Computer Security Symposium 2020
26 - 29 October 2020

コインベースプロトコルの初期配置誤りに関する考察

駒野 雄一[1]　水木 敬明[2]

概要：コインベースプロトコルは身近なものを用いるマルチパーティ計算（物理的暗号技術）の一つであり、複数人のプレイヤーがコイン（硬貨）を利用して秘密計算を行う。CSS2018 において、我々は、コインベースプロトコルを実行するプレーヤーがプロトコルが指定する操作を誤った際に、秘密入力に関する情報が漏洩するか否かを議論した。コインベースプロトコルでは、入力に対応するコインはプレイヤーの掌中に初期配置されるため、その正しさは明らかではない。本稿では、コインの初期配置を誤った際のプロトコルの挙動を解析するとともに、初期配置の誤りに対して頑健なプロトコルの構成を議論する。

キーワード：マルチパーティ計算、物理的暗号、コイン（硬貨）、初期配置誤り

A note on Erroneous Setting in Coin-based Protocol

Yuichi Komano[1]　Takaaki Mizuki[2]

Abstract: Coin-based protocol is a multi-party computation where players securely compute, by using a physical coins, a function value for their secret input. At CSS 2018, we discussed the information leakage of secret information when a player executes a wrong operation. In the coin-based protocol, coins for a secret input are grabbed in the player's hands and invisible. Hence, the correctness of coins (or, and initial setting of coins) cannot be checked obviously. In this paper, we will analyze the protocol with wrong setting and discuss measures for enhancing the robustness of the coin-based protocol against the wrong initial setting.

Keywords: multi-party computation, coin, wrong initial setting

1. はじめに

マルチパーティ計算は、複数のユーザが互いの入力を秘匿したまま所望の値を計算するための技術であり、企業間のマーケティング情報の集計などへの応用が議論されている。一方、計算機を利用せずにマルチパーティ計算を実現する技術として、**カードベースプロトコル** [1], [3] や**コインベースプロトコル** [6], [7] が提案されている。これらの物理的暗号技術は、計算機をブラックボックス利用しないためにユーザが安心してプロトコルを実行できることに加えて、プロトコルの原理を直感的に理解しやすいために情報セキュリティ教育にも有用である。

物理的暗号技術は簡単に実装（実行）できる処理で構成されるが、ユーザはプロトコルが指定する操作や手順などを誤ってしまうことがある。文献 [5] は、プロトコルの各ステップにおいて、秘匿すべき入力の漏洩情報量を評価するための指標として**確率トレース**の概念を提案し、Koch らのダイアグラム [2] を拡張した**拡張ダイアグラム**を提案した。そして、6 枚のカードを用いる AND プロトコル [4] を例として、ユーザがカード巡回の向きを誤った場合に入力の情報が漏れる場合があることを示し、情報漏洩への対策を提案した。

本稿は、コインベースプロトコルにおいて、コインの初期配置に誤りがある場合に、プロトコルがどのような値を出力するか、初期配置の誤りを検知することができるのか、といったことを、プロトコルの挙動を解析して考察する。さらに、初期配置の誤りに対して頑健なプロトコルの構成

[1] 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

を議論する。

2. コインベースプロトコル

コインベースプロトコルは、コイン (硬貨) を利用するマルチパーティ計算技術である。1 枚のカードを裏返せば情報を秘匿できるカードベースプロトコルと異なり、コインベースプロトコルは、コインを積み重ねたり手で覆ったりして、情報を秘匿しながら計算を実行する。コインの表と裏を、それぞれ $\circ$ と $\bullet$ であらわす。コインベースプロトコルでは、ユーザは、自身の入力を

$$\bullet = 0, \quad \circ = 1 \tag{1}$$

のようにエンコードして、コインを掌中やテーブルに配置し、所定の規則で操作する。

2.1 プロトコル例：コインベース AND プロトコル

6 枚のコインを用いる AND プロトコル $\mathcal{P}^{\mathsf{AND}}_{\mathsf{Coin}}$ [7] は、以下の手順で実行される。

---

6 枚コイン AND プロトコル $\mathcal{P}^{\mathsf{AND}}_{\mathsf{Coin}}$

入力集合:

$$\Big\{\Gamma_0^{00} = (\circ, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon), \Gamma_0^{01} = (\circ, \circ|\bullet\bullet, \circ\bullet|\epsilon, \epsilon, \epsilon, \epsilon),$$
$$\Gamma_0^{10} = (\bullet, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon), \Gamma_0^{11} = (\bullet, \circ|\bullet\bullet, \circ\bullet|\epsilon, \epsilon, \epsilon, \epsilon)\Big\}$$

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) $(\mathsf{move}, \mathbf{A}_L \rightarrow \mathbf{T}_1, 3)$
(4) $(\mathsf{move}, \mathbf{A}_R \rightarrow \mathbf{T}_2, 3)$
(5) $(\mathsf{shuffle}, \mathbf{T}_1, \mathbf{T}_2)$
(6) $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_3, 1)$
(7) $(\mathsf{move}, \mathbf{T}_2 \rightarrow \mathbf{T}_4, 1)$
(8) **if** $(\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2)) = (\circ, \bullet)$
$(\mathsf{result}, \mathsf{bottom}(\mathbf{t}_1))$
(9) **else if** $(\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2)) = (\bullet, \circ)$
$(\mathsf{result}, \mathsf{bottom}(\mathbf{t}_2))$

コインベースプロトコルでは、各ユーザの掌中とテーブルにコインを配置して処理を実行する。例えば、$\mathcal{P}^{\mathsf{AND}}_{\mathsf{Coin}}$ の入力 Γ_0^{10} に対しては、次のようにコインが配置される。Alice は、(1) 式に従って、$\bar{a} = 0$（ここで、$\bar{a}$ は a の反転ビットを表す、b についても同様）に対応するコイン $\bullet$ を左手 $\mathbf{A}_L$ に隠し持ち、右手 $\mathbf{A}_R$ にはコイン $\circ$ を持つ。Bob は、$b = 0$ に対して、左手 $\mathbf{B}_L$ にコイン $\bar{b}\bullet = \circ\bullet$ を隠し持ち、右手 $\mathbf{B}_R$ に $b\bullet = \bullet\bullet$ を隠し持つ。上述のプロトコルではテーブル $\mathbf{T}$ を 4 つの領域に分けて使用するが、入力の段階ではコインを配置しない。

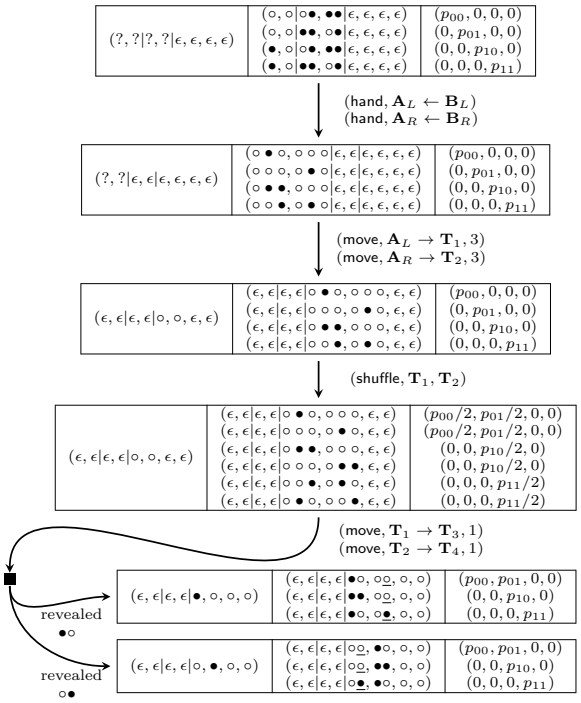

図 1 6 枚コイン AND プロトコル $\mathcal{P}^{\mathsf{AND}}_{\mathsf{Coin}}$ の状態遷移

上の実行例において、$(\mathsf{hand}, \mathbf{P}_1 \leftarrow \mathbf{P}_2)$ は、$\mathbf{P}_2$ の掌中にあるコインを $\mathbf{P}_1$ の掌中に移す操作をあらわす。具体的には、$\mathbf{P}_2$ を握ったまま $\mathbf{P}_1$ の上に重ねて、両方の手を同時に開き、コインが見えないように $\mathbf{P}_1$ を閉じる。$\mathbf{P}_1$ と $\mathbf{P}_2$ にコインを隠し持っていた場合には、$\mathbf{P}_1$ のコインの上に $\mathbf{P}_2$ のコインが反転して積み重なる。$(\mathsf{move}, \mathbf{P}_1 \rightarrow \mathbf{P}_2, n)$ は、$\mathbf{P}_1$ に配置されたコインの上から n 枚をつまみ上げて、$\mathbf{P}_2$ のコインの上に移す。$(\mathsf{shuffle}, \mathbf{P}_1, \mathbf{P}_2)$ は、$\mathbf{P}_1$ と $\mathbf{P}_2$ に置かれたコインをランダムに入れ替える。ステップ (8) および (9) は、テーブルの第 1 と第 2 の領域のコインが $\circ, \bullet$（あるいは $\bullet, \circ$）であるならば、第 1（あるいは第 2）の領域のコインのテーブル面に AND 結果が格納されていることをあらわす。次節にて、プロトコルのモデルと記法の定義を述べる。

図 1 は、$\mathcal{P}^{\mathsf{AND}}_{\mathsf{Coin}}$ への拡張ダイアグラム [5] の適用例である。ここで、ϵ は、コインが存在しないことを表す空文字列である。この図を用いると、プロトコルの完全性と安全性を確認できる。詳細は文献 [7] を参照されたい。

2.2 コインベースプロトコルの定義

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、$\circ\bullet$ は、最前面が $\circ$ であり、底面が（$\bullet$ の裏面である）$\circ$ 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、コインベースプロトコルは、以下のように定式化される [7]。

定義 1 (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 2 (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。

2.3 その他のプロトコル

文献 [6], [7] では、2.1 節で紹介した 6 枚のコインを用いる AND を計算するプロトコルのほかに、OR や XOR を計算するプロトコル、COPY を実行するプロトコルが提案されている。

3. コインベースプロトコルの初期配置誤り

コインベースプロトコルでは、初期配置として、プレイヤーが掌の上にコイン（の束）を載せて手を握る。このとき、プレイヤーが上下（裏表）を間違えてコイン（の束）を掌の上に載せる可能性がある。本節では、そのような場面における、プロトコルの実行結果について考察する。

3.1 初期配置誤りのある AND プロトコル

AND プロトコルでは、初期配置として、Alice は左手と右手に 1 枚ずつのコインを保持し、Bob は左手と右手に 2 枚ずつのコインを保持する。このとき、Alice は、左手と右手のコインのそれぞれについて、正しい向きと誤った向きのコインを掌の上に載せる可能性がある。

一方、Bob は、左手と右手のそれぞれに、掌に接するコイン (下のコイン) は裏面が上向きになるようにして、2 枚のコインを掌に載せる。このとき、枚数を間違える、あるいは、下のコインの向きを間違えると、AND プロトコルのステップ 3 または 4 で、初期配置の誤りを検知することができる。実際、正しいプロトコルにおいては、ステップ 3 および 4 を行うと、テーブルの上に表向きのコイン (もともと、Bob の掌に接していたコイン) が一番上に見えるように、3 枚のコインの束が置かれることになる。上述の誤りが生じると、コインの束の枚数が 3 枚とは異なる、あるいは、一番上のコインが裏向きになるため、誤りを検知することができる。そのため、ここでは、Bob が左手と右手に載せる 2 枚のコインのうち、1 枚目のコイン (上のコイン) について、正しい向きと誤った向きのコインを載せる場合を考える。

すなわち、Alice の左手と右手のそれぞれのコイン、Bob の左手と右手のそれぞれ 1 枚目のコイン、の 4 枚について、誤りの有無を考える。このとき、考えられる誤りのパタン数としては、Alice の左手が正しい/誤り、Alice の右手が正しい/誤り、Bob の左手が正しい/誤り、Bob の右手が正しい/誤りの組み合わせで、$2^4 = 16$ パタンとなる。

それぞれのパタンにおけるプロトコルの挙動を、表 1 にまとめる。表 1 の、“A_L”、“A_R”、“B_L”、“B_R” は、それぞれ、Alice の左手、Alice の右手、Bob の左手、Bob の右手を表す。また、“Alice の手” と “Bob の手” は、それぞれ、Alice の左手と右手に握られたコインが、入力値 (Alice の入力ビット a と Bob の入力ビット b) に対して、どのような値になっているかを示している。ここで、上付き線はビットの反転を表し、$\perp$ は入力値に対応づかない不正値であることを表す。“出力” は、誤りを含む入力を与えたときに、プロトコルが出力する値を示す。“配置誤り検知” は、プロトコルの実行中に配置誤りを (上述のステップ 3 や 4 のように) 検知することができるかどうかを示している。

まず、#1 のパタンは、入力が誤りを含まない状態であり、正しくプロトコルが実行される。そのため、“Alice の手” と “Bob の手”、“出力” は正しい値であり、“配置誤り検知” は不要であるため “—” と記載した。

その他の 15 通りのパタンに関しては、入力に何らかの誤りが含まれる。このうち、#2、#13、#14 に関しては、掌中のコインが誤りにより本来の入力を反転した値に対応し、もともとその値 (反転した値) を入力してプロトコルを実行する場合との区別をつけることができない。ここでは、もともとその値でプロトコルを実行していると見做して、“配置誤り検知” の欄は “不可能” と記載した。

また、#5、#6、#9、#10 は、ステップ 3 と 4、あるいは、ステップ 8 と 9 において、プロトコルを正しく実行した際には生じえない向きのコインが目で見える形で現れることにより、配置誤りを検知することができる。このとき、プロトコルが停止して出力を導かないため、“出力” は “—” と記載した。

残る、#3、#4、#7、#8、#11、#12、#15、#16 は、前述の AND プロトコルでは、配置誤りを検知することができず、不正な値 (入力値やプロトコル中のシャッフルの有無に応じて、正しい出力値 $a \wedge b$ を偶然、あるいは、その反転値) を得る。ただし、このパタンでは、プロトコルのステップ 1 を開始する前に、Alice が右手を開いて A_R が正しいこと (Alice の入力に依らず、常に表向きのコインが一枚載せられていること) を示すことで、誤りを検知することができる。すなわち、プロトコルを微修正することにより、配置誤りに対して頑健なプロトコルを構成できる。そこで、“配置誤り検知” の欄には、検知可能となることを括弧付きで記載した。

3.2 その他のプロトコル

文献 [7] では、AND のほかに、OR、COPY、XOR のコインベースプロトコルが示されている。これらのプロトコルに関しても、前節と同様に、配置誤りの影響と検知可能性を議論することができる。実際、それらのプロトコルに関しても、入力に依らない向きが固定のコインを事前に開示するような微修正を加えることにより、配置誤りに対して頑健なプロトコルを構成することができる。

例えば、コインベース OR プロトコルにおいては、AND プロトコルと同様に、

(1) 正しい値が計算される（表 1 の#1）
(2) 配置誤りを検知可能（表 1 の#5 など）
(3) 配置誤りを検知することができない（配置誤りにより本来の値の反転値が入力されたとみなさざるを得ない場合、表 1 の#2 など）
(4) 向きが入力に依存しないコインを見せる（手を開いてコイン束の一番上のコインを見せる、表 1 の#3 など）

である。

コインベース XOR プロトコルにおいては、上の (1) か

表 1　初期配置誤りのあるコインベース AND プロトコルの挙動

パタン	A_L	A_R	B_L	B_R	Alice の手	Bob の手	出力	配置誤り検知
#1	正	正	正	正	a	b	$a \wedge b$	—
#2	誤	正	正	正	$\overline{a}$	b	$\overline{a} \wedge b$	不可能
#3	正	誤	正	正	$\perp$	b	$\perp$	(Alice が右手を開けば可能)
#4	誤	誤	正	正	$\perp$	b	$\perp$	(Alice が右手を開けば可能)
#5	正	正	誤	正	a	$\perp$	—	検知可能
#6	誤	正	誤	正	$\overline{a}$	$\perp$	—	検知可能
#7	正	誤	誤	正	$\perp$	$\perp$	$\perp$	(Alice が右手を開けば可能)
#8	誤	誤	誤	正	$\perp$	$\perp$	$\perp$	(Alice が右手を開けば可能)
#9	正	正	正	誤	a	$\perp$	—	検知可能
#10	誤	正	正	誤	$\overline{a}$	$\perp$	—	検知可能
#11	正	誤	正	誤	$\perp$	$\perp$	$\perp$	(Alice が右手を開けば可能)
#12	誤	誤	正	誤	$\perp$	$\perp$	$\perp$	(Alice が右手を開けば可能)
#13	正	正	誤	誤	a	$\overline{b}$	$a \wedge \overline{b}$	不可能
#14	誤	正	誤	誤	$\overline{a}$	$\overline{b}$	$\overline{a} \wedge \overline{b}$	不可能
#15	正	誤	誤	誤	$\perp$	$\overline{b}$	$\perp$	(Alice が右手を開けば可能)
#16	誤	誤	誤	誤	$\perp$	$\overline{b}$	$\perp$	(Alice が右手を開けば可能)

ら (4)（ただし、(4) に関しては Alice と Bob のそれぞれが、プロトコルを実行する前に左手を開く）だけでは、誤りのある初期配置から、誤りが検知されることなく、無効な出力値が出力されてしまう場合がある。コインベース XOR プロトコルでは、Alice と Bob は左手に 3 枚のコイン束を入力において配置する。ここで、それぞれのコイン束の一番上と一番下（掌に接するコイン）のコインの向きは、入力の値に依存しない。(4) の操作では、Alice と Bob が手を開くことで 3 枚のコイン束の一番上のコインの向きが正しいことを示しているが、手を開いたうえでコイン束を反転させて一番下のコインを見せる（向きが正しいことを示す）ことで、初期配置の誤りを検知することができ、無効な出力値が出力されることを防ぐことができる。

COPY を実行するプロトコルにおいては、複製したいビットに対応する 1 枚のコインを Alice が左手に握り、Alice の右手およびテーブル上に、秘密入力に依存しない 2 枚のコイン束がそれぞれ初期配置される。このとき、Alice の右手およびテーブルの上のコインが正しく配置されていることを（Alice が右手を開いて）確認することで、初期配置の誤りを検知することができる。 したがって、AND プロトコルなどに対して議論したような不具合は生じない（誤りを検知するための追加操作は必要ない）。

4. 初期配置に頑健なプロトコルに向けて

前節で述べたように、コインベースプロトコルにおいて初期配置に誤りがあるとき、オリジナルのプロトコルであっても途中で誤りを検知することができる場合がある（表 1 の#5、#6、#9、#10 など）。

一方、プロトコルの途中では誤りを検知することができず、不正な（正しい、または誤った）出力値を導く場合もある（表 1 の#3、#4、#7、#8、#11、#12、#15、#16 など）。文献 [6], [7] で提案されているプロトコルに関しては、秘密入力に向きが依存しないコインが正しく配置されていることを示すこと（本稿では、**フォーマットチェック**と呼ぶ）で、そのような場合を排除することができる。

フォーマットチェックの方法として、前節では、Alice または Bob がプロトコルの Step1 を実行する前に、手を開いてコイン束の最上面を見せる、または、手を開いてコイン束の向きを反転させて最上面（もともと掌に接していた面）を見せる、という手法を紹介した。

この他にも、コインはテーブルの上（だけ）に置かれるようにコインの初期配置を変更して、机上でフォーマットチェックを行ってもよい。具体的には、コインベースプロトコルに以下の変更を加える。(1)Alice および Bob の掌中に初期配置されるコイン束を、掌中の代わりにテーブルの上に配置する。(2) このとき、最上面から秘密入力の情報が漏れてしまう場合には、ダミーのコインを一番上に配置する。(3) 秘密入力に依存しないコインの向きが正しいことをプレイヤー同士で確認する。例えば、コイン束の上から 3 枚目が秘密入力に依存しない場合には、コイン束の上の 2 枚を（そのまま）摘み上げて脇に動かし、3 枚目のコインの向きを確認したうえで、摘み上げた 2 枚のコインを元に戻す。(4) 文献 [6], [7] のピッキング操作によってコイン束を Alice と Bob の掌中に移して、オリジナルのプロトコルの初期配置の状態とする。(5) オリジナルのプロトコルを Step1 から実行する。

5. まとめ

本稿は、コインベースプロトコルに対して、コインの初期配置に誤りがある場合のプロトコルの挙動を考察した。

その結果、プロトコルの途中で可視化されるコインが、本来は現れえない向きとなることで、初期配置の誤りを検知することができる場合があることを示した。また、その他の場合においても、プロトコルにフォーマットチェックの手順を追加することで、配置誤りに対して頑健なプロトコルを構成できることを示した。初期配置誤りを含む異常処理に対して頑健で高機能なプロトコルや、情報セキュリティ教育に適したプロトコルと教材の開発は、今後の課題である。

謝辞. 本研究は JSPS 科研費 JP18H05289 の助成を受けたものです.

参考文献

[1] den Boer, B.: More efficient match-making and satisfiability: the five card trick. In Quisquater, J.J., Vandewalle, J., eds.: Advances in Cryptology — EUROCRYPT '89. Volume 434 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1990) 208–217

[2] Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In Iwata, T., Cheon, J., eds.: Advances in Cryptology – ASIACRYPT 2015. Volume 9452 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2015) 783–807

[3] Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E100.A**(1) (2017) 3–11

[4] Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In Deng, X., Hopcroft, J.E., Xue, J., eds.: Frontiers in Algorithmics. Volume 5598 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 358–369

[5] Mizuki, T., Komano, Y.: Analysis of information leakage due to operative errors in card-based protocols. In Iliopoulos, C.S., Leong, H.W., Sung, W., eds.: Combinatorial Algorithms - 29th International Workshop, IWOCA 2018. Volume 10979 of Lecture Notes in Computer Science., Springer (2018) 250–262

[6] 駒野 雄一, 水木 敬明: コインを用いる新たなマルチパーティ計算. マルチメディア、分散、協調とモバイル (DICOMO2018) シンポジウム, 2018 年, 2H-2, 441–447

[7] Komano, Y., Mizuki, T.: Multi-party computation based on physical coins. In: D. Fagan, C. Martín-Vide, M. O'Neill, M.A. Vega-Rodríguez (eds.) Theory and Practice of Natural Computing - 7th International Conference, TPNC 2018, Dublin, Ireland, December 12-14, 2018, Proceedings, *Lecture Notes in Computer Science*, vol. 11324, pp. 87–98. Springer (2018)