軽量Nパーティ秘匿関数計算の一般解と情報銀行の分散型セキュアストレージへの応用

滝 雄太郎[1]　藤田 茂[2]　宮西 洋太郎[3]　白鳥 則郎[4]

概要：本稿では 3 主体の協調計算による秘匿関数計算プロトコルである “軽量 3 パーティ秘匿関数計算” [1] を $n > 3$ の場合の軽量 N パーティ秘匿関数計算として拡張，一般解を導いた結果を示す．また，提案方式の一般解を導く際には Lee Distance を効果的に導入してる．本稿の結果より主体数 n とデータの復元に必要な主体数 k を一般解の条件下で自由に選択することが可能なためシステム運用者は耐障害性・対攻撃性を自由に選択することが可能となる．そして，本稿の具体的な応用例として情報銀行の構築技術の一つとして重要である分散型セキュアストレージの構成を述べる．

1. はじめに

セキュリティ対策やプライバシー保護等，情報の秘密を守ることが，情報システムを構築する上で，重要な課題である．この課題を解決する一つの手法として，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，分散した情報を一か所に集めて処理すると，その一か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．

この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．

秘匿関数計算 (k-out-of-n) において，n=3 の特別の場合の解が与えられている [1]．しかし一般の n>3 については，未解決の問題として知られている．

我々は，秘匿関数計算 (k-out-of-n) の一般化を検討してきた [2]．

本稿では, まず未解決の n>3 に対する一般解を与える．ここで, Lee Distance[3] を効果的に導入することにより, 一般解を得ることに成功している．この一般解により適用範囲が格段に広がり，セキュリティ保持に大きな貢献が期待できる．

得られた一般解の応用として安心・安全が大きな課題となっている情報銀行，クラウドサービス，IoT などが考えられる．ここでは応用例として，情報銀行 [4] を構築するための，分散型セキュアストレージの構成を示す．

情報銀行は，政府 [5] で検討され，2017 年度には実証実験が始まる情報活用基盤である．一方で情報銀行に対しては，情報漏洩への危惧や個人を特定されるのではないかという不安感が大きい．これは情報銀行に限った話ではなく，現在でもサーバからの情報漏洩が常に危惧されている．この危惧に対して，秘匿関数計算は，たとえ一定数のサーバからの情報漏洩があったとしても，秘密が保たれることから，対策の実現技術として魅力的である．

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 5 章では提案方式の応用例の一つとして，情報銀行などにおいて重要技術となっている分散型セキュアストレージの構成を述べる．6 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．

文献 [1] の秘匿関数計算は全体の主体数を 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とした場合には 2-out-of-3 のものである．

また，分散したデータを復元すること無く加減算・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

[1] 千葉工業大学大学院情報科学研究科
Graduate School of Chiba Institute of Technology 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, JAPAN
[3] 株式会社アイエスイーエム
ISEM, Inc, JAPAN
[4] 中央大学研究開発機構
Research and Development Initiative, Chuo University, JAPAN

る．以下に一例として乗算処理の手順を示す．なお，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 $ab = c_1 + c_2 + c_3$ となるような 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 章で示す条件下で動作する秘匿関数計算である．また，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．なお，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．なお，記述の簡略化のため a_i の添字の i は全て n で割った余りと見なす．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．

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 $(\alpha_{i,i}, \ldots, \alpha_{i,n-k+i}) := (a_i, \ldots, a_{n-k+i})$ を開示する．
(2) $i = 0, \ldots, n-1$ について $\alpha_{i,i} \neq \{\alpha_{i+1,i}, \ldots, \alpha_{i+n-k+1,i}\}$ となるような $\alpha_{i,i}, \ldots, \alpha_{i+n-k+1,i}$ が存在すれば，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事により各々の k, n の組み合わ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二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一度だけ行えば以降の乗算処理の際には必要無い．

以降，本節では準備として乗算処理で行いたいことを説明し処理の作成，実際の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 の二値を乗じた $c = ab$ 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次のようになる．

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

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わかる．

では具体的には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 $a_2 b_2$ 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 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 $a_i b_j (i \neq j)$ の計算を行う事，$a_i b_i$ と a, b 双方の添字が一致する項は最後に c_i を計算する時に用いられる事である．つまり，どの主体がどの $a_i b_j (i \neq j)$ の各項を計算するのか決定する事が必要である．

以上のことより次にどの主体がどの $a_i b_j (i \neq j)$ の項を計算するか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良いかを図 1 に示す．

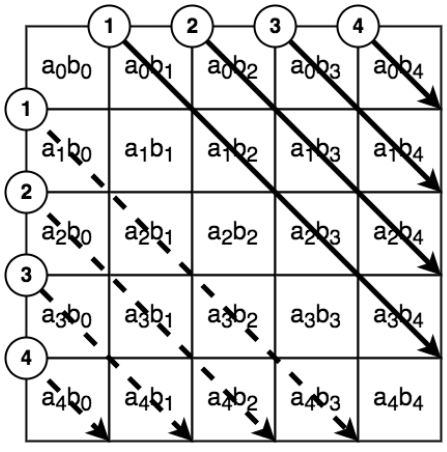

図 1 $n = 5$ の場合の Algorithm1 の動作

図 1 中で重要なのがある主体が $a_i b_j$ となる項を計算可能であるという事は $a_j b_i$ も計算可能であるという事である．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良いと言える．この点を踏まえて Algorithm1 を用いると各主体に処理を割り振る事が可能となる．

Algorithm1 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
$a_{row}, b_{col}, b_{col}, a_{row}$ **：** 処理として割り振られる項
P_i **：** 計算主体

Algorithm1 の処理手順は $(a_0 b_1)$ 要素から右斜下に各 $a_i b_j$ 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ものとなっている．

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
```

Algorithm1 の処理が終わると各主体に計算しなければならない項 $a_i b_j (i \neq j)$ が全て割り振られる事が期待される．しかし例えば 4-out-of-5 のような場合には $a_0 b_2$ や $a_1 b_3$ 等の割り振られない項が出てきてしまう．このようないずれかの主体に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 4 章で一般解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** は各主体が協調計算を行い実行される．また，各項の処理内容は先ほど求めた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} 同士で 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，

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

ここで先程の Algorithm1 が問題なく行われていれば式中の $a_{row}b_{col} + a_{col}b_{row}$ 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，

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

となり，$c_0, \ldots, c_{n-1}$ 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みなせるため $c_0, \ldots, c_{n-1}$ は正しく ab のシェアとなっていることがわかる．

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事は，全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事である．

また **Share** の手順より以下の事が定義される．

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 と二つの値が関わっており工夫をしないと以下の証明が煩雑になる．そこで a_i, b_j の添字 i, j に着目しこれの Lee Distance を求める事により一つの値にまとめ簡略化を行えるようにする．なお a_i, b_j の添字 i, j の Lee Distance は以下によって定義される．

定義 4.3. a_i, b_j の二つの値の添字 i, j の Lee Distance $Lee(i, j)$ は主体数 n を用いて，

$$Lee(i, j) = \min(|i-j|, n-|i-j|) \tag{2}$$

である．

ここで何故 Lee Distance を用いると簡略化が行えるかについて図 2,3 を用いて説明する．図 2,3 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 Distance 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全て Lee Distance が 2 である．

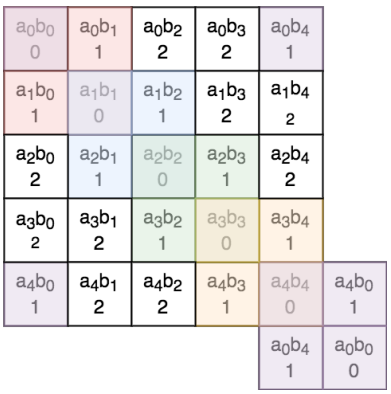

図 2 4-out-of-5 の時の計算可能項と計算必要項の関係

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

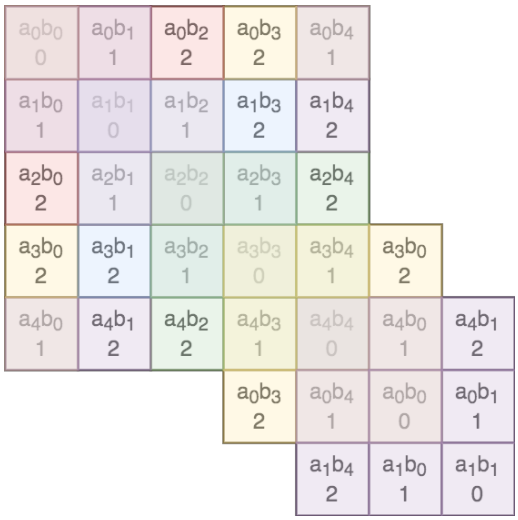

図 3 3-out-of-5 の時の計算可能項と計算必要項の関係

命題 4.2. 各主体で計算できる項 a_ib_j の最大の Lee Distance 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．この時の最大の Lee Distance 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．この時の最大の Lee Distance は 2 である．

(3) 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．この時の最大の Lee Distance は $i-j$ である．

(4) 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．この時の最大の Lee Distance 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 Distance 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 Distance はわかっているため n が偶数の場合に計算しなければならない項の Lee Distance 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 Distance 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 Distance 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 Distance 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 Distance 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 Distance 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 Distance 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．

$$\begin{aligned} n/2 &\leq n-k \\ n &\leq 2n-2k \\ -n &\leq -2k \\ n &\geq 2k \end{aligned} \tag{3}$$

□（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 Distance はわかっているため n が奇数の場合に計算しなければならない項の Lee Distance 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 Distance 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 Distance 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 Distance 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 Distance 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

Distance 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 Distance 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．

$$\begin{aligned} \frac{n-1}{2} &\leq n-k \\ n-1 &\leq 2n-2k \\ -n &\leq -2k+1 \\ n &\geq 2k-1 \end{aligned} \tag{4}$$

□（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）

また実用上は命題 4.4 の場合，つまり $n \geq 2k-1$ 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. 本方式の応用

提案方式は，Internet of Things, IoT，エッジコンピューティング，Information Fllow of Things, IFoT，などに応用可能であり，本稿ではその一例として，情報銀行の分散型セキュアストレージについて述べる．

情報銀行では，取り扱うデータの内容から，高いセキュティが求められている．

本稿で示した軽量 N パーティ秘匿関数計算による秘密分散/秘密計算を用いることで，パーティ数を増やすことが出来，分散型セキュアストレージを構成することが出来る．図 4 に，2-out-of-5 として構成した分散ストレージと，その利用に基づく，利用者の購買履歴を秘密分散によって保存し，秘密計算によって，購買履歴を利用するシステムのイメージを示す．

図 4 では，利用者の持つ携帯端末上で動作するアプリから，直接分散ストレージに秘密分散されたシェアが，5 個のパーティに送信される．購買履歴の利用者は，2 つのパーティを使うことで，購買履歴を利用できる．また，5 つのパーティの一つがクラックされてデータが漏洩しても，利用者の購買履歴は復元不能であり，一つがクラックされデータが改竄された場合には，改竄の検出が検出可能である．

6. 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，一般解を導いた．提案方式では主体数 n とデータの復元に必要な主体数 k を一般解の条件下で示すことが可能となる．また，この一般解は Lee Distance を導入し偶数の場合には $n \geq 2k$，奇数の場合には $n \geq 2k-1$ の条件をみたす場合であることを示した．

また，本稿の具体的な応用例として情報銀行の構築技術の一つとして重要である分散型セキュアストレージの構成の一例を示した．

今後の課題としては，提案方式の安全性について拡張以前と同等であることを示す事や実際に本手法を実装し検証を行っていく点があげられる．

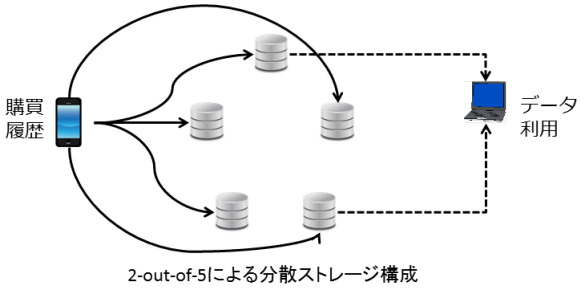

図 4 軽量 N パーティ秘匿関数計算の応用

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] 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).

[3] Deza, M. M. and Daza, E.: *Encyclopedia of Distances(3rd ed.)*, Springer (2014).

[4] 砂原秀樹，山内正人， 金杉洋，柴崎亮介：「情報銀行」構想とその技術的課題，*DICOMO*, Vol. 2014, pp. 1024–1026 (2014).

[5] データ流通環境整備検討会：AI, IoT 時代におけるデータ活用ワーキンググループ (第 9 回) 議事次第 (2017).