Vol.2016-CSEC-73 No.19
Vol.2016-IOT-33 No.19
2016/5/27

効率の良い秘密分散断片の誤り位置特定および訂正プロトコル

高橋　慧[1]　濱田　浩気[1]　菊池　亮[1]　五十嵐　大[1]　桐淵　直人[1]

概要：秘密分散に基づく秘密計算は，複数のサーバに元データから生成した断片を分散保管し，各サーバが協調して演算を行うことで，元データを秘匿したまま演算結果だけを出力することができる．しかし，通信障害等により，あるサーバが持つ断片と他のサーバが持つ断片に不整合が発生した場合，利用者に正しい計算結果を返すことができなくなる．また，データの不整合を確認するために，断片を集めてしまうと元データの秘匿性が保てないという課題がある．そこで本研究では元データを秘匿したまま不整合の発生した断片を特定し，復旧する方式を提案する．

従来の断片不整合検出手法は，断片の集合における不整合の有無のみを出力するものであった．この手法で不整合の位置を特定するためには，二分探索を繰り返し不整合の有無を調べて位置を特定するため，多くの通信と計算が発生する．これに対して，提案方式では，誤り位置と値を出力可能な符号理論を秘密計算に適用し，二分探索を不要にすることで，通信量・計算量の削減を実現した．提案方式を適用することで，復旧時にサーバ間でやり取りするデータ量を約 80%削減するなど処理の効率化に貢献することができる．

An Efficient Technique for Specifying Error Locations and Correcting the Errors in Secretly-Shared Data

Satoshi Takahashi[1]　Koki Hamada[1]　Ryo Kikuchi[1]　Dai Ikarashi[1]　Naoto Kiribuchi[1]

1. はじめに

近年，クラウド事業者に対してデータを秘匿した状態での保存・演算を可能にする技術として，秘密計算が注目されている．主な秘密計算の方式としては，大まかに Yao の Garbled Circuit，準同型暗号によるもの，秘密分散を用いるものが知られている．

Yao の Garbled Circuit [1] は回路を暗号化する Garbler と，暗号化された回路を計算する Evaluator の 2 者間による秘密計算である．近年は幾つかのアプリケーションにおいて実用的な速度を出しているが，多者間の秘密計算への拡張方法は自明ではなく，後述する秘密分散による方式と比較すると処理速度で劣ることが多い．

準同型暗号による方式では，Paillier 暗号 [2] や ElGamal 暗号 [3] を用いた方式が典型的な例として挙げられる．2009 年には，乗算と加算の両方が可能な完全準同型暗号が提案され [4]，理論的には任意の回路を暗号化したまま演算可能であることが証明された．ただし，実用レベルでの性能評価においては，更なるブレイクスルーが必要であると言われている．

秘密分散による方式では，(k, n)-秘密分散技術を用いて平文から n 個の断片を生成し，うち k 個未満が漏えいした場合でも元のデータに関する情報が得られないという秘匿性を保ちながら，断片を保管したサーバが協調して演算を行うことで，演算結果を出力することができる．この方式は，乗算と加算の両方が同時に扱えるという機能性に加え，実装評価においても，LAN 環境での 10 万件のソート処理が約 0.7 秒という実用的な性能を達成している [5]．しかし，秘密分散による秘密計算においては，元データを複数のサーバで保管する性質上，断片不整合という課題が発生する可能性がある．

[1] NTT セキュアプラットフォーム研究所，〒 180-8585 東京都武蔵野市緑町 3-9-11，NTT Secure Platform Laboratories，3-9-11，Midori-cho Musashino-shi，Tokyo 180-8585 Japan

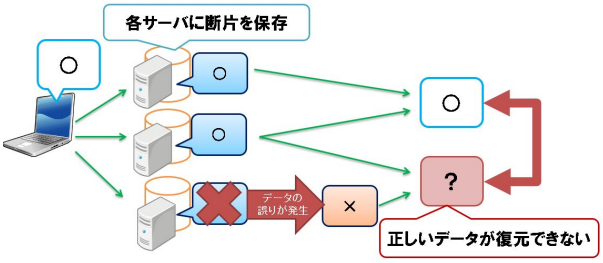

図 1　秘密計算における断片不整合

なお，本稿では Yao の Garbled Circuit や準同型暗号を含む秘匿化した演算技術の総称を広義の秘密計算とし，秘密分散に基づいたものを狭義の秘密計算とする．以降，単に秘密計算と言えば後者を指すものとする．

1.1　秘密計算における断片不整合

秘密計算では，複数のサーバに断片を分散保管しているが，図 1 のようにあるサーバが持つ断片に誤りが発生した場合，復元時にどのサーバに保管されている断片を用いるかによって得られる復元結果が異なってしまう他，この状態で秘密計算を実施すると，誤った演算結果が出力されることになる．このような状態を秘密計算における“断片の不整合”が発生した状態と定義する．

本稿ではこのような現象が発生する状況としてサーバ障害からの復旧時を想定する．従来一台のサーバで発生した障害から復旧する技術としては，バックアップを用いたリストア等が挙げられる．この技術を用いることで，サーバ障害が発生した場合でもバックアップを取得した時点までは正しく復旧することができる．しかしながら，この技術が保証できるのは一台のサーバの中で，正しいデータに復旧したという点のみであるため，秘密計算の様に複数のサーバ間で，正当性が確保できているか，つまり断片不整合が発生していないかという点は保障することができない．

ここで，これらの断片不整合の問題を解決するために，複数の元データが秘密分散されてる状況において，元データを秘匿したまま以下の 3 点を実現する必要がある．

- 不整合の発生検知
- 不整合の発生した断片の特定
- 不整合の発生した断片の復旧

本稿では，前述の 3 点を実現するためのアプローチとして，符号理論における一般的な誤り訂正手法である「リードソロモン符号」を秘密計算に適用した方式を提案する．提案方式は，断片不整合発生の有無だけではなく，その発生箇所や正しい断片の値まで一度の処理で出力することができる．

2.　従来手法

2.1　誤り特定手法

秘密計算技術において発生した断片の不整合を検知する従来手法として，2014 年に五十嵐らによって提案された手法がある [6]．この手法では他のサーバと不整合の発生した断片群を入力すると，入力した断片群に不整合，つまり誤りが有るか，無いかという情報のみを出力する．このため，複数の断片を入力として，不整合の発生箇所を特定するためには，この方式を繰り返し適用し二分探索を行う必要がある．

この手法は，入力した断片群のチェックサムを生成し，各サーバ間でそれらを比較することで不整合の検出を行う．よって，方式の適用回数に比例して通信回数が増加してしまうという課題がある．

2.2　誤り訂正手法

断片の誤り位置の特定後に実施する訂正処理として，五十嵐らによって 2013 年に提案された手法が知られている [7]．この手法では k 個以上の正しい断片から不整合の発生した断片を元のデータを復元することなく再構築することができる．

3.　提案手法

3.1　概要

提案手法では，秘密計算において発生する，断片の誤り検知及び発生位置特定・訂正を実施するために，一般的な誤り訂正手法として用いられている符合理論を秘密計算に適用する．ここで，利用する符号理論の技術はリードソロモン符号 [8] を採用する．この理由としてリードソロモン符号は 2 元の BCH 符号を多元に拡張したものであり，$GF(p)$ 上で演算を実施する秘密計算技術との相性が良いと考えられた為である．

3.2　検討方針

誤り位置特定及び訂正の新たな手法の提案にあたり下記の 2 点を検討方針とした．

(1) 秘密計算が実現する秘匿性を担保する
(2) 通信量・計算量を従来よりも削減する

1 つ目の方針は，秘密計算技術が従来担保する秘匿性である「各サーバの管理者へデータ漏えいを起こさない」という点を守ったまま断片の誤り検知及び発生位置特定・訂正を実現する．

また，2 つ目の方針については，秘密計算では，処理時に通信や演算が大量に発生するため，リードソロモン符号の全ての処理を秘密計算で行うのは非効率である．そこで，(1) の方針を守りつつ，秘密計算の適用範囲をできるだけ狭くし，必要な通信や計算を削減する．

3.3　準備

3.3.1　記法・定義

- p: 奇素数．

- n: サーバの台数．ただし，n は $2 < n < p$ の整数．
- P_i: i 番目のサーバ．ただし，i は $1 \leq i \leq n$ の整数．
- k:復元に必要な断片の個数．ただし，$1 \leq k \leq \frac{(n+1)}{2}$ の整数．
- $\mathbb{Z}_p$: 0 以上 p 未満の整数の集合．
- $[\![a]\!]$: 元データ $a \in \mathbb{Z}_p$ の断片の集合．
- $[\![a]\!]_i$: サーバ P_i が所有する元データ a の断片．

3.4 前提

提案方式は断片の誤りを検出し，その位置の特定及び訂正を行うものであり，あらかじめ，元データを断片の形式で各サーバが所有していることを前提とする．ここでは 3 パーティの秘密計算を想定し，元データ a は $n = 3, k = 2$ の Shamir 型秘密分散 [9] をされている物とする．また，誤った断片を持つサーバは特定されており，それ以外のサーバは正しい断片を所有するものとする．

- 元データの数を m とする．ただし，m は $1 \leq m < p$ の整数とする．
- 元データの値を a_i とする．ただし，i は $0 \leq i < m$ の整数とする．
- 誤りの発生した断片の数を t とする．ただし，t は $1 \leq t < m/2$ の整数とする．
- 正しい断片を所有するサーバを P_j とする．ただし，j は $j = 1, 2$ の整数とする．
- 誤りの生じた断片を所有するサーバを P_3 とする．
- P_3 の断片に付加される誤りの値を $e_{j_l}(0 \leq l \leq t-1)$ とする．ただし，j_l は誤り位置を表す．

以上により，P_3 のサーバが持つ m 個の断片の内，t 個の断片に誤りが含まれており，誤りの発生した断片 $[\![a_{j_l}]\!]'_3$ $(0 \leq l \leq t-1)$ は下記の様に書くことができる．

$$[\![a_{j_l}]\!]'_3 = [\![a_{j_l}]\!]_3 + e_{j_l} \tag{1}$$

3.5 方式手順

提案方式の流れは，

(1) 生成多項式 $G(x)$ の決定
(2) パリティ多項式 $R(x)$ の算出
(3) 受信多項式 $Y(x)$ の算出
(4) シンドロームの算出
(5) 誤り多項式，誤り数値多項式の算出
(6) 誤り位置の特定
(7) 誤り訂正

である．検討の結果，本提案手法では前述の 3.2 で述べた 2 つの検討方針を達成するために，「(2) パリティ多項式 $R(x)$ の算出」部分のみ秘密計算を用いて行い，残りの処理は平文で実施する．これにより，各サーバの持つ断片情報を秘匿しながら誤り位置の特定・復旧を実現し，必要な通信は正しい断片を持つサーバ P_1, P_2 から誤りの発生したサーバ P_3 へのパリティ送信部分のみに圧縮することができる．

ここでは，秘密計算に適用したために特殊な処理が必要となる，(4) まで説明を行うが，残りの (5)〜(7) の処理については，秘密計算特有の処理はなく，通常のリードソロモン符号での処理と同様であり，ピーターソン復号法 [10] やユークリッド復号法 [11] などの手法が知られている．

3.5.1 生成多項式 $G(x)$ の決定

α を $GF(p)$ の原始元とする．リードソロモン符号の生成多項式 $G(x)$ は，訂正可能な誤りの個数 t（ただし，$1 \leq t < \frac{p-1}{2}$）に対して適当な整数 b(ただし，$0 \leq b$) に対して $\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+2t-1}$ を根に持つ $GF(p)$ 上の多項式として次の式で与えられる．

$$G(x) = \prod_{i=0}^{2t-1} (x - \alpha^{b+i}) \tag{2}$$

提案方式では，秘密分散に用いるパラメータ p に適合するパラメータ t, α を事前に選び，全てのサーバで共有する．

3.5.2 パリティ多項式 $R(x)$ の算出

本方式では，断片の不整合が発生したサーバ P_3 が持つ断片を一般的な符号理論における，誤りを含んだ受信符号とみなし，リードソロモン符号を応用した手法により誤り位置の特定及び訂正を実施する．

そのために，正しい断片を持つサーバ P_1, P_2 は P_3 の持つ断片の誤り位置の特定及び訂正に必要なパリティ多項式 $R(x)$ を算出する．ここで，パリティ多項式 $R(x)$ は以下のように定義される $2t-1$ 次の多項式であり，

$$R(x) = \sum_{i=0}^{2t-1} r_i x^i \tag{3}$$

サーバ P_3 の正しい断片から作られる情報符号多項式 $C_3(x)$ を式 (4) と定義するとき，

$$C_3(x) = \sum_{i=0}^{m-1} [\![a_i]\!]_3 x^{(i+2t)} \tag{4}$$

式 (5) の関係が成立する．

$$X(x) = C_3(x) + R(x) = G(x)U(x) \tag{5}$$

ここで，$X(x)$ を送信符号と呼ぶものとする．

次に，$C_3(x)$ の係数である，P_3 が本来持つ誤りの無い断片 $[\![a_i]\!]_3 (0 \leq i \leq m-1)$ は P_1 および P_2 の断片を用いて下記の様に書くことができる．

$$[\![a_i]\!]_3 = \beta_1 [\![a_i]\!]_1 + \beta_2 [\![a_i]\!]_2 \tag{6}$$

β_j は P_3 の断片 $[\![a_i]\!]_3$ を P_1, P_2 の持つ断片から算出する Lagrange 補間するための係数である．

式 (6) の関係式から $C_3(x)$ は以下のように書くことができる．

$$C_3(x) = \sum_{i=0}^{m-1} (\beta_{i1}[\![a_i]\!]_1 + \beta_2[\![a_i]\!]_2)x^{i+2t} = \sum_{i=0}^{m-1} (\beta_1[\![a_i]\!]_1 x^{i+2t} + \beta_2[\![a_i]\!]_2 x^{i+2t}) \quad (7)$$

これより，送信符号 $X(x)$ は以下の様になる．

$$X(x) = C_3(x) + R(x) = \sum_{i=0}^{m-1} (\beta_1[\![a_i]\!]_1 x^{i+2t} + \beta_2[\![a_i]\!]_2 x^{i+2t}) + \sum_{i=0}^{2t-1} r_i x^i \quad (8)$$

ここで，$R(x)$ を求めるには，$r_i(0 \leq i \leq 2t-1)$ を求めればよい．式 (5) より，$x = \alpha^b, \alpha^{b+1}, \cdots, \alpha^{b+2t-1}$ とすると，$X(x) = 0$ となることから，$2t$ 本の連立方程式を解くことにより，$r_0, \cdots, r_{2t-1}$ を求めることができる．

ここで，$R(x)$ の係数 r_i から作られる行列を $\mathbb{R} = [r_0, \cdots, r_{2t-1}]^T$ とすると，

$$\mathbb{R} = -\mathbb{C}_1\mathbb{A}_C\mathbb{A}_R^{-1} - \mathbb{C}_2\mathbb{A}_C\mathbb{A}_R^{-1} = \mathbb{D}_1 + \mathbb{D}_2 \quad (9)$$

と書くことができる．ここで，$\mathbb{C}_i$ および $\mathbb{A}_C, \mathbb{A}_R$ は以下の様に定義する．

$$\mathbb{C}_i = [\beta_i[\![a_{m-1}]\!]_i, \cdots, \beta_i[\![a_0]\!]_i] \quad (10)$$

$$\mathbb{A}_C = \begin{bmatrix} \alpha^b & \cdots & \alpha^{(b+2t-1)(m-1+2t)} \\ \alpha^b & \cdots & \alpha^{(b+2t-1)(m-2+2t)} \\ \vdots & \ddots & \vdots \\ \alpha^b & \cdots & \alpha^{(b+2t-1)2t} \end{bmatrix} \quad (11)$$

$$\mathbb{A}_R = \begin{bmatrix} \alpha^b & \cdots & \alpha^{(b+2t-1)(2t-1)} \\ \alpha^b & \cdots & \alpha^{(b+2t-1)(2t-2)} \\ \vdots & \ddots & \vdots \\ \alpha^b & \cdots & 1 \end{bmatrix} \quad (12)$$

また，

$$\mathbb{D}_i = -\mathbb{C}_i\mathbb{A}_C\mathbb{A}_R^{-1} \ (i = 1, 2) \quad (13)$$

とすると，$\mathbb{D}_i$ は正しい断片を持つサーバ $P_i(i = 1, 2)$ が持つ，断片 $[\![a_j]\!]_i(0 \leq j \leq m-1)$ と定数の線形結合によって表すことができるため，サーバ P_i によって独立に算出可能である．

求められた，$\mathbb{D}_1, \mathbb{D}_2$ の要素 $[d_{(0,i)}, \ldots, d_{(2t-1,i)}]$ を係数とする，多項式を

$$D_i(x) = d_{(2t-1,i)}x^{2t-1} + \cdots + d_{(0,i)} \quad (i = 1, 2) \quad (14)$$

とするとき，求める $R(x)$ は以下のように書くことができる．

$$R(x) = D_1(x) + D_2(x) \quad (15)$$

上記の関係を元にアルゴリズム 1 示す手順によって $R(x)$ を算出する．

Algorithm 1 符号多項式 $X(x)$ の生成

1: 入力：P_1, P_2 の持つ断片 $[\![a_{m-1}]\!]_i \ldots [\![a_0]\!]_i (i = 1, 2)$
2: 出力：パリティ多項式 $R(x)$
3: 多項式 $R(x) = D_1(x) + D_2(x)$ とする．
4: **for** $i = 1$ **to** 2 **do in parallel**
5: 　P_i は以下の処理を実施．
6: 　式 (13) から，$\mathbb{D}_i$ を算出．
7: 　$\mathbb{D}_i(i = 1, 2)$ を P_3 に送信．
8: **end for**
9: P_3 は以下の処理を実施
10: $P_i(i = 1, 2)$ から受け取った $\mathbb{D}_i$ から $D_i(x)$ を算出
11: $R(x) = D_1(x) + D_2(x)$ を算出

Algorithm 2 受信符号多項式 $Y(x)$ の算出

1: 入力：断片 $[\![a_0]\!]'_3, \ldots, [\![a_{m-1}]\!]'_3$，パリティ多項式 $R(x)$
2: 出力：受信符号多項式 Y(x)
3: P_3 は以下の処理を実施．
4: 多項式 $C'_3(x) = \sum_{j=0}^{m-1}[\![a_j]\!]'_3 x^{(j+2t)}$ とおく．
5: アルゴリズム 1 で算出した $R(x)$ を用いて，受信符号多項式 $Y(x) = C'_3(x) + R(x)$ を生成．

Algorithm 3 シンドロームの算出

1: 入力：受信符号多項式 $Y(x)$，$\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+2t-1}$
2: 出力：シンドローム $s_{(j)}$ $(0 \leq j \leq 2t-1)$
3: P_n は以下の処理を実施
4: $x = \alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+2t-1}$ として，$s_{(j)} = Y_i(x)$ $(0 \leq j \leq 2t-1)$ を算出．

3.5.3 受信符号多項式 $Y(x)$ の算出

誤った断片を持つサーバ P_3 はアルゴリズム 1 で生成した $R(x)$ を用いて，受信符号多項式 $Y(x)$ を生成する．生成手順についてはアルゴリズム 2 に示す．

3.5.4 シンドロームの算出

破損した断片を持つサーバ P_3 は 3.5.3 節で生成した受信多項式 $Y(x)$ からシンドローム $s_{(j)} = Y(\alpha^j)$ を算出する．具体的な算出手順についてはアルゴリズム 3 に示す．

3.6 手法の一般化

本節ではこれまで述べて来た提案手法を (k, n) の Shamir 型秘密分散に拡張する手法について述べる．これまでの説明では $(2, 3)$ 秘密分散で生成された断片を想定して行っていたが，これを限定せずに (k, n) に拡張するのは容易である．具体的には 3.5.2 節で述べた，パリティ多項式 $R(x)$ の算出の部分を拡張することで実現することができる．

(k, n) 秘密分散においても式 (6) と同様，あるサーバ P_j が持つデータ a に関する断片 $[\![a]\!]_j$ は n 台中 k 台のサーバが持つ断片を用いて以下の様に表すことができる．

$$[\![a]\!]_j = \sum_{l=0}^{k-1} \beta_l[\![a]\!]_l \quad (16)$$

これにより，(k, n) 秘密分散を用いた場合パリティ多項式 $R(x)$ の係数 r_i $(1 \leq i \leq 2t-1)$ から作られる行列 $\mathbb{R}$ は以下の様に書くことができる．

$$\mathbb{R} = -\sum_{l=0}^{k-1} \mathbb{C}_l \mathbb{A}_C \mathbb{A}_R^{-1} \tag{17}$$

上記の処理によりパリティ多項式 $R(x)$ を算出した後の処理は $(2,3)$ 秘密分散の断片に適用する場合と同様である．

4. 評価

本章では，まず提案方式の安全性について評価を行う．その後，提案方式と従来方式を計算量・ラウンド数・通信量の 3 つの観点から比較した結果を示す．秘密計算のプロトコルは各サーバに閉じた演算およびサーバ間のデータ通信によって構成される．そのため，これら 3 つの観点について評価することで，必要な処理時間に関する評価を行うことができる．本稿ではそれぞれの評価軸を以下のように定義し評価を行う．

- 計算量：各サーバ内部で行われる乗算の回数
- 通信量：サーバ間でやり取りされるデータの総量
- ラウンド数：最大限並列化した上で必要な通信の回数

ここでは評価の簡略化のため，M 個の元データは $(2,3)$ 秘密分散で分散されており，これらの断片群に T 個の誤りが均等に混入しているというモデルを想定し，評価を行う．

4.1 安全性

本節では提案方式の安全性について考える．ここでは安全性の評価基準として，秘密計算が実現する秘匿性を担保できるかという観点で評価を行う．具体的には，(k,n)-秘密分散が保証する秘匿性つまり，断片を持つ各サーバに対して，元データの情報が漏洩しないかという点を評価する．

提案方式は (k,n)-秘密分散を拡張した秘密計算について，発生した断片不整合の位置特定及び訂正を行う手法であり，各サーバが持つ断片から情報が漏れないということは，(k,n)-秘密分散の安全性で担保されるものである．そのため，提案方式の安全性は，不整合発生サーバが k 台の正常な断片を持つサーバから受け取るパリティ情報から元データを復元することができるかという点に帰着することができる．

ここで，不整合の発生したサーバが正しい断片を持つサーバから受け取るパリティ情報は，式 (13) に示す情報である．ここで，$\mathbb{D}_i$ の要素 $[d_{(0,i)}, \ldots, d_{(2t-1,i)}]$ は，断片 $[\![a_j]\!]_i (0 \leq j \leq m-1)$ の線形結合によって表されるものである．これより，$2t < m$ という条件の下では，$\mathbb{D}_i$ の要素からそれぞれの断片 $[\![a_j]\!]_i (0 \leq j \leq m-1)$ を一意に定めることはできない．これより，本提案方式は $2t < m$ という条件の下では，(k,n)-秘密分散法と同等の安全性を持つ．

4.2 計算量

従来手法は誤りの位置特定及び復旧という大きく 2 つの処理に分けられる．まず，誤り特定においては，ある H 個の断片群を入力としたとき誤りの有無を出力するのに必要な乗算の回数は

$$H + k \tag{18}$$

と表すことができる．ここで，k はデータの復元に必要な断片の個数を示す．M 個の断片に誤りが T 個含まれる場合，二分探索的に誤り位置を特定する為，手法を適用する回数は

$$(T-1) + T\log_2 \frac{M}{T} \tag{19}$$

と書くことができる．この時，一項目の $(T-1)$ 回については M 個の断片全てに対して誤りの有無をチェックし，二項目の $T\log_2 \frac{M}{T}$ 回の部分でチェックする断片の個数は $M \times (\frac{1}{2})^{(T-1)}$ を初項とし，公比が $\frac{1}{2}$，項数 $T\log_2 \frac{M}{T}$ の等比数列として表すことができる．これにより必要な乗算の回数はその等比数列の和で表される．よって誤り特定に必要な計算量は，

$$\begin{aligned} &M(T-1) + 2M(\frac{1}{2})^{(T-1)}(1 - \frac{T}{M}) \\ &+ k((T-1) + T\log_2 \frac{M}{T}) \end{aligned} \tag{20}$$

となる．また，従来手法において T 個の誤りの復旧に必要な計算量は，

$$T \times k \tag{21}$$

となる．

以上により従来手法を用いて M 個の断片から T 個の誤りを特定し，訂正するために必要な乗算の回数は以下の様になる．

$$\begin{aligned} &M(T-1) + 2M(\frac{1}{2})^{(T-1)}(1 - \frac{T}{M}) \\ &+ k((T-1) + T\log_2 \frac{M}{T}) + Tk \end{aligned} \tag{22}$$

なお，従来手法においては，全てのサーバで上記の処理を行う必要がある．

次に提案手法について考える．提案方式では計算量を圧縮するために，M 個の断片群を m 個づつに分割 $(2 < m < M)$ し，処理を行う．ここで分割数は $L = \frac{M}{m}$ と表し，各 m 個づつの断片群には t 個の誤りが含まれる．また，t と T の関係は $T = L \times t$ と表すことができる．提案手法では正しい断片を持つサーバ P_1, P_2 と復旧対象のサーバすなわち誤った断片を持つサーバ P_3 で行う乗算の回数が異なる．まず，正しい断片を持つサーバ P_1, P_2 について考える．提案方式ではこれらのサーバは復旧対象サーバ P_3 に送るパリティの生成を行う．このために必要な乗算の回数は以下の様になる．

$$2 \times t \times m \times L \tag{23}$$

次に，復旧対象サーバ P_3 が行う処理について考える．復旧対象サーバは P_1, P_2 から受け取ったパリティ及び自身の持つ誤りを含んだ断片を用いて 3.5 節で述べた手順で誤り

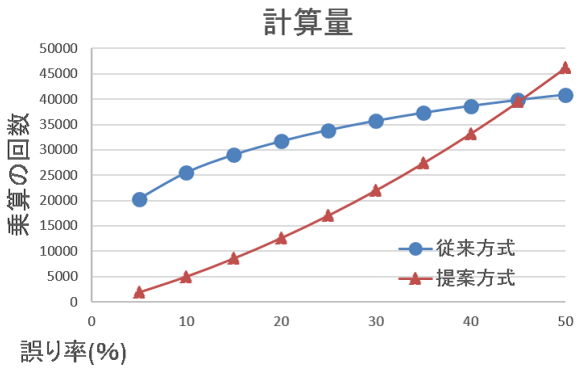

図 2　従来手法と提案手法の計算量についての比較結果

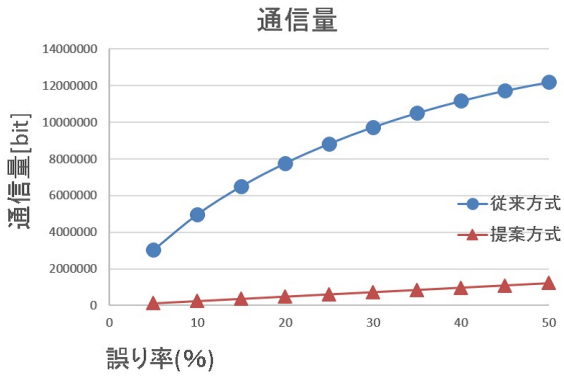

図 3　従来手法と提案手法の通信量についての比較結果

の位置の特定，訂正を行い，全体で必要な計算量は以下の通りとなる．

$$L\{2tm + (2t-1)^2 + m(t-1) + t(t-1) + O(t^2)\} \quad (24)$$

ここで，$O(t^2)$ は誤り位置・数値多項式の生成においてユークリッド復号法を用いた場合の計算量のオーダーを表している．上記のそれぞれの算出結果に対して，具体的な数値を代入し計算量に関する評価を行った結果を図 2 に示す．

評価の条件は $M = 10,000, L = 1,000$ とし，含まれる誤りを $T = 500$ から $5,000$ まで 500 ずつと変化させて評価を行った．この結果より，提案手法は一定の誤り率以下では従来手法と比較し計算量の削減効果があるといえる．

4.3　ラウンド数

従来手法については、前述の計算量の評価と同様，誤りの特定及び訂正という処理に分けて考える．まず，ある任意の H 個の断片群について誤りの有無を出力するために必要なラウンド数は 2 ラウンドである．ここで，M 個の断片群から T 個の誤りを特定するために，二分木による探索を行った場合，誤りを特定するためには，木の深さ分，つまり $\log_2 M$ ラウンド必要となる．次に，1 つの断片の誤りの訂正に必要なラウンド数は 1 ラウンドであり，T 個の断片の誤り訂正は全て同時に行うことができる．これより，従来手法において，M 個の断片群における T 個の誤りを訂正するために全体で必要なラウンド数は以下の通りとなる．

$$2 \times \log_2 M + 1 \quad (25)$$

次に提案手法について考える．提案手法の場合必要な通信は P_1, P_2 から P_3 パリティを送信する部分のみであり，m 個の誤り位置特定及び訂正に必要なラウンド数は 1 ラウンドである．M 個のデータに対して適用した場合も各 m 個ずつについて並列にパリティ送信を行うことで，ラウンド数を 1 ラウンドにすることができる．この評価結果より，提案手法は誤り訂正を行う断片の個数に依存せずにラウンド数を決定することができると言える．

4.4　通信量

従来手法についてはこれまで同様，誤りの特定及び訂正という処理に分けて考える．従来手法では一度の方式適用で必要な通信量は適用する断片群の個数に関わらず，以下のように書くことができる．

$$3 \times n \times (n-1) \quad (26)$$

また，誤り訂正手法については 1 個の断片を復旧するために，$2k$ の通信量が必要となる．これまでと同様，M 個中 T 個の誤りについて特定・訂正をする場合に必要な通信量は，以下の様になる．

$$3n(n-1) \times ((T-1) + T \log_2 \frac{M}{T}) \quad (27)$$

次に提案手法について考える．提案手法では P_1, P_2 から P_3 へパリティを送信する場合のみ通信を行い，前述の計算量同様 L 個のブロックに分割した場合に全体で必要な通信量は以下の様になる．

$$2tk \times L \quad (28)$$

ここで，上記の算出結果に対して，4.2 節で述べた計算量の評価と同様の条件で実施した評価結果を図 3 に示す．

この結果より，提案手法は従来手法と比較し，誤り率などの条件によらず大きな通信量削減効果を期待することができると言える．

4.5　具体的な削減効果

ここでは，より具体的な削減効果を評価するため，通信がボトルネックになる環境を想定し，現実的な通信速度と通信遅延から従来手法と提案手法の処理時間の理論値を算出する．

具体的には，データ数：1 万件，誤り率：35 %，データサイズ：61bit，通信帯域：42.2Mbps，通信遅延：150ms という条件の下，復旧時に必要な時間の削減効果を算出した．その結果，従来手法の処理時間は 52 分程度だが，提案手法を用いることで，2 分 30 秒程度まで削減することができ

る．ただし，本評価は 4.3，4.4 節で算出した結果に前述の条件を当てはめた結果であり，実際には並列処理の適用等により結果が変動する可能性がある．

5. まとめ

本稿では，秘密計算技術において発生する断片不整合に対応するために，秘密計算の実現する安全を担保しながら，以下の 3 点を実現する手法として一般に誤り訂正技術として利用されているリードソロモン符号のアルゴリズムを秘密計算に適用した手法を提案した．

- 不整合の発生検知
- 不整合発生箇所の特定
- 不整合の発生した断片の復旧

また，通信量・計算量についても従来の手法よりも削減することができる．特に通信量やラウンド数について大きな削減効果を得ることができ，この結果，通信がボトルネックになる環境においては障害からの復旧時間を 50 分程度から 3 分程度まで削減できる見込みを得た．また，提案手法の適用範囲は理論上，準同型性を持つ秘密分散ベースの秘密計算全て (例：Additive な秘密分散 [12]) に適用可能である．

参考文献

[1] A. Yao, “How to Generate and Exchange Secrets (Extended Abstract),” FOCS, pp.162–167, 1986.
[2] P. Paillier. “Public-key Cryptosystems Based on Composite Degree Residuosity Classes” EUROCRYPT’99, pp.223-238.
[3] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” Information Theory, IEEE Transactions on 31(4), pp.469-472, 1985.
[4] C. Gentry, “A fully homomorphic encryption scheme,” PhD Thesis, Stanford University, 2009.
[5] 濱田浩気, 五十嵐大, 菊池亮, 千田浩司, 諸橋玄武, 冨士仁, 高橋克巳, “実用的な速度で統計分析が可能な秘密計算システム MEVAL,” CSS 2013, 2013.
[6] D.Ikarashi, R.Kikuchi, K.Hamada and K．Chida． “Actively Private and Correct MPC Scheme in $t < n/2$ from Passively Secure Schemes with Small Overhead，” Cryptology ePrint Archive, Report 2014/304, 2014.
[7] 五十嵐 大，菊池 亮，濱田 浩気，千田 浩司．“マルチパーティ計算可能な秘密分散におけるデータ改ざん検知方法，”SCIS2013，2013.
[8] 西村 芳一．“改訂新版 データの符号化技術と誤り訂正の基礎，” CQ 出版社，pp109-132， 2011.
[9] A.Shamir．“How to share a secret，” Communication of the ACM 22.11 pp612-613， 1979.
[10] 和田山 正．“誤り訂正技術の基礎，” 森北出版，pp76-83，2010.
[11] 平澤 茂一，笠原 正雄．“ユークリッド復号法，” 電子情報通信学会 基礎・境界ソサイエティ Fundamentals Review Vol. 4 (2010) No. 3 P 183-191，2011.
[12] 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.