Vol.2021-CSEC-94 No.14
Vol.2021-SPT-43 No.14
2021/7/19

サーバ台数$n < 2k - 1$において実数演算可能な秘匿計算法の提案

納所勇之介[†1]　岩村惠市[†1]　稲村勝樹[†2]

概要：近年，電子機器の IoT 化が急速に普及し始め，それらから得られる各種データを，多面的かつ時系列で蓄積し，処理・解析するビッグデータの活用技術に注目が集まっている．その技術の一つとして秘匿計算の研究がある．秘匿計算では素数 p を法とした演算が行われるため，実数を扱う場合，指数と仮数部に分けて秘匿計算を行う場合が多い．それでも有限体上の演算と通常の実数演算では異なる部分も多く，特に有限体上で実数ベースの除算を行う場合，複雑な処理を行わなくてはならない．そこで本論文では，TUS 方式で用いられている秘密情報に乱数を掛けるというアプローチのもと，実数上で直接計算を行うことができる秘匿計算法の提案を行う．ただし，この研究は実数の秘匿計算に対して新しいアプローチを提案するものであり，その詳細な性能評価などは今後の課題である．

キーワード：秘密分散，秘匿計算，マルチパーティ計算，実数演算，$n < 2k - 1$

Proposal of a Secure Computation Method with Real Number Arithmetic for $n < 2k - 1$ Servers

YUNOSUKE NOSO[†1]　KEIICHI IWAMURA[†1]
MASAKI INAMURA[†2]

Abstract: In recent years, the IoT of electronic devices has begun to spread rapidly, and attention has been focused on technologies for utilizing big data that accumulate, process, and analyze various data obtained from these devices in a multifaceted and time-series manner. One of these technologies is the study of secure computation. In secure computation, operations are performed using a prime number p as a law, so when dealing with real numbers, secure computation is often divided into exponential and significand parts. Nevertheless, there are many differences between arithmetic operations on finite bodies and ordinary real number arithmetic, and, when performing real number-based division on finite bodies, complicated operations must be performed. In this paper, we propose a secure computation method that can be directly computed on real numbers, based on the approach of multiplying the secret information used in the TUS method by random numbers. However, since this research proposes a new approach to the secure computation of real number arithmetic, the detailed performance evaluation of the proposed method is a subject for future work.

Keywords: Secret Sharing Scheme, Secure Computation, Multiparty Computation, Real Number Arithmetic

1. はじめに

近年，IoT 化された電子機器から得られる各種データを，多面的かつ時系列で蓄積し，処理，解析するビッグデータの活用技術に注目が集まっている[1]．ビッグデータの活用では多種多様なデータを用い，その中には個人に紐付く情報を扱うこともある．そのことから，プライバシー問題へ対応するため，データを暗号化したままで処理，解析を行うことができる秘匿計算技術が注目されている．秘匿計算技術には準同型暗号や秘密分散法[2]を用いることが多いが，どちらも有限体上の演算であり，実数をそのまま扱うことができなかった．

そこで，実数を固定小数点表示や浮動小数点表示として秘匿計算を行う方式が研究されている．Catrina らは実数を固定小数点表示とし，仮数部と指数部に分け，仮数部を秘密情報として計算を行う方式を提案し[3][4]，Aliasgari らは実数を浮動小数点表示とし，符号ビット，ゼロフラグ，仮数部，指数部のそれぞれを秘密情報として扱う秘匿計算を行う方式を提案した[5]．さらに，天田らは Aliasgari らが提案した浮動小数点における秘匿計算内で生じるビット長を削減し，通信量を削減した方式を提案している[6]．いずれの固定小数点表示，浮動小数点表示における秘匿計算手法も，各情報を有限体上の値と対応づけており，加減乗算は有限体上の演算を拡張するだけで実現できる．しかし，除算は有限体上の演算と実数上の演算ではその答えが異なるため，秘匿除算においてはゴールドシュミット法と呼ばれる，除算を乗算で近似する複雑な処理を行わなければならなかった．さらに，近年，機械学習を秘匿計算で行う SecureML[7]や ABY3[8]といった方式が提案されているが，除算に関してはいずれも公開値による除算を行うため，純粋な秘匿除算とは言い難い．

また，我々が調査した限り，秘匿計算を有限体上ではな

†1 東京理科大学
Tokyo University of Science
†2 広島市立大学
Hiroshima City University

Vol.2021-CSEC-94 No.14
Vol.2021-SPT-43 No.14
2021/7/19

く実数上で行っているのは金岡らが提案した軽量秘密分散法[9]を基本とするものだけである.しかし, この軽量秘密分散法は用いる乱数の範囲が具体的には指定されておらず, 乱数の設定範囲では桁落ちや情報落ちが発生する可能性があるとされている．さらに，乗除算と加減算が混在する場合，乗除算，または加減算の結果を一旦クライアントに戻した後，加減算，または乗除算を行う必要がある．この問題を解決するため，高橋らは途中の演算結果にサーバで生成した乱数を掛ける方式[10]を提案しているが，サーバへの委託計算の形式を取っているため，秘密情報の入力者が複数人いる場合を想定していない.

一方，秘密分散法を用いた秘匿演算では一般に$n \geq 2k-1$という制限を持つが，$n < 2k-1$においても秘匿演算が可能な TUS 方式が研究されている．TUS 方式の特徴は秘密情報に乱数をかけて秘匿する点にある．2 つの実数をかける場合，定められた小数桁で四捨五入を行うが，四捨五入された値を 2 つの実数に分解しようとする場合，多くの候補が考えられ，特定の値を定められない．よって，本論文では使用する乱数の範囲を指定し，また，上記 TUS 方式の特徴を生かすことで, 桁落ちや情報落ちを防ぐことができ, 実数上で秘匿加算と秘匿乗除算を組み合わせて行うことができる秘匿計算法の提案を行う.

本論文の構成は, 2 章において関連研究について説明し, 3 章において提案方式とその安全性を示し，4 章では提案方式の誤差の評価を行う.

2. 関連研究

2.1 金岡らの軽量秘密分散法

金岡らは秘密分散を有限体上ではなく，実数上で行うことで，実数上での乗算と除算を可能にする秘密計算手法を提案した[9]．この手法では以下のように乗除算用，加減算用のシェア(分散値)を作成することで，実数上での秘密計算を可能にしている．ただし, 使用する乱数の範囲が指定されていないため，乱数の値によっては秘密情報が漏洩する可能性が高まる恐れがある.

2.1.1 乗除算用のシェアの作成

実数上のもとのデータをx，分散させるサーバ台数をnとした場合以下となるような乱数$x_i(i=0,\dots n-2)$を選択する．なお，x_{n-1}は$x/\prod_{i=0}^{n-2} x_i$で求める.

$$x_0 \cdot x_1 \cdot \dots \cdot x_{n-1} = x$$

[n out of n の場合]

すべてのシェアが集まることで秘密計算が可能となる n out of n を実現するためには各サーバS_iにx_iを配布する.

[2 out of 3 の場合]

2 つのパーティが集まることで秘密計算が可能になる 2 out of 3 を実現するためには，各サーバS_iに$x_i, x_{(i+1)\ mod\ 3}$を配布する.

2.1.2 乗算(除算)

元のデータをx，yとし，n out of n，2 out of 3 の場合において, 復元者が$z = x \cdot y$を求める方法は以下の通りである. なお, 除算の場合は・を/に変えることで, 乗算と同様の方法で求めることができる.

[n out of n の場合]

サーバS_iは, 復元者に$z_i = x_i \cdot y_i$を送る. 復元者は集まったz_iより$z = \prod_{i=0}^{n-1} z_i$を求める.

[2 out of 3 の場合]

$j = (i+1)\ mod\ 3$としたとき，サーバS_iは復元者に，$c_i = x_i \cdot y_i$，$c_j = x_j \cdot y_j$と，どのシェアを持っているかの情報である(i,j)を送る．復元者はどのシェアを持っているかの情報と集まったc_i, c_jから$z = \prod_{i=0}^{2} c_i$を求める.

2.1.3 加算(減算)のシェアの作成

実数上のもとのデータをx，分散させるサーバ台数をnとした場合以下となるような乱数$x_i(i=0,\dots n-2)$を選択する．なお，x_{n-1}は$x - \sum_{i=0}^{n-2} x_i$で求める.

$$x_0 + x_1 + \dots + x_{n-1} = x$$

[n out of n の場合]

すべてのシェアが集まることで秘密計算が可能となる n out of n を実現するためには各サーバS_iにx_iを配布する.

[2 out of 3 の場合]

2 つのパーティが集まることで秘密計算が可能になる 2 out of 3 を実現するためには，各サーバS_iに$x_i, x_{(i+1)\ mod\ 3}$を配布する.

2.1.4 加算(減算)

元のデータをx，yとし，n out of n，2 out of 3 の場合において，復元者が$z = x \pm y$を求める方法は以下の通りである.

[n out of n の場合]

サーバS_iは, 復元者に$z_i = x_i \pm y_i$を送る. 復元者は集まったz_iより$z = \sum_{i=0}^{n-1} z_i$を求める.

[2 out of 3 の場合]

$j = (i+1)\ mod\ 3$としたとき，サーバS_iは復元者に，$c_i = x_i \pm y_i$，$c_j = x_j \pm y_j$と，どのシェアを持っているかの情報である(i,j)を送る．復元者はどのシェアを持っているかの情報と集まったc_i, c_jから$z = \sum_{i=0}^{2} c_i$を求める.

2.2 高橋らの方式

高橋らは金岡らの課題を解決する手法を提案した[10]. 2.1 節で示した方法でa, b, c, dの 4 つのデータを 2 つのサーバS_0, S_1に乗算用のシェアとして秘密分散するとき，それぞれの積のシェアをa_0, b_0, c_0, d_0，a_1, b_1, c_1, d_1とする．このとき, $y = ab + cd$となるようなyの計算を行う場合, 以下の手順で計算を行う.

① サーバS_0は$x_0 = a_0 b_0, y_0 = c_0 d_0$，サーバ$S_1$は$x_1 = a_1 b_1, y_1 = c_1 d_1$を計算する.

② サーバS_0は乱数r_0，サーバS_1は乱数r_1をそれぞれ生成し，クライアントに送る.

③ サーバS_0はx_0, y_0に乱数r_0を掛けた$r_0 x_0, r_0 y_0$をサーバ

Vol.2021-CSEC-94 No.14
Vol.2021-SPT-43 No.14
2021/7/19

S_1に，サーバS_1はx_1, y_1に乱数r_1を掛けたr_1x_1, r_1y_1をサーバS_0に送る．

④ 各サーバは$T = r_0x_0 \times r_1x_1 + r_0y_0 \times r_1y_1 = (r_0r_1)(x_0x_1 + y_0y_1) = (r_0r_1)(a_0b_0a_1b_1 + c_0d_0c_1d_1) = (r_0r_1)(ab + cd)$を計算し，クライアントに送る．

⑤ クライアントは$T \times \frac{1}{r_0r_1} = ab + cd$で$y$を求めることができる．

2.3 TUS 方式

まず，神宮らによって TUS1 方式と呼ばれる，秘密情報に乱数を掛けて秘匿化した秘匿化秘密情報を Shamir 法によって分散することによって，サーバ台数$n < 2k - 1$の場合でも秘匿乗算可能な方式が提案された[11]. しかし，この方式は秘匿積和演算といった異なる演算の組み合わせに対して安全でない方式であった．この問題を解決するため，ムハンマドカマルらによって TUS2 方式と呼ばれる，攻撃者が知らない乱数を用いた 1 に対する分散値集合を導入することで，異なる演算の組み合わせに対しても安全な秘匿演算方式が提案された[12]. ただし, TUS2 方式は Shamir 法の分散，復元を複数回行うことから従来の秘匿計算手法より計算量が多く，効率的でないという問題があった．そこで，鴇田らによって TUS3 と呼ばれる， XOR 法を導入することで，効率的に秘匿計算を行う手法が提案された[13]. そして，岩村らによって TUS4 方式と呼ばれる，秘匿計算に必要となる乱数の計算を事前処理に集中させることで，秘匿計算から通信の発生を排除し，通信回数が乗算回数に依存しない秘匿計算方式が提案された[14]. この方式は Semi-honest な攻撃者に対して「攻撃者が知らない乱数とそれを構成する乱数の分散値を各サーバが持つ」という条件のもと，非常に高速に秘匿演算を実行することができる．

3. 提案方式

本提案方式は，実数上で秘密分散を行う手法である．したがって，有限体上ではなく，実数上の多項式を用いて秘密情報の分散を行うものとして定義する．具体的には以下のようになる．

【分散処理】

① ディーラは秘密情報sとして任意の整数を選ぶ.

② ディーラは異なるn個の整数$x_i (i = 1,2, \dots, n)$を選び，サーバIDとする．

③ ディーラは$k-1$個の整数である乱数$a_j (j = 1,2, \dots, k-1)$を選び, 以下の式を生成し, 分散値$f(x)$を計算する．なお，以下の計算は有限体上ではなく，実数上で計算を行う．

$$f(x) = s + a_1x_i + a_2x_i^2 + \cdots + a_{k-1}x_i^{k-1}$$

④ ディーラは上記式のxに各サーバIDを代入して分散値$f_i = f(x_i)$を計算し，各サーバS_iにf_iを配布する．

【復元処理】

① 復元者は任意のk第のサーバからk個の分散値f_iを収集する．

② 復元者はk個の分散値から連立方程式を立て，多項式$f(x)$を復元し，$f(0) = s$より秘密情報sを得る．

また，秘密情報を秘匿するための基本的なアイディアを説明する．TUS 方式は秘密情報に乱数をかけて秘匿するが, 小数の桁数は定まっているため, 乗算結果を分解する場合, 多くの候補が存在する．例えば，小数点第 1 位までの実数を乗算して，その乗算結果を小数点第 1 位で四捨五入した値が21.0である場合，正確な乗算値の候補は20.95～21.04となる．また，乗算値が21.0である場合，乗算した実数の候補は0.1～210.0まで選択可能となる. したがって, 考慮する小数点以下の桁数を増加させると，その候補は膨大な数とできるので，秘密情報の特定が非常に困難となる．

そこで，TUS 方式で用いられている秘密情報に乱数を掛けるというアプローチのもと，実数上における秘匿計算手法の提案を行う．なお，提案方式では先述の定義に加え，以下の前提条件を置く．

- 全ての演算は有限体上ではなく，実数上で行われる．
- $[a]_i$はサーバS_iが保有する分散値とする．
- 事前にサーバS_jは変換用乱数組（$[\varepsilon_h]_j, \varepsilon_{h,j}$）を持つ．ここで，$\varepsilon_h = \prod_{j=0}^{k-1} \varepsilon_{h,j}$ (ε_h， $\varepsilon_{h,j}$は整数)であり，特に，下記に示す積和演算，除和演算では$h = 1,2$である．
- 秘密情報や乱数に特に記述がない場合，整数部分の桁数に上限はなく，小数部分の下限の桁数をt桁とする．
- 秘密情報に負の値を含む場合，それを秘匿する乱数も負の値を含むが乱数，秘密情報ともに 0 は含まない．
- 乗算後は必ず四捨五入を行う．ただし，四捨五入を行うことができない場合，乱数を変える．

3.1 秘匿積和演算

以下では$n = k = 2$の場合について述べる．

【秘密情報の秘匿】

1. ユーザ A は小数点第t位の乱数α_0, α_1を生成し，$\alpha = \alpha_0 \times \alpha_1$を計算し，小数点第$t$位未満を四捨五入する．
2. ユーザ A はα_0をサーバS_0に，α_1をサーバS_1に送り，秘密情報aに対してαaを計算し, 小数点第t位未満を四捨五入した後，全サーバに送る．
3. 秘密情報b, cをもつユーザ B, C も同様の処理を行う．

【秘匿計算】

1. サーバS_jは乱数$\delta_{1,j}, \dots, \delta_{q,j}$を生成し，$\frac{\delta_{1,j},\dots,\delta_{q,j}}{\alpha_j\beta_j\varepsilon_{1,j}}$，$\frac{\delta_{1,j},\dots,\delta_{q,j}}{\gamma_j\varepsilon_{2,j}}$を計算し，小数点第$t$位未満を四捨五入した後，1 台のサーバ（ここでは，サーバS_0）に送る．ここで，qは分母の乱数の個数の最大値+1個とする．したがって，積和演算では，$q = 4$となる．また，$\delta_j = \delta_{1,j}, \dots, \delta_{q,j}$の小数点第$t$位未満を四捨五入した後，復元者に送る．
2. サーバS_0は以下を計算して全サーバに送る．

Vol.2021-CSEC-94 No.14
Vol.2021-SPT-43 No.14
2021/7/19

$$\frac{\delta}{\alpha\beta\varepsilon_1}=\frac{\delta_{1,0}\delta_{2,0}\delta_{3,0}\delta_{4,0}}{\alpha_0\beta_0\varepsilon_{1,0}}\times\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\alpha_1\beta_1\varepsilon_{1,1}}$$

$$\frac{\delta}{\gamma\varepsilon_2}=\frac{\delta_{1,0}\delta_{2,0}\delta_{3,0}\delta_{4,0}}{\gamma_0\varepsilon_{2,0}}\times\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\gamma_1\varepsilon_{2,1}}$$

3. 各サーバS_jは以下を計算し，復元者に送る．

$$[\delta(ab+c)]_j=\frac{\delta}{\alpha\beta\varepsilon_1}\times\alpha a\times\beta b\times[\varepsilon_1]_j+\frac{\delta}{\gamma\varepsilon_2}\times\gamma c\times[\varepsilon_2]_j$$

【復元処理】

1. 復元者は，k台のサーバS_jから受け取った$[\delta(ab+c)]_j$，δ_jを集めて，$\delta(ab+c)$，δを復元し，以下を計算し，$ab+c$を得る．

$$ab+c=\frac{\delta(ab+c)}{\delta}$$

3.2 秘匿積和演算の安全性

秘匿積和演算は 3 入力 1 出力演算であることから，どんなに安全な秘匿演算手法を用いても 2 入力 1 出力が分かれば残りの 1 入力が漏洩し，また，3 入力が分かれば 1 出力が漏洩するため，以下に定義する以上の攻撃者を想定する必要はない．よって，以下に定義する攻撃者 1，2 に対して安全であれば，提案方式は安全な秘匿積和演算を実現すると言える．ただし，本提案方式は Semi-honest な攻撃者を想定している．

攻撃者 1:

秘匿積和演算に関する 1 入力 1 出力を知るものが攻撃者となり，自らが入力した秘密情報，秘密情報の秘匿に用いた乱数，および復元時に得られる情報を知る．さらに，$k-1$台のサーバが知る情報も知ることができ，それらをもとに残りの 2 入力を個々に知ろうとする．

攻撃者 2:

秘匿積和演算に関する 2 入力知るものが攻撃者となり，自らが入力した秘密情報と秘密情報の秘匿に用いた乱数を知る．さらに，$k-1$台のサーバが知る情報も知ることができ，それらをもとに残りの 2 入力を個々に知ろうとする．

【秘匿積和演算の安全性】

<攻撃者 1 に対する安全性>

ユーザ B，復元時に得られる情報，およびサーバS_0の情報を知るものが攻撃者となった場合，攻撃者 1 は以下の情報を得る．ただし，$(*)$がついたものは四捨五入された情報である．

$$B=\{b,\beta_0,\beta_1,\alpha_0,\alpha a(*),\gamma_0,\gamma c(*),\delta_{1,0},\delta_{2,0},\delta_{3,0},\delta_{4,0},$$
$$\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\alpha_1\beta_1\varepsilon_{1,1}}(*),\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\gamma_1\varepsilon_{2,1}}(*),[\delta(ab+c)]_0,$$
$$[\delta(ab+c)]_1,\delta(ab+c),ab+c,\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}(*)\}$$

$\alpha a(*)$は小数点第t位未満を四捨五入していることから，αaの候補は10^t通り存在する．また，α_0から$\alpha_1 a$が求まるが，この値も四捨五入した値であるので，その候補は10^t通り存在する．したがって，aの候補は$10^t\times10^t=10^{2t}$通り存在することになる．ここで，有限体上での秘密分散における法pを 128 ビット程度の素数としたとき，秘密情報として考えられる値の候補と同程度の候補をもつには$2^{128}-1\approx10^{39}<10^{2t}$となる．よって，提案方式における乱数の下限の小数点以下の桁数を$t=20$桁とすればよい．同様の議論は$\gamma c(*)$にも言える．

また，乱数α_1,γ_1を知ろうとするために，$\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\alpha_1\beta_1\varepsilon_{1,1}}(*),\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\gamma_1\varepsilon_{2,1}}(*),\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}(*)$から$\frac{1}{\alpha_1\varepsilon_{1,1}},\frac{1}{\gamma_1\varepsilon_{2,1}}$を得ようとするが，$\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}(*)$は小数点以下第$t=20$位未満で四捨五入されることを考慮すると，$\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}$の候補は$10^{60}$通り存在することになり，これは$10^{60}>10^{39}\approx2^{128}-1$となるため，乱数$\alpha_1,\gamma_1$を推定することは非常に困難となる．

以上の議論は攻撃者 1 がユーザ B ではなく，ユーザ A，C となった場合でも同様であるため，提案した積和演算方式は攻撃者 1 に対して秘密情報の特定は非常に困難であると言える．

<攻撃者 2 に対する安全性>

ユーザ B，C，およびサーバS_0の情報を知るものが攻撃者となった場合，攻撃者 2 は以下の情報を得る．ただし，$(*)$がついたものは四捨五入された情報である．

$$BC=\{b,\beta_0,\beta_1,c,\gamma_0,\gamma_1,\alpha a(*),\delta_{1,0},\delta_{2,0},\delta_{3,0},\delta_{4,0},$$
$$\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\alpha_1\beta_1\varepsilon_{1,1}}(*),\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\gamma_1\varepsilon_{2,1}}(*),[\delta(ab+c)]_0\}$$

$\alpha a(*)$の安全性に関しては攻撃者 1 に対する安全性と同様であるため，省略する．また，$\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\alpha_1\beta_1\varepsilon_{1,1}}(*),\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\gamma_1\varepsilon_{2,1}}(*)$から乱数$\alpha_1$の情報を得ようとするが，$\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\alpha_1\beta_1\varepsilon_{1,1}}(*),\frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}}{\gamma_1\varepsilon_{2,1}}(*)$はともに小数点以下第$t=20$位未満で四捨五入されることを考慮すると，$\delta_{1,1}\delta_{2,1}\delta_{3,1}\delta_{4,1}$の候補は$10^{60}$通り存在することになり，これは$10^{60}>10^{39}\approx2^{128}-1$となるため，各乱数$\delta_{1,1},\delta_{2,1},\delta_{3,1},\delta_{4,1}$を推定することは非常に困難となる．

また，攻撃者 2 は$\delta_{1,0},\delta_{2,0},\delta_{3,0},\delta_{4,0},[\delta(ab+c)]_0$のみしかわからず，$[\delta(ab+c)]_1$を知らないため，$\delta(ab+c)$を求めることができず，$ab+c$を求めることは困難である．

以上の議論は攻撃者 2 がユーザ B，C ではなく，ユーザ A，B，またはユーザ A，C となった場合でも同様であるため，提案した積和演算方式は攻撃者 2 に対して秘密情報の特定は非常に困難であると言える．したがって，提案した積和演算方式は攻撃者 1，2 に対して秘密情報の特定は非常に困難であることから，Semi-honest な攻撃者に対して安全となる．

3.3 秘匿除和演算

【秘密情報の秘匿】

1. ユーザ A は小数点第t位の乱数α_0,α_1を生成し，$\alpha=$

$\alpha_0 \times \alpha_1$を計算し，小数点第t位未満を四捨五入する．

2. ユーザ A はα_0をサーバS_0に，α_1をサーバS_1に送り，αaを計算し，小数点第t位未満を四捨五入した後，全サーバに送る．
3. ユーザ B，C も同様の処理を行う．

【秘匿計算】

1. サーバS_jは乱数$\delta_{1,j}, \dots, \delta_{q,j}$を生成し，$\frac{\delta_{1,j},\dots,\delta_{q,j}}{\alpha_j \beta_j \varepsilon_{1,j}}$，$\frac{\delta_{1,j},\dots,\delta_{q,j}}{\gamma_j \varepsilon_{2,j}}$を計算し，小数点第$t$位未満を四捨五入した後，1 台のサーバ（ここでは，サーバS_0）に送る．ここで，qは分母の乱数の個数の最大値+1個とする．したがって，除和演算では，$q = 3$となる．また，$\delta_j = \delta_{1,j}, \dots, \delta_{q,j}$の小数点第$t$位未満を四捨五入した後，復元者に送る．
2. サーバS_0は以下を計算して全サーバに送る．

$$\frac{\beta\delta}{\alpha\varepsilon_1} = \frac{\beta_0\delta_{1,0}\delta_{2,0}\delta_{3,0}}{\alpha_0\varepsilon_{1,0}} \times \frac{\beta_1\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\alpha_1\varepsilon_{1,1}}$$

$$\frac{\delta}{\gamma\varepsilon_2} = \frac{\delta_{1,0}\delta_{2,0}\delta_{3,0}}{\gamma_0\varepsilon_{2,0}} \times \frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\gamma_1\varepsilon_{2,1}}$$

3. 各サーバS_jは以下を計算し，復元者に送る．

$$[\delta(a/b+c)]_j = \frac{\beta\delta}{\alpha\varepsilon_1} \times \alpha a/\beta b \times [\varepsilon_1]_j + \frac{\delta}{\gamma\varepsilon_2} \times \gamma c \times [\varepsilon_2]_j$$

【復元処理】

1. 復元者は，k台のサーバS_jから受け取った$[\delta(a/b+c)]_j$，δ_jを集めて，$\delta(a/b+c)$，δを復元し，以下を計算し，$a/b+c$を得る．

$$a/b+c = \frac{\delta(a/b+c)}{\delta}$$

3.4 秘匿除和演算の安全性

秘匿積和演算と同様の攻撃者 1，2 に対して秘匿除和演算の安全性を示す．

【秘匿除和演算の安全性】

<攻撃者 1 に対する安全性>

ユーザ B，復元時に得られる情報，およびサーバS_0の情報を知るものが攻撃者となった場合，攻撃者 1 は以下の情報を得る．ただし，$(*)$がついたものは四捨五入された情報である．

$$B = \{b, \beta_0, \beta_1, \alpha_0, \alpha a(*), \gamma_0, \gamma c(*), \delta_{1,0}, \delta_{2,0}, \delta_{3,0},$$
$$\frac{\beta_1\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\alpha_1\varepsilon_{1,1}}(*), \frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\gamma_1\varepsilon_{2,1}}(*), [\delta(a/b+c)]_0,$$
$$[\delta(a/b+c)]_1, \delta(a/b+c), a/b+c, \delta_{1,1}\delta_{2,1}\delta_{3,1}(*)\}$$

$\alpha a(*), \gamma c(*)$の安全性に関しては，3.2 節で示した秘匿積和演算における$\alpha a(*), \gamma c(*)$の安全性と同様であるため，省略する．また，乱数α_1, γ_1を知ろうとするために，$\frac{\beta_1\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\alpha_1\varepsilon_{1,1}}(*), \frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\gamma_1\varepsilon_{2,1}}(*), \delta_{1,1}\delta_{2,1}\delta_{3,1}(*)$から$\frac{1}{\alpha_1\varepsilon_{1,1}}, \frac{1}{\gamma_1\varepsilon_{2,1}}$を得ようとするが，$\delta_{1,1}\delta_{2,1}\delta_{3,1}(*)$は小数点以下第$t = 20$位未満で四捨五入されることを考慮すると，$\delta_{1,1}\delta_{2,1}\delta_{3,1}$の候補は$10^{40}$通り存在することになり，これは$10^{40} > 10^{39} \approx 2^{128} - 1$となるため，乱数$\alpha_1, \gamma_1$を推定することは非常に困難となる．

以上の議論は攻撃者 1 がユーザ B ではなく，ユーザ A，C となった場合でも同様であるため，提案した積和演算方式は攻撃者 1 に対して秘密情報の特定は非常に困難であると言える．

<攻撃者 2 に対する安全性>

ユーザ B，C，およびサーバS_0の情報を知るものが攻撃者となった場合，攻撃者 2 は以下の情報を得る．ただし，$(*)$がついたものは四捨五入された情報である．

$$BC = \{b, \beta_0, \beta_1, c, \gamma_0, \gamma_1, \alpha a(*), \delta_{1,0}, \delta_{2,0}, \delta_{3,0},$$
$$\frac{\beta_1\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\alpha_1\varepsilon_{1,1}}(*), \frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\gamma_1\varepsilon_{2,1}}(*), [\delta(a/b+c)]_0\}$$

$\alpha a(*)$の安全性に関しては，3.2 節で示した秘匿積和演算における$\alpha a(*)$の安全性と同様であるため，省略する．また，$\frac{\beta_1\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\alpha_1\varepsilon_{1,1}}(*), \frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\gamma_1\varepsilon_{2,1}}(*)$から乱数$\alpha_1$の情報を得ようとするが，$\frac{\beta_1\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\alpha_1\varepsilon_{1,1}}(*), \frac{\delta_{1,1}\delta_{2,1}\delta_{3,1}}{\gamma_1\varepsilon_{2,1}}(*)$はともに小数点以下第$t = 20$位未満で四捨五入されることを考慮すると，$\delta_{1,1}\delta_{2,1}\delta_{3,1}$の候補は$10^{40}$通り存在することになり，これは$10^{40} > 10^{39} \approx 2^{128} - 1$となるため，各乱数$\delta_{1,1}, \delta_{2,1}, \delta_{3,1}$を推定することは非常に困難となる．

また，攻撃者 2 は$\delta_{1,0}, \delta_{2,0}, \delta_{3,0}, [\delta(a/b+c)]_0$のみしかわからず，$[\delta(a/b+c)]_1$を知らないため，$\delta(a/b+c)$を求めることができず，$a/b+c$を求めることは困難である．

以上の議論は攻撃者 2 がユーザ B，C ではなく，ユーザ A，B，またはユーザ A，C となった場合でも同様であるため，提案した積和演算方式は攻撃者 2 に対して秘密情報の特定は非常に困難であると言える．したがって，提案した除和演算方式は攻撃者 1，2 に対して秘密情報の特定は非常に困難であることから，Semi-honest な攻撃者に対して安全となる．

4. 誤差の評価

4.1 秘密情報の秘匿における誤差と安全性

秘密情報の秘匿において用いる乱数α_0, α_1，秘密情報aを以下のように設定し，誤差と安全性を議論する．以下では$s = 10$とし，簡単のため，整数部及び小数部共に 3 桁の範囲を想定する．変数$a_{l,m}$は乱数α_lのs^mの係数を表す．また，変数a_lと記述した場合は，対応する乱数a_lのs^mの係数を表す．なお，以下の議論は$\beta b, \gamma c$に関しても同様の議論となる．

$$\alpha_0 = a_{0,3}s^3 + a_{0,2}s^2 + a_{0,1}s + a_{0,0} + a_{0,-1}s^{-1} + a_{0,-2}s^{-2} + a_{0,-3}s^{-3}$$
$$\alpha_1 = a_{1,3}s^3 + a_{1,2}s^2 + a_{1,1}s + a_{1,0} + a_{1,-1}s^{-1} + a_{1,-2}s^{-2} + a_{1,-3}s^{-3}$$
$$a = a_3s^3 + a_2s^2 + a_1s + a_0 + a_{-1}s^{-1} + a_{-2}s^{-2} + a_{-3}s^{-3}$$

上記設定からαaを求める．ここで，直接αaを計算した後四

Vol.2021-CSEC-94 No.14
Vol.2021-SPT-43 No.14
2021/7/19

捨五入する場合(以降方式①と呼ぶ)と，αを計算して四捨五入した後，αaを計算して再び四捨五入した場合(以降方式②と呼ぶ)，αaは係数A_l, A'_lを用いてそれぞれ以下のようになる．ただし，{}は四捨五入部を表す．

【αaを計算した後四捨五入する場合(方式①)】

$$\begin{aligned}\alpha a = {} & A_9 s^9 + A_8 s^8 + A_7 s^7 + A_6 s^6 + A_5 s^5 + A_4 s^4 + A_3 s^3 \\ & + A_2 s^2 + A_1 s + A_0 + A_{-1} s^{-1} + A_{-2} s^{-2} \\ & + A_{-3} s^{-3}\{+A_{-4} s^{-4} + A_{-5} s^{-5} + A_{-6} s^{-6} \\ & + A_{-7} s^{-7} + A_{-8} s^{-8} + A_{-9} s^{-9}\}\end{aligned}$$

【αを計算，四捨五入した後，αaを計算，四捨五入する場合(方式②)】

$$\begin{aligned}\alpha a = {} & A'_9 s^9 + A'_8 s^8 + A'_7 s^7 + A'_6 s^6 + A'_5 s^5 + A'_4 s^4 + A'_3 s^3 \\ & + A'_2 s^2 + A'_1 s + A'_0 + A'_{-1} s^{-1} + A'_{-2} s^{-2} \\ & + A'_{-3} s^{-3}\{+A'_{-4} s^{-4} + A'_{-5} s^{-5} + A'_{-6} s^{-6}\}\end{aligned}$$

ここで，各係数の差は以下のようになる．

$$A_9 - A'_9 = A_8 - A'_8 = \cdots = A_0 - A'_0 = 0$$

$$A_{-1} - A'_{-1} = \left(a_{0,-1}a_{1,-3} + a_{0,-2}a_{1,-2} + a_{0,-3}a_{1,-1}\right)a_3$$

$$\begin{aligned}A_{-2} - A'_{-2} = {} & \left(a_{0,-1}a_{1,-3} + a_{0,-2}a_{1,-2} + a_{0,-3}a_{1,-1}\right)a_2 \\ & + \left(a_{0,-2}a_{1,-3} + a_{0,-3}a_{1,-2}\right)a_3\end{aligned}$$

$$\begin{aligned}A_{-3} - A'_{-3} = {} & \left(a_{0,-1}a_{1,-3} + a_{0,-2}a_{1,-2} + a_{0,-3}a_{1,-1}\right)a_1 \\ & + \left(a_{0,-2}a_{1,-3} + a_{0,-3}a_{1,-2}\right)a_2 \\ & + a_{0,-3}a_{1,-3}a_3\end{aligned}$$

$$\begin{aligned}A_{-4} - A'_{-4} = {} & \left(a_{0,-1}a_{1,-3} + a_{0,-2}a_{1,-2} + a_{0,-3}a_{1,-1}\right)a_0 \\ & + \left(a_{0,-2}a_{1,-3} + a_{0,-3}a_{1,-2}\right)a_1 \\ & + a_{0,-3}a_{1,-3}a_2\end{aligned}$$

$$\begin{aligned}A_{-5} - A'_{-5} = {} & \left(a_{0,-1}a_{1,-3} + a_{0,-2}a_{1,-2} + a_{0,-3}a_{1,-1}\right)a_{-1} \\ & + \left(a_{0,-2}a_{1,-3} + a_{0,-3}a_{1,-2}\right)a_0 \\ & + a_{0,-3}a_{1,-3}a_1\end{aligned}$$

$$\begin{aligned}A_{-6} - A'_{-6} = {} & \left(a_{0,-1}a_{1,-3} + a_{0,-2}a_{1,-2} + a_{0,-3}a_{1,-1}\right)a_{-2} \\ & + \left(a_{0,-2}a_{1,-3} + a_{0,-3}a_{1,-2}\right)a_{-1} \\ & + a_{0,-3}a_{1,-3}a_0\end{aligned}$$

したがって，方式①のほうが四捨五入の回数が少なく，考慮できる係数の数が増加するため，方式②より精度の高い演算を行えることが分かる．ただし，方式①の場合，攻撃者 1, 2 において乱数α_0(またはα_1)を知ると，αaしか四捨五入を行っていないため，αaの四捨五入前の値と乱数α_0(またはα_1)から割り切ることのできるαaを推定できるため，安全性が低下し，秘密情報が漏洩してしまう可能性がある．これを具体値で説明する．簡単のため，用いる乱数・秘密情報の小数の下限を$t = 1$桁として，$\alpha_0 = 9.3, \alpha_1 = 21.5, a = 15.7$とする．ここで方式①では，$\alpha a = 9.3 \times 21.5 \times 15.7 = 3139.215 \approx 3139.2$となる．したがって，攻撃者が乱数$\alpha_0$を知ったとすると，$\alpha a \div \alpha_0 = 337.5483\ldots$となり，$\alpha$が四捨五入された値でないため，$\alpha_1 a$の候補は337.55となる．これを素因数分解し，乱数・秘密情報の小数の下限が$t = 1$桁であることを考慮すると$337.55 = 0.1 \times 3375.5, 0.5 \times 675.1, 4.3 \times 78.5, 15.7 \times 21.5$となり，$a$の候補を8個に絞り込むことができる．一方，方式②では，$\alpha = 9.3 \times 21.5 = 199.95 \approx 200.0, \alpha a = 200.0 \times 15.7 = 3140.00 \approx 3140.0$となる．ここで，$\alpha a$の候補が3139.95～3140.04の 10 個あることに加えて，仮に攻撃者が乱数α_0を知っていることから，$\alpha a \div \alpha_0 = 3140.00 \div 9.3 = 337.6\ldots$を求めたとしても，この値は四捨五入された正確でない値であるため，α_1, aが共に正確でなく，その候補は337.55～337.64の 10 通り存在することになり，秘密情報aの候補は10^2通りとなり，方式①に比べて秘密情報の候補を絞り込むことは非常に困難である．

以上より，方式②より方式①の方が精度の高い演算を行うことができるが，秘密情報が漏洩してしまう可能性があるため，安全な方式だとは言えない．したがって，安全性を確保するため，方式②が適切な方式だと言える．

4.2 秘匿計算の分子の乱数の増加における誤差と安全性

秘匿計算の分子の乱数を増加させたとき，演算結果の誤差と安全性がどのように変化するか議論する．以下では簡単のため，秘匿計算の分子の乱数を 1 個にした場合と 2 個にした場合で議論する．ただし，4.1 節と各乱数における係数の定義は同様とし，加えて，変数$d_{l,0,m}$は乱数 $d_{l,0}$のs^mの係数を表すものと定義する．

【秘匿計算の分子の乱数が 1 個の場合】

秘密情報Aを乱数α_0で秘匿した後，乱数$\frac{\delta_{1,0}}{\alpha_0}$を掛け，最後に乱数$\delta_{1,0}$で割り秘密情報$A$を復元する場合を考える．なお，秘密情報と各乱数は以下のように設定する．

$$\begin{aligned}\mathrm{A} &= a_3 s^3 + a_2 s^2 + a_1 s + a_0 + a_{-1} s^{-1} + a_{-2} s^{-2} + a_{-3} s^{-3} \\ \alpha_0 &= a_{0,3} s^3 + a_{0,2} s^2 + a_{0,1} s + a_{0,0} + a_{0,-1} s^{-1} + a_{0,-2} s^{-2} \\ &\quad + a_{0,-3} s^{-3} \\ \delta_{1,0} &= d_{1,0,3} s^3 + d_{1,0,2} s^2 + d_{1,0,1} s + d_{1,0,0} + d_{1,0,-1} s^{-1} \\ &\quad + d_{1,0,-2} s^{-2} + d_{1,0,-3} s^{-3}\end{aligned}$$

したがって，上記設定から，$\frac{\delta_{1,0}}{\alpha_0}$は，以下のようになる．

$$\frac{\delta_{1,0}}{\alpha_0} = q_0 + q_{-1} s^{-1} + q_{-2} s^{-2} + q_{-3} s^{-3}\{+q_{-4} s^{-4} + \cdots\}$$

ただし，q_0からq_{-3}は以下のように算出できる．

$$q_0 = \frac{d_{1,0,3}}{a_{0,3}},$$

$$q_{-1} = \frac{d_{1,0,2} - q_0 a_{0,2}}{a_{0,3}},$$

$$q_{-2} = \frac{d_{1,0,1} - q_0 a_{0,1} - q_{-1} a_{0,2}}{a_{0,3}},$$

$$q_{-3} = \frac{d_{1,0,0} - q_0 a_{0,0} - q_{-1} a_{0,1} - q_{-2} a_{0,2}}{a_{0,3}}$$

次に，秘密情報Aを乱数α_0で秘匿した後，乱数$\frac{\delta_{1,0}}{\alpha_0}$を掛けると，以下のようなる．なお，簡単に表現するため，各係数をΔ_lに置き換えている．

$$\frac{\delta_{1,0}}{\alpha_0} \times \alpha_0 A = \delta_{1,0} A = (q_0 + q_{-1} s^{-1} + q_{-2} s^{-2} + q_{-3} s^{-3}) \times \alpha_0 A$$

Vol.2021-CSEC-94 No.14
Vol.2021-SPT-43 No.14
2021/7/19

$$= \Delta_6 s^6 + \cdots + \Delta_{-3} s^{-3}$$

最後に乱数$\delta_{1,0}$で割り秘密情報を復元すると,以下のようになる.

$$\frac{\delta_{1,0} A}{\delta_{1,0}} = A = Q_3 s^3 + \cdots + Q_{-3} s^{-3}$$

ここで，Q_3からQ_{-1}は以下のように算出できる.

$$Q_3 = \frac{\Delta_6}{d_{1,0,3}}$$

$$Q_2 = \frac{\Delta_5 - Q_3 d_{1,0,2}}{d_{1,0,3}}$$

$$Q_1 = \frac{\Delta_4 - Q_3 d_{1,0,1} - Q_2 d_{1,0,2}}{d_{1,0,3}}$$

$$Q_0 = \frac{\Delta_3 - Q_3 d_{1,0,0} - Q_2 d_{1,0,1} - Q_1 d_{1,0,2}}{d_{1,0,3}}$$

$$Q_{-1} = \frac{\Delta_2 - Q_3 d_{1,0,-1} - Q_2 d_{1,0,0} - Q_1 d_{1,0,1} - Q_0 d_{1,0,2}}{d_{1,0,3}}$$

上式を計算すると，$Q_3 = a_3, Q_2 = a_2, Q_1 = a_1, Q_0, = a_0$となるが，$Q_{-1}$は割り切れず，$Q_{-1} \neq a_{-1}$となる．したがって，秘匿計算の分子の乱数が 1 個の場合，秘密情報を復元するとQ_{-1}からQ_{-3}，つまり，a_{-1}からa_{-3}で誤差が生じることになる.

【秘匿計算の分子の乱数が 2 個の場合】

秘密情報Aを乱数α_0で秘匿した後，乱数$\frac{\delta_{1,0}\delta_{2,0}}{\alpha_0}$を掛け，最後に乱数の積$\delta_{1,0}\delta_{2,0}$で割り秘密情報を復元する場合を考える．なお，乱数$\delta_{2,0}$を追加で以下のように設定する.

$$\delta_{2,0} = d_{2,0,3} s^3 + \cdots + d_{2,0,-3} s^{-3}$$

したがって，上記設定から，$\frac{\delta_{1,0}\delta_{2,0}}{\alpha_0}$は，以下のようになる.

$$\frac{\delta_{1,0}\delta_{2,0}}{\alpha_0} = q'_3 s^3 + q'_2 s^2 + q'_1 s + q'_0 + q'_{-1} s^{-1} + q'_{-2} s^{-2} + q'_{-3} s^{-3}\{+ q'_{-4} s^{-4} + \cdots\}$$

となる．ただし，q_3からq_{-3}は以下のように算出できる.

$$q'_3 = \frac{D_6}{a_{0,3}},$$

$$q'_2 = \frac{D_5 - q_3 a_{0,2}}{a_{0,3}},$$

$$q'_1 = \frac{D_4 - q_3 a_{0,1} - q_2 a_{0,2}}{a_{0,3}},$$

$$q'_0 = \frac{D_3 - q_3 a_{0,0} - q_2 a_{0,1} - q_1 a_{0,2}}{a_{0,3}},$$

$$q'_{-1} = \frac{D_2 - q_3 a_{0,-1} - q_2 a_{0,0} - q_1 a_{0,1} - q_0 a_{0,2}}{a_{0,3}},$$

$$q'_{-2} = \frac{D_1 - q_3 a_{0,-2} - q_2 a_{0,-1} - q_1 a_{0,0} - q_0 a_{0,1} - q_{-1} a_{0,2}}{a_{0,3}},$$

$$q'_{-3} = \frac{D_0 - q_3 a_{0,-3} - q_2 a_{0,-2} - q_1 a_{0,-1} - q_0 a_{0,0} - q_{-1} a_{0,1} - q_{-2} a_{0,2}}{a_{0,3}}$$

次に，秘密情報Aを乱数α_0で秘匿した後，乱数$\frac{\delta_{1,0}\delta_{2,0}}{\alpha_0}$を掛けると，以下のようなる．なお，簡単に表現するため，各係数をΔ'_lに置き換えている.

$$\frac{\delta_{1,0}\delta_{2,0}}{\alpha_0} \times \alpha_0 A = \delta_{1,0}\delta_{2,0} A$$
$$= (q'_3 s^3 + q'_2 s^2 + q'_1 s + q'_0 + q'_{-1} s^{-1} + q'_{-2} s^{-2} + q'_{-3} s^{-3}) \times \alpha_0 A$$
$$= \Delta'_9 s^9 + \cdots + \Delta'_{-3} s^{-3}$$

最後に乱数の積$\delta_{1,0}\delta_{2,0}$で割り秘密情報を復元すると，以下のようになる.

$$\frac{\delta_{1,0}\delta_{2,0} A}{\delta_{1,0}\delta_{2,0}} = A = Q'_3 s^3 + \cdots + Q'_{-3} s^{-3}$$

ここで，Q_3からQ_{-1}は以下のように算出できる.

$$Q'_3 = \frac{\Delta'_9}{D_6}$$

$$Q'_2 = \frac{\Delta'_8 - Q_3 D_5}{D_6}$$

$$Q'_1 = \frac{\Delta'_7 - Q_3 D_4 - Q_2 D_5}{D_6}$$

$$Q'_0 = \frac{\Delta'_6 - Q_3 D_3 - Q_2 D_4 - Q_1 D_5}{D_6}$$

$$Q'_{-1} = \frac{\Delta'_5 - Q_3 D_2 - Q_2 D_3 - Q_1 D_4 - Q_0 D_5}{D_6}$$

$$Q'_{-2} = \frac{\Delta'_4 - Q_3 D_1 - Q_2 D_2 - Q_1 D_3 - Q_0 D_4 - Q_{-1} D_5}{D_6}$$

$$Q'_{-3} = \frac{\Delta'_3 - Q_3 D_0 - Q_2 D_1 - Q_1 D_2 - Q_0 D_3 - Q_{-1} D_4 - Q_{-2} D_5}{D_6}$$

上式を計算すると，$Q'_3 = a_3, Q'_2 = a_2, Q'_1 = a_1, Q'_0 = a_0, Q'_{-1} = a_{-1}, Q'_{-2} = a_{-2}, Q'_{-3} = a_{-3}$となる．したがって，秘匿計算の分子の乱数が 2 個の場合，秘密情報を復元すると誤差は生じない．したがって，秘匿計算における分子の乱数を増加させると，秘密情報復元時の誤差が小さくなると言える．また，安全性に関しては秘匿計算における分子の数が 1 つ増加するごとに候補が10^t通りずつ増加する.ゆえに，秘匿計算における分子の数を最低でも 3 つ以上とすれば，その候補の数は10^{2t}通りとなり，前提条件における小数点以下の桁数$t = 20$桁を考慮すると，$10^{40} > 10^{39} \approx 2^{128} - 1$となるため，分子の各乱数を推定することは困難となる．したがって，秘匿計算における分子の数が増加すると，演算精度が向上するとともに，安全性も向上すると言える.

5. まとめ

TUS 方式で用いられている秘密情報に乱数を掛けるというアプローチのもと，実数上で直接計算を行うことができる秘匿計算法の提案を行った．より詳細な誤差の検討や性能評価は今後の課題であるが，アルゴリズムの実装にあ

Vol.2021-CSEC-94 No.14
Vol.2021-SPT-43 No.14
2021/7/19

たってはコンピュータ内部で小数が浮動小数点数として扱われるため，その誤差を考慮しなければならない．

参考文献

[1] 「総務省｜平成 28 年版 情報通信白書｜新たな ICT がもたらす社会経済へのインパクト」，https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h28/html/nc111220.html
[2] Shamir Adi, "How to share a secret.", Communications of the ACM 22.11 (1979): pp.612-613.
[3] Catrina Octavian, and Amitabh Saxena. "Secure computation with fixed-point numbers." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2010.
[4] Catrina Octavian. "Round-efficient protocols for secure multiparty fixed-point arithmetic." 2018 International Conference on Communications (COMM). IEEE, 2018.
[5] Aliasgari Mehrdad, et al. "Secure Computation on Floating Point Numbers." NDSS. 2013.
[6] 天田拓磨, 奈良成泰, 西出隆志, 吉浦裕, “通信量を削減した浮動小数点演算のためのマルチパーティ計算”, 情報処理学会論文誌, 2019 年, 60 巻号, pp.1433-1447
[7] Mohassel Payman, and Yupeng Zhang. "Secureml: A system for scalable privacy-preserving machine learning." 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017.
[8] Mohassel Payman, and Peter Rindal. "ABY3: A mixed protocol framework for machine learning." Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018.
[9] 金岡晃, 宮西洋太郎, 韓嘯公, 北上眞二, 佐藤文明, 浦野義頼, 白鳥則郎, “実数演算可能な軽量秘密計算法の一考察”, コンピュータセキュリティシンポジウム 2014 論文集, pp.682-687
[10] 高橋康太, 佐藤文明. "簡易的秘密計算によるクラウド利用インタフェースの開発と応用.", 第 24 回マルチメディア通信と分散処理ワークショップ論文集, 2016 年, pp.246-252.
[11] 神宮武志, 青井健, 岩村惠市, “秘密分散法を用いた次数変化のない秘匿計算手法”, 情報処理学会論文誌, 2018 年, 第 59 巻, 第 3 号, pp.1038-1049
[12] ムハンマド カマル アフマド アクマル アミヌディン, 岩村惠市, "秘密分散法を用いた四則演算の組み合わせに対して安全な次数変化のない秘匿計算." 情報処理学会論文誌, 2018 年, 第 59 巻第 9 号, pp.1581-1595.
[13] 鴇田恭平, 岩村惠市, "高速かつ n<2k-1 において秘密情報に 0 を含んでも実行可能な秘密分散法による秘匿計算法", 2018 年, 電気学会論文誌 C, Vol.138, No.12, pp.1634-1645.
[14] Keiichi Iwamura, Ahmad Akmal Aminuddin Mohd Kamal, “Secure Computation by Secret Sharing Using Input Encrypted with Random Number (Full Paper)”, 18th International Conference on Security and Cryptography (SECRYPT 2021), 入手先: https://eprint.iacr.org/2021/548.pdf