コインを用いる新たなマルチパーティ計算

駒野 雄一[1]　水木 敬明[2]

概要：カードベースプロトコルは、トランプなどのカードを利用して、お互いの入力を秘匿したまま AND 演算などのマルチパーティ計算を実現する手法である。計算機をブラックボックスとして利用することがないために、プロトコルの原理を直感的に理解しやすく、情報セキュリティの非専門家への教育にも有用である。しかし、カードベースプロトコルは、同一の模様のカード（例えば、$\boxed{\clubsuit}$, $\boxed{\heartsuit}$）をそれぞれ複数用意しなければならず、一般には、一組の市販のトランプのカードセットでは実現できなかった。本稿は、カードと同様になじみのあるコインを用い、新たなマルチパーティ計算の実現手法として**コインベースプロトコル**を提案する。カードと異なりコインは同一の模様の個体を準備しやすいため、より日常的に実行しやすくなることが期待される。一方、裏面を背にすれば情報を秘匿できるカードと異なり、コインは一方の面から他方の面の情報が漏れてしまう。そのため、情報を秘匿したままマルチパーティ計算を実行するためには、カードベースプロトコルとは異なる工夫が必要となる。本稿は、情報を秘匿しながら AND や OR などを演算するためのコインベースプロトコルの概念を提案し、具体的なプロトコルの構成も示す。

1. はじめに

カードベースプロトコルとは、二人以上のユーザがトランプなどのカードを用いて、マルチパーティ計算を行う技術である。具体的には、各ユーザは、自身の秘密情報に応じてカード列を作り、テーブルに背面を上にして並べる。そして、所定の規則に従って、カードを手で並べ替えたりすることで、お互いの秘密情報を漏らさずに、それらのビット積やビット和などを計算する。例えば、Mizuki らは 6 枚のカードを利用してビット積 (AND) を計算するプロトコル [1] や、出力の形式を工夫してカード枚数を 4 枚に削減した AND プロトコル [2] を提案している。

例えば、6 枚のカードを用いると、以下に示す AND プロトコル [1] を構成できる。このプロトコルでは、0 と 1 を、

$$\boxed{\clubsuit}\boxed{\heartsuit} = 0, \quad \boxed{\heartsuit}\boxed{\clubsuit} = 1$$

と符号化し、コミットメントと呼ぶ。このとき、AND を以下の手順で計算できる。

(1) 入力コミットメントの間に黒と赤のカードを 1 枚ずつ置き、裏返す（この 2 枚は 0 のコミットメントになる）。

$$\underbrace{\boxed{?}\boxed{?}}_{a}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{b} \quad \rightarrow \quad \underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{b}$$

(2) 次のように並べ替える。

$$\begin{array}{c}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\\ \searrow\!\!\searrow\quad\searrow\!\!\searrow\\ \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\end{array}$$

(3) カード列を半分に割り、それぞれの部分列内の並びは固定したまま、2 つの部分列をシャッフルする。

$$\left[\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\right] \quad \rightarrow \quad \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}$$

このランダム二等分割カットは人間が安全に実装できることが知られている [3]。

(4) 次のように並べ替える。

$$\begin{array}{c}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\\ \searrow\!\!\searrow\quad\searrow\!\!\searrow\\ \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\end{array}$$

(5) 左側の 2 枚のカードをめくり、その並びによって

$$\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{a\wedge b}\boxed{?}\boxed{?} \quad \text{または} \quad \boxed{\heartsuit}\boxed{\clubsuit}\boxed{?}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\wedge b}$$

のように、（秘匿した状態で）$a \wedge b$ の値を得る。

カードベースプロトコルは、計算機などのブラックボックスが不要であり、身近なカードを用いて容易に実装できる。さらに、原理を直感的に理解しやすく、秘密情報を漏らさずにプロトコルを実行できることが理解しやすい。そのため、本音を隠したままでの合意形成といった日常的な問題に対して安心してプロトコルを実行できるほか、セキュリティの非専門家への教育にも有用である [4]。

[1] （株）東芝 研究開発センター
Corporate R&D Center, Toshiba Corporation
[2] 東北大学 サイバーサイエンスセンター
Cyberscience Center, Tohoku University

図 1　コインベースプロトコルで使用するコインの例

1.1　本稿の貢献

本稿は、カードの代わりにコインを用いたコインベースプロトコルの概念を提案して、AND プロトコルなどを実際に構成する。コインとしては、例えば硬貨にように表面と裏面が異なるものを想定する（図 1）。硬貨には製造年が記載されたり、硬貨ごとに異なる柄が描かれることがあるが、本稿では、コインはそれぞれ区別がつかないと仮定する。

コインベースプロトコルは、プレーヤーの掌中かテーブルの上でコインを操作し、マルチパーティ計算を実現する。本稿では、表面を上にした状態のコインを ∘ であらわし、裏面を上にした状態のコインを • であらわす。例えば、∘ と • をそれぞれ 1 と 0 にエンコードするとき、NOT プロトコルはコインを反転することで実現される。

裏面を上にしてテーブルに置けば表面の情報を秘匿できるカードと異なり、コインは上の面から下の面の情報が特定されるため、テーブルの上でそのまま実行すると、プロトコル実行中にユーザの秘密情報が漏れてしまう。そこで、情報を隠しながらプロトコルを実行するために、プレーヤーの手でコインを隠したり、コインの上に別のコインを重ねたりするなどの、工夫が必要となる。

本稿は、まず、コインプロトコルを実現するための操作を定式化する。そして、それらの操作を用いて NOT, AND, OR, XOR を計算する各プロトコルと、秘匿入力をコピーするためのプロトコルの実現例を示す。さらに、それらのプロトコルが完全性と安全性をみたすことを示す。

1.2　本稿の構成

はじめに、2 節において、関連研究であるカードベースプロトコル、および、カードベースプロトコルの評価に用いる確率トレースとダイアグラムを説明する。その後、3 節において、コインベースプロトコルのアイデアと定義を与える。さらに、4 節において、NOT, AND, OR, XOR, コピーの各プロトコルの具体的な構成例を与える。5 節では、コインベースプロトコルの実装と、プロトコルの拡張可能性を議論する。最後に、6 節でまとめを述べる。

2.　関連研究

本節では、カードベースプロトコルについて説明する。

2.1　カードベースプロトコル

1 節で述べたように、カードベースプロトコルは、身近にあるカードを用いてマルチパーティ計算を実現する手法であり、数多くのプロトコルの構成例が報告されている。本節では、プロトコルの完全性と安全性を議論する上で必要となる**可視化列**と**原始列**の概念を（インフォーマルに）説明する。正確な定義は、文献 [5] を参照されたい。

1 節で述べた 6 枚のカードを用いた AND プロトコルを考えよう。入力 🂠🂠♣♡🂠🂠 や、出力 ♡♣🂠🂠🂠🂠 のようなカード列に対して、目に見える記号の列を**可視化列**と呼ぶ。例えば、🂠🂠♣♡🂠🂠 の可視化列を、??♣♡?? と記す。

同様に、カード列に対して、カードの裏表に関わらず、カードの表面に記載された記号の列を**原始列**と呼ぶ。例えば、前述の 6 枚のカードを用いた AND プロトコルの $(a, b) = (0, 1)$ に対応する入力 🂠🂠♣♡🂠🂠 の原始列を、♣♡♣♡♡♣ と記す。

2.2　確率トレースと拡張ダイアグラム

本節では、カードベースプロトコルの完全性と安全性の確認に有用な確率トレースと、確率トレースを利用した拡張ダイアグラム [6] を説明する。

定義 1 (確率トレース)　カードベースプロトコル $\mathcal{P}$ に対して、入力集合 U に含まれる要素の数を $n = |U|$ とする。このとき、

$$q_{i,j} = \Pr[M = \Gamma^i, G_j = s | V_j = v]$$

をみたす n 個の成分からなる組 $(q_{1,j}, \cdots, q_{n,j})$ を、ステップ番号 j、原始列 s、可視列トレース v に対する**確率トレース**と呼ぶ。ここで、Γ^i は、第 i 番目の入力をあらわす。また、M、G_j、V_j は、それぞれ、入力の確率変数、第 j ステップ終了後の原始列の確率変数、第 j ステップ終了後の可視列トレースの確率変数をあらわす。

Koch、Walzer、Härtel [7] は、カードベースプロトコルの完全性を確認するために、カードの状態遷移をあらわすダイアグラム（KWH ダイアグラムと呼ぶ）を提案した。我々 [6] は、（誤操作を含むことも許す）カードベースプロトコルの安全性を評価するために、上述の確率トレースを併記して、KWH ダイアグラムを拡張した（拡張ダイアグラムと呼ぶ）。

図 2 は、6 枚のカードを用いる AND プロトコルへの拡張ダイアグラムの適用例である。この図において、“Step i” の行は、第 i ステップの処理の直後におけるカード列の可視化列、原始列と確率トレースを示している。下線で示した原始記号は、プロトコルの出力をあらわす。確率トレースにおいて、p_{ab} は入力 Γ^{ab} が生じる確率をあらわす。

プロトコルの完全性は、図のステップ 5 において、非零となる p_{ij} に対する $i \wedge j$ と、下線で示したプロトコルの出力が一致することにより確認できる。

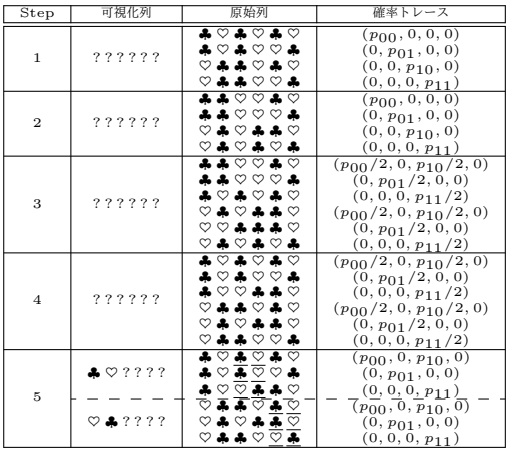

図 2 6 枚カード AND プロトコルへの拡張ダイアグラムの適用例

安全性は、以下のように確認できる：図のステップ 5 において、左側 2 枚のカードが ♣♡ となる場合を調べよう。この場合に該当する確率トレースの和（図中の破線の上に示した 3 つのベクトル成分）は、$(p_{00}, p_{01}, p_{10}, p_{11})$ となる。すなわち、この場合においては、プロトコルの実行前後で確率分布が変わらない、つまり、プロトコルを実行しても情報が漏れていない。ステップ 5 で左側 2 枚のカードが ♡♣ となる場合にも同様の議論がなりたつため、プロトコルが安全であることが確認できる。

3. コインベースプロトコル

本節は、コインベースプロトコルを導入するためのアイデアを紹介し、コインベースプロトコルの定義を与える。

3.1 アイデア

本稿は、カードの代わりに、 図 1 に示したようなコインを用いて、コインベースプロトコルの概念を提案する。1.1 節で述べたように、本稿では、コインは表と裏でデザインが異なるが、すべてのコインで表および裏のデザインは同一であり、それぞれのコインは区別がつかないと仮定する。

安全性を無視すれば、カードの ♣ と ♡ をコインの ● と ○ で置き換えて、カードベースプロトコルを実行すれば、コインを用いて AND 計算などを実行できる。

あるいは、カードベースプロトコルで用いるカードの 2 倍の枚数のコインを用意して、上と同様にカードベースプロトコルのカードをコインに置き換えて、そのコインの上に表面を上にしたダミーのコインを重ねれば、自明なコインベースプロトコルを構成できる。自明なプロトコルにおいては、プロトコルを通じてダミーのコインの表面のみが人目に触れるが、すべてで表面が上になっているために情報が漏れることはない。自明なコインベースプロトコルでは、カードプロトコルにおけるカードの反転処理の代わりに、ダミーのコインを取り除く処理を実行する。

本稿は、すべてのコインにダミーのコインを載せるのではなく、コインを手で握って隠しながら実行するプロトコルを提案する。例えば、4.2 節で紹介するコインベースプロトコルでは 6 枚のコインで AND 計算を実行できる。本稿は、コインベースプロトコルの計算モデル（操作）を定式化し、5 節ではその実装について議論する。

3.2 定義

コインの表と裏をそれぞれ ○ と ● であらわす。2 枚のコイン $c, d \in \{\circ, \bullet\}$ に対して、d の上に c が重なった状態を 2 枚のコイン c, d の束と呼び、その状態を

$$cd$$

であらわす。例えば、○● は、最前面が ○ であり、底面が（● の裏面である）○ であるような、2 枚のコインが重なった束をあらわす。自然数 k に対して、$\mathcal{S}$ を、k 枚のコイン束の集合とする。すなわち、

$$\mathcal{S} = \{\circ, \bullet\}^k = \{\epsilon, \circ, \bullet, \circ\circ, \circ\bullet, \bullet\circ, \bullet\bullet, \circ\circ\circ, \circ\circ\bullet, \cdots\}$$

とする。すなわち、$\mathcal{S}$ はアルファベット $\{\circ, \bullet\}$ 上の長さ k 以下の文字列の集合とみなすことができる。ここで、ϵ は、コインが存在しない状態をあらわす。

2 人のプレーヤーにより実行されるコインベースプロトコルにおけるコインの状態を記述するために、以下の対を利用する。

$$(\mathbf{a}_L, \mathbf{a}_R | \mathbf{b}_L, \mathbf{b}_R | \mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_k)$$

ここで、$\mathbf{a}_L, \mathbf{a}_R, \mathbf{b}_L, \mathbf{b}_R, \mathbf{t}_i \in \mathcal{S}$ は、それぞれ、Alice の左手、Alice の右手、Bob の左手、Bob の右手、テーブルの i 番目のエリア（$i \in [1, k]$）を表す。それぞれの変数（コインがおかれる場所）を $\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R, \mathbf{T}_i$ とする。以下では、プロトコルを通じてコインを置くことのない $\mathbf{T}_j$ がある場合には、簡単のために $\mathbf{t}_j$ や $\mathbf{T}_j$ の記号を省略する。

このとき、コインの各状態の集合を

$$\mathsf{Arg}^k = \{\Gamma = (\mathbf{c}_1, \mathbf{c}_2 | \mathbf{c}_3, \mathbf{c}_4 | \mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_k); \mathbf{c}_i, \mathbf{d}_i \in \mathcal{S}\}.$$

であらわす。特に、入力の集合を $U \subseteq \mathsf{Arg}^{\mathcal{S}}$ であらわす。

コイン束 $\mathbf{c} \in \mathcal{S}$ に対して、$\mathsf{top}(\mathbf{c})$, $\mathsf{bottom}(\mathbf{c})$, $\mathsf{turn}(\mathbf{c})$ は、それぞれ、$\mathbf{c}$ の表面（最前面）、$\mathbf{c}$ の底面（最下面）、$\mathbf{c}$ の反転を返すような関数とする。

次に、コイン束の列 $\Gamma = (\mathbf{c}_1, \mathbf{c}_2 | \mathbf{c}_3, \mathbf{c}_4 | \mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_k)$ の可視化列を定義しよう。まず、top を以下のように拡張する。$\mathbf{c} \in \mathcal{S}$ に対して、$\overline{\mathsf{top}}(\mathbf{c})$ は、$\mathbf{c}$ が（閉じた）手の中にあるときには “?” を返し、テーブルの上にあるときには $\mathsf{top}(\mathbf{c})$ を返す関数とする。さらに、$\Gamma = (\mathbf{c}_1, \mathbf{c}_2 | \mathbf{c}_3, \mathbf{c}_4 | \mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_k) \in \mathsf{Arg}^k$ に対して、$\overline{\mathsf{top}}(\Gamma) = (\overline{\mathsf{top}}(\mathbf{c}_1), \overline{\mathsf{top}}(\mathbf{c}_2) | \overline{\mathsf{top}}(\mathbf{c}_3), \overline{\mathsf{top}}(\mathbf{c}_4) | \overline{\mathsf{top}}(\mathbf{d}_1), \overline{\mathsf{top}}(\mathbf{d}_2), \cdots, \overline{\mathsf{top}}(\mathbf{d}_k))$ とおく。例えば、$\mathbf{c}_1 = \mathbf{c}_3 = \mathbf{d}_1 = \circ\bullet$ かつ $\mathbf{c}_2 = \mathbf{c}_4 = \mathbf{d}_2 = \bullet\circ$ のとき、$\overline{\mathsf{top}}(\Gamma) = (?, ?; ?, ?; \circ, \bullet)$ であ

る。このとき、可視化列は以下で定義される。

$$\mathsf{Vis}^k = \{\overline{\mathsf{top}}(\Gamma); \Gamma \in \mathsf{Arg}^k\}.$$

このとき、以下のようにコインベースプロトコルを導入する。

定義 2 (コインベースプロトコル) コインベースプロトコルは、4 つ組 $\mathcal{P} = (k, U, Q, A)$ で定義される。

- k は、コインの枚数である。
- $U \subseteq \mathsf{Arg}^k$ は、入力の集合である。ただし、Arg^k は k 枚のコインで構成されるコイン束の列の集合をあらわす。
- Q は、**状態の集合** である。ただし、**初期状態** を $q_0 \in Q$、**終了状態** を $q_{\mathrm{f}} \in Q$ と書く。
- $A : (Q - \{q_{\mathrm{f}}\}) \times \mathsf{Vis}^k \to Q \times \mathsf{Action}$ は、**動作関数**をあらわす。ただし、Action は、以下のコイン操作からなる動作の集合である。
  - $(\mathsf{move}, \mathbf{P}_1 \to \mathbf{P}_2, n)$: $\mathbf{P}_1, \mathbf{P}_2 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R, \mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$ かつ $\mathbf{P}_1 \neq \mathbf{P}_2$ とする。プレーヤーは、$\mathbf{P}_1$ に置かれたコイン束 $\mathbf{p}_1$ の $m(\geq n)$ 枚のコインのうち、上から n 枚を $\mathbf{P}_2$ に移動する。移動する n 枚のコインと、移動されずに残る $m-n$ 枚のコインからなる束を、$\mathbf{p}^{(u)}$ と $\mathbf{p}^{(l)}$ と書くことにしよう。この操作により、$(\cdots, \mathbf{p}_1, \cdots, \mathbf{p}_2, \cdots)$ は、$(\cdots, \mathbf{p}_1^{(l)}, \cdots, \mathbf{p}_1^{(u)}\mathbf{p}_2, \cdots)$ に変化する。
    $\mathbf{P}_1 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R\}$ であるときには、プレーヤーは move の実行開始時に手を開く。あるいは、$\mathbf{P}_2 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R\}$ の場合には、$\mathbf{p}_1^{(u)}$ を受け取るために、プレーヤーは手を開く。コイン束を移動したら、プレーヤーは手を閉じる。move を実行すると、$\mathsf{top}(\mathbf{p}_1)$, $\mathsf{top}(\mathbf{p}_1^{(l)})$, $\mathsf{top}(\mathbf{p}_2)$ に関する情報が漏れることに注意しよう。プレーヤーは、その他のコインの情報が漏れないように注意しながら、move を実行するものとする。
  - $(\mathsf{hand}, \mathbf{P}_2 \leftarrow \mathbf{P}_1)$: $\mathbf{P}_1, \mathbf{P}_2 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R\}$ かつ $\mathbf{P}_1 \neq \mathbf{P}_2$ とする。プレーヤーは、コイン束 $\mathbf{p}_1 \in \mathcal{S}^k$ を握った手 $\mathbf{P}_1$ を、掌側が向き合うように、別の手 $\mathbf{P}_2$ の上に重ねる。そして、それらの手を同時に開き、重ね合わされたコインが見えなくなるように、下の手を閉じる。この操作により、$(\cdots, \mathbf{p}_1, \cdots, \mathbf{p}_2, \cdots)$ は、$(\cdots, \epsilon, \cdots, \mathsf{turn}(\mathbf{p}_1)\mathbf{p}_2, \cdots)$ に変化する。
  - $(\mathsf{shuffle}, \mathbf{P}_1, \mathbf{P}_2)$: $\mathbf{P}_1, \mathbf{P}_2 \in \{\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$ かつ $\mathbf{P}_1 \neq \mathbf{P}_2$ とする。プレーヤーは、$\mathbf{P}_1$ と $\mathbf{P}_2$ に置かれたコイン束の位置をランダムに入れ替える。
  - $(\mathsf{flip}, \mathbf{P})$: $\mathbf{P} \in \{\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$ とする。プレーヤーは、$\mathbf{P}$ に置かれたコイン束を反転する。
  - $(\mathsf{rflip}, \mathbf{P})$: $\mathbf{P} \in \{\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$ とする。プレーヤーは、$\mathbf{P}$ に置かれたコイン束をランダムに反転する。

プロトコルを正しく実行したときに、最終状態において正しい演算結果が出力されるとき、コインベースプロトコル $\mathcal{P}$ は**完全性**をみたすという。

次に、コインベースプロトコルの安全性を定義する。

定義 3 (コインベースプロトコルの安全性) コインベースプロトコル $\mathcal{P}$ が実行時に入力に関する情報をもらさない（すなわち、入力が可視列のトレースと独立である）とき、プロトコル $\mathcal{P}$ が **安全** であるという。

4. コインベースプロトコルの具体例

本節では、コインベースプロトコルの具体例と、それぞれの性質を示す。

4.1 NOT プロトコル

1 枚のコインを用いて情報の符号化を行う場合、NOT の計算はコインを反転すればよい。例えば、Alice が右手に持っているコインを左手に移し替えると、コインが反転して NOT を計算できる。このとき、完全性、安全性は自明である。

4.2 AND プロトコル

Alice と Bob が AND を計算するプロトコルを示す。Alice と Bob の入力 $a, b \in \{0, 1\}$ に対して、プロトコルは 4 種類の対 Γ^{ab} を入力とする。

---

6 枚コインベース AND プロトコル $\mathcal{P}_{\mathsf{Coin}}^{\mathsf{AND}}$
入力:

$$\big\{\Gamma^{00} = (\circ, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon), \Gamma^{01} = (\circ, \circ|\bullet\bullet, \circ\bullet|\epsilon, \epsilon, \epsilon, \epsilon),$$
$$\Gamma^{10} = (\bullet, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon), \Gamma^{11} = (\bullet, \circ|\bullet\bullet, \circ\bullet|\epsilon, \epsilon, \epsilon, \epsilon)\big\}$$

ステップ:
(1) $(\mathsf{hand}, \mathbf{A}_L \leftarrow \mathbf{B}_L)$
(2) $(\mathsf{hand}, \mathbf{A}_R \leftarrow \mathbf{B}_R)$
(3) $(\mathsf{move}, \mathbf{A}_L \to \mathbf{T}_1, 3)$
(4) $(\mathsf{move}, \mathbf{A}_R \to \mathbf{T}_2, 3)$
(5) $(\mathsf{shuffle}, \mathbf{T}_1, \mathbf{T}_2)$
(6) $(\mathsf{move}, \mathbf{T}_1 \to \mathbf{T}_3, 1)$
(7) $(\mathsf{move}, \mathbf{T}_2 \to \mathbf{T}_4, 1)$
(8) **if** $(\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2)) = (\circ, \bullet)$
$(\mathsf{result}, \mathsf{bottom}(\mathbf{t}_1))$
(9) **else if** $(\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2)) = (\bullet, \circ)$
$(\mathsf{result}, \mathsf{bottom}(\mathbf{t}_2))$

2.2 節で説明した拡張ダイアグラムを用いると、図 3 に示すように、$\mathcal{P}_{\mathsf{Coin}}^{\mathsf{AND}}$ が完全性と安全性をみたすことを確認

ステップ	可視化列	原始列	確率トレース
0	$(?,?\|?,?\|\epsilon,\epsilon,\epsilon,\epsilon)$	$(\circ,\circ\|\circ\bullet,\bullet\bullet\|\epsilon,\epsilon,\epsilon,\epsilon)$ $(\circ,\circ\|\bullet\bullet,\circ\bullet\|\epsilon,\epsilon,\epsilon,\epsilon)$ $(\bullet,\circ\|\circ\bullet,\bullet\bullet\|\epsilon,\epsilon,\epsilon,\epsilon)$ $(\bullet,\circ\|\bullet\bullet,\circ\bullet\|\epsilon,\epsilon,\epsilon,\epsilon)$	$(p_{00},0,0,0)$ $(0,p_{01},0,0)$ $(0,0,p_{10},0)$ $(0,0,0,p_{11})$
2	$(?,?\|\epsilon,\epsilon\|\epsilon,\epsilon,\epsilon,\epsilon)$	$(\circ\bullet\circ,\circ\circ\circ\|\epsilon,\epsilon\|\epsilon,\epsilon,\epsilon,\epsilon)$ $(\circ\circ\circ,\circ\bullet\circ\|\epsilon,\epsilon\|\epsilon,\epsilon,\epsilon,\epsilon)$ $(\circ\bullet\bullet,\circ\circ\circ\|\epsilon,\epsilon\|\epsilon,\epsilon,\epsilon,\epsilon)$ $(\circ\circ\bullet,\circ\bullet\circ\|\epsilon,\epsilon\|\epsilon,\epsilon,\epsilon,\epsilon)$	$(p_{00},0,0,0)$ $(0,p_{01},0,0)$ $(0,0,p_{10},0)$ $(0,0,0,p_{11})$
4	$(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ,\circ,\epsilon,\epsilon)$	$(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\bullet\circ,\circ\circ\circ,\epsilon,\epsilon)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\circ\circ,\circ\bullet\circ,\epsilon,\epsilon)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\bullet\bullet,\circ\circ\circ,\epsilon,\epsilon)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\circ\bullet,\circ\bullet\circ,\epsilon,\epsilon)$	$(p_{00},0,0,0)$ $(0,p_{01},0,0)$ $(0,0,p_{10},0)$ $(0,0,0,p_{11})$
5	$(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ,\circ,\epsilon,\epsilon)$	$(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\bullet\circ,\circ\circ\circ,\epsilon,\epsilon)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\circ\circ,\circ\bullet\circ,\epsilon,\epsilon)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\bullet\bullet,\circ\circ\circ,\epsilon,\epsilon)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\circ\circ,\circ\bullet\bullet,\epsilon,\epsilon)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\circ\bullet,\circ\bullet\circ,\epsilon,\epsilon)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\bullet\circ,\circ\circ\bullet,\epsilon,\epsilon)$	$(p_{00}/2,p_{01}/2,0,0)$ $(p_{00}/2,p_{01}/2,0,0)$ $(0,0,p_{10}/2,0)$ $(0,0,p_{10}/2,0)$ $(0,0,0,p_{11}/2)$ $(0,0,0,p_{11}/2)$
9	$(\epsilon,\epsilon\|\epsilon,\epsilon\|\bullet,\circ,\circ,\circ)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ,\bullet,\circ,\circ)$	$(\epsilon,\epsilon\|\epsilon,\epsilon\|\bullet\underline{\circ},\circ\underline{\circ},\circ,\circ)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\bullet\bullet,\circ\underline{\circ},\circ,\circ)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\bullet\underline{\circ},\circ\underline{\bullet},\circ,\circ)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\underline{\circ},\bullet\underline{\circ},\circ,\circ)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\underline{\circ},\bullet\bullet,\circ,\circ)$ $(\epsilon,\epsilon\|\epsilon,\epsilon\|\circ\underline{\bullet},\bullet\underline{\circ},\circ,\circ)$	$(p_{00},p_{01},0,0)$ $(0,0,p_{10},0)$ $(0,0,0,p_{11})$ $(p_{00},p_{01},0,0)$ $(0,0,p_{10},0)$ $(0,0,0,p_{11})$

図 3 コインベース AND プロトコルへの拡張ダイアグラムの適用

できる。本図において、“Step 0” は入力直後、すなわち、ステップ 1 の処理を実行する直前における、可視化列、原始列、確率トレースをあらわす。

まず、完全性を確認しよう。図 2 と同様に、プロトコルの出力を下線で示した。本図において、$(a,b)=(1,1)$ である場合、すなわち、p_{11} が非零である場合に限り、出力が $\mathsf{bottom}(\underline{\bullet})=\circ$ である。したがって、プロトコルは完全性をみたす。

次に、安全性を確認しよう。本図のステップ 13 において、$(\mathsf{top}(\mathbf{t},L),\mathsf{top}(\mathbf{t}_R))$ が $(\bullet,\circ)$ となる場合、および $(\circ,\bullet)$ となる場合のいずれにおいても、確率トレースの和は $(p_{00},p_{01},p_{10},p_{11})$ となる。したがって、2.2 節と同様に、プロトコルは安全であることが確認できる。

4.3 OR プロトコル

Alice と Bob が OR を計算するプロトコルを示す。OR は、AND プロトコルと NOT プロトコルを組み合わせて、$\mathcal{P}^{\mathsf{OR}}_{\mathsf{Coin}}(\Gamma^{\overline{ab}})$ の反転値として計算できる。プロトコルを以下に示す。Alice と Bob の入力 $a,b\in\{0,1\}$ に対して、プロトコルは 4 種類の対 Γ^{ab} を入力とする。

---

6 枚コインベース OR プロトコル $\mathcal{P}^{\mathsf{OR}}_{\mathsf{Coin}}$

入力:

$$\big\{\Gamma^{00}=(\bullet,\circ|\bullet\bullet,\circ\bullet|\epsilon,\epsilon,\epsilon,\epsilon,\epsilon),\Gamma^{01}=(\bullet,\circ|\circ\bullet,\bullet\bullet|\epsilon,\epsilon,\epsilon,\epsilon,\epsilon)$$
$$\Gamma^{10}=(\circ,\circ|\bullet\bullet,\circ\bullet|\epsilon,\epsilon,\epsilon,\epsilon,\epsilon),\Gamma^{11}=(\circ,\circ|\circ\bullet,\bullet\bullet|\epsilon,\epsilon,\epsilon,\epsilon,\epsilon)\big\}$$

ステップ:

(1) execute $\mathcal{P}^{\mathsf{AND}}_{\mathsf{coin}}(\Gamma^{ab})$
(2) **if** $(\mathsf{top}(\mathbf{t}_1),\mathsf{top}(\mathbf{t}_2))=(\circ,\bullet)$
  $(\mathsf{move},\mathbf{T}_1\to\mathbf{T}_3,2)$
(3) **else if** $(\mathsf{top}(\mathbf{t}_1),\mathsf{top}(\mathbf{t}_2))=(\bullet,\circ)$
  $(\mathsf{move},\mathbf{T}_2\to\mathbf{T}_3,2)$
(4) $(\mathsf{flip},\mathbf{T}_3)$
(5) $(\mathsf{move},\mathbf{T}_3\to\mathbf{T}_5,2)$
(6) $(\mathsf{result},\mathsf{bottom}(\mathbf{t}_5))$

AND プロトコルと同様に、OR プロトコルも完全性と安全性をみたすことを確認できる。

4.4 XOR プロトコル

次に、Alice と Bob が XOR を計算するためのプロトコルの具体例を与える。$a\oplus b$ を計算するために、rflip でコイン束をランダムに反転する。のコインの反転回数の偶奇に対応した乱数 r に対して、まず、$a'=a\oplus r$ と $b'=b\oplus r$ を計算する。そして、$a'\oplus b'=a\oplus b$ により、XOR を計算する。

具体的には、以下の処理を実行する。まず、b に対応するコインを a に対応するコインの上に重ね、そのコイン束をランダムに反転する。r' 回の反転を行うと、$r=r' \bmod 2$ に対して、これらのコインは $a'=a\oplus r$ と $b'=b\oplus r$ に対応したコインとなる。次に、a' か b' のコインを開示する。開示したコインが 0 に対応する場合には、他方のコインが XOR 計算の結果に対応する。一方、開示したコインが 1 に対応する場合には、他方のコインの反転が XOR 計算の結果に対応する。情報の漏洩を防ぐために、ダミーのコインを使用する。

Alice と Bob の入力 $a,b\in\{0,1\}$ に対して、プロトコルは 4 種類の対 Γ^{ab} を入力とする。

---

6 枚コインベース XOR プロトコル $\mathcal{P}^{\mathsf{XOR}}_{\mathsf{Coin}}$

入力:

$$\big\{\Gamma^{00}=(\circ\bullet\bullet,\epsilon|\circ\circ\bullet,\epsilon|\epsilon,\epsilon,\epsilon),\Gamma^{01}=(\circ\bullet\bullet,\epsilon|\circ\bullet\bullet,\epsilon|\epsilon,\epsilon,\epsilon)$$
$$\Gamma^{10}=(\circ\circ\bullet,\epsilon|\circ\circ\bullet,\epsilon|\epsilon,\epsilon,\epsilon),\Gamma^{11}=(\circ\circ\bullet,\epsilon|\circ\bullet\bullet,\epsilon|\epsilon,\epsilon,\epsilon)\big\}$$

ステップ:

(1) $(\mathsf{hand},\mathbf{A}_L\leftarrow\mathbf{B}_L)$
(2) $(\mathsf{move},\mathbf{A}_L\to\mathbf{T}_1,6)$
(3) $(\mathsf{rflip},\mathbf{T}_1)$
(4) $(\mathsf{move},\mathbf{T}_1\to\mathbf{T}_2,3)$
(5) $(\mathsf{shuffle},\mathbf{T}_1,\mathbf{T}_2)$
(6) $(\mathsf{move},\mathbf{T}_1\to\mathbf{T}_3,1)$
(7) **if** $\mathsf{top}(\mathbf{t}_1)=\bullet$
  $(\mathsf{move},\mathbf{T}_2\to\mathbf{T}_3,2)$
(8) **else**
  (a) $(\mathsf{flip},\mathbf{T}_2)$
  (b) $(\mathsf{move},\mathbf{T}_2\to\mathbf{T}_3,2)$
(9) $(\mathsf{result},\mathsf{bottom}(\mathbf{t}_3))$

AND プロトコルと同様に、XOR プロトコルも完全性と安全性をみたすことを確認できる。

4.5 コピープロトコル

コピープロトコルは、上述の XOR プロトコルを拡張することで実現できる。入力 a に対応するコインに加えて、$b = c = 0$ に対応する 2 枚のコインを利用する。そして、XOR プロトコルと同様に rflip を実行し、それらのコインを $a' = a \oplus r$ と $b' = c' = r$ に対応するコインに変換する。そして、$a' \oplus b'$ と $a' \oplus c'$ を実行すると、a のコピーを得ることができる。コピープロトコルは、入力 $a \in \{0, 1\}$ に対して、2 種類の対 Γ^a を入力とする。

---

5 枚コインベース XOR プロトコル $\mathcal{P}^{\mathsf{COPY}}_{\mathsf{Coin}}$

入力:

$$\left\{\Gamma^0 = (\bullet, \circ\bullet|\epsilon, \epsilon|\bullet\bullet, \epsilon, \epsilon), \Gamma^1 = (\circ, \circ\bullet|\epsilon, \epsilon|\bullet\bullet, \epsilon, \epsilon)\right\}$$

ステップ:

(1) $(\mathsf{hand}, \mathbf{A}_L \leftarrow \mathbf{A}_R)$
(2) $(\mathsf{move}, \mathbf{A}_L \rightarrow \mathbf{T}_1, 3)$
(3) $(\mathsf{rflip}, \mathbf{T}_1)$
(4) $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_2, 2)$
(5) **if** $\mathsf{top}(\mathbf{t}_1) = \circ$
　　$(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_3, 2)$
(6) **else**
　(a) $(\mathsf{flip}, \mathbf{T}_1)$
　(b) $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_3, 2)$
　(c) $(\mathsf{move}, \mathbf{T}_2 \rightarrow \mathbf{T}_1, 2)$
　(d) $(\mathsf{flip}, \mathbf{T}_1)$
　(e) $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_2, 2)$
(7) $(\mathsf{result}, \mathsf{bottom}(\mathbf{t}_2)\mathsf{bottom}(\mathbf{t}_3))$

上述のコピープロトコルは、1 枚の入力コインに対して、4 枚のコインを追加して、入力と同じ情報をもつ 2 枚のコインを出力する。追加するコインの枚数を $2k + 2$ 枚にすると、1 枚の入力コインに対して、入力コインと同じ情報を持つ $2k$ 枚のコインを出力することができる。具体的には、上述のプロトコルの入力のうち、$\mathbf{a}_R, \mathbf{t}_L$ を次のように置き換える。$\mathbf{a}_R$ は、2 枚のコイン束 $\circ\bullet$ の代わりに、$k+1$ 枚のコイン束 $\circ\circ\cdots\circ\bullet$ に置き換える。一方、$\mathbf{t}_L$ は、2 枚のコイン束 $\bullet\bullet$ の代わりに、$k+1$ 枚のコイン束 $\bullet\bullet\cdots\bullet$ に置き換える。入力を置き換えた上で、上述のプロトコルを実行すると、$\mathbf{t}_L$ と $\mathbf{t}_R$ において、それぞれの最前面のコインを除いた k 枚のコインのテーブル側の面が入力と同じ値となる。

上述のプロトコルと同様に、コピープロトコルも完全性と安全性をみたすことを確認できる。

5. 議論

本節では、コインベースプロトコルの実装と、拡張可能性を議論する。

図 4　hand の実装例

図 5　rflip の実装例

5.1 実装

まず、定義 2 で挙げた 5 つの操作の実装可能性を議論する。これらの操作のうち、move と flip は自然に実装可能であるため、説明は省略する。本節では、hand, shuffle, rflip の 3 つの操作の実装を議論する。

hand の実装例を、図 4 に示す。まず、プレーヤーは、コイン束を移動する手を近付ける（図 4 の左上）。次に、プレーヤーは、（反転した）コイン束を移動させるために、そのコイン束を握った手を移動先の手の上に重ねる（同図の中央上）。そして、プレーヤーは、二つの手を同時に開き、下の手の上に（反転した）コイン束を重ねる（同図の右上）。その後、プレーヤーは、重なったコイン束を隠すように、下の手を閉じる（同図の左下）。最後に、プレーヤーは上の手を元に戻す（同図の右下）。この操作により、移動したコイン束は上下が反転することに注意しよう。

shuffle は、カードベースプロトコルのシャッフル操作と同様に実行可能である。プレーヤーは、コイン束の動きが追跡できなくなるまで、2 つのコイン束の位置を入れ替える。

最後に、rflip の 2 つの実装方法を議論する。一方はコイン以外の道具を用いずに rflip を実行し、もう一方は W クリップなどの道具を利用する。前者の実装では、プレーヤーは、コイン束を握り、コイン束の回転数が追跡できなくなるまで、コイン束を回転させる（図 5 の左）。後者の実装では、プレーヤーは、コイン束を W クリップなどで留めて（同図の右）、コインフリップと同様に、クリップ留めしたコイン束を空中に放り投げる。

5.2 プロトコルの拡張可能性

4 節において、本稿は 5 つのコインベースプロトコルの

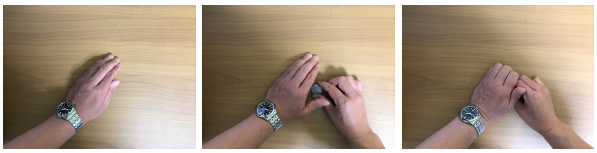

図 6 コインの取り出し (pick)

実現例を示した。それらは、コインベースプロトコルの実現可能性を示すものである。最適なプロトコル、すなわち、より少ない枚数でプロトコルを実現可能であるかどうかの検討は、今後の課題である。

以下では、コインの枚数ではなく、機能の拡張可能性を議論しよう。4 節で示したプロトコルの実現例における出力は、他のコインにより隠されたコインの下の面にあらわれる。したがって、プレーヤーがプロトコルの出力を明らかにすることなく再び掌中に収めることができれば、複数のプロトコルの合成が可能となる。

ここで、{NOT, AND} は完備集合となることに注意しよう。NOT 計算は、コインがプレーヤーの手の中にあるのであれば、容易に実行可能である。したがって、AND 計算の出力を再びプレーヤーが掌中に収める（pick 操作と呼ぶ）ことができれば、任意の演算が実行可能となる。pick 操作の実装例を図 6 に示す。プレーヤーは、プロトコルが出力したコイン束の上に手をかざす（図 6 の左）。そして、コイン束のコインのうち、下のコインの情報が漏れないように注意しながら、上に重なったコインを取り除く（同図の中央）。最後に、プロトコルの出力に対応するコインを拾い上げる（同図の右）。

6. まとめ

本稿は、コインを用いてマルチパーティ計算を実現可能な新たな概念として、コインンベースプロトコルを提案して、NOT や AND 等を計算する具体的なプロトコルの構成を示した。さらに効率の良いプロトコルを開発したり、文献 [6] を拡張して誤操作を含むコインベースプロトコルからの漏洩情報量を評価したりすることなどは、今後の課題である。

謝辞

本研究は JSPS 科研費 17K00001 の助成を受けたものです.

参考文献

[1] T. Mizuki and H. Sone, “Six-Card Secure AND and Four-Card Secure XOR”, Frontiers in Algorithmics, LNCS 5598, Springer, pp.358-369, 2009.

[2] T. Mizuki, M. Kumamoto, and H. Sone, “The Five-Card Trick Can Be Done with Four Cards”, ASIACRYPT 2012, LNCS 7658, Springer, pp.598-606, 2012.

[3] I. Ueda, A. Nishimura, Y.Hayashi, T. Mizuki, and H. Sone, “How to Implement a Random Bisection Cut”, Theory and Practice of Natural Computing, LNCS, 10071, Springer, pp. 58-69, 2016

[4] A. Marcedone, Z. Wen, and E. Shi, “Secure Dating with Four or Fewer Cards”, Cryptology ePrint Archive, Report 2015/1031, 2015

[5] T. Mizuki, and H. Shizuya, “Computational Model of Card-Based Cryptographic Protocols and Its Applications”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E100.A(1), pp. 3-11, 2017

[6] T. Mizuki, and Y. Komano, “Analysis of Information Leakage due to Operative Errors in Card-based Protocols”, IWOCA 2018, LNCS, Springer, (To appear).

[7] A. Koch, S. Walzer Stefan, and K. Härtel, “Card-Based Cryptographic Protocols Using a Minimal Number of Cards”, ASIACRYPT 2015, LNCS 9452, Springer, pp. 783-807, 2015