Vol.2013-CSEC-63 No.3
2013/12/9

秘密分散法を用いたサーバ台数変化がない乗算手法

渡辺 泰平[1,a)]　岩村 惠市[1]

概要：秘密分散法の大きなメリットとして, サーバ n 台のうち,k 台が稼働していれば, 正常な動作が可能であるという点があげられる. 秘密分散法を用い, データを秘匿したまま演算する技術として, 秘密計算があり，その秘密計算における, 重要な演算の一つとして乗算がある. 乗算の手法はいくつか提案されているが (k,n) 秘密分散を基本としていても, 乗算演算実行時及び乗算結果復元時に必要なサーバの台数か k 台以上となるものが多い．この場合，k 台のサーバが稼働していても，乗算時のみ計算できないという事態が生じるため，乗算時も元の前提条件を崩さないように，最低 k 台のサーバが稼働していれば，秘密計算可能とすることが望まれる. 我々は CSS2013 において上記の問題を解決する手法を提案した．しかし, この手法は一時的に閾値未満のサーバがもつ分散情報から秘密情報を復元できてしまうという欠点があった. 本論文では, 一時的にも情報漏洩が生じず，かつ，サーバの台数変化がない手法を提案する．

キーワード：秘密分散法，秘密計算，ビッグデータ，乗算，プライバシー保護，Shamir 閾値秘密分散法

1. はじめに

近年，ビッグデータの活用が注目されている．ビッグデータの活用とは，主に事業に役立つ知見を導出するためのデータ活用である [1]．このビッグデータを活用するためには，ビッグデータを分析し [2]，予測するなどの使用方法が考えられる．その時に，自身の欲している予測結果や推測結果を導き出す場合，おおもとのデータの量が少ないと目的の予測・推測結果は導けない．そのため，データの共有をすることにより，おおもとのデータを増やし，ビッグデータを作り出すことが必要である [3]．そのことで自身の欲している予測結果，推測結果を得ることができる．しかし，ここで問題になるのがデータの共有である．データの共有とは，他者に自己のデータを渡すことである．そのデータには個人情報等も含まれている場合もある．そういった場合，秘密情報が他者に公開されては困るため，保護が必要である．これらより，データの保護と演算をまとめてできる技術がビッグデータの活用には必要不可欠になる [18]．その問題の有力な解決手段として各種計算の入力となるデータを秘匿しつつ計算結果を求める秘密計算がある．例えば，秘密情報を所持している人を，情報提供者と呼び，計算結果を要求している人を情報収集者と呼ぶとする．情報提供者 A と B は各々が持つ秘密情報 a と b を暗号化して暗号文 $E(a)$ と $E(b)$ を作り，サーバなどの複数の装置に暗号文を預ける．そのサーバは，暗号文を使い，あるプロトコルに従い計算することで，秘密情報 a や b を知ることなく，$E(a+b)$ や $E(a\times b)$ といった各種計算を実行できる．また，情報収集者は，いくつかのサーバから $E(a+b)$ などの計算結果を集め，復号することにより計算結果 $a+b$ を得られるという仕組みである．四則演算において秘密計算の加算や減算のプロトコルは，比較的簡単に実行できるが，乗算や除算のプロトコルは処理が複雑で難しいことが知られている．我々は，その秘密計算の中でも，秘密分散法を用いた秘密計算の乗算を CSS2013 で提案した．

まず，従来の乗算法における問題点を説明する．従来の乗算に関しての論文は，Shamir ベースの乗算として M.Ben-Or らの [6]，R.GENNARO らの [7] 等があり，また協調演算を必要とする乗算は，五十嵐らの [8]，I.Dangård らの [9]，布川らの [10] がある．Shamir ベースの乗算に関しては，乗算時に 2k-1 台のサーバが必要になり，サーバの構成が $n \geq 2k-1$ と制限される．そのため，例えば Shamir(3,4) しきい値秘密分散法でシステムを構成する場合，このシステムは加算やデータの復元は実現できるが乗算は実現できず k=3 とする場合，n=5 以上にする必要がある．これは，システム構成の柔軟性が失われることを意味する．また，協調演算を必要とする乗算は，乗算時に n 台すべてのサーバが通信し計算する必要がある．そのため，n 台のサーバが正常に稼働していなければ乗算は秘密分散

[1] 東京理科大学工学部
Tokyo University of Science
6-3-1 niijyuku, Katsushika-ward, Tokyo 125-0051, JAPAN
[a)] watanabe@sec.ee.kagu.tus.ac.jp

の利点を生かしていないと言える．まとめると，これらの従来の乗算手法は，乗算時に k 台を超えるサーバを必要とする，サーバの台数変化をすることが問題である．

上記秘密計算の乗算時にサーバが台数変化をするポイントとして，1. 乗算演算実行時，2. 乗算結果復元時の 2 点があり，各々を問題点 1，問題点 2 とする．例えば，Shamir 閾値秘密分散法 [5] に基づく乗算 [6][7] は，各サーバが分散情報どうしを掛けるだけで乗算結果の分散情報を生成できるが，乗算結果の分散情報の次数は 2k-2 に上昇するため，乗算結果の分散情報を生成するサーバの台数は 2k-1 台 ($> k$) 必要になる (問題点 1)．また，乗算結果の復元時には，2k-1 台 ($> k$) のサーバから乗算結果の分散情報を集めなければならない (問題点 2)．このようなサーバの台数変化があることが秘密分散法のメリットを生かし切れていない所である．それに対して，CSS2013 で発表した方式の長所としては，上記 2 点の問題点を改善し，乗算しても乗算実行時と乗算結果復元時に台数変化しないという点である．

しかし，一時的に閾値未満のサーバが持つ分散情報から秘密情報を復元できてしまうという欠点があった．これは，乗算を行う手順が秘密情報に関する分散情報の秘匿化を解いた後に乗算結果の分散情報を生成するためである．今回は，秘密情報に関する秘匿化を解かずに乗算結果の分散情報を生成する方式を提案することで，この欠点を改善する．

以降，2 章では Shamir の秘密計算の乗算についての従来手法を紹介し，その問題点であるサーバの台数変化について説明する．3 章ではその問題を解決する．CSS2013 で発表した方式を紹介し，さらにその問題点をあげる．4 章でそれらを改善した提案方式を紹介する．5 章では，提案手法に関する評価や検討を行う．

2. 従来方式

本章では，秘密分散法を用いた乗算の従来方式として，Shamir の (k,n) 閾値秘密分散法 [5] を用いる手法について説明を行う．また，M.Ben-Or らの [6]，R.GENNARO らの [7] もこの乗算を用いる．これらは，分散情報の次数が上がったものを，再び計算することで次数を下げ，次数の上昇を防ぎながら乗算する方式である．

2.1 Shamir(k,n) 閾値秘密分散法 [5]

Shamir(k,n) 閾値秘密分散法は，n 個の分散情報のうち，任意の k 個を集めることで秘密情報 s が復元できるが，k-1 個以下の分散情報では秘密情報についての情報が全く得られない方式である．さらに乗算をすると閾値が 2k-1 に変化することを示す．簡単に手順を以下に示す．

[分散]

(1) $s < p$ かつ $n < p$ である任意の素数 p を選ぶ．

(2) $GF(p)$ の元から，異なる n 個の $x_i (i = 1, \cdots, n)$ を選びだし，サーバ ID とする．

(3) $GF(p)$ の元から，$k-1$ 個の乱数 $a_l (l = 1, 2, \cdots, k-1)$ を選んで，以下の式を生成する．

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

(4) 上記式 (1) の x_i に各サーバ ID を代入して，分散情報 W_i を計算し，各サーバにこれらのユーザ IDx_i と生成した分散情報 W_i を配付する．

[乗算]

情報提供者 A は秘密情報 a を，情報提供者 B は秘密情報 b を上の分散式により，各々秘密分散する．サーバは，分散された情報を以下のように計算する．

(1) サーバ x_i に対する秘密情報 a, b の分散情報をそれぞれ W_{ai}, W_{bi} とする．サーバ xi は，W_{ai}, W_{bi} 用いて式 (4) の乗算をする．

$$W_{ai} = a + a_1 x_i + a_2 x_i^2 + \cdots + a_{k-1} x_i^{k-1} \quad (2)$$

$$W_{bi} = b + b_1 x_i + b_2 x_i^2 + \cdots + b_{k-1} x_i^{k-1} \quad (3)$$

$$W_{ai} W_{bi} = ab + (ab_1 + ba_1) x_i + \cdots + a_{k-1} b_{k-1} x_i^{2k-2} \quad (4)$$

(2)(3) 式は k-1 次多項式であったが，(4) 式は 2k-2 次多項式となり，式 (4) を解くために必要な分散情報の数が k から 2k-1 に上昇する．

[復元]

復元は，全てのサーバの台数 n とすると n ≥2k-1 でないとできない．

(1) 復元に用いる分散情報を $W_i (i = 1, 2, \cdots, 2k-1)$ とする．また，その分散情報に対応するユーザ ID を x_i とする．

(2) 分散式に x_i と W_i を代入し，2k-1 個の連立方程式を解いて，乗算結果 ab を得る．乗算結果 ab の復元の際には，Lagrange の補間公式を用いると便利である．

2.2 Shamir(k,n) 閾値秘密分散法に基づく乗算の問題点

2.1 章に示す乗算においては，2k-1 台のサーバから分散情報を集めなければならない．これは，多項式どうしの乗算により，次数が上がってしまうためである．よって，乗算結果復元時や乗算演算実行時にサーバの台数変化が生じる．これを詳細に見てみると以下のようになる．

本来 Shamir ベースの乗算は各々のサーバが分散情報どうしを乗算することにより，乗算結果の分散情報を計算できる．しかし，上記の乗算では，乗算結果復元時に 2k-1 台のサーバから乗算結果の分散情報を集めなければならないため，2k-1 台のサーバが乗算実行時に乗算を計算しなければならない．

よって，従来方式は 1. 乗算演算実行時と 2. 乗算結果復

元時にサーバの台数変化が起こるという 2 つの問題点が発生する．よって従来方式では k 台のサーバが正常に稼働していても，乗算演算や乗算結果復元ができないため秘密分散法を用いた乗算の方式としては問題であることが明確である．

3. CSS2013 の方式とその解決すべき問題点

この章では, CSS2013 で発表した方式 [23](以降, CSS2013 方式) を説明し，その問題点をあげる．

3.1 CSS2013 方式の概要

CSS2013 で提案した方式は，Shamir(k,n) 閾値秘密分散法の形を崩さずに構成している．この方式と Shamir(k,n) 閾値秘密分散法の違う点は，以下のようになっている．通常 Shamir(k,n) 閾値秘密分散法では，サーバ 1 台に対してサーバ ID は 1 つしか持たせない．それにより，分散情報はサーバ 1 台に対して 1 つとなるが，本方式では，サーバ 1 台に対してサーバ ID を 2 つ持たせることにより，分散情報はサーバ 1 台に対して 2 つとなる．この違いがあるため，閾値 k の分散情報を乗算し閾値が 2k-1 に上昇しても，k 台のサーバを集めたとき，2k 個の分散情報を集めることができ，乗算結果が復元可能である．

しかし，サーバ ID を 2 つ持たせることには問題がある．それは，1 台のサーバが 2 つずつ分散情報を持っているため，k/2 台のサーバを攻撃しただけで，乗算結果ではなく秘密情報が復元できてしまうという点である．その問題の解決策として，2 つあるサーバ ID の片方の秘密情報の分散情報を秘匿化することにより，Shamir(k,n) 閾値秘密分散法と同じようにサーバ 1 台に分散情報を 1 つしか持っていないように見える．そのため秘密情報については，攻撃者は 1 台のサーバからは 1 つの分散情報しか知ることができず，k 台のサーバを集めなければ元の秘密情報を復元できなくすることができる．また乗算結果を復元する時には情報収集者は 2k-1 個以上の分散情報が必要なため秘匿化をしてある分散情報も集め，秘匿化を解くことで乗算結果を得ることができる．

3.2 CSS2013 方式

まず，CSS2013 で我々が発表した方式 [23] を示す．ここでは情報提供者 A が秘密情報 a を持ち，情報提供者 B が秘密情報 b を持つ．そして，乗算結果 ab を情報収集者 C が知りたいとする．

[分散]

(1) 各サーバに 2 つずつサーバ ID$x_i(i = 1, 2, \cdots, 2n)$ を振る．

(2) A,B さんは $2n$ 台分の分散情報 W_{iA}, W_{iB} を shamir(k,n) 閾値秘密分散法により生成.

(3) n 個の分散情報 $W_{iA}, W_{iB}(i = 1, 2, \cdots, n)$ を各サーバ x_i に 1 つずつ渡す.

(4) A,B さんはそれぞれ真性乱数 $r_{lA}, r_{lB}(l = 1, 2, \cdots, n)$ を生成.

(5) A,B さんはそれぞれ $W_{jA}, W_{jB}(j = n+1, n+2, \cdots, 2n)$ の分散情報に 1 つずつ乱数 r_{lA}, r_{lB} を足し，その分散情報 W_{jlA}, W_{jlB} を各サーバに 1 つずつ渡す.

(6) A,B さんは乱数鍵 $r_{lA}, r_{lB}(l = 1, 2, \cdots, n)$ を shamir(k,n) 秘密分散法により，分散情報 $R_{lmA}, R_{lmB}(m = 1, 2, \cdots, n)$ を各サーバへ渡す.

[乗算]

(1) サーバ $x_i(i = 1, 2, \cdots, n)$ は，自身が持っている分散情報 W_{iA}, W_{iB} を乗算し乗算結果の分散情報 W_{Mi} を作る．

(2) サーバ $x_i(i = 1, 2, \cdots, n)$ は，k-1 台のサーバに乱数鍵 $r_{lA}, r_{lB}(l = 1, 2, \cdots, n)$ に関する分散情報 R_{lmA}, R_{lmB} を要求し，乱数鍵 r_{lA}, r_{lB} を復元する．

(3) それぞれ分散情報 W_{jlA}, W_{jlB} を乱数鍵 r_{lA}, r_{lB} で引き，分散情報 $W_{jA}, W_{jB}(j = n+1, n+2, \cdots, 2n)$ を生成する．

(4) サーバ $x_i(i = 1, 2, \cdots, n)$ は，自身が持っている秘密情報の分散情報 $W_{jA}, W_{jB}(j = n+1, n+2, \cdots, 2n)$ を乗算し乗算結果の分散情報 $W_{Mj}(j = n+1, n+2, \cdots, 2n)$ を作る．

(5) サーバ $x_i(i = 1, 2, \cdots, n)$ は，復元した乱数鍵 r_{lA}, r_{lB} と分散情報 $W_{jA}, W_{jB}(j = n+1, n+2, \cdots, 2n)$ を消去する．

[復元]

(1) 任意の k 台のサーバから C さんは，乗算結果の分散情報 W_{Mi}, W_{Mj} を集める．

(2) サーバは，C さんに乗算結果の分散情報を渡した後，乗算結果を消去する．

(3) C さんは 2k 個の乗算結果の分散情報 W_{Mi}, W_{Mj} と，それに対応する各サーバ IDx_i を分散式に代入し，連立方程式を解くことで乗算結果 ab を得る．

3.3 CSS2013 方式の問題点

CSS2013 で我々が発表した方式は 3.1 章で示した問題点を解決した．しかし，[乗算] の (3) で乗算演算実行時に秘匿化を解くため，その瞬間に分散情報を集めることが可能であれば，秘密情報が漏えいするという問題点がある．

ただし，CSS2013 方式において秘匿された分散情報が復元されるのは，[乗算](3) において乗算を行うその瞬間のみであり，非常に短い時間であるとしている．よって，現実的に攻撃者がその瞬間のみを狙って分散情報を集めることは困難といえる．しかし，その瞬間を狙う攻撃ができれば k 個以下のサーバから秘密情報が漏えいする．よって，この点が CSS2013 方式の問題点である．

4. 提案方式

4.1 提案方式の概要

本提案方式の概要を説明する．本方式は，3.2 章の CSS2013 方式の問題点を改善した方式である．CSS2013 方式の秘匿化は，秘密情報の分散情報に乱数を加算していたため，秘匿化したまま分散情報どうしを乗算できなかった．そのため，秘匿化を解くのはサーバになる．しかし，今回の方式では，秘匿化を分散情報に乱数の乗算をして実現している．すなわち，秘匿化したままの分散情報どうしを乗算し，その結果を情報収集者に送り，後でその乗算結果から乱数を除算する．

4.2 提案方式

情報提供者 A，B は秘密情報 $a, b \in \mathbf{Z}/p\mathbf{Z}(p$ は素数）を持つ．また，計算は基本的に $\mathbf{Z}/q\mathbf{Z}(q > 2p$ の素数) 上で行われるものとする．

[分散]

(1) 各サーバ n 台にサーバ ID$x_i(i = 1, 2, \cdots, n)$ を割り当て，さらにサーバ ID$x_i(i = n+1, n+2, \cdots, 2n)$ を割り当てる．

(2) A,B さんは自身の持っている秘密情報を，サーバ ID$x_i(i = 1, 2, \cdots, 2n)$ により各々 $2n$ 個の分散情報 $W_{iA}, W_{iB}(i = 1, 2, \cdots, 2n)$ を shamir(k,n) 閾値秘密分散法により生成．

(3) 分散情報 $W_{iA}, W_{iB}(i = 1, 2, \cdots, n)$ を各サーバ x_i に 1 つずつ渡す．

(4) A,B さんは真性乱数で秘匿化鍵 $r_A, r_B \in \mathbf{Z}/q\mathbf{Z}$ を生成．

(5) A,B さんはそれぞれ $W_{iA}, W_{iB}(i = n+1, n+2, \cdots, 2n)$ の分散情報を秘匿化するため，各分散情報に秘匿化鍵 r_A, r_B を乗算し，その秘匿化した分散情報 $W'_{jA} = W_{iA}\ r_A, W'_{jB} = W_{iB}\ r_B(j = 1, 2, \cdots, n)$ を各サーバに 1 つずつ渡す．

(6) A,B さんは秘匿化鍵 r_A, r_B をサーバ ID$x_i(i = 1, 2, \cdots, n)$ を使い Shamir(k,n) 秘密分散法により，分散情報 R_{iA}, R_{iB} を各サーバへ渡す．

[乗算]

(1) 各サーバは，サーバ ID$x_i(i = 1, 2, \cdots, n)$ に対応する分散情報 W_{iA}, W_{iB} を乗算し，乗算結果の分散情報 $W_{Mi} = W_{iA}W_{iB}(mod\ p^2)(i = 1, 2, \cdots, n)$ を作る．

(2) 各サーバは，サーバ ID$x_i(i = n+1, n+2, \cdots, 2n)$ に対応する分散情報 $W'_{jA}, W'_{jB}(j = 1, 2, \cdots, n)$ を乗算し，秘匿化した乗算結果の分散情報 $W'_{Mj} = W'_{jA}W'_{jB}$ を作る．

4.2.1 乗算結果 [復元]

(1) C さんは，任意の k 台のサーバから乗算結果の分散情報 W_{Mi} を集める．

(2) C さんは，任意の k 台のサーバから秘匿化された乗算結果の分散情報 $W'_{Mj}(j = 1, 2, \cdots, n)$ を集める．

(3) C さんは，k 台のサーバから秘匿化鍵の分散情報 R_{iA}, R_{iB} を集め秘匿化鍵 r_A, r_B を復元し，秘匿化してある乗算結果の分散情報 W'_{Mj} を各々割ることで秘匿化を解く．

(4) C さんは集めた 2k 個の乗算結果の分散情報 $W_{Mi}, W'_{Mj}/r_A r_B$ から，連立方程式を解くことで乗算結果 ab を得る．

4.3 考察

ここでは，提案方式の乗算時における法の変換，因数分解についての安全性や情報収集者 C さんの攻撃に関する記述，対策を述べる．

4.3.1 因数分解の安全性について

一般に情報の秘匿化には乱数を加算することによって実現される．それに対して，本提案方式の [分散] の (5) では，秘密情報の分散情報に秘匿化鍵 r_A, r_B を乗算することにより秘匿化している．しかし，法とする数が素数であれば必ず乗算に関する逆元が存在するため，乗算によっても情報の秘匿化が可能であると考える．例として，表 1 に法を 7 として，秘匿化結果が 5 のときの秘密情報と乱数の関係を加算と乗算について示す．すなわち，秘匿化結果を a=5(mod 7), 秘密情報を s(mod 7), 乱数を r(mod 7) とした場合，$s \times r = a$ について全ての組み合わせが実現されていることを示す．(乗算では 0 は扱わないとする．)

表 1 秘匿化の加算と乗算の比較表

演算	秘密情報 s	乱数 r	秘匿化結果 a
加算	6	6	5
	5	0	
	4	1	
	3	2	
	2	3	
	1	4	
	0	5	
乗算	6	2	5
	5	1	
	4	3	
	3	4	
	2	6	
	1	5	

以上より，乗算についても全ての組み合わせが実現されているため，秘匿化結果を攻撃者が得ても情報を特定できない．乗算について因数分解が問題となるのは，素数を法としない場合と考えられる．

4.3.2 情報収集者による攻撃

4.2 章で記述した提案方式が使用される状況としては，情報収集者が正当であるという事をパスワード [13][14] など

を使い証明した後，秘密計算の結果を知ることが出来るシステム [15] 等の通常のクラウド [16] を想定した物である．そういったシステムには 4.2 章で示した方式で十分であると考えられる．しかし，情報収集者が，その攻撃者としてサーバから情報を得ることが出来る場合，4.2 章の方式は乗算結果の復元後にサーバ事業者 k/2 台を攻撃すると，秘密情報を知る事ができる．それを以下に示す．

ここでは，情報収集者が乗算結果を受け取り復元した後の情報収集者とサーバ k-1 台が保持している情報を見てみる．簡単のために，乗算結果の復元時はサーバ $x_i(1,2,\cdots,k)$ を使ったとする．さらに，情報収集者の攻撃時は，復元時に使ったサーバ k 台のうちサーバ $x_i(1,2,\cdots,k/2)$ を攻撃するものとする．また，秘密情報 a,b 等の分散情報の k,n は任意に決められるが，簡単のために k=2 で分散されているとする．

この時，サーバが保持している情報と情報収集者が保持している情報を上げると以下のようになる．

サーバ 1 台が保持している情報

(1) サーバ ID:x_i, x_{i+n}
(2) 秘密情報の分散情報:W_{iA}, W_{iB}
(3) 秘匿化した秘密情報の分散情報:W'_{jA}, W'_{jB}
(4) 秘匿化鍵の分散情報:R_{iA}, R_{iB}
(5) 乗算結果の分散情報:W_{Mi}
(6) 秘匿化した乗算結果の分散情報:W'_{Mj}

情報収集者が保持している情報

I) 乗算結果の分散情報:$W_{Mi}(i=1,2,\cdots,k)$
II) 秘匿化した乗算結果の分散情報:$W'_{Mj}(j=1,2,\cdots,k)$
III) 秘匿化鍵の分散情報:$R_{iA}, R_{iB}(i=1,2,\cdots,k)$
IV) 秘匿化鍵 r_A, r_B

ここで注目したいのが，サーバが保持している (2)(3) と情報収集者が保持している IV) である．(3) 秘匿化した秘密情報の分散情報 W'_{jA}, W'_{jB} は，秘匿化鍵 r_A, r_B を用いて秘匿化されていた．しかし，いま情報収集者は IV) 秘匿化鍵 r_A, r_B を保持している．サーバから (2)(3) の秘密情報の分散情報 W_{iA}, W_{iB} と，秘匿化した秘密情報の分散情報 $W'_{jA} = W_{i+nA}\, r_A, W'_{jB} = W_{i+nB}\, r_B(j=n+1,\cdots,n+k)$ を使い，秘密情報の分散情報 $W_{iA}, W_{iB}, W_{i+nA}, W_{i+nB}$ がわかる．よって，秘密情報 a,b が求まることがわかる．

4.3.3 情報収集者による攻撃への対策

情報収集者による攻撃が成功した理由は，情報収集者が知った秘匿化鍵と，サーバに保存されている秘匿化鍵が同じであったためと考えられる．秘匿化鍵は，秘匿化された分散情報を割るために存在するため，秘匿化された分散情報と対応していれば常に同じ値である必要はない．そのため，各サーバは一時的な乱数を発生させ，それを秘匿化鍵と秘匿化分散情報に掛けて情報収集者に送信すればよい．これを実現するためには，4.2.1 の乗算結果の復元を以下のものに変えれば良い．

[復元] [乗算結果]

ここでは，一時的に使う共通鍵 $P_{k1} = \prod_{h=1}^{k} g_h$，$P_{k2} = \prod_{h=1}^{k} v_h$ とし，$P_k = P_{k1}P_{k2}$ とする．

(1) 復元に使うサーバ k 台は，それぞれ 2 つずつ一時的に使う乱数 $v_h, g_h(h=1,\cdots,k)$ を生成する．
(2) k 台のサーバは，それぞれ秘匿化した乗算結果の分散情報 W'_{Mj} に乱数 $v_h g_h$ を乗算し，ランダムにサーバ k 台で分散情報を渡し合い $W'_{MjP_k} = W'_{Mj}P_k$ を作る．
(3) k 台のサーバは，それぞれ自身が持っている秘匿化鍵の分散情報 R_{iA} に乱数 g_h を秘匿化鍵の分散情報 R_{iB} に乱数 v_h を乗算し，ランダムにサーバ k 台で分散情報を渡し合い P_{k1} または P_{k2} 倍した秘匿化鍵の分散情報 $R_{iAP_{k1}} = R_{iA}P_{k1}$ と，$R_{iBP_{k2}} = R_{iB}P_{k2}$ を作る．
(4) サーバは，一時的な乱数 $g_h, v_h(h=1,\cdots,k)$ を消去する．
(5) C さんは，k 台のサーバから乗算結果の分散情報 W_{Mi} を集める．
(6) C さんは，k 台のサーバから秘匿化された乗算結果の分散情報 W'_{MjP_k} を集める．
(7) C さんは，k 台のサーバから秘匿化鍵の分散情報 $R_{iAP_{k2}}, R_{iBP_{k1}}$ を集め秘匿化鍵 $P_{k1}r_A, P_{k2}r_B$ を復元し，秘匿化してある乗算結果の分散情報 W'_{MjP_k} を各々割ることで秘匿化を解く．
(8) C さんは集めた 2k 個の乗算結果の分散情報 $W_{Mi}, W'_{MjP_k}/P_k r_A r_B$ から，連立方程式を解くことで乗算結果 ab を得る．

4.4 記憶容量及び計算量，通信量

提案方式では，k 台で乗算が実現できるよう，乱数と分散情報にその乱数を掛けた秘匿化分散情報を導入した．よって，サーバ 1 台あたりの記憶容量は台数変化を生じる手法に比べて 3 倍の記憶容量を必要とする．ただし，台数変化が生じる手法では 2k-1 台に分散情報を持たせる必要があることを考えると，全体的には約 1.5 倍の記憶容量の増加となる．

また，乱数の分散，復元，秘匿化分散情報の生成，復元を考えると計算量も台数変化を生じる手法よりも増加することが明らかであり，4.3.3 の手法により一時的な乱数を発生させる場合には，通信量も増加する．

しかし，提案方式は台数変化をさせない，すなわちシステム構成に柔軟性を持たせることが出来る．CRYPTO2012 にも同様に k 台のサーバによって乗算を行う手法 [9] が提案されているが，提案方式の方が計算量などの点において優れていると考えられる．他の手法との記憶容量，計算量，

通信量に関する詳細な比較は今後の課題である．

4.5 まとめ

今回は，CSS2013 で発表した方式は，秘密情報を k 台未満のサーバを集めると知ることが出来る瞬間があるという問題点を解決した．また前回と同じように Shamir ベースで乗算を可能としているため，閾値 k とサーバ台数 n は，任意に変化させることが出来るシステムである．サーバが k 台しか稼働していなくても乗算演算実行可能，乗算結果復元可能なシステムを作った．

さらに，アクセス制御があり情報収集者が信頼できる場合と，アクセス制御がなく情報収集者が信頼できない場合の 2 パターンを作った．情報収集者が信頼できる場合の方式は，計算量，通信量を抑えられたと考える．

今後の課題としては，安全性の証明をつけ，実装もしたいと考える．また，計算量，通信量の比較をし，さらに記憶容量の削減のために，[19][20][21][22] などを適用し記憶容量の削減も図っていきたい．

参考文献

[1] 鈴木良介,”ビッグデータビジネス時代　現実にイノベーションを生み出すポスト・クラウドの戦略”, 飛泳社 (2011)
[2] 古関　聰，佐藤　直人,”Hadoop を活用した大規模データ解析の動向と今後の展望”，オペレーションズ・リサーチ:経営の科学，Vol.56(6)(2011)
[3] 柴田　英寿,”「クラウド化」と「ビッグデータ活用」はなぜ進まないのか？”，東洋経済新報社 (2012)
[4] 藤井 康広，佐藤 尚宜，吉野 雅之，原田 邦彦,”クラウドコンピューティング・ビッグデータ利活用を支える先進セキュリティ技術”, 日立評論 94(10), 731-735, 2012-10-00
[5] A. Shamir. How to share a secret. Communications of the ACM, 22, (11), pp.612-613(1979)
[6] 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)
[7] ROSARIO.G,MICHAEL.O.R,TAL.R,Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography.PODC ’98 Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing Pages 101-111 .
[8] 五十嵐 大, 千田 浩司, 高橋 克己,”高効率 3 パーティ秘匿関数計算の情報理論的安全性” Vol.2010-CSEC-50 No.46
[9] I.Damgård,V.Pastro,N.Smart,and S.Zakarias.Multiparty Computation from Somewhat Homomorphic Encryption. CRYPTO 2012, Vol. 7417 of Lecture Notes in Computer Science, pp 643-662.
[10] 布川敦史，渡辺泰平，須賀祐治，岩村惠市”計算主体を限定しない汎用的で軽量な秘匿関数計算の安全性とその拡張”SCIS2013
[11] 五十嵐大，濱田浩気，菊池亮，千田浩司,”小パーティの秘密分散ベース秘密計算のための $O(l)$ ビット通信ビット分解および $O(|p'|)$ ビット通信 Modulus 変換法”，CSS2013
[12] 加藤遼，桐淵直人，西脇雄高，吉浦裕,”定数ラウンド (k,n) 秘匿モジュロ変換プロトコルの提案”，CSS2011
[13] 尾形わかは.”情報理論的に安全なパスワード付秘密分散法の安全性と効率化”．SCIS2013
[14] Ali Bagherzandi, Stanisaw Jarecki, Saxena, Nitesh Saxena, Yanbin Lu,” Password-protected secret sharing.” Proceedings of the ACM CCS 2011, pp.433-444.
[15] 諸橋玄武，五十嵐大，菊池亮，千田浩司.”実用的な秘密分散法/マルチパーティ計算システムのための認証方式”. SCIS2013
[16] 坂崎尚生，側高幸治，長谷部高行，山田朝彦，大岩寛,”個人情報や企業情報を安全に活用するためのクラウドコンピューティング基盤の整備”CSS2011
[17] 志村正法，宮崎邦彦，西出隆志，吉浦裕,”秘密分散データベースの構造演算を可能にするマルチパーティプロトコルを用いた関係代数演算”，情報処理学会論文誌,Vol.51,No.9,1563-1578(Sep.2010)
[18] 江口　宏，尾形 わかは”n-1 人の不正者に対して安全なより効率の良い (k,n) 閾値法”．SCIS2012
[19] H. Krawczyk. Secret Sharing Made Short. Crypto’93,pp.136-146 (1994)
[20] 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)
[21] G. R. Blakley. Security of ramp schemes. Crypto’84, pp.242-268 (1984)
[22] 高橋　慧，岩村惠市,”クラウドコンピューティングに適した計算量的安全性を持つ秘密分散法” CSS2012
[23] 渡辺泰平，岩村惠市”秘密分散法を用いた乗算に関する一手法”CSS2013