情報処理学会論文誌 Vol.59 No.10 1895–1902 (Oct. 2018)

推薦論文

# 軽量Nパーティ秘匿関数計算の一般化

滝 雄太郎[1] 藤田 茂[2] 宮西 洋太郎[3] 白鳥 則郎[4]

受付日 2018年5月7日, 採録日 2018年7月10日

**概要**：本稿では “軽量 3 パーティ秘匿関数計算” を軽量 N パーティ秘匿関数計算として一般化を行った．その結果，分散の主体数を $n$，復元および計算に必要な主体数を $k$ とすると，$n$ が偶数のときに $n \geq 2k$，奇数のときに $n \geq 2k-1$ であることが秘密計算可能であるための十分条件であることを示した．この一般化の条件を導く際に Lee 距離を効果的に用いた．本稿の結果より主体数 $n$ と主体数 $k$ を一般化の条件下で選択することが可能となる．これにより，システム運用者は耐障害性・対攻撃性に応じて，$n$，$k$ を広範囲に選択することが可能となる．

キーワード：クラウド，秘密分散，秘密計算，セキュリティ，分散処理

# Lightweight N-party Seure Function Evaluation with Error Detection

Yutaro Taki[1] Shigeru Fujita[2] Yohtaro Miyanishi[3] Norio Shiratori[4]

Received: May 7, 2018, Accepted: July 10, 2018

***Abstract:*** In this paper, “A Lightweight Three-party secure function evaluation” is generalized as “A N-party secure function evaluation”. As a result of generalization, it was shown that $n \geq 2k$ when $n$ is even number and $n \geq 2k-1$ when it is odd number. The Lee distance was used effectively when deriving the condition of this generalization. From the results of this paper, it becomes possible to select the number of party “$k$” and the number of party “$n$” necessary for restoring data under generalization conditions. As a result, the system operator can freely select $n$, $k$ according to fault tolerance/anti-aggressiveness.

***Keywords:*** cloud, secret sharing, secure computation, security, distributed processing

## 1. はじめに

セキュリティ対策やプライバシ保護等，情報の秘密を守ることが，情報システムを構築するうえで，重要な課題である [2]．この課題を解決する 1 つの手法として，情報を複数箇所に分散し，秘密を守る秘密分散が注目・実用化されている．しかし，分散した情報を 1 カ所に集めて処理すると，その 1 カ所にリスクが集中する．このため，情報を分散させるだけでなく，分散したままで処理を実行する秘密計算，秘匿関数計算が提案されている [3], [4], [5]．

一般的に秘匿関数計算では情報を $n$ 個のデータに分散し，その中の $k$ 個のデータを用いて目的の処理を達成する手法を，k-out-of-n と記す．この k-out-of-n は $n = 3$ の特別な場合の処理が与えられている [1]．しかし一般の $n > 3$ については，その条件が明らかになっていない，未解決の問題として知られている．そこで我々は，秘匿関数計算（k-out-of-n）の一般化を検討してきた [6]．

本稿では，まず未解決の $n > 3$ に対する一般化の条件を与える．ここで，Lee 距離 [7] を効果的に導入することにより，一般化に成功している．この一般化により適用範囲

[1] 大日本印刷株式会社
Dai Nippon Printing Co., Ltd., Shinjuku, Tokyo 162–8001, Japan
[2] 千葉工業大学情報科学部情報工学科
Department of Computer Science, Chiba Institute of Technology, Narashino, Chiba 275–0016, Japan
[3] 株式会社アイエスイーエム
ISEM, Inc., Chuou, Tokyo 103–0008, Japan
[4] 中央大学研究開発機構
Research and Development Initiative, Chuo University, Bunkyo, Tokyo 112–8551, Japan

本稿の内容は 2017 年 6 月のマルチメディア，分散，協調とモバイル（DICOMO2017）シンポジウムで報告され，マルチメディア通信と分散処理研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

が広がり，セキュリティ対策と可用性のバリエーションを選択できるという貢献が期待できる．

以下，本稿では，2 章で関連研究について述べ，3 章で提案方式について述べ，4 章で一般解の成立条件について述べる．そして最後に，5 章でまとめを述べる．

## 2. 関連研究

この章では提案方式の元となる秘匿関数計算 [1] とその乗算方法について述べる．この方法は全体の主体数を $n$，データの復元に必要な主体数を $k$ とした k-out-of-n のマルチパーティプロトコルにおいて，$n=3$，$k=2$ とした，2-out-of-3 のものである．

秘匿関数計算 [1] で示された 2-out-of-3 では，分散したデータを復元することなく加減算・定数倍，乗算，論理演算等の各計算処理を行うことが可能である．

以下に計算処理の一例として乗算処理の手順を示す．本稿での，計算主体やシェアの記法は文献 [1] に準拠している．たとえば，$P_i$ は各計算主体であり $[a]_i$ や $[b]_i$ は $a$ や $b$ のデータのシェアである．

**Mul**：$a$，$b$ のシェアから $ab$ のシェアを作成

入力：$P_i([a]_i, [b]_i)$
出力：$P_i([ab]_i)$
(1) $P_0$ は $r_1, r_2, c_0 \in \mathbf{Z}/m\mathbf{Z}$ をランダムに選択してから，$c_1 := (a_0+a_1)(b_0+b_1) - r_1 - r_2 - c_0$ を計算して $P_1$，$P_2$ にそれぞれ $(r_1, c_1)$，$(r_2, c_0)$ を送信し，$[ab]_0 := (c_0, c_1)$ とする．
(2) $P_1$，$P_2$ はそれぞれ $y := a_1b_2 + a_2b_1 + r_1$，$z := a_2b_0 + a_0b_2 + r_2$ を計算して $P_2$，$P_1$ に送信する．
(3) $P_1$，$P_2$ は $c_2 := y + z + a_2b_2$ を計算してそれぞれ $[ab]_1 := (c_1, c_2)$，$[ab]_2 := (c_2, c_0)$ とする．

この一連の処理が行われると各主体には $ab = c_0 + c_1 + c_2$ となる $c$ のシェアが作成される．

## 3. 提案方式

提案方式は文献 [1] を拡張し主体数 $n$ と復元に必要な主体数 $k$ を 4 章で示す条件下で成立する秘匿関数計算である．乗算以外の各処理は容易に拡張が可能であるため，本稿では，乗算の説明の際に必要な秘密分散・復元，容易に拡張が可能であることを示すために加減算・定数倍，そして乗算の処理手順のみを示す．

### 3.1 秘密分散・復元

入力 $a \in \mathbf{Z}/m\mathbf{Z}$ の秘密分散 **Share** は，主体 $P_i$ が持つシェアを $[a]_i = (a_i, \ldots, a_{n-k+i})$ とする．また，シェアの中の添字 $i, \ldots, n-k+1$ は主体数 $n$ の剰余をとったものとする．このシェアに含まれるデータの個数は $n-k+1$ 個であり，主体数 $n$ と復元に必要な主体数 $k$ との関係より導かれる．たとえば，$k=3$，$n=5$ の場合に $i=4$ であれば $[a]_4 = (a_4, a_0, a_1)$ である．また，シェア内の最後の 2 つの値 $a_0$，$a_1$ は $a_5$，$a_6$ ではなく $n=5$ の剰余をとったものとなっている．

**Share**：k-out-of-n 秘密分散

入力：$a \in \mathbf{Z}/m\mathbf{Z}$
出力：$P_i([a]_i)$
(1) $a_0, \ldots, a_{n-1} \in \mathbf{Z}/m\mathbf{Z}$ をランダムに選択する．
(2) $a_{n-1} := a - \sum_{i=0}^{n-2} a_i$ を計算する．
(3) $i = 0, \ldots, n-1$ について，$[a]_i := (a_i, \ldots, a_{n-k+i})$ として $P_i$ に送信する．

$a = \sum_{i=0}^{n-1} a_i$ より，任意の異なる $k$ 個の $[a]_i$ から $a$ を復元することができる．また明らかに $k$ 個未満の $[a]_i$ からは $a$ は復元できない．すなわち，k-out-of-n 秘密分散の性質を有している．また，エラー検出を含む復元の手続き **Dec** は次のようになる．

**Dec**：k-out-of-n 秘密分散のエラー検出を含む復元

入力：$P_i([a]_i)$
出力：$a$ or $\perp$
(1) $P_i$ は $(\alpha_i, \ldots, \alpha_{n-k+i}) := (a_i, \ldots, a_{n-k+i})$ を開示する．
(2) $i = 0, \ldots, n-1$ において各主体 $P_i$ の $\alpha_i$ に対応する $a_i$ について $\alpha_i \neq a_i$ となる $\alpha_i$，$a_i$ が 1 つでも存在すれば，異常を示す $\perp$ を返して終了する．
(3) $a = \sum_{i=0}^{n-1} a_i$ を計算する．

### 3.2 加減算・定数倍

加減算や定数倍の手続きは以下の **Add/Sub**，**CoMul** のようになる．

**Add/Sub**：$a$，$b$ のシェアから $a \pm b$ のシェアを作成

入力：$P_i([a]_i, [b]_i)$
出力：$P_i([a \pm b]_i)$
- $P_i$ は $[a \pm b]_i = (a_i \pm b_i, \ldots, a_{n-k+i} \pm b_{n-k+i})$ を計算する．

**CoMul**：$a$ のシェア，定数 $c$ から $ca$ のシェアを作成

入力：$P_i([a]_i), c$
出力：$P_i([ca]_i)$
- $P_i$ は $[ca]_i = (ca_i, \ldots, ca_{n-k+i})$ を計算する．

### 3.3 乗算

乗算処理は文献 [1] と同様に各主体間の協調計算が必要となる．ここで重要な点は文献 [1] が 2-out-of-3 に特化した方式であったのに対して本方式では k-out-of-n に一般化したという点である．この一般化を行ったことにより，各々

の $k$，$n$ の組合せで各主体が行う処理内容が変わってくる．

このため乗算の手続きは以下のような 2 段階に分けて行われる．

(1) 各主体が行う処理の分担を決める．

(2) 実際に乗算処理を行う．

なお，最初の各主体が行う処理の分担を決める手順は 1 度だけ行えば以降の乗算処理の際には必要ない．

以降，本節では準備として乗算処理で行うことを説明し，処理分担の作成，実際の乗算処理について述べる．

#### 3.3.1 準備

本方式での乗算処理は $a$，$b$ の 2 つの値を乗じた $c = ab$ という値を作る処理である．また $a$，$b$ は **Share** により $a = \sum_{i=0}^{n-1} a_i$，$b = \sum_{i=0}^{n-1} b_i$ となるように分割された後に各主体にシェア $[a]_i$，$[b]_i$ として分散されている．そして $c$ も同様に **Mul** の手続きが行われると $c = \sum_{i=0}^{n-1} c_i$ となるように作成された後に各主体にシェア $[c]_i = [ab]_i$ として分散される．

つまり **Mul** の手続きにより最終的に各主体が得る値は式 (1) のようになる．

$$\sum_{i=0}^{n-1} c_i = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j \tag{1}$$

これより **Mul** の手続きでは各主体が協調して $\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j$ の計 $n^2$ 個の項を計算することが必要となることが分かる．

次に，どのように協調計算を行うかについて述べる．ここで 2 章の最後に説明した文献 [1] の乗算の手続きを確認する．

**Mul**：$a$，$b$ から $ab$ を作成（2-out-of-3）

(1) $P_0$ は複数の値をランダムに選択して，自身のシェアの値を用いて $c_0$，$c_1$ を計算する．その後，$P_1$，$P_2$ に値を送信して $[ab]_0 := (c_0, c_1)$ とする．

(2) $P_1$，$P_2$ は $P_0$ から送信された値と，自身のシェアの値を用いて $y$，$z$ を計算する．その後，$P_2$，$P_1$ で値を送信し合う．

(3) $P_1$, $P_2$ は受信した値と自身の計算した値と $a_2b_2$ を用いて $c_2$ を計算する．その後，$[ab]_1 := (c_1, c_2)$，$[ab]_2 := (c_2, c_0)$ とする．

ここで重要になるのが，1) $P_0$ 以外の各主体は自身が持っているシェアの値を用いて $a_ib_j$（$i \neq j$）の計算を行うこと，2) $a_ib_i$ と $a$，$b$ 双方の添字が一致する項は最後に $c_i$ を計算するときに用いられることである．つまり，どの主体がどの $a_ib_j$（$i \neq j$）の各項を計算するのかを決定することが必要である．

次にどの主体がどの $a_ib_j$（$i \neq j$）の項を計算するかを決定する手順を述べる．

#### 3.3.2 各主体の処理分担の決定

ここではまずはじめに例として $n = 5$ の場合にどのように各項を計算する主体を決めればよいかを**図 1** に示す．

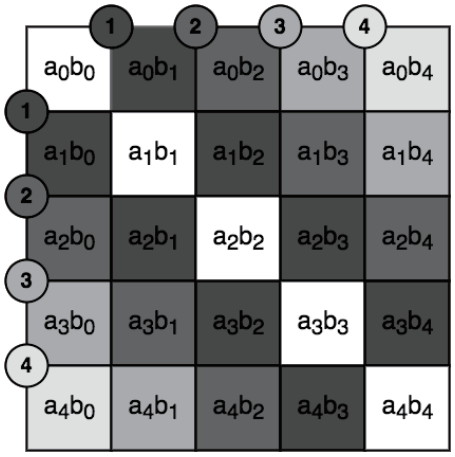



**図 1** $n = 5$ の場合の Algorithm 1 の動作
**Fig. 1** The action of Algorithm 1 when $n = 5$.

**Algorithm 1**
各主体への処理分担を行う手順

```
if n % 2 == 0 then
    max_party = n - 2
else
    max_party = n - 1
end if
for start_col=1; start_col<n; start_col++ do
    for row=0; row<(n-start_col); row++ do
        row = row, col = row + start_col
        for i=0; i<max_party; i++ do
            if ((a_row, b_col, b_col, a_row) ∈ P_i([a]_i, [b]_i)) then
                a_row b_col + b_col a_row は P_i が計算する
                break
            end if
        end for
    end for
end for
```

丸付きの数字および図の中の濃淡によって，**Mul** 内の処理で $a_ib_j$ の組合せを各主体に割り当てていく順序を示している．この場合，1 回目：①，2 回目：②，3 回目：③，4 回目：④，となる．

図 1 の場合は，$a_0b_1, a_1b_0, a_1b_2, a_2b_1, a_2b_3, a_3b_2, a_3b_4, a_4b_3, a_0b_2, a_2b_0, \ldots, a_1b_4, a_4b_1, a_0b_4, a_4b_0$ の順番で各組合せを各主体に割り当てて行く．

図 1 中で注意すべきことは，ある主体が $a_ib_j$ となる項を計算可能であるということは $a_jb_i$ も計算可能であるということである．つまり各主体への各項の処理分担を行う際には全体の半分の項を確認すればよいといえる．この点をふまえて Algorithm 1 を用いると各主体に処理を割り振ることが可能となる．Algorithm 1 中に出てくる値の説明は，スペースの都合で次ページに記載した．

Algorithm 1 の処理手順は $(a_0b_1)$ 要素から右斜下に各 $a_ib_j$ の要素が計算できる主体 $P_i$ があるかどうか確認していく．

Algorithm 1 中に出てくる値

**n**：主体の個数
**max_party**：探索する主体の最大値
**start_col**：そのループ内で一番最初に対象とする列
**row**：対象とする行
**col**：対象とする列
$a_{row}$，$b_{col}$，$b_{col}$，$a_{row}$：row，col は図 1 のなかの行と列の番号を示す．
$P_i$：計算主体

Algorithm 1 の処理が終わると各主体に計算しなければならない項 $a_ib_j$（$i \neq j$）が，すべて割り振られることが期待される．

しかし，たとえば 4-out-of-5 のような場合には，$a_0b_2$，$a_1b_3$，$a_2b_4$，$a_0b_3$，$a_1b_4$ のように元々主体に分散されていない項が出現する．このようにいずれかの主体にも，割り振られない処理すべき項が発生すると，乗算処理は行えない．この点については，3.3.3 項で協調計算について，および 4 章で一般化が成立する条件として説明を行う．

### 3.3.3 乗算の手続き

実際の乗算の手続き **Mul** は各主体どうしが協調計算を行うことで実行される．また，各項の処理内容は，Algorithm 1 による，各主体への処理の割り振りによって決められている．

**Mul**：$a$，$b$ のシェアから $ab$ のシェアを作成

入力：$P_i([a]_i, [b]_i)$
出力：$P_i([ab]_i)$
(1) $P_0$ の操作
　(a) $r_1, \ldots, r_{n-1}, c_0, \ldots, c_{n-k-1} \in \mathbf{Z}/m\mathbf{Z}$ をランダムに選択する．
　(b) $c_{n-k} := a_{row}b_{col} + a_{col}b_{row} + \cdots + \sum_{i=0}^{n-k} a_ib_i - \sum_{i=1}^{n-1} r_i - \sum_{i=0}^{n-k-1} c_i$ を計算する．
　(c) $(r_i, c_i, \ldots, c_{n-k})$ を必要とするパーティに送信する．
　(d) $[ab]_0 := (c_0, \ldots, c_{n-k})$ とする．
(2) $P_i$ の操作
　(a) $P_i$ は $[a]_i$，$[b]_i$，$r_i$ を用いて $S_i := a_{row}b_{col} + a_{col}b_{row} + \cdots + r_i$ を計算する．
　(b) $P_{2i-1}$，$P_{2i}$ どうしで $S_i$ を送信し合う．
　(c) $P_{2i-1}$，$P_{2i}$ は $c_{n-k+i} := S_{2i-1} + S_{2i} + a_{n-k+i}b_{n-k+i}$ を計算する．
　(d) $c_{n-k+i}$ を必要とするパーティに送信する．
　(e) $[ab]_i := (c_i, \ldots, c_{n-k+i})$ とする．

シェアの正当性について式 (2)，(3) に示す．

$$\begin{aligned}\sum_{i=0}^{n-1} c_i &= a_{row}b_{col} + a_{col}b_{row} + \cdots + \sum_{i=0}^{n-k} a_ib_i - \sum_{i=1}^{n-1} r_i \\ &\quad + \sum_{i=1}^{\frac{n-1}{2}} (S_{2i-1} + S_{2i} + a_{n-k+i}b_{n-k+i}) \\ &= a_{row}b_{col} + a_{col}b_{row} + \cdots + \sum_{i=0}^{n-k} a_ib_i \\ &\quad + \sum_{i=1}^{\frac{n-1}{2}} (a_{row}b_{col} + a_{col}b_{row} + \cdots + a_{n-k+i}b_{n-k+i})\end{aligned} \tag{2}$$

ここで先ほどの Algorithm 1 により，式中の $a_{row}b_{col} + a_{col}b_{row}$ には計算しなければならない $a_ib_j$（$i \neq j$）の項が含まれている．したがって式 (3)

$$\sum_{i=0}^{n-1} c_i = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_j \tag{3}$$

に示すように，$c_0, \ldots, c_{n-1}$ の任意の 2 つの元は $\mathbf{Z}/m\mathbf{Z}$ 上の互いに独立な乱数と見なせるため，$c_0, \ldots, c_{n-1}$ は正しく $ab$ のシェアとなっていることが分かる．

### 3.4 論理回路演算

論理回路演算は，否定，論理積，論理和，排他的論理和の 4 種類である．また，論理回路演算でも，**Mul** のように協調計算が必要であり，各主体が持っているシェアのみでは演算できない．否定の論理回路演算は以下のようになる．

**Not**：$a \in \{0, 1\}$ のシェアから $\bar{a}$ のシェアを作成

入力：$P_i([a]_i)$
出力：$P_i([\bar{a}]_i)$
- 各パーティは $\bar{a_0} := 1 - a_0$，$\bar{a_i} := a_i$ を計算し $[\bar{a}]_i$ とする．

また，論理積，論理和，排他的論理和は $a, b \in \{0, 1\}$ について，

- $a \wedge b = ab$
- $a \vee b = a + b - ab$
- $a \oplus b = a + b - 2ab$

となるため，$a$，$b$ のシェアについて加減算，定数倍，乗算の各演算を組み合わせればよい．具体的には以下のようになる．

**And**：$a$，$b$ のシェアから $a \wedge b$ のシェアを作成

入力：$P_i([a]_i, [b]_i)$
出力：$P_i([a \wedge b]_i)$
- $P_i$ は **Mul**$([a]_i, [b]_i)$ を実行する．

**Or**：$a$，$b$ のシェアから $a \vee b$ のシェアを作成

入力：$P_i([a]_i, [b]_i)$
出力：$P_i([a \vee b]_i)$
- $P_i$ は $\mathbf{Sub}(\mathbf{Add}([a]_i, [b]_i), \mathbf{Mul}([a]_i, [b]_i))$ を実行する．

**Xor**：$a$，$b$ のシェアから $a \oplus b$ のシェアを作成

入力：$P_i([a]_i, [b]_i)$
出力：$P_i([a \oplus b]_i)$
- $P_i$ は $\mathbf{Sub}(\mathbf{Add}([a]_i, [b]_i), \mathbf{CoMul}(\mathbf{Mul}([a]_i, [b]_i), 2))$ を実行する．

## 4. 一般解の成立条件

### 4.1 準備

まずはじめに一般解を求める際に重要であるのは **Mul** が実行可能であるかである．このことを以下に示す．

**命題 4.1.** 3 章で示したプロトコルのうち **Mul** を必要としないものは k-out-of-n の条件を満たすならば実行可能である．

*Proof.* **Share**，**Dec** は秘密分散・復元プロトコルであり k-out-of-n の条件を満たすならば実行可能である．また，**Mul** を必要としないプロトコルは各パーティ内部で処理が完結しているため同様に k-out-of-n の条件を満たすならば実行可能である． □（命題 4.1）

そして **Mul** が実行可能であるかどうかは以下によって定義される．

**定義 4.1. Mul** が実行可能であるということは，すべての計算が必要な項が各主体間の協調計算により計算可能であることである．

また **Share** の手順より以下のことが定義される．

**定義 4.2. Share** によって分散された各計算主体のシェアは $n-k+1$ 個のデータを持つ．

ここからの証明を行うには乗算の際に出てくる項 $a_ib_j$ が重要となってくる．しかしこれは $a_i$，$b_j$ と 2 つの値が関わっており以下の証明が煩雑になる．そこで本稿では，$a_i$，$b_j$ の添字 $i$，$j$ に着目し，これの Lee 距離を求めることにより 1 つの値にまとめ簡略化を行った．$a_i$，$b_j$ の添字 $i$，$j$ の Lee 距離は以下によって定義される．

**定義 4.3.** $a_i$，$b_j$ の 2 つの値の添字 $i$，$j$ の Lee 距離 $Lee(i,j)$ は主体数 $n$ を用いて，

$$Lee(i,j) = \min(|i-j|, n-|i-j|) \tag{4}$$

である．

ここでなぜ Lee 距離を用いると簡略化が行えるかについて**図 2**，**図 3** を用いて説明する．図 2，図 3 の格子内の上部は計算すべき項 $a_ib_j$ であり，下部は添字 $i$，$j$ の Lee 距離である．重要な点として，図 2 の色が塗られていない，各主体で計算が不可能な部分に着目するとすべて Lee 距離が 2 である．

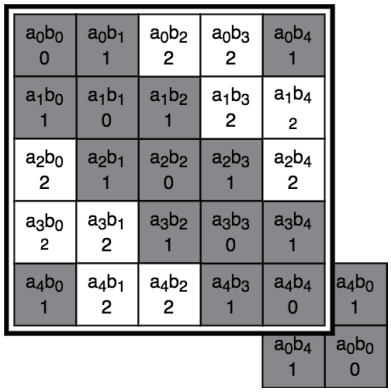



**図 2** 4-out-of-5 のときの計算可能項と計算必要項の関係
**Fig. 2** The relationship between calculable terms and must-calculate terms at 4-out-of-5.

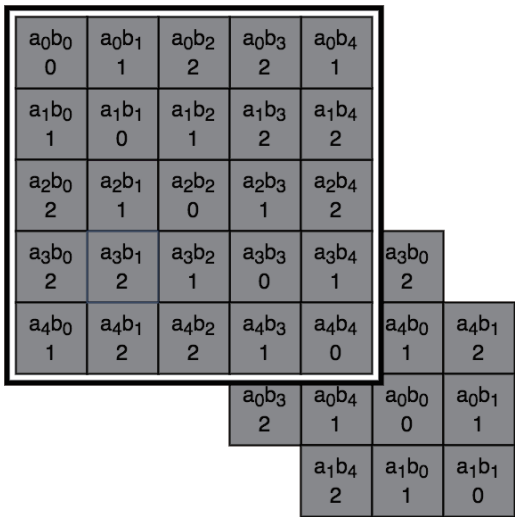



**図 3** 3-out-of-5 のときの計算可能項と計算必要項の関係
**Fig. 3** The relationship between calculable terms and must-calculate terms at 3-out-of-5.

そして各主体で計算できる項 $a_ib_j$ の最大の Lee 距離は以下のようになる．

**命題 4.2.** 各主体で計算できる項 $a_ib_j$ の最大の Lee 距離は $n-k$ である．

*Proof.* まずはじめに具体例として 2-out-of-3 と 2-out-of-4 の場合を説明した後に，j-out-of-i と j-out-of-(i+1) の場合を説明し数学的帰納法により示す．

(1) 2-out-of-3 の場合
各主体は 2 個のデータを持つシェアを保持している．このときの最大の Lee 距離は 1 である．

(2) 2-out-of-4 の場合
各主体は 3 つのデータを持つシェアを保持している．このときの最大の Lee 距離は 2 である．

(3) 2-out-of-i の場合
各主体は $i-2+1$ 個のデータを持つシェアを保持している．このときの最大の Lee 距離は $i-1$ である．
(4) 2-out-of-(i+1) の場合
各主体は $(i+1)-2+1$ 個のデータを持つシェアを保持している．このときの最大の Lee 距離は $i$ である．
(5) j-out-of-i の場合
各主体は $i-j+1$ 個のデータを持つシェアを保持している．このときの最大の Lee 距離は $i-j$ である．
(6) j-out-of-(i+1) の場合
各主体は $(i+1)-j+1$ 個のデータを持つシェアを保持している．このときの最大の Lee 距離は $(i+1)-j$ である．
(3)，(4) より各主体で計算できる項 $a_ib_j$ の最大の Lee 距離は $n-k$ である． □（命題 4.2）

### 4.2 k-out-of-n における一般解

**定理 4.1.** k-out-of-n の条件を満たし，$n$ が偶数の場合なら $n \geq 2k$，奇数の場合なら $n \geq 2k-1$ を満たすならば 3 章で示したプロトコルは実行可能である．

*Proof.* 本定理は偶数の場合と奇数の場合に分かれているため，各々の場合に証明を行っていく．

**命題 4.3.** $n$ が偶数かつ k-out-of-n の条件を満たすならば $n \geq 2k$ ならば **Mul** は実行可能である．

*Proof.* （命題）
定義 4.1 により計算しなければならない項の最大の Lee 距離は分かっているため $n$ が偶数の場合に計算しなければならない項の Lee 距離の最大値を導く．

**補題 4.1.** $n$ が偶数の場合には計算しなければならない項の Lee 距離の最大値は $n/2$ である．

*Proof.* （補題）
まずはじめに具体例として $n=4$，$n=6$ の場合を説明した後に，$n=i$，$n=i+2$ の場合を説明し数学的帰納法により示す．
(1) $n=4$ の場合計算しなければならない項の Lee 距離の最大値は 2 である．
(2) $n=6$ の場合計算しなければならない項の Lee 距離の最大値は 3 である．
(3) $n=i$ の場合計算しなければならない項の Lee 距離の最大値は $i/2$ である．
(4) $n=i+2$ の場合計算しなければならない項の Lee 距離の最大値は $i/2+1$ である．
(3)，(4) より $n$ が偶数の場合には計算しなければならない項の Lee 距離の最大値は $n/2$ である． □（補題 4.1）

定義 4.1，命題 4.2，補題 4.1 より $n$ が偶数の場合に **Mul** が実行可能な条件は以下のように求まる．

$$\begin{aligned} n/2 &\leq n-k \\ n &\leq 2n-2k \\ -n &\leq -2k \\ n &\geq 2k \end{aligned} \tag{5}$$

□（命題 4.3）

**命題 4.4.** $n$ が奇数かつ k-out-of-n の条件を満たすならば $n \geq 2k-1$ ならば **Mul** は実行可能である．

*Proof.* （命題）
定義 4.1 により計算しなければならない項の最大の Lee 距離は分かっているため $n$ が奇数の場合に計算しなければならない項の Lee 距離の最大値を導く．

**補題 4.2.** $n$ が奇数の場合には計算しなければならない項の Lee 距離の最大値は $(n-1)/2$ である．

*Proof.* （補題）
まずはじめに具体例として $n=3$，$n=5$ の場合を説明した後に，$n=i$，$n=i+2$ の場合を説明し数学的帰納法により示す．
(1) $n=3$ の場合計算しなければならない項の Lee 距離の最大値は 1 である．
(2) $n=5$ の場合計算しなければならない項の Lee 距離の最大値は 2 である．
(3) $n=i$ の場合計算しなければならない項の Lee 距離の最大値は $(i-1)/2$ である．
(4) $n=i+2$ の場合計算しなければならない項の Lee 距離の最大値は $(i-1)/2+1$ である．
(3)，(4) より $n$ が偶数の場合には計算しなければならない項の Lee 距離の最大値は $(n-1)/2$ である．
□（補題 4.2）

定義 4.1，命題 4.2，補題 4.1 より $n$ が偶数の場合に **Mul** が実行可能な条件は以下のように求まる．

$$\begin{aligned} \frac{n-1}{2} &\leq n-k \\ n-1 &\leq 2n-2k \\ -n &\leq -2k+1 \\ n &\geq 2k-1 \end{aligned} \tag{6}$$

□（命題 4.4）

命題 4.3，4.4 より k-out-of-n の条件を満たし，$n$ が偶数の場合なら $n \geq 2k$，奇数の場合なら $n \geq 2k-1$ を満たすならば 3 章で示したプロトコルは実行可能であるという定理 4.1 が成立する．

□（定理 4.1）

また実用上の $n$，$k$ の関係は，
(1) $n = 3$ の場合 $2 \geq k$
(2) $n = 4$ の場合 $2 \geq k$
(3) 3 以上の奇数 $even$ が $n$ の場合 $\frac{even+1}{2} \geq k$
(4) $even + 1$ が $n$ の場合 $\frac{even+1}{2} \geq k$
となる．つまり命題 4.4 の場合，$n \geq 2k - 1$ の場合を満たすならば 3 章で示したプロトコルは実行可能である．

## 5. おわりに

本稿では “軽量 3 パーティ秘匿関数計算” [1] を軽量 N パーティ秘匿関数計算として拡張し，一般化を行った．一般化した結果，主体数 $n$ とデータの復元に必要な主体数 $k$ について，

$$\begin{cases} n \geq 2k & (n\ is\ even) \\ n \geq 2k-1 & (n\ is\ odd) \end{cases}$$

の条件を満たす場合であることを示した．

今後の課題は，提案方式の安全性について拡張以前と同等であること，実際に本手法を実装し性能評価を行っていく点があげられる．

また，$n$ を大きくしたときに $k$ を大きくするならば安全性が，小さくするならば可用性が高くなる点を定量的に示す．

そして，アプリケーションやサーバ，IoT デバイスといった $n$ にあたる計算主体のスケーラビリティについて評価を行う．

**謝辞** 本稿の執筆においては，中央大学白鳥研ゼミメンバである，荻野正様（明星大学教授），北上眞二様（福井工業大学教授），浦野義頼様（元早稲田大学教授），松山泰男様（早稲田大学名誉教授）の皆様と議論させていただきました．ここに記して深く感謝いたします．

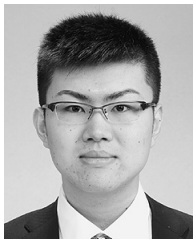
### 参考文献

[1] 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).
[2] 総務省：情報セキュリティ対策の必要性，入手先 ⟨http://www.soumu.go.jp/main_sosiki/joho_tsusin/security/business/executive/01.html⟩（参照 2017-10-28）.
[3] NEC：機密情報の漏えいを強固に防止する秘密計算の高速化手法を開発，入手先 ⟨http://jpn.nec.com/press/201612/20161215_02.html⟩（参照 2017-10-28）.
[4] NTT：秘密計算システム，入手先 ⟨http://www.ntt.co.jp/RD/active/201602/jp/pf/pf013.html⟩（参照 2017-10-28）.
[5] Lu, W., Kawasaki, S. and Sakuma, J.: Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data, *IACR Cryptology ePrint Archive*, Vol.2016, p.1163 (2016).
[6] 滝雄太郎，藤田 茂，宮西洋太郎，白鳥則郎ほか：k out of n 秘密計算プロトコルの一考察，研究報告マルチメディア通信と分散処理（DPS），Vol.2016, No.5, pp.1–7 (2016).
[7] Deza, M.M. and Daza, E.: *Encyclopedia of Distances* (3rd ed.), Springer (2014).

### 推薦文

DICOMO2017 の発表論文の中で特に評価が高かったため．この研究は，軽量 3 パーティ秘匿関数計算を軽量 N パーティ秘匿関数計算として拡張し，一般解を導出している．このことにより，たとえば分散型セキュアストレージとしての応用ができることが示されている．

（マルチメディア通信と分散処理研究会主査 重野 寛）



**滝 雄太郎** （正会員）

2017 年千葉工業大学大学院情報科学研究科博士課程前期修了．修士（情報科学）．在学中，秘密分散，秘密計算の研究に従事．現在，大日本印刷株式会社勤務．

**藤田 茂** （正会員）

1997 年千葉工業大学大学院工学研究科博士課程後期退学，1998 年博士（工学），1997 年千葉工業大学工学部助手，2012 年同学情報科学部教授，現在に至る．秘密分散，秘密計算，IoT セキュリティ，エージェントシステムとその応用に興味を持つ．電子情報通信学会，人工知能学会，IEEE 各会員．本会シニア会員．

**宮西 洋太郎** （正会員）

1968 年神戸大学大学院工学研究科電気工学専攻修了（工学修士）．1968～2000 年三菱電機株式会社勤務．1997 年静岡大学大学院電子科学研究科電子応用工学専攻修了（工学博士）．2000～2004 年公立はこだて未来大学システム情報科学部教授．2004～2009 年宮城大学事業構想学部教授，2009～2011 年同大学客員教授，2009 年 5 月から 2011 年 3 月まで仙台ソフトウェアセンター嘱託を経て，現在，株式会社アイエスイーエム代表取締役．セキュリティとその評価に興味を持つ．電子情報通信学会，計測自動制御学会，システム制御情報学会各会員．本会シニア会員．

**白鳥 則郎** （名誉会員）

1977年東北大学大学院工学研究科修了（工学博士），1977年4月東北大学電気通信研究所助手，1984年11月東北大学電気通信研究所助教授，1990年4月東北大学工学部情報工学科教授，1993年4月東北大学電気通信研究所教授，1997年7月～8月UCLA客員教授，2010年4月東北大学名誉教授，2010年4月～2013年3月東北大学客員教授，2010年4月～2013年3月公立はこだて未来大学理事，2012年4月～2017年3月早稲田大学大学院国際情報通信研究科教授，2017年4月～現在，中央大学研究開発機構教授．1996年度～1997年度本会理事，2004年度～2005年度本会副会長，2009年度～2010年度本会会長，本会25周年記念論文賞，平成8年度本会論文賞，1999年度本会フェロー，2007年度本会功績賞 2010年文部科学省・平成21年度文部科学大臣表彰科学技術賞「研究部門」，2011年電子情報通信学会功績賞，2012年電子情報通信学会名誉員，2013年日本工学会フェロー，2016年度「情報化促進貢献個人表彰」文部科学大臣賞，2017年IEEE Life Fellow．