特集
Special Feature

［安全なデータ活用を実現する秘密計算技術］

# ③秘密分散法を用いた 3 者秘密計算の有用性

応
専

荒木俊則 | NEC　森田　啓 | 産業技術総合研究所　花岡悟一郎 | 産業技術総合研究所

1986 年に Yao（FOCS'86）がミリオネア問題と呼ばれる金持ちの財産比べプロトコルを提案したのが秘密計算の始まりとされる．これは，金持ち同士が自分たちの財産の額は明かさずに，どちらが金持ちであるかのみを知ることのできるプロトコルである．そのプロトコルを構成するため，本稿で扱う秘密分散のほかにも，Garbled circuit や準同型写像を利用したものなど，さまざまな研究がなされている．すでに計算機で表現できる関数のすべては秘密計算が可能であることが知られているが，そのほとんどが計算や通信のコストが大きく，実用には程遠かった．金持ちであれば，多大な金額を計算機に投入し財産比べプロトコルを行えるが，そうでない者にとっては，秘密計算をしているうちに財産を失ってしまいかねない！

**3 者秘密計算の性能向上．** データ処理を秘匿したいという現代社会の要求に応え，技術の実用化を図るにはまず，こうしたコストの削減が望まれる．現在，任意の処理を可能とする 3 者秘密計算の性能が大きく向上し，アプリケーションによっては実用に耐える性能が確保可能となりつつある．本稿ではそのような 3 者秘密計算のうち，NEC の取り組む複製型秘密分散を用いた 3 者秘密計算と，産業技術総合研究所の取り組むクライアント補助型秘密計算を紹介する．

## 複製型秘密分散を用いた 3 者秘密計算

本章では，複製型秘密分散を用いた 3 者秘密計算について説明する．秘密分散，複製型秘密分散，複製型秘密分散を用いた秘密計算，秘密計算の安全性について述べる．

### 秘密分散とは

秘密分散は，機密情報を複数のシェアと呼ばれる情報に加工するプロトコルである．シェアは，あらかじめ定められた組合せのみから機密情報が復元できるように作られる．いくつかのシェアが流出した場合でも機密情報の機密性は守られ，いくつかのシェアが失われても，秘密情報が復元できる．

**$(k, n)$ しきい値法**

$(k,\ n)$ しきい値法と呼ばれる秘密分散について説明する．これは，$n$ 人の参加者 $\mathcal{P}=\{P_1,\dots,P_n\}$ とディーラ $\mathcal{D}$ によって行われるプロトコルであり，2 つのアルゴリズム ShareGen と Reconst によって構成される．ShareGen は秘密情報 $s$ を分散するためのアルゴリズムであって，ディーラ $\mathcal{D}$ によって実行されるものとする．ShareGen は秘密情報 $s$ を入力として，シェアのリスト $(v_1,\dots,v_n)$ を出力する．シェアは，$k$ 人以上の参加者が集まれば秘密情報を復元でき，$k-1$ 人以下の参加者が集まっても秘密情報に関する情報を得られないように作られる．次項では，本稿で用いる複製型秘密分散を用いた (2, 3) しきい値法について説明する（図 -1）．

**複製型 (2, 3) しきい値法**

$N$ を整数とする．また，秘密情報 $x$ を $N$ 未満の整数として，複製型 (2, 3) しきい値法の ShareGen と Reconst を記載する．

-ShareGen：秘密情報 $x \in \mathbb{Z}_N$ を入力として，以下の処理に従って $(v_1, v_2, v_3)$ を計算し，出力する[☆1]．

1. $x_1+x_2+x_3 = x \bmod N$ を満たす $x_1, x_2, x_3 \in \mathbb{Z}_N$ を一様ランダムに選択する[☆2]．
2. $v_1 = (x_1, x_3), v_2 = (x_2, x_1), v_3 = (x_3, x_2)$ を出力する．

$(v_1, v_2, v_3)$ からどの 2 つを選んでも $x_1, x_2, x_3$ が含まれていることを確認されたい．

-Reconst：シェアのリスト $(v_{i_1},\ldots,v_{i_k})$ を入力として，秘密情報を以下の処理に従って出力する．

1. 入力されたシェアから $x_1, x_2, x_3$ を抽出する（$x_1, x_2, x_3$ は，どの 2 つ以上のシェアからでも揃う）．
2. $x = x_1+x_2+x_3 \bmod N$ を出力する．

$x_1, x_2, x_3$ と $x$ の関係から，分散した値が復元される．また，各シェアは $x_1, x_2, x_3$ のどれかが欠けている．残りの値を推測することで $x$ を類推できるが，$x_i$ がランダムに選ばれることから $x$ 自体を推測することと相違ない．したがって，各シェアからは秘密情報 $x$ に関する情報が漏れない．

本節では，数値を分散・復元する方法を示した．次節では，分散された数値を復元をすることなく計算する方法を示す．

## 複製型 (2, 3) しきい値法を用いた秘密計算

分散された値に関して加減算と乗算が実行できれば，任意の計算が実行可能となる（図 -2）．ただし，計算は法 $N$ の下での計算とする[☆3]．本節では，まず加減算の方法について説明し，次に乗算に関して説明する．$x, y \in \mathbb{Z}_N$ が複製型 (2, 3) しきい値法によって $P_1, P_2, P_3$ の間で分散されているものとする．具体的には，$x = x_1+x_2+x_3 \bmod N$, $y = y_1+y_2+y_3 \bmod N$ を満たす $x_1, x_2, x_3, y_1, y_2, y_3$ に関して $P_i (i = 1, 2, 3)$ は以下の情報を有する．

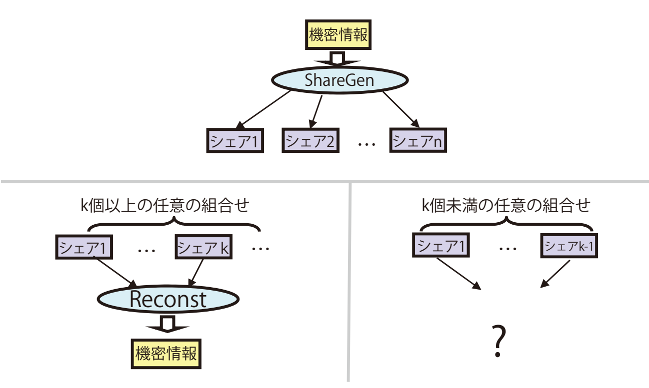
− $P_1$：$x$ のシェア $(x_1, x_3)$，$y$ のシェア $(y_1, y_3)$
− $P_2$：$x$ のシェア $(x_2, x_1)$，$y$ のシェア $(y_2, y_1)$
− $P_3$：$x$ のシェア $(x_3, x_2)$，$y$ のシェア $(y_3, y_2)$

**加減算**

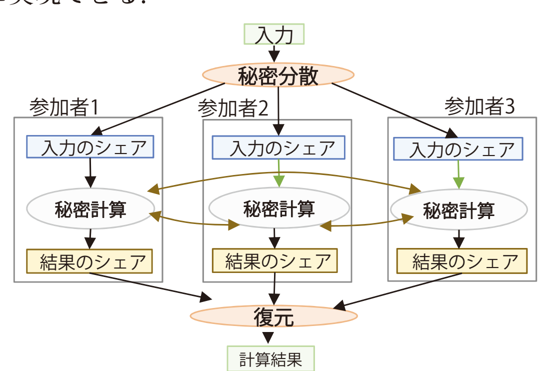
$i = 1, 2, 3$ について，$z_i = x_i+y_i \bmod N$ とする．このとき，各参加者 $P_i (i = 1, 2, 3)$ は $(z_i, z_{i-1})$ を $x+y$ のシェアとする．なお，下付き文字において 0 は 3 に置き換える．この処理は，各参加者が自身が所有する情報だけを用いて計算することができる．この値が $x+y \bmod N$ のシェアとなっているかは，$z_1+z_2+z_3 = x+y \bmod N$ が成立することから確認できる．

$$\begin{aligned} &z_1+z_2+z_3 \bmod N \\ &= (x_1 + y_1) + (x_2 + y_2) + (x_3 + y_3) \bmod N \\ &= (x_1 + x_2 + x_3) + (y_1 + y_2 + y_3) \bmod N \\ &= x+y \bmod N \end{aligned}$$

この計算の過程に参加者同士のやりとりがないため，$x$ や $y$ に関する情報が計算過程で漏洩することはない．減算は，$z_i = x_i - y_i \bmod N$ とすれば同様に実現できる．




■図 -1 $(k, n)$ しきい値法




■図 -2 3 者への秘密分散を用いた秘密計算

☆1 $\mathbb{Z}_N=\{0, 1, ..., N-1\}$ とする．
☆2 $x \bmod N$ とは $x$ を $N$ で割った余りである．
☆3 法 $N$ の下での計算とは，計算結果は $N$ で割った余りとするということである．

**乗算**

乗算においては，参加者間の通信が必要である．なお，下付き文字において 0 は 3 に，4 は 1 に置き換えるものとする．

1. 各 $P_i$ は，$r_i \in \mathbb{Z}_N$ をランダムに選択し，$r_i$ を $P_{i+1}$ に送付する．
2. 各 $P_i$ は，$z_i = x_i \cdot y_i + x_i \cdot y_{i-1} + x_{i-1} \cdot y_i + r_i - r_{i-1} \bmod N$ を計算し，$P_{i+1}$ に送付する．
3. 各 $P_i$ は，$(z_i, z_{i-1})$ を $z$ のシェアとする．

上記の処理で計算した値が $x \cdot y \bmod N$ のシェアとなっていることは，$z_1+z_2+z_3 = x \cdot y \bmod N$ が成立していることから確認できる．

$$
\begin{aligned}
&z_1+z_2+z_3 \bmod N \\
&\quad = x_1 \cdot y_1 + x_2 \cdot y_2 + x_3 \cdot y_3 \\
&\quad + x_1 \cdot y_2 + x_1 \cdot y_3 + x_2 \cdot y_1 + x_2 \cdot y_3 \\
&\quad + x_3 \cdot y_1 + x_3 \cdot y_2 \\
&\quad + r_1 + r_2 + r_3 - (r_1 + r_2 + r_3) \\
&\quad = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) \bmod N \\
&\quad = x \cdot y \bmod N
\end{aligned}
$$

各参加者が計算結果 $x, y, x \cdot y$ について追加の情報を得ていないことを確認する．プロトコルの対称性から $P_1$ に着目する．$P_1$ が受信する値は，$r_3$ と $z_3 = x_3 \cdot y_3 + x_3 \cdot y_2 + x_2 \cdot y_3 + r_3 - r_2 \bmod N$ であり，これらから $x_2$ か $y_2$ や，これらの値の関係性が定まると追加の情報が漏洩していることになるが，$P_1$ が $r_2$ を知らないために防がれている．

**乗算に関する通信量の削減**

前節で説明した秘密計算法において通信にかかる時間は大きいといわれている．可能な限り，通信回数と量を減らすことが効率的な処理につながる．本節では，前節で示した方法において通信量を削減するための工夫を述べる．

前節で示した方法は，各参加者が 2 つの要素をほかの参加者に送っている．まず，1. の処理は乗算を行う値と無関係に実行することができるので，計算を行う前にまとめて実行してもよい．また，$F : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \mathbb{Z}_N$ を鍵長 $k$, $n$ ビットの数値（ID）を入力として $\mathbb{Z}_N$ の要素を出力する疑似ランダム関数☆4 とすると以下のような方法で初期化処理を行った後は通信を行うことなく乗算の 1. の処理を実行することが可能となる．なお，4 は 1 に，0 は 3 に置き換えるものとする．なお，次に計算される乗算に対応する ID については，通し番号を用いるなどの方法で同期がとられているものとする．

- 各 $P_i$ は，$k_i \in \{0, 1\}^k$ をランダムに選択し，$k_i$ を $P_{i+1}$ に送付する．これを初期化処理とする．
- 各 $P_i$ が，ID に対応する $r_i$, $r_{i-1}$ を計算する場合，$r_i = F(k_i, \mathrm{ID})$, $r_{i-1} = F(k_{i-1}, \mathrm{ID})$ によって計算する．

以上の初期化処理を行うと，乗算 1 回あたり，各参加者は 1 つの要素を送受信するだけとなる．プロトコルにおいて $N=2$ とすると，加減算は XOR 演算，乗算は AND 演算となる．AND 演算は各参加者が 1 ビットだけの情報を送受信すれば良く非常に効率的となる．

2016 年，このプロトコルを利用して標準暗号である AES-128 を秒間 120 万回以上処理するという結果が発表された [1]．AES を用いている認証プロトコル Kerberos に当てはめると秒間 35,000 回以上の認証処理となり，大きな組織で利用される認証サーバとして用いても十分な性能が示されている．その後は，より複雑な処理を効率的に実現可能とするための補助プロトコルの開発も進み，統計処理への適用が始まっており，処理によっては高い性能を示している．

## 秘密計算法の安全性

これまで秘密計算プロトコルの実行過程で発生する情報漏洩には触れてきたが，そのほかにも考慮すべき重要な問題がある．それは参加者の不正な行動である．たとえば，乗算のプロトコルにおいて，$P_1$ 自身が計算した値 $z_1$ を改ざんし，$z'_1 = z_1 + \delta$ とし

☆4 鍵を知らない場合，その出力を ID から予測できない関数とする．

て $P_2$ に送付することを考える．このとき，乗算結果 $z' = z'_1 + z_2 + z_3 = z_1 + z_2 + z_3 + \delta \bmod N$ となり，ちょうど $\delta$ だけ増えてしまう．このような改ざんが入った計算結果を復元したとき，改ざんを行った参加者は自身が付与した差分を考慮して正しい値を取得するが，ほかの参加者はその事実にも気づくことができない．このように，プロトコルに従わない参加者は malicious adversary と呼ばれる．それに対し，プロトコルに従うが可能な限り情報を手に入れようとする参加者は semi-honest adversary と呼ばれる．

Malicious adversary による不正な振舞いがあったことが検知できる機能や，不正な振舞いをした参加者を特定する機能に関する研究も非常に盛んに行われている．

文献 2）においては，$N$ が素数である場合の不正検知方法が示されている．この方法では，不正の検知能が $N$ のサイズによっており，高い検知能を得るためには大きな $N$ を選択する必要がある．文献 3）においては $N$ が $2^x$ である場合の不正検知方法が示されている．この方法は $x$ のサイズによらず，通信量が 7 倍となることと引き換えに $1 - 2^{-40}$ という非常に高い検知能を得る．

どのような用途において，どのような方法が効率的なのかといった問題は，さらに効率の良い方式の開発も含めて，今後の重要な課題である．

## クライアント補助型秘密計算

本章では，クライアント補助型 2 者秘密計算☆5 と呼ばれる 3 者秘密計算を取り上げる．これは，クライアントと呼ばれる第三者が 2 者秘密計算を補助する 3 者秘密計算である．クライアント補助型 2 者秘密計算は比較的軽量であり，実行時間や計算コストの面で実用的な方式を提供できることについて，例を挙げながら説明する．

### 2 者秘密計算とは

2 者秘密計算として代表的なものは，GMW 秘密分散法☆6 である．GMW 秘密分散法を用いて「2 者で秘密 $x$ を秘密分散する」ならば，1 人がシェアと呼ばれるある値 $x_1$ を持ち，もう 1 人はシェアと呼ばれるある値 $x_2$ を持つ状態となる．なお，シェア $x_1$ および $x_2$ は，十分大きな正の整数 $N$ に対して，以下のような関係を満たす非負整数である：

$$x_1 < N, x_2 < N \text{かつ} x = x_1 + x_2 \bmod N.$$

では，実際にはどのように $x_1$, $x_2$ は作られるのだろうか？

秘密 $x$ は 2 者のどちらかの持つ秘密の値と考えてもよいし，第三者が持つ秘密の値と考えてもよい．ここでは第三者が持つ秘密の値と捉えることにする．すなわち，第三者が秘密 $x$ を持っており，乱数 $r$ を集合 $\{0, 1, 2, \cdots, 2^n - 1\}$ から選び，

$$x_1 := r, x_2 := x - r$$

とする．$x_1$ が 2 者のうちどちらか 1 人に，$x_2$ はもう 1 人に送信（秘密分散）される．乱数を用いているため，それぞれの値だけからは $x$ の値は漏れないが，足し合わせると $x$ になるように秘密分散されていることが確認できる．このように第三者のいる設定は，クライアント・サーバモデルと呼ばれ，2 者が 2 つのサーバに相当し，第三者がクライアントに相当する．クライアントが値 $x$ を秘匿したまま所望の演算を行いたい場合，上記のような方法で 2 つのサーバに秘密をシェアし，この 2 者に計算してもらう方法が，秘密分散に基づいた 2 者秘密計算（以下では単に 2 者秘密計算と呼ぶ）である．

### なぜ 2 者秘密計算は実用的でないのか？

2 者秘密計算では，入力を秘匿したままの加算（秘

☆5 実際に秘匿データ処理・演算を行う 2 者と，入出力にかかわるクライアントを合わせた 3 者と捉えられる．

☆6 提案者達の名前 Goldreich, Micali, Wigderson の頭文字をとってこう呼ばれる．

匿加算☆7）は非常に低コストに実行可能である．たとえば，秘密 $x$ と $y$ がすでに2者で秘密分散されているとする．すなわち，$x = x_1+x_2 \bmod N$, $y = y_1+y_2 \bmod N$ に対し，サーバ1は $x_1, y_1$ を持ち，サーバ2は $x_2, y_2$ を持っているとする．加算 $x+y$ を秘匿したまま計算したい場合，各サーバは手元にある2つのシェアをそのまま足せばよい．すなわち，秘匿加算の実行後にはサーバ1は $x_1+y_1$ を持ち，サーバ2は $x_2+y_2$ を持つ．サーバがそれぞれ持つシェアの値からは，$x$ や $y$ に関する情報は何も漏れていないことが分かるはずだ．さらに，上記の処理で計算した値が $x+y$ のシェアとなっていることは，$(x_1+y_1)+(x_2+y_2) = x+y \bmod N$ が成り立つことから分かる．

一方で，秘匿乗算は秘匿加算のように単純ではない．仮に各サーバが手元でシェアを乗算したとする．このときサーバ1は $x_1y_1$ を持ちサーバ2は $x_2y_2$ を持つ．これらが $xy$ のシェアになっていないことは $x_1y_1+y_2y_2 \neq xy \bmod N$ であることより分かる．

シェアの生成法より $xy = x_1y_1+y_2y_2+x_1y_2+y_2y_1 \bmod N$ であるが，$x_1y_2$ や $x_2y_1$ といった項は各サーバが手元で計算できる値ではないため，手元にあるシェア同士を掛け算しただけでは秘匿乗算を行うことはできなかったわけである．この問題を解決するためには，紛失通信と呼ばれる公開鍵暗号系の技術を用いればよいが，それは通信および計算のコストが大きく，実用的ではない．

## 秘匿乗算の軽量化の試み

Beaver は，Multiplication Triple（MT）と呼ばれる，秘匿乗算のための乱数の三つ組を2者で秘密分散して用いれば，秘匿乗算の実行の過程で紛失通信を行わずに済む手法を提案した．実際は MT の生成において紛失通信を行うことになるが，乗算したい数値とは独立な乱数の三つ組を秘密分散すればよいため，実際に行いたい秘匿乗算をするまでの任意のタイミングで MT を生成することが可能である．

こうした MT の生成は言い換えると，実行の過程での通信コストや計算コストを軽減し，事前計算にコストの大半を押しつけることができる手法であるといえる．

事前計算に通信・計算コストを押しつけることは，実用性を考えた場合には必ずしも最良の選択肢というわけではない．たとえば，2つのサーバはそれぞれ，Web 上にて健康診断をするサービス提供者と秘密計算用の依頼計算サーバとし，クライアントはそのサービスの利用者とする．このとき，クライアントは自身の遺伝子情報など秘密の情報をシェアの形でサーバに送り，その情報を漏らさないままに診断してもらう状況を考える．このとき，確かにクライアントが自身の秘密情報を秘密分散してから，健康診断結果をサーバから送り返してもらうまでの通信時間・計算時間的コストは軽量化しているが，サーバ自身の事前計算を含む実際の演算実行にかかる総コストはそれほど軽量化しているわけではない．これはすなわち，サービスを提供する際の提供コストは相変わらず高いままということである．利用者は，2つのサーバが秘密計算に用いた提供コストに見合う高い利用料を払う必要が出てしまう．これは（GMW 秘密分散ベースの）3者秘密計算を用いても同様である．

## クライアント補助型秘密計算により問題解決！

上述したような，事前計算でも通信・計算コストが大きいという問題点を解決する方法の1つとして着目されているのが，クライアント補助型2者秘密計算である（図-3）．本稿で主として取り上げたクライアント・サーバモデルでは，クライアントが自身の秘密をシェアの形に変換してサーバに送信していた．そこで，クライアントがさらに MT を生成し，それをシェアの形に変換してサーバに送ることにすれば，サーバが事前計算して MT を生成するより

☆7　前章の複製型秘密分散とはシェアの形式が異なるため，秘匿加算や秘匿乗算の方法に違いがあることに注意．

もはるかに効率的にMTを生成することができる．そのからくりの理解のために，以下にMTの生成について説明する．

## Multiplication Triple（MT）の生成

MTは$ab = c$を満たす乱数の三つ組（$a, b, c$）のシェアのことである．ここで，$a = a_1+a_2$, $b = b_1+b_2$, $c = c_1+c_2$のように秘密分散されるとする．具体的には，

$$(a_1+a_2)(b_1+b_2) = (c_1+c_2) \tag{1}$$

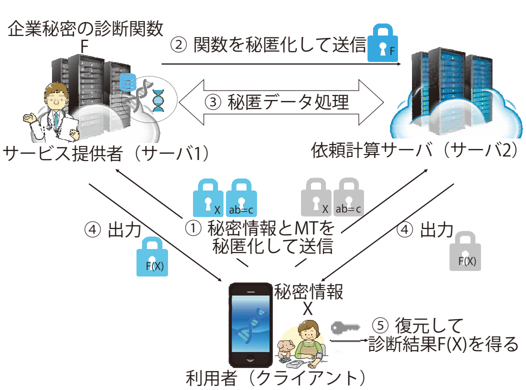
を満たす乱数（$a_1, a_2, b_1, b_2, c_1, c_2$）に対して，サーバ1は（$a_1, b_1, c_1$）をシェアとして持ち，サーバ2は（$a_2, b_2, c_2$）をシェアとして持つ．式(1)を満たすような乱数（$a_1, a_2, b_1, b_2, c_1, c_2$）を，$a, b, c$を知らずに2つのサーバ間のやりとりだけで生成することは（可能ではあるが紛失通信を行うなど通信・計算コストがかかり）容易ではない．一方，クライアントが$ab = c$なる乱数の三つ組を選んでから，$a$, $b$, $c$をそれぞれシェアの形にしてサーバに送ることは効率的である．これがクライアント補助型2者秘密計算の主なメリットである．

## 効率的な秘匿比較プロトコル

ここまで，秘匿加算が2者の通信なしで実行できること，また秘匿乗算もクライアント補助型2者秘密計算を用いることで低コストで実行できることを記した．さらに高度な演算を秘匿したまま行いたいという要求が，データ処理をはじめとした応用分野からなされている．その中でも特に重要な演算が，秘匿比較プロトコルである．具体的な利用例としては，最大値，最小値を求める問題や，ニューラルネットワークの分野における活性化関数の計算などを，秘匿したまま実行する際に道具として用いられる．秘匿比較プロトコルはYaoが秘密計算を提案して以来，さまざまな研究がなされており，表-1に示すように通信ラウンド数の効率化が図られている．通信ラウンドとは，2者間での1人あたりの通信回数と考えてほぼ差し支えない．秘密分散を用いた秘匿計算でとりわけ通信ラウンド数を気にするのは，1回の通信に20～80ミリ秒ほどの遅延が発生し，それが実行時間に対して支配的に影響するためである．そこで，通信する総データ量をそれほど増やすことなく，通信ラウンド数の少ないプロトコルをいかに構成するかが焦点になる．

秘匿比較プロトコルは，本質的にはビットごとの処理が必要なプロトコルであり，算術的に処理できる秘匿加算や秘匿乗算とは異なり，対象とする値のビット長の対数オーダーに比例する程度の通信ラウンド数が必要とされていた．

ところがDamgård ら[4]は，ビット分解と呼ばれる手法でビットごとに算術演算を可能にし，さらに逆数の関係にある相関乱数を利用することで，複数




■図-3　クライアント補助型2者秘密計算を用いて健康診断サービスを提供するモデル

■表-1　秘匿比較プロトコルの性能比較

| 秘匿比較プロトコル | 通信ラウンド | 通信する総要素数☆8 | 実行時間☆9 (ms) |
|---|---|---|---|
| Damgård ら[4] | 79 | $176n \log n + 80n \log\log n + 70n$ | 5,688 |
| 西出・太田[5] | 28 | $96n + 120 \log n + 4$ | 2,016 |
| Damgård ら + クライアント補助型 | 70 | $144n \log n + 64n \log\log n + 52n$ | 5,040 |
| 西出・太田 + クライアント補助型 | 14 | $36n + 48 \log n + 7$ | 1,008 |
| 森田ら[6] | 5 | $12n^2 + 301$ | 360 |

☆8　1要素あたり$n$ビット．

☆9　総時間はネットワーク遅延が72msであるWAN環境を仮定している．これは米国の東海岸と西海岸とをつなぐネットワークに相当する．

の OR 演算や複数の乗算を定数回の通信ラウンドで実行可能な手法を提案した．そして，それを構成要素として用いることで，秘匿比較プロトコルの通信ラウンドを定数回におさえられることを示した．さらに西出・太田[5]は，処理の重いビット分解を利用せず，ビットごとの乱数シェアを生成するプロトコルを構成要素として用いて，さらに少ない通信ラウンド数で秘匿比較プロトコルが構成できることを示した．森田ら[6]は，上記の 2 つの構成法を踏まえて，クライアント補助型 2 者秘密計算のモデルのもとで，データの表現形式に木構造を用いることでビットごとの処理を並列化できる方法を示し，通信ラウンドが 5 回の実用的な秘匿比較プロトコルを構築した．このプロトコルは，これまで最良の定数ラウンドプロトコルであった西出・太田の秘匿比較プロトコルと比べて，通信ラウンド数は 5 分の 1 以下である．また，西出・太田のプロトコルをクライアント補助型のモデルで実行した場合と比べても，通信ラウンド数を 3 分の 1 程度に削減している．秘匿比較プロトコルがより実用的な速度で実行可能となり，応用実装に利用できる段階にようやくたどり着いたといえる．

**参考文献**

1) Araki, T., Furukawa, J., Lindell, Y., Nof, A. and Ohara, K.: High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority, ACM-CCS 2016, pp.805-817 (2016).
2) Ikarashi, D., Kikuchi, R., Hamada, K. and Chida, K.: Actively Private and Correct MPC Scheme in $t < n/2$ from Passively Secure Schemes with Small Overhead, IACR Cryptology ePrint Archive : Report 2014/304 (2014).
3) Araki, T., Barak, A., Furukawa, J., Lichter, T., Lindell, Y., Nof, A., Ohara, K., Watzman, A. and Weinstein, O.: Optimized Honest-Majority MPC for Malicious Adversaries - Breaking the 1 Billion-Gate Per Second Barrier, IEEE S&P 2017, pp.843-862 (2017).
4) Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J. and Toft, T.: Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation, TCC 2006, pp.285-304 (2006).
5) Nishide, T. and Ohta, K.: Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol, PKC 2007, pp.343-360 (2007).
6) Morita, H., Attrapadung, N., Teruya, T., Ohata, S., Nuida, K. and Hanaoka, G.: Constant-Round Client-Aided Secure Comparison Protocol, ESORICS (2) 2018, pp.395-415 (2018).

(2018 年 7 月 2 日受付)

**荒木俊則**　t-araki@ek.jp.nec.com
2005 年東京工業大学理工学研究科集積システム専攻博士前期課程修了．同年，NEC 入社．現在，同社セキュリティ研究所主任．2011 年同大学院イノベーションマネジメント研究科イノベーション専攻博士後期課程修了．博士（工学）．

**森田　啓**　hiraku.morita@aist.go.jp
2017 年名古屋大学工学研究科計算理工学専攻博士後期課程修了．現在，産業技術総合研究所情報技術研究部門高機能暗号研究グループ特別研究員．博士（工学）．専門は暗号理論．

**花岡悟一郎（正会員）**　hanaoka-goichiro@aist.go.jp
2002 年東京大学大学院工学系研究科電子情報工学専攻博士課程修了．現在，産業技術総合研究所情報技術研究部門高機能暗号研究グループ研究グループ長．博士（工学）．専門は暗号理論．