = (\alpha_1)^0 || (\alpha_1)^1$$
$$(\beta_1)^0 = W_{b_1 0} - W_{b_1 1}$$
$$(\beta_1)^1 = W_{b_1 1}$$
$$\beta_1 = (\beta_1)^0 || (\beta_1)^1$$

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

$$W_{c_0 0} = (\alpha_0\beta_0)^0 + (\alpha_0\beta_0)^1$$
$$W_{c_0 1} = (\alpha_0\beta_0)^1$$

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

$$W_{c_1 0} = (\alpha_1\beta_1)^0 + (\alpha_1\beta_1)^1$$
$$W_{c_1 1} = (\alpha_1\beta_1)^1$$

(8) 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.

[除算]

入力：$[a]_i, [b]_i\ (i = 0,1)$

出力：$[e]_i \coloneqq [b/a]_i$

(1) P_0, P_1 はそれぞれ $(W_{a_1 0}, W_{b_1 0}), (W'_{a1}, W'_{b0}, W_{a_0 1}, W_{b_0 1})$ を P_1, P_0 に送信する.

(2) P_0 は$\alpha a, \beta b$ を復元する.

$$(\alpha a)^0 = W'_{a0} - W'_{a1}$$
$$(\alpha a)^1 = W'_{a1}$$
$$\alpha a = (\alpha a)^0 || (\alpha a)^1$$
$$(\beta b)^0 = W'_{b0} - W'_{b1}$$
$$(\beta b)^1 = W'_{b1}$$
$$\beta b = (\beta b)^0 || (\beta b)^1$$

(3) P_0 は $\beta b/\alpha a$ を計算し以下のように秘密分散する (P_1 に$W'_{e_0 1}$を送信する). その後, $\beta b/\alpha a$ を破棄する.

$$W'_{e0} = (\beta b/\alpha a)^0 + (\beta b/\alpha a)^1$$
$$W'_{e1} = (\beta b/\alpha a)^1$$

(4) P_0 は(α_0, β_0)を復元する.

$$(\alpha_0)^0 = W_{a_0 0} - W_{a_0 1}$$
$$(\alpha_0)^1 = W_{a_0 1}$$
$$\alpha_0 = (\alpha_0)^0 || (\alpha_0)^1$$
$$(\beta_0)^0 = W_{b_0 0} - W_{b_0 1}$$
$$(\beta_0)^1 = W_{b_0 1}$$
$$\beta_0 = (\beta_0)^0 || (\beta_0)^1$$

(5) P_1 は(α_1, β_1)を復元する.

$$(\alpha_1)^0 = W_{a_1 0} - W_{a_1 1}$$
$$(\alpha_1)^1 = W_{a_1 1}$$
$$\alpha_1 = (\alpha_1)^0 || (\alpha_1)^1$$
$$(\beta_1)^0 = W_{b_1 0} - W_{b_1 1}$$
$$(\beta_1)^1 = W_{b_1 1}$$
$$\beta_1 = (\beta_1)^0 || (\beta_1)^1$$

(6) P_0は β_0/α_0 を計算し以下のように秘密分散する（P_1 に$W_{e_0 1}$を送信する).

$$W_{e_0 0} = (\beta_0/\alpha_0)^0 + (\beta_0/\alpha_0)^1$$
$$W_{e_0 1} = (\beta_0/\alpha_0)^1$$

(7) P_1は β_1/α_1 を計算し以下のように秘密分散する（P_0 に$W_{e_1 0}$を送信する).

$$W_{e_1 0} = (\beta_0/\alpha_0)^0 + (\beta_0/\alpha_0)^1$$
$$W_{e_1 1} = (\beta_0/\alpha_0)^1$$

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

5 評価

基本方式と提案方式の特徴は (k, n)を限定せず，かつ秘密情報を秘匿化したままで一旦復元するため，乗算だけでなく除算にも容易に対応可能という点である．千田らによる方式 [8] は乗算に対応しているが，$(k, n) = (2,3)$に限定されており，除算は実現しない．よって，千田らの方式と基本方式, 提案方式の機能を比較すると表 1 のようになる（効率は後述）．

以下では，秘密分散を用いた乗算において現時点で最も効率的と思われる千田らによる方式を従来方式として，基本方式，提案方式について計算量，記憶容量，通信量に関する効率の比較を行う．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乗算し，任意の 2 台のサーバより演算結果の復元を行う場合に特化されている. 提案方式は (k, n) は限定されていないが，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場合について考える．なぜならば，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, 欠損耐性と情報漏洩耐性という観点からは同等と考えられるためである．ただし，従来方式は秘匿除算を実現しないので, その比較は省略するが, 基本方式と提案方式は有限体上の除算と乗算がほぼ同じ計算量とすれば(一般に逆元の対応表をもつとすれば, 除算は乗算と同様の計算量で実現できる)，秘匿除算を秘匿乗算とほぼ同じ計算量，記憶容量（逆元の対応表の容量を含める場合はその分増加する），通信量で実現可能である．また，秘密情報を l ビットとして比較を行う．

表 1　各方式の機能比較

	千田方式	基本方式	提案方式
(k, n)の自由度	×	○	○
除算	×	○	○
効率	○	×	○

5.1 計算量

従来方式, 基本方式, 提案方式において最も計算量を必要とする演算は乗算，除算の計算で

ある. そこで, 以下のようにして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. 表 2 に計算量の比較を示す.

$M(l)$：l ビット同士の乗算の計算量（$M\left(\frac{l}{2}\right) = \frac{1}{4}M(l)$）

$I(l)$：l ビット同士の除算の計算量

$R(l)$：Ramp 秘密分散における復元の計算量

表 2　サーバの計算量の比較

	従来方式	基本方式	提案方式
分散	なし	$4M(l)$	$4M(l)$
復元	なし	$M(l)+I(l)$	$M(l)+I(l)$
乗算	$6M(l)$	$3M(l)$	$3M(l)$

表 2 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，基本方式と提案方式では 3 回の乗算で秘匿乗算が可能であり, 繰り返しの秘匿乗算により分散・復元の計算量が無視出来るような場合，基本方式と提案方式は $1/2$ の計算量で秘匿乗算が可能である. また，サーバ 1 台当たりの計算量も従来方式が最大 3 回の乗算であるのに対し，基本方式と提案方式は最大 2 回でよい. また，基本方式と提案方式は (k,n) を限定しないため，k に対して全体として $k+1$ 回の乗算で秘匿乗算が可能である.

5.2　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．表 3 に全体でのサーバの記憶容量の比較を示す.

表 3　サーバの記憶容量の比較

	従来方式	基本方式	提案方式
記憶容量	$6l$	$6l$	$3l$

表 3 より，従来方式と基本方式の記憶容量は同じだが，Ramp 方式を適用した提案方式では必要な記憶容量は $1/2$ となる．サーバ 1 台当たりでは，従来方式が $2l$ であるのに対して，基本方式は $3l$，提案方式は $1.5l$ となる．よって，全体としても任意の n に対して基本方式は $3nl$，提案方式は $1.5nl$ となる.

5.3　通信量

l ビットのデータの通信に必要な通信量を $S(l)$ として比較を行う．表 4 に全体でのサーバ間の通信量の比較を示す．なお，$S(l/2) = (1/2)S(l)$ とする.

表 4　サーバ間の通信量の比較

	従来方式	基本方式	提案方式
分散	$6S(l)$	$6S(l)$	$3S(l)$
復元	$4S(l)$	$6S(l)$	$3S(l)$
乗算	$6S(l)$	$9S(l)$	$\frac{9}{2}S(l)$

表 4 より，基本方式に比べ従来方式の方が全てのアルゴリズムにおいて通信量が少なくすむことが分かる. しかし, 提案方式では分散情報のサイズを $1/2$ にしているため，従来方式より通信量を少なくすることができる. 基本方式において (k,n) を変化させた場合，分散では $n(k+1)S(l)$，復元では $(k+1)kS(l)$，乗算では $(k+1)(k-1+n)S(l)$ となる．提案方式では各々 $1/k$ となる.

6　提案方式の安全性

6.1　ランプ型秘密分散適用に関する安全性

提案方式では，ランプ型秘密分散を用いて $\alpha a, \alpha_0, \cdots, \alpha_{k-1}$ を各サーバに秘密分散するが, ランプ型秘密分散はしきい値 k 以下の分散情報から秘密情報が漏洩する可能性がある. よって, ランプ型秘密分散適用に関する安全性を検討する. まず，αa に関しては αa が漏洩しても, α が分からなければ秘密情報である a は漏洩しない.

よって，α に関する安全性を考える．ここで，[4]において，秘密情報のビット数を 100，Ramp 秘密分散の分割数を 2 としたとき，(k,n) 秘密分散 C_1 と Ramp 秘密分散 C_2 における，集まった分散情報の個数に対する秘密情報のあいまいさ（値として可能性のある個数）の比較が行われおり，以下のようになる．

- 分散情報の個数が k のとき → $C_1: 1, C_2: 1$
- 分散情報の個数が $k-1$ のとき → $C_1: 2^{100}, C_2: 2^{50}$
- 分散情報の個数が $k-2$ のとき → $C_1: 2^{100}, C_2: 2^{100}$

次に，同じ条件で，提案方式にて $\alpha_0, \cdots, \alpha_{k-1}$ の分散情報がそれぞれ $k-1$ 個漏洩した場合の安全性について考える．すなわち，$k=2$ の場合 α_0, α_1 はそれぞれ 2^{50} 個の値が考えられる．よって，$\alpha = \prod_{i=0}^{k-1} \alpha_i$ で与えられる α は $(2^{50})^k$ 個の値が考えられ，$k=2$ の場合，α は 2^{100}個の値となる．すなわち，提案方式では 1 つの乱数 α を特定するためには k 個の乱数を特定する必要であるので，乱数の各ビット数を 100とすれば，それを定めるために必要な情報量は $100k$ となる．それに対して，分散情報のサイズを $1/k$ にするランプ型秘密分散を適用すると必要な情報量は$100k/k = 100$となり，元の α の情報量と等しくなる．

以上より，基本方式にランプ型秘密分散を適用しても，安全性は，a に対して(k,n) 秘密分散を適用した場合と同等であることが言える．

7　まとめ

本論文では，従来提案されていなかった秘匿除算が実行可能な秘密分散法を用いた秘匿計算手法に対する効率化の提案を行った．特に $(k,n) = (2,2)$ のシステムにおいて，提案方式では秘匿乗算を従来方式の $1/2$ の計算量で実行可能なことを示した．また，ランプ型秘密分散法を適用することにより，通信量，記憶容量も従来方式より小さくすることができた．今回は頁数の都合から秘匿加減算についての説明は省略したが，今後の課題としては，秘匿加減算の効率化と除算を有限体上の元でなく，整数または実数に対して行えるようにすることがある．

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)