

簡易的秘算によるクラウド利用インタフェースの開発と応用

高橋康太^{1,a)}

佐藤文明^{1,b)}

クラウドに保存されたデータの漏えいや不正使用を防ぐために、データを暗号化したまま計算できる秘算法の利用が研究されている。初期の秘算方式では計算量が多いため実用的ではなかったが、近年、計算量の小さい秘算方式であるセキュアマルチパーティ法や簡易的秘算法といったマルチパーティ型秘算が提案されている。しかし、ユーザはデータを分散させたり統合するための複雑な手順をプログラムする必要がある。本研究ではマルチパーティ型秘算における利用インタフェースを開発し、具体的な問題に応用することである。本研究では、フーリエ変換、範囲検索、そして外れ値検出に適用する場合の簡易的秘算法の利用方法について示す。

1. はじめに

近年、クラウドコンピューティングが普及しつつある。クラウドコンピューティングは計算能力を柔軟に拡大、縮小でき、総合的に情報システムのコストを下げうるなどの長所がある反面、データの保管や情報処理ノウハウ(プログラム)を外部企業(クラウド事業者)に委託することにより、データや処理方法の不正使用や漏洩のリスクをもつという短所がある。この短所ゆえに、委託するデータやプログラムの性質によっては、利用者は十分な安心感をもって利用することができない。

この課題に対して、委託するデータの秘算保持や、計算結果の秘算保持を目指したセキュアマルチパーティ法[1]や簡易的秘算[2][3][4]が提案されている。また、この簡易的秘算法が平均、分散、相関係数といった統計処理に対して適用できることが分かっている。本研究では、この簡易的秘算法をさらに複雑な応用に適用する際の課題を検討する。従来の計算では、加減算と乗除算が混在した計算や、範囲検索などの比較処理については、一旦中間結果をクライアントに戻す処理が必要となっていたが、その処理をサーバ間で交換することで、クライアント側には最終結果のみを返す仕組みを導入した。その仕組みを使った計算として、位置情報における近傍ノードの検索処理について検討する。また、実用的な問題として、フーリエ変換への応用について検討した結果を報告する。

2. 軽量秘算

2.1 セキュアマルチパーティ法による秘算

セキュアマルチパーティ法[1]では、秘算の保持と信頼性を両立させるために、秘算にしたい数値データ a, b (0 以上 m 未満の整数) について、 n 個のサーバに分散させ、それらのサーバのうち、異なる k 個のサーバからデータを得て、

元の数値 a, b を復元する仕組みとなっている(本稿では、 k out of n と略する場合がある)。本研究での提案方式を説明する上で、本稿では、既存方式の説明が必要である。文献[1]に従い、 $k=2, n=3$ の場合について、基本的動作を紹介する。

数値 a, b を、乱数を用いるなどで、 $a=a_0+a_1+a_2$ 、 $b=b_0+b_1+b_2$ となるように、 (a_0, a_1, a_2) 及び (b_0, b_1, b_2) に自社サーバにおいて分割し、 (a_0, a_1) 、 (b_0, b_1) を外部サーバ 0 に、 (a_1, a_2) 、 (b_1, b_2) を外部サーバ 1 に、 (a_2, a_0) 、 (b_2, b_0) を外部サーバ 2 に分散配置する。これらのデータは 3。1 で触れたが、それぞれ a, b の「シェア」と称される。

この a と b についての分割、分散配置を図 1 に示す。c については、乗算用であり、本節後半に述べる。

このような準備の下に、ある計算要求が発生したとする。まずは、 a と b の加算(減算も同様)の場合、図 1 において、サーバ n ($n=0, \dots, 2$) では a_n と b_n の加算及び a_{n+1} と b_{n+1} の加算を計算する($n+1$ は mod 3 の計算)。それぞれの結果は、サーバ n に対して、秘算が保持されている。それぞれの結果がユーザ側(自社サーバ)に返答される。3 個のサーバのうち 2 個から計算結果が得られれば、求める結果が得られる。ユーザ側では、それぞれの結果を加算することによって、計算要求の結果を得ることができる。

$$(a_0+b_0)+(a_1+b_1)+(a_2+b_2)=(a_0+a_1+a_2)+(b_0+b_1+b_2) \\ =a+b$$

係数のかかった重み付け加減算についても同様である。

次に、計算要求が乗算の場合、動作は複雑になるが、概略、次のようなプロセスで結果を得ることができる。この場合にも、3 個のサーバのうち、2 個のサーバから計算結果を得ることにより、要求された計算結果を得ることができるが、途中のプロセスでは、3 個のサーバはすべて正常に動作していることが必要である。

(1)サーバ 0 において

1)ランダム整数 (0 以上 m 未満) c_0, r_1, r_2 を生成する。

2) $c_1 = (a_0+a_1) \cdot (b_0+b_1) - r_1 - r_2 - c_0 \pmod{m}$ を生成する。

3)サーバ 1 に (c_1, r_1) を送る。

4)サーバ 2 に (c_0, r_2) を送る。

1 東邦大学理学部
Toho University, Faculty of Science,
2-2-1 Miyama, Funabashi, Chiba, 274-8510, Japan.

a) 5513065t@nc.toho-u.ac.jp
b) fsato@is.sci.toho-u.ac.jp

- 5)サーバ0 にシェアとして (c0, c1) を保持する。
 (2)サーバ1 において
 1)y = a1・b2+a2・b1+r1(mod m)を計算する。
 2)y をサーバ2 に送る。
 (3)サーバ2 において
 1)z = a2・b0+a0・b2+r2(mod m)を計算する。
 2)z をサーバ1 に送る。
 (4)サーバ1、2 において
 1)c2 = y+z+a2・b2 を計算する。
 2)サーバ1 にシェアとして (c1, c2) を保持する。
 3)サーバ2 にシェアとして (c2, c0) を保持する。以上の準備のもとで、異なる2個のサーバから得られる c0, c1, c2 を加算することによって、a と b の積 a・b が得られる。

$$\begin{aligned} & c0+c1+c2 \\ &=c0+(a0+a1) \cdot (b0+b1) - r1 - r2 - c0 \\ & +(a1 \cdot b2+a2 \cdot b1+r1)+(a2 \cdot b0+a0 \cdot b2+r2)+a2 \cdot b2 \\ &=a0 \cdot (b0+b1+b2)+a1 \cdot (b0+b1+b2)+a2 \cdot (b0+b1+b2) \\ &=(a0+a1+a2) \cdot (b0+b1+b2) \\ &=a \cdot b \end{aligned}$$

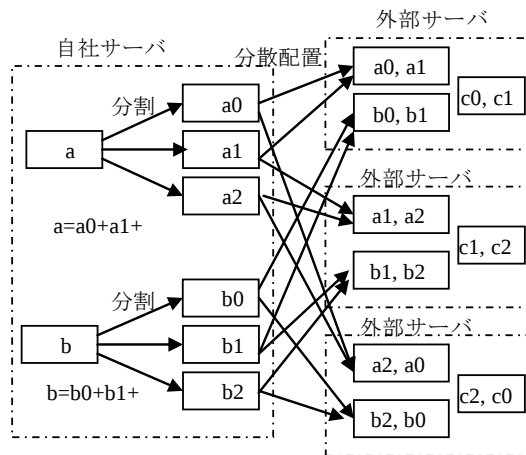


図1 秘密にすべきデータの分割と分散配置

2.2 簡易的秘密計算法

マルチパーティ法による秘密計算はデータが2つの場合は、あらかじめ (c0, c1)、(c1, c2)、(c2, c0) を計算し、その結果を1組のシェアとして、各サーバに保持すればいい。しかし、データが多くなるに従い、2項の乗算の場合に限っても、組み合わせ数の2分の1の数の組をシェアとして保持する必要がある。データ種類(属性)の数をpとすると、シェアの組の数qは $q=p \cdot (p-1)/2$ となる。RDBの場合、これに行の数が増える。この(c0, c1)、(c1, c2)、(c2, c0)をシェアとして保持する方式では、データの量が課題になるという問題がある。

そこでこれらの問題を回避するために簡易的秘密計算法

[2]が提案された。この簡易的秘密計算法はセキュアマルチパーティ法の加減算用シェアに加え乗除算用のシェアを作ることによって問題を回避している(図2)。

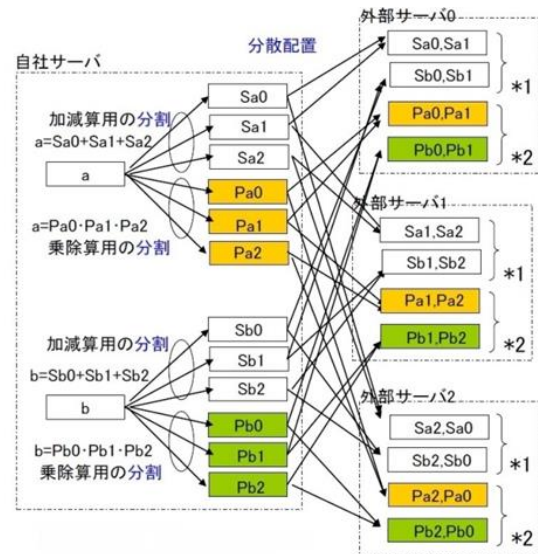


図2 簡易的秘密計算法

加減算用のシェアは従来のセキュアマルチパーティ法と同様に、数値a、bを、乱数を使って $a=Sa_0+Sa_1+Sa_2$ 、 $b=Sb_0+Sb_1+Sb_2$ となるように(Sa_0, Sa_1, Sa_2)および(Sb_0, Sb_1, Sb_2)を自社サーバで分割し、(Sa_0, Sa_1)、(Sb_0, Sb_1)を外部サーバ0に、(Sa_1, Sa_2)、(Sb_1, Sb_2)を外部サーバ1に、(Sa_2, Sa_0)、(Sb_2, Sb_0)を外部サーバ2に分散配置する。

次に乗除算用のシェアとして数値a、bを乱数を使って $a=Pa_0 \cdot Pa_1 \cdot Pa_2$ 、 $b=Pb_0 \cdot Pb_1 \cdot Pb_2$ となるように(Pa_0, Pa_1, Pa_2)および(Pb_0, Pb_1, Pb_2)を自社サーバで分割し、(Pa_0, Pa_1)、(Pb_0, Pb_1)を外部サーバ0に、(Pa_1, Pa_2)、(Pb_1, Pb_2)を外部サーバ1に、(Pa_2, Pa_0)、(Pb_2, Pb_0)を外部サーバ2に分散配置する。

従来の方式と同じように $k=2, n=3$ としたときを考える。加減算は同じである。乗算はサーバnでPanとPbnおよびPan+1とPbn+1の乗算を行う。3つのうち2つから結果を得られれば $Pa_0 \cdot Pb_0$ 、 $Pa_1 \cdot Pb_1$ 、 $Pa_2 \cdot Pb_2$ が揃うのでそれを用いればa・bを求めることができる。

$$\begin{aligned} & (Pa_0 \cdot Pb_0) \cdot (Pa_1 \cdot Pb_1) \cdot (Pa_2 \cdot Pb_2) \\ &= (Pa_0 \cdot Pa_1 \cdot Pa_2) \cdot (Pb_0 \cdot Pb_1 \cdot Pb_2) \\ &= a \cdot b \end{aligned}$$

除算についても同様に、サーバnでPanとPbnおよびPan+1とPbn+1の除算を行う。3つのうち2つから結果を得られれば Pa_0/Pb_0 、 Pa_1/Pb_1 、 Pa_2/Pb_2 が揃うのでそれを用いればa/bを求めることができる。

$$(Pa_0/Pb_0) \cdot (Pa_1/Pb_1) \cdot (Pa_2/Pb_2)$$

$$= (Pa_0 \cdot Pa_1 \cdot Pa_2) / (Pb_0 \cdot Pb_1 \cdot Pb_2) \\ = a/b$$

これら全ての場合も各サーバが持っているシェアは3つのうちの2つのみであり、秘密は保持される。また、 k 、 n に対しても単純な計算方法なので、自由に選択し、システムを構成することができる。また、この簡易的秘密計算法では一度分けた和用乱数、積用乱数をそのまま使用するため、保存データ量は $O(\text{データ数})$ となる。

3. 簡易的秘密計算法の課題

簡易的秘密計算法では、加減算と乗除算が混在する計算、範囲検索のような比較計算では、中間結果を一旦クライアントに戻す必要がある。中間結果を一度クライアントに戻さない計算方法としては、サーバ間で情報を交換してサーバで最終結果まで計算してもらう仕組みが必要となる。

単にサーバで情報を交換すると、計算結果が漏れてしまうことになるため、中間結果に乱数を加えるかあるいは掛けることで、中間結果が漏れないようにする。また、サーバが中間結果を、交換して計算したのち、その最終結果をクライアントに戻すときには、中間結果に加えた乱数の逆数を加えることで、最終的な計算結果を得るものとする。

具体的には次のように計算する。

例えば、 a 、 b 、 c 、 d の4つのデータを2つのクラウドサーバに秘密分散しているとする。それぞれ積のシェアを ap_0 , ap_1 , bp_0 , bp_1 , cp_0 , cp_1 , dp_0 , dp_1 とする。このときに、 $y = a \cdot b + c \cdot d$ となるような y の計算を要請したとすると、クラウドサーバ0では、 $x_0 = ap_0 \cdot bp_0$, $y_0 = cp_0 \cdot dp_0$ 、クラウドサーバ1では、 $x_1 = ap_1 \cdot bp_1$, $y_1 = cp_1 \cdot dp_1$ を計算するが、これをクラウド上で足すことはできず、クライアントに返して加算をする必要があった。この状況では、2つの配列データの積の総和といった計算がクラウド上でできない。

そこで、クラウドサーバ0に r_0 、クラウドサーバ1に r_1 という乱数を配置し、サーバ間で情報を交換する。つまり、クラウドサーバ0上の x_0 , y_0 に r_0 をかけたものをクラウドサーバ1に送り、クラウドサーバ1上の x_1 , y_1 に r_1 を掛けたものをクラウドサーバ0に送る。その結果、各クラウドには、 $r_0 \cdot x_0$, $r_0 \cdot y_0$, $r_1 \cdot x_1$, $r_1 \cdot y_1$ がある。これから、

$$T = r_0 \cdot x_0 + r_0 \cdot y_0 + r_1 \cdot x_1 + r_1 \cdot y_1 \\ = (r_0 + r_1)(x_0 + y_0 + x_1 + y_1) \\ = (r_0 + r_1)(ap_0 \cdot bp_0 + ap_1 \cdot bp_1 + cp_0 \cdot dp_0 + cp_1 \cdot dp_1) \\ = (r_0 + r_1)(a \cdot b + c \cdot d)$$

が計算できる。T をクライアントに返せば、 $1/(r_0 + r_1)$ で y が求まる。また、この計算を使うことで、クライアントに返すことなく、2つの配列データの積の総和といった計算ができるようになる。

4. 簡易的秘密計算法の応用

4.1 離散フーリエ変換

離散フーリエ変換 (DFT) は、時系列データを周波数領域に変換して分析、処理するために広く用いられている計算である。例えば、スマートフォンの加速度データをクラウドに蓄積し、それを DFT によって各周波数成分を求め、特徴分析をすることができる。

DFT の関係式は、以下の通りである。

N 個の入力データ ($f(0)$, $f(1)$, ..., $f(N-1)$) に対して、変換式 $F(k) = \sum_{n=0}^{N-1} f(n)e^{-j2\pi kn/N}$ ($k=0, 1, \dots, N-1$) で定義される

N 個の数列 F を f に対する離散フーリエ変換という。ただし、 j は虚数単位である。また、次のような逆変換式

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k)e^{j2\pi kn/N} \quad (n=0, 1, \dots, N-1)$$

で定義される数列 f を F に対する離散逆フーリエ変換 (Inverse Discrete Fourier Transform; IDFT) という。これらは変換と逆変換の関係にある。記法を簡単にするために

$W_N = e^{-j2\pi/N} = \cos(2\pi/N) - j\sin(2\pi/N)$ と置くことが多い。このように置き換えると DFT、IDFT の式はそれぞれ

$$F(k) = \sum_{n=0}^{N-1} f(n)W_N^{kn} \quad (k=0, 1, \dots, N-1)$$

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k)W_N^{-nk} \quad (n=0, 1, \dots, N-1)$$

と表せる。この W_N は回転因子や、ひねり因子と呼ばれる。

時系列データがクラウドサーバに秘密分散によって保存されているものとする。そのデータを用いて DFT を計算するには、まず時系列データを乱数を用いて、足し合わせたときに元に戻るよう分割する。例えば、時系列データの実部を $fi1[i]$ ($i=0, 1, \dots, N-1$)、時系列データの虚部を $fi2[k]$ ($k=0, 1, \dots, N-1$) とし、実部用の乱数を r_1 、虚部用の乱数を r_2 とする。これらのデータをそれぞれ2つに分割しようとしたならば、式はそれぞれ $fi1[i]*r_1$ と $fi1[i]*(1-r_1)$ (実部の式)、 $fi2[i]*r_2$ と $fi2[i]*(1-r_2)$ (虚部の式) となる。分割した実部の時系列データと虚部の時系列データをつづつ持つように分散すればよい。後はそれぞれのクラウドサーバで DFT を計算し、最後にそのデータを足し合わせることで求めることができる。

4.2 位置情報による近隣ノード検索

ノードの位置情報による近隣ノード検索の例を示す。図3に示したのは、 i 番目のクライアント $Client[i]$ から位置情報が2つのクラウドサーバ $Cloud1$ と $Cloud2$ に秘密分散で保存されている状況である。 $Client[i]$ の緯度は $a[i]$ 、経度は

$b[i]$ として表記している。サーバには、この緯度経度情報が $(a[i]_1, b[i]_1)$ および $(a[i]_2, b[i]_2)$ として保持されているものとする。

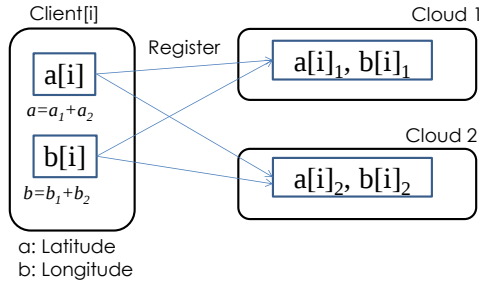


図3 位置情報のクラウドへの保存

ここで、ある緯度経度の範囲あるノードを検索する問題を考える。例えば、緯度が t 度から s 度、経度が p 度から q 度の間にあるノードを検索する。この検索要求は、クライアントから分割されて、クラウドに送付される。つまり、4つの乱数を用いて p, q, s, t が $p_1, p_2, q_1, q_2, s_1, s_2, t_1$ および t_2 に分割され、それぞれ p_1, q_1, s_1 および t_1 が Cloud1 に送信され、 p_2, q_2, s_2 および t_2 が Cloud2 に送信される。

Cloud1 では、 p_1 と a_1 そして q_1 との差分が計算される。同様に、 s_1 と b_1 そして t_1 との差分が計算される。Cloud2 についても同様に差分が計算される。その後、Cloud1 の計算結果を Cloud2 へ、Cloud2 の計算結果を Cloud1 へ送信する。

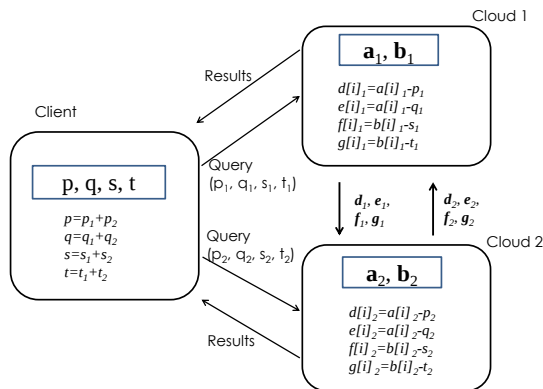


図4 位置情報の範囲検索

例えば、図4にそって説明する。ここで、検索範囲の表現で $p > q$ そして $s > t$ の関係があるとする。また、秘密計算の前提から、次の条件が成り立っている。

$$p = p_1 + p_2$$

$$\begin{aligned} q &= q_1 + q_2 \\ s &= s_1 + s_2 \\ t &= t_1 + t_2 \end{aligned} \quad (2)$$

(p_1, q_1, s_1, t_1) は Cloud1 に送られて、 $a[i]_1, b[i]_1$ との差分 $d[i]_1, e[i]_1, f[i]_1$ および $g[i]_1$ が計算される。

$$\begin{aligned} d[i]_1 &= a[i]_1 - p_1 \\ e[i]_1 &= a[i]_1 - q_1 \\ f[i]_1 &= b[i]_1 - s_1 \\ g[i]_1 &= b[i]_1 - t_1 \end{aligned} \quad (3)$$

(p_2, q_2, s_2, t_2) は Cloud2 に送られて、 $a[i]_2, b[i]_2$ との差分 $d[i]_2, e[i]_2, f[i]_2$ および $g[i]_2$ が計算される。

$$\begin{aligned} d[i]_2 &= a[i]_2 - p_2 \\ e[i]_2 &= a[i]_2 - q_2 \\ f[i]_2 &= b[i]_2 - s_2 \\ g[i]_2 &= b[i]_2 - t_2 \end{aligned} \quad (4)$$

これらの結果は Cloud1 と Cloud2 に送られて、その結果が統合される。

$$\begin{aligned} d[i] &= d[i]_1 + d[i]_2 \\ e[i] &= e[i]_1 + e[i]_2 \\ f[i] &= f[i]_1 + f[i]_2 \\ g[i] &= g[i]_1 + g[i]_2 \end{aligned} \quad (5)$$

ここで

$$\begin{aligned} d[i] &= d[i]_1 + d[i]_2 \\ &= (a[i]_1 - p_1) + (a[i]_2 - p_2) \\ &= (a[i]_1 + a[i]_2) - (p_1 + p_2) \\ &= a[i] - p \end{aligned} \quad (6)$$

従って、もし $d[i] < 0$ であつ $e[i] > 0$ であれば、それは $p > a[i] > q$ つまり緯度は範囲内にあることになる。同様に、もし $f[i] < 0$ であつ $g[i] > 0$ であれば、それは $s > b[i] > t$ つまり経度は範囲内にあることになる。従って、各クラウドは位置情報が指定された範囲内にあるかどうかを判定できることになる。

ここで、それぞれの計算過程で情報が漏れないことを確認する。まず、情報の保存においては各クライアントの位置情報は秘密分散されており、各クラウドが元の情報を知ることはできない。また、検索において、検索範囲の値は秘密分散されており、元の検索範囲に

についての情報を知ることとはできない。また、各クラウドで計算された検索範囲のとの差分情報であるが、この計算結果は乱数と同様の性質を持つため、この情報からもとのクライアントの位置情報を得ることはできない。ただし、検索範囲にそのクライアントが入っていることはわかるため、検索範囲に入るだけの近さにクライアント同士が存在していることは情報として得られる。

4.3 外れ値検出

データが他の値から大きく外れた値を検出する問題に適用する場合を考える。外れ値検出は、例えば工場の生産ラインにおける不良品の検出、株価の下落による不景気の予測、あるいは Web アクセス数の急激な増加によるクラッキング検出などに応用できる。

最もよく用いられるのは、データが正規分布に従うと仮定して、平均値から 2σ 、あるいは 3σ (σ は標準偏差) より離れているデータを外れ値として検出するものである。そのほかに、Smirnov-Grubbs 検定は、外れ値を検定で棄却して検出する方法である。帰無仮説「全てのデータは同じ母集団に属する」を、有意水準 α で片側検定を行っていく。一番平均値から離れているデータから、順に仮説検定していく。

いま、データ $x[i]$ ($i=0, \dots, N-1$) のそれぞれが、 3σ を超えているかどうかで、外れ値を検出するプロセスを、簡易型秘密計算で行う方法について述べる。いま2つのクラウドサーバにデータを分散して保存すると仮定する。それぞれのクラウドには、和のシェアを $x[i]_{s1}$ と $x[i]_{s2}$ 、積用のシェア $x[i]_{p1}$ 、 $x[i]_{p2}$ がそれぞれ配置されているものとする。また、クラウド間のデータ交換用の乱数 $r11$ と $r12$ をそれぞれのクラウドに配置しておく。また、クライアントは、 $r11 \cdot r12 \cdot r13=1$ になる乱数 $r13$ を持っているとする。

外れ値の計算は、(1) データの平均値を求める、(2) データの分散を求める、(3) 平均からのずれが 3σ を超えているかを判定する、3つのステップからなる。

(1) の平均値 E を求める手順は、各クラウドで和のシェアの総和 ($e1$ 、 $e2$ とする) を計算し、クライアントでその和を N で割って求める ($E=(e1+e2)/N$)。

$$\begin{aligned} (e1+e2)/N &= (\sum x[i]_{s1} + \sum x[i]_{s2})/N \\ &= (\sum x[i])/N \\ &= E \end{aligned}$$

(2) の分散 V を求める手順は、まず各クラウドで、 $x2[i]_{p1}=x[i]_{p1} \cdot x[i]_{p1} \cdot r1$ 、 $x2[i]_{p2}=x[i]_{p2} \cdot x[i]_{p2} \cdot r2$ を計算する。次に、 $x2[i]_{p1}$ と $x2[i]_{p2}$ とを交換する。そして、各クラウドで、 $d=\sum x2[i]_{p1} \cdot x2[i]_{p2}$ を計算し、クライアントに返す。クライアントは、分散 $V=d/r13/N-E^2$ を計算する。

$$\begin{aligned} d-E^2 &= (\sum x2[i]_{p1} \cdot x2[i]_{p2})/r13/N-E^2 \\ &= (\sum (x[i]_{p1} \cdot x[i]_{p1} \cdot r1)(x[i]_{p2} \cdot x[i]_{p2} \cdot r2))/r13/N-E^2 \end{aligned}$$

$$\begin{aligned} &= (\sum (x[i]_{p1} \cdot x[i]_{p1})(x[i]_{p2} \cdot x[i]_{p2}))/N-E^2 \\ &= (\sum (x[i])^2)/N-E^2 \\ &= V \end{aligned}$$

(3) 平均からのずれが 3σ を超えているかを判定するには、 $x[i] > E+3\sigma$ か、 $x[i] < E-3\sigma$ を満たすことを調べる。いま、 $E+3\sigma=A$ 、 $E-3\sigma=B$ とおく。 $x[i]$ が A より大きいか、 B より小さいかを調べる。

A を $A1$ と $A2$ に秘密分散し、 B を $B1$ と $B2$ に秘密分散する。 $A1$ 、 $B1$ をクラウド1に、 $A2$ 、 $B2$ をクラウド2に配布する。クラウド1は $x[i]_{s1}-A1$ 、 $x[i]_{s2}-A2$ を計算し、クラウド2は $x[i]_{s1}-B1$ 、 $x[i]_{s2}-B2$ を計算してクライアントに返す。クライアントでは、 $(x[i]_{s1}-A1)+(x[i]_{s2}-A2)=x[i]-A > 0$ と $(x[i]_{s1}-B1)+(x[i]_{s2}-B2)=x[i]-B < 0$ とを計算し、どちらかが満たされるかどうかを判定する。

クラウドで判定まで行うことも考えられるが、サーバ間で情報交換を行うと、 $A-B$ 、つまり 6σ の値が漏れてしまうことが問題である。

5. 簡易的秘密計算法の利用インタフェース

5.1 基本コンセプト

本研究では、利用者が簡単に使える簡易的秘密計算法の利用インタフェースを開発した。図5に開発した簡易的秘密計算法の利用インタフェースのシステム構成図を示す。

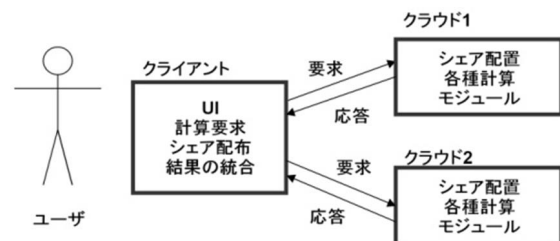


図5 利用インタフェースの構成図

本研究で開発した利用インタフェースのコンセプトは次のとおりである。

1) 利用者の負担を軽減

演算用のボタンを押すと結果が出てくるような画面にすることで、自分で秘密分散等の面倒な処理を自分で行う必要がない。

2) 扱いやすいインタフェース

Excel はパソコンを使っている人にとって身近なアプリケーションであり、Excel さえパソコンにインストールされていればプログラミングのための環境構築のインストールが不要だ。また、プログラミングを使ったことがない人でも早くに慣れることができる。

3) 拡張しやすい

クライアントからの要求に対する計算処理をモジュール化したため、機能を増やすときの拡張作業がしやすい。

4) 後処理がしやすい

このプログラムから違う計算処理、統計処理を加工するときの書き換えが簡単に行える。

以上の理由から、今回の実装には Excel の VBA を採用した。

5.2 インタフェースの設計

クライアントとサーバの通信手順は図 6 となる。

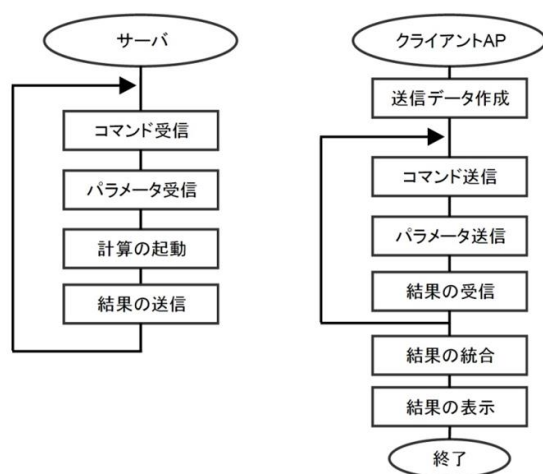


図 6 クライアントとサーバの通信手順

提案方式では、socket による TCP/IP を使用した Excel のライブラリを利用している[5]。TCP/IP はあらゆるコンピュータや異なる OS で相互に通信することが可能である。その性質を利用して、どのパソコンの Excel でも起動できるようにした。

クライアントは預けたいデータを秘密分散でサーバ数に分割する。各サーバへはコマンドを送信してデータの種類やサイズを各サーバに知らせてから要求を送る。そして返ってきた結果を統合して表示させる。一方サーバ側では要求のコマンドを受信したらそのコマンドに対する計算を起動させ、結果をクライアントに返す。

クライアントの画面設計は図 7 に示す。今回は実装結果が見えるよう、シート 1 にボタン配置と結果、シート 2 に送信データと受信データを書き込んだ。

クライアント側にはサーバ 1、サーバ 2 の IP アドレスとポート番号を書き、接続、切断というボタンでサーバとの接続を行う。サーバ側にも接続、切断ボタンを配置した。

また、サーバ側のボタンの配置を図 8 に示す。サーバ側では接続のボタンを押すとローカルポート番号と接続 IP 番号を読み込んでクライアントと接続する。また、サーバ

側のボタンや処理要求は最低限にしたかったため、接続、切断以外の命令はクライアントからの要求に対して自動で返せるようなプログラムを組んだ。

実装された計算としてシェアの配信、平均、分散値、相関係数がある。図 9 にクライアント側の配信用データの作成シートを示す。また、図 10 にサーバの受信データと計算シートを示す。

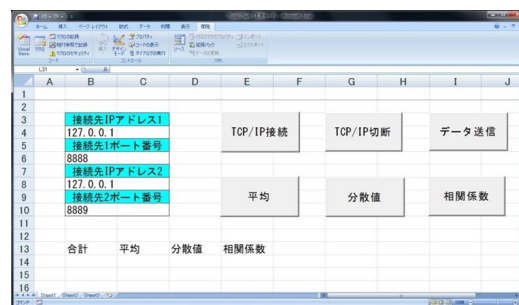


図 7 クライアント側ボタン配置の様子

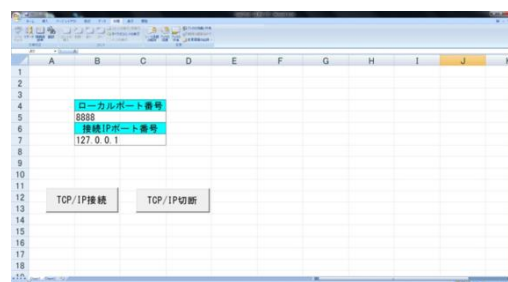


図 8 サーバ側ボタン配置の様子

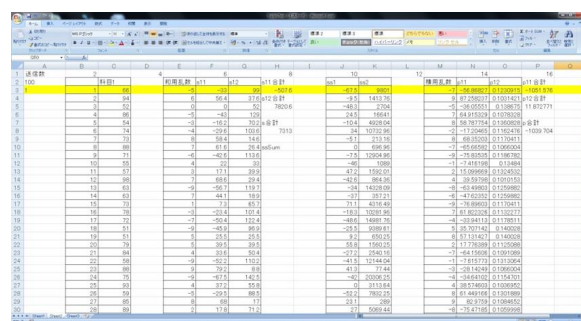


図 9 クライアント側計算シート

図 10 サーバ側計算シート

6. まとめと今後の課題

簡易的秘密計算法によって複雑な計算、および中間結果を持つような問題に適用する際の課題とサーバ間情報交換方式について述べた。今後は、これらの計算方法をクラウドサーバ上に実装していく際に、共通に使える計算方法をライブラリ化するなどしていく予定である。

参考文献

- 1) 千田浩司, 安全な情報処理を目指す秘密計算技術の研究動向と実用化に向けた取り組み, 情報処理 Vol. 54 No. 11 pp. 1130-1134 Nov. 2013
- 2) 宮西洋太郎他, クラウドサービス利用者の安心感を高める簡易的秘密計算法の提案, 信学技報, Vol. 114 No. 49 SWIM2014-4 pp19-24, 2014/5
- 3) 金岡晃他, 実数演算可能な軽量秘密計算法の一考察, CSS2014(Computer Security Symposium 2014), Oct. 2014 Sapporo pp682-687
- 4) 宮西洋太郎他, セキュアマルチパーティ秘密計算法におけるユーザ安心感定量化の試み, 情報処理学会研究報告, Vol2015-DPS-162 No. 41 2015/3
- 5) Excel 通信テスト(RS232C/TCPIP/UDPIP)
<http://homepage2.nifty.com/nonnon/Download/ExcelComTest/>