シングルカットフルオープンカードベースプロトコル

品川 和雅[1,2,3,a)]　縫田 光司[4]

概要：シングルカットフルオープンカードベースプロトコルとは、入力情報を保持したカード列にランダムカットを一回だけ施し、そのカード列を全てオープンするタイプのカードベースプロトコルである。この設定は極めてシンプルであるにも関わらず、これまでに Five-Card Trick など数種のプロトコルしか提案されていない。本研究では、シングルカットフルオープンカードベースプロトコルの構成を、文字列の共役化問題という組合せ論的な問題として新しく捉え直す。技術的な貢献として、プロトコルに必要なカード枚数の下界や、プロトコルの探索方法について述べる。プロトコル探索の結果として、三入力の排他的論理和のプロトコルなど、複数の新種のプロトコルが得られたので、これらを紹介する。

キーワード：秘密計算、カードベースプロトコル、シングルカット、フルオープン

Single-Cut Full-Open Card-Based Protocols

Kazumasa Shinagawa[1,2,3,a)]　Koji Nuida[4]

Abstract: Single-cut full-open card-based protocol is a type of card-based protocols in which a random cut is applied only once to a card sequence holding input information and then, the card sequence is fully opened. Although it is extremely simple, only a few protocols including the Five-Card Trick have been proposed so far. In this study, we provide a new perspective to consider the problem of constructing single-cut full-open card-based protocol as a combinatorial problem of string conjugation. As technical contributions, we give the lower bound of the number of cards and the method for protocol search. As a result of protocol search, we found several new protocols including a three-bit XOR protocol.

Keywords: Secure computation, Card-based protocol, Single-cut, Full-open

1. イントロダクション

1.1 カードベースプロトコル

カードベースプロトコル [3,4] とは、物理的に実行できる、カード組を用いた秘密計算プロトコルのことである。通常、計算はコンピュータ内部でブラックボックス的に実行されるが、カードベースプロトコルでは全ての計算がカードの操作という形で視覚化されるため、プロトコルの流れを理解しやすく、安全性についても納得感が高いという特徴がある。このため、暗号理論教育およびセキュリティ教育への応用が考えられている [13]。

カードベースプロトコルの研究は、Crépeau と Kilian [3] によって任意の関数の計算可能性が示されて以降、いかに効率的にプロトコルを構成できるかという課題が中心的であった。効率性の指標の代表的なものはカード枚数である。例えば、2 ビットの論理積 $x_1 \wedge x_2$ のカード枚数の削減には長い研究の歴史がある [1,3,5,6,8,9,12]。

カード枚数と並んで、重要な効率性の指標であると考えられているのが、シャッフル回数である。シャッフルとは、カード列を確率的に並べ替える操作であり、並べ替え方を選ぶ際の乱数は誰にも操作できず、かつ、全プレイヤーにとって秘密になっていなければならない。シャッフル回数に関する結果として、我々 [11] は任意の関数を 1 回の

1 東京工業大学（Tokyo Institute of Technology）
2 産業技術総合研究所（National Institute of Advanced Industrial Science and Technology）
3 第一著者は JSPS 科研費 17J01169 の助成を受けている。
4 東京大学（The University of Tokyo）
a) shinagawakazumasa@gmail.com

シャッフルで計算する汎用的なプロトコルを示した。そのプロトコルのカード枚数は $2n + 24q$ 枚（n は入力長、q は回路の素子数）であり、n と q の線形枚数であるため、効率的である。したがって、シャッフル回数とカード枚数の両方が効率的な計算は実現可能である。

プロトコルの効率性は、シャッフル回数とカード枚数の二つの指標に基づいて評価されるが、実際にプロトコルを実行する場合、シャッフルの種類も重要である。実行しやすいシャッフルの代表例には、**ランダムカット**（次節で詳述）[4] やランダム二等分割カット [8] がある。特に、ランダムカットは最も単純に実行できるシャッフルであると考えられている。シャッフルの種類の観点から述べると、我々の論文 [11] の 1 回シャッフルのプロトコルは、性質の良いシャッフルのクラス[*1]に含まれるシャッフルであるが、ランダムカットやランダム二等分割カット等と比べると実行しやすいとは言い難い[*2]。

1.2 シングルカットフルオープンカードベースプロトコル

本研究が対象とするプロトコルは、シングルカットフルオープンカードベースプロトコル [10] である。これは以下で定義する**シングルカット性**と**フルオープン性**を同時に満たすカードベースプロトコルである。

(1) シャッフルはランダムカットを 1 回だけ用いること（シングルカット性）
(2) プロトコルの最後で全てのカードを表にめくること（フルオープン性）

ここで、ランダムカットとは〈左端のカードを右端に移動する〉という巡回操作をランダムな回数だけ行うシャッフルである。ℓ 枚のカード列にランダムカットを適用したとき、巡回操作の回数 r は $\{0, 1, 2, \cdots, \ell - 1\}$ から一様ランダムに選ばれる。さらに、r 自体は誰にも（シャッフルを実行しているプレイヤーたちにも）知ることはできないようにシャッフルを行う。（これがランダムカットの安全性である。どのように安全なランダムカットを実現するかについては、4 章を参照されたい。）例えば、以下の 3 枚のカード列 ABC にランダムカットを適用すると、適用後のカード列は $\{ABC, BCA, CAB\}$ の中から一様ランダムに（確率 1/3 で）選ばれる。

$$\boxed{A}\boxed{B}\boxed{C} \xrightarrow{\text{RandomCut}} \begin{cases} \boxed{A}\boxed{B}\boxed{C} & \text{確率 } 1/3 \\ \boxed{B}\boxed{C}\boxed{A} & \text{確率 } 1/3 \\ \boxed{C}\boxed{A}\boxed{B} & \text{確率 } 1/3 \end{cases}$$

ここで、上記のカード列の実際の見た目は $\boxed{?}\boxed{?}\boxed{?}$ である。

[*1] 一様閉シャッフル（uniform and closed shuffles）というクラスに入っている。ランダムカットやランダム二等分割カットなど、全ての効率的なシャッフルはこのクラスに入っている。

[*2] ただし、2 回のシャッフルを要する代わりに、パイルスクランブルシャッフル [2] で実行できる等価なプロトコルも提案している [11]。

シングルカットフルオープンカードベースプロトコルの特徴は以下のとおりである。

- 実装が簡単であること。ランダムカットは最も実装しやすいシャッフルの一つであると考えられている。また、シャッフル回数が 0 回では非自明な関数は計算できない。したがって、ランダムカット 1 回は、シャッフルの観点から最も単純な設定であると言える。
- プロトコル終了時に秘密情報が残っていないこと。フルオープン性を満たさないプロトコルの場合、プロトコル終了時に裏返したままのカード列（それをめくってしまうと入力情報が漏れるカード列）が残ってしまう。その場合、カードを片付けるときに再びシャッフルする等の工夫が必要となってしまう。

したがって、シングルカットフルオープンカードベースプロトコルは最も単純な設定のカードベースプロトコルであると言ってよい。

1.3 既存研究

歴史的に最初に提案されたカードベースプロトコルである Five-Card Trick [4] は、論理積 $x_1 \wedge x_2$ を計算するシングルカットフルオープンカードベースプロトコルである。記号 ♣, ♡ をそれぞれ $0, 1$ に対応させたとき、以下のように 5 枚のカードを並べる。

$$\underset{\overline{x}}{\boxed{?}}\underset{x}{\boxed{?}}\underset{\heartsuit}{\boxed{?}}\underset{y}{\boxed{?}}\underset{\overline{y}}{\boxed{?}}$$

例えば、$(x, y) = (1, 1)$ のとき、カード列は ♣♡♡♡♣ となる。このカード列にランダムカットを適用し、全てめくり、巡回的に ♣♡♡♡♣ と同一のパターンになったら 1 を出力し、巡回的に ♡♣♡♡♣ と同一ならば 0 を出力する。ポイントは、出力結果が 0 となる全ての場合（入力が $(x, y) \in \{(0, 0), (0, 1), (1, 0)\}$）において、巡回的に同一のパターンになることである。

この研究以降、さまざまなカードベースプロトコルが提案されてきたが、シングルカットフルオープンカードベースプロトコルは最近まで全く提案されていなかった。

約 30 年（1989 年から 2018 年）の月日を経て、次に提案されたシングルカットフルオープンカードベースプロトコルは、Shinagawa と Mizuki [10] によって提案された 3 ビットの等号判定 $(x = y = z)$? を行うものである。このプロトコルでは、以下のようにカード列を並べる。

$$\underset{x}{\boxed{?}}\underset{\overline{y}}{\boxed{?}}\underset{z}{\boxed{?}}\underset{\overline{x}}{\boxed{?}}\underset{y}{\boxed{?}}\underset{\overline{z}}{\boxed{?}}$$

出力結果が 0 のとき（$x = y = z$ が成り立つとき）は巡回的に ♣♡♣♡♣♡ と同一になり、1 のときは巡回的に ♣♣♣♡♡♡ と同一になる。また、2 ビットの等号判定 $(x = y)$? は以下のカード列で実現できることも同じ論文で言及されている。

$$\underset{x}{\boxed{?}}\underset{\overline{x}}{\boxed{?}}\underset{y}{\boxed{?}}\underset{\overline{y}}{\boxed{?}}$$

(なお、この論文はシングルカット性とフルオープン性[*3]を初めて定義した論文でもある。)

したがって、現在までに提案されているシングルカットフルオープンカードベースプロトコルは、著者らの知る限り、以下の 3 つの関数を計算するものしか存在しない。

- 2 ビットの論理積 $x \wedge y$
- 2 ビットの等号判定 $(x = y)$?
- 3 ビットの等号判定 $(x = y = z)$?

1.4 本研究の貢献

本研究の貢献は、シングルカットフルオープンカードベースプロトコルの基本性質を示したことと、計算機を用いたプロトコル探索を行ったことである。

2 章では、基本性質を示す。まず、プロトコルの流れを限定して考えても問題ないことを示す（2.1 節)。次に、プロトコルを構成する問題と、文字列の共役化問題との間の関連について示す（2.2 節)。この視点に基づいて、カード枚数の下界を示す（2.3 節)。

3 章では、計算機を用いたプロトコル探索を行う。まず、プロトコル探索アルゴリズムのアイディアを示す（3.1 節)。これを用いて、3 変数の場合（3.4 節)、4 変数の場合（3.5 節)、5 変数以上の場合（3.6 節）についての探索を行う。特に、3 ビットの XOR 関数 $x \oplus y \oplus z$ を計算する 8 枚のプロトコルを初めて発見した。

4 章では、シングルカットフルオープンカードベースプロトコルの実装方法のアイディアについて述べる。

2. プロトコルの基本性質

本章では、シングルカットフルオープンカードベースプロトコルの基本性質を示す。

2.1 プロトコルの典型パターンの考察

本節の議論は、以下の前提[*4]に基づく。

- プロトコル中に許される操作は以下に限る。
  - カード列を並べること。(並べ方は入力情報に依存してよい。ただし、各カードの記号が入力の高々 1 ビットに依存するような依存の仕方でなければならない。)
  - カード列にシャッフルを施すこと。
  - カード列をめくる（あるいは伏せる）こと。
  - カード列を並び替えること。
- ランダムカットは、全て裏向きのカード列 $\boxed{?}\boxed{?}\cdots\boxed{?}$ に適用するものとし、表向きのカードを含むカード列に対する同様のシャッフルは、異なるシャッフルであるとみなす。
- 出力情報から入力情報の少なくとも 1 ビットを必ず推測できる関数を、自明な関数と呼ぶこととする。

一つ目は、カードベース暗号の最も標準的な設定である。二つ目は、$\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{?}$ のようなカード列に対する巡回的なシャッフルを、ランダムカットとは別のシャッフルであると考えるという意味である[*5]。三つ目は、自明な関数の定義であり、例えば $f(x_1, x_2) = x_1$ は自明な関数である。

次の命題は、シングルカットフルオープンカードベースプロトコルは特定の形式に限定してよいことを述べている。

命題 1. 非自明な関数 $f(x_1, x_2, \cdots, x_n)$ を計算するシングルカットフルオープンカードベースプロトコルが存在するならば、f を計算する以下の形式のプロトコルが存在する。
(1) カード列を裏向きで並べる。
(2) カード列全体にランダムカットを適用する。
(3) 全てのカードをめくり、出力情報を決定する。

証明. シングルカットフルオープンカードベースプロトコルは、一般的に以下の形式になる。
(i) カード列を（表向き裏向き混在で）並べる。
(ii) めくる操作と並べ替え操作を任意の回数適用する。
(iii) カード列にランダムカットを適用する。
(iv) めくる操作と並べ替え操作を任意の回数適用する。
(v) 全てのカードをめくり、出力情報を決定する。
なぜなら、最初にカード列を並べることと、途中でランダムカットを適用することと、最後にフルオープンして出力情報を決定することは決まっているので、めくる操作／並べ替え操作をその間に挿入する自由度しかないからである。

まず、(iii) のランダムカットはカード列の全体に適用すると考えてよい。なぜなら、ランダムカットを適用しないカードを (v) でめくったとき、もしそのカードが変数カード（x_i または $\overline{x_i}$ のカード）なら安全性を損ない、定数カード（♣ または ♡ のカード）なら何の意味も持たないためそのカードを除外しても問題ないからである。

次に、(ii) と (iv) が実質的な意味を持たないことを示す。(i) においてカード列は表裏混在で並べられているが、安全性の観点から変数カードは全て裏向きである。(ii) の並べ替え操作は、(i) の並べ方に吸収されると考えてよい。(ii) のめくる操作は、安全性の観点から定数カードにしか適用できないが、その操作に意味はなく、(i) の段階で全て裏向きになっているとしてよい。(iv) のめくる操作は (v) のめくる操作に吸収されると考えてよい。(iv) の並べ替え操作は、出力情報の決定の仕方に吸収されると考えてよい。

[*3] 論文 [10] では、フルオープン性は garbage-freeness という用語が使われている。
[*4] この前提自体はシングルカットフルオープンカードベースプロトコルに限ったものではなく、カードベースプロトコル全般を想定したものである。
[*5] このタイプのシャッフルを含めた考察は今後の研究課題とする。なお、このタイプのシャッフルを認めると、ランダム二等分割カット（カード列を等枚数の 2 つのブロックに分割しランダムに入れ替えるシャッフル）およびパイルシフティングシャッフル（カード列を等枚数の ℓ 個のブロックに分割しランダムに巡回させるシャッフル）が可能となる。

したがって、(i) においてカード列は全て裏向きで、(ii) と (iv) は考えなくてよく、(iii) のランダムカットはカード列全体に適用されるとしてよいため、命題が示された。□

この命題から、シングルカットフルオープンカードベースプロトコルの構成の自由度は、最初の並べ方のみであることが分かる。なお、出力情報の決定の仕方は、安全性の観点から、巡回的に同じパターンには同じ出力値を割り当てなければならないため、(等価な関数同士を同一視するのなら) 一意に定まる。

2.2 文字列の共役化問題との関連

本節では、シングルカットフルオープンカードベースプロトコルを構成する問題と、文字列の共役化問題という組合せ論的な問題との関連について述べる。なお、文字列の**共役**は語の組合せ論という分野で知られている概念であるが、文字列の**共役化**は本稿で新しく定義する概念である。

まず、文字列の共役と共役化の定義を述べる。

共役とは、二つの文字列が巡回的に同じパターンでるという性質のことである。

定義 1 (共役 [7]). 文字列 $s,t \in \{0,1\}^n$ に対して、$u,v \in \{0,1\}^*$ が存在して、$s = uv$ かつ $t = vu$ となるとき、s と t は互いに共役であるといい、$s \approx t$ と表す。

互いに同じ長さの文字列の集合が与えられたときに、その全ての文字列の**同じ位置**に 0 (または 1) の定数列を挿入することを、0 挿入 (1 挿入) と呼ぶ。例えば、$s = 0011$ および $t = 0101$ に対して、先頭に 1 を追加して $\tilde{s} = \underline{1}0011$ および $\tilde{t} = \underline{1}0101$ とするのは 1 挿入であるが、それぞれ別の位置に 1 を挿入して $s' = 0\underline{1}011$ および $t' = 0101\underline{1}$ とするのは 1 挿入ではない。0 挿入と 1 挿入を任意回繰り返して、文字列の全てのペアが互いに共役になるとき、その文字列の集合を共役化可能という。

定義 2 (共役化可能). 文字列の集合 $S \subset \{0,1\}^n$ が与えられたときに、0 挿入と 1 挿入を任意位置/任意回施して、その全てが共役になるとき、S を共役化可能という。

例 1. $s = 110000$ と $t = 101000$ は共役化可能である。

$$\begin{array}{l} s = 110000 \\ t = 101000 \end{array} \rightarrow \begin{array}{l} \tilde{s} = 110\underline{1}0\underline{1}0\underline{1}0 \\ \tilde{t} = 101\underline{1}0\underline{1}0\underline{1}0 \end{array}$$

共役化問題とは、文字列の集合 $S \subset \{0,1\}^n$ が与えられたときに、S が共役化可能か否かを判定し、共役化可能な場合は 0 挿入／1 挿入の仕方を見つける問題である。

共役化問題は、シングルカットフルオープンカードベースプロトコルを構成する問題と関連がある。これを見るためには、プロトコルの真理値表を書くとよい。例として、Five-Card Trick の $\overline{x}\,x\,1\,y\,\overline{y}$ の真理値表を載せる。

$\overline{x}$	x	1	y	$\overline{y}$
1	0	1	0	1
1	0	1	1	0
0	1	1	0	1
0	1	1	1	0

上三段は出力結果 0 に対応し、互いに共役である。ここで、このプロトコルの設計方法として、以下の真理値表

$\overline{x}$	x	y	$\overline{y}$
1	0	0	1
1	0	1	0
0	1	0	1
0	1	1	0

の真ん中に 1 挿入をしたと解釈することもできる。この場合、上三段の文字列 $\{1001, 1010, 0101\}$ の共役化問題を解いたことになる[*6]。

変数 $x_1, x_2, \cdots, x_n$ の順序付き真理値表 (重複列があってもよいもの) の全体の集合を $\mathcal{T}_n$ とする。上記の観察から、次の命題が成り立つ。

命題 2. 関数 $f : \{0,1\}^n \to \{0,1\}$ を計算するシングルカットフルオープンカードベースプロトコルが存在するのは、以下の条件を満たす順序付き真理値表 $T \in \mathcal{T}_n$ が存在するとき、かつ、そのときに限る。

(1) 出力値が 0 の行に対応する文字列の集合は、ある 0/1 挿入で共役化可能である。

(2) 出力値が 1 の行に対応する文字列の集合は、ある 0/1 挿入で共役化可能である。

(3) *(1)* と *(2)* を同時に満たす 0/1 挿入が存在して、かつ、*(1)* と *(2)* は異なる共役類に属する。

2.3 カード枚数の下界

本節では、シングルカットフルオープンカードベースプロトコルのカード枚数の下界を求める。

$f : \{0,1\}^n \to \{0,1\}$ を n 変数のブール関数とする。$b \in \{0,1\}$ に対して、$N_b(f) = |\{x \mid f(x) = b\}|$ を定義する。すなわち、N_b はブール関数を入力として受け取り、出力値が b となる入力の総数を出力する汎関数である。定義より、任意の f に対して $N_0(f) + N_1(f) = 2^n$ が成り立つ。

例 2. n ビットの論理積 $f(x_1, x_2, \cdots, x_n) = x_1 \wedge x_2 \wedge \cdots \wedge x_n$ に対して、$N_0(f) = 2^n - 1$ および $N_1(f) = 1$ である。

定理 1. 任意の関数 $f : \{0,1\}^n \to \{0,1\}$ に対して、f を計算するシングルカットフルオープンカードベースプロトコルは $\max(N_0(f), N_1(f))$ 枚以上のカードを必要とする。

証明. f を計算するプロトコルの真理値表を T とする。このとき、T の $N_0(f)$ 行と $N_1(f)$ 行はそれぞれ共役になっ

[*6] 正確には、上三段が互いに共役になり、最下段がそれらと共役でないような 0/1 挿入を与えたことになる。

ている。一方で、真理値表の性質より、それぞれの行は互いに異なる文字列になっている。長さ k の文字列は、高々 k 種類の共役な文字列しか持てないので、真理値表の列数は $\max(N_0(f), N_1(f))$ 以上必要である。 □

系 1. 任意の関数 $f : \{0,1\}^n \to \{0,1\}$ に対して、f を計算するシングルカットフルオープンカードベースプロトコルは 2^{n-1} 枚以上のカードを必要とする。

3. 計算機を用いたプロトコル探索

本章では、計算機を用いたシングルカットフルオープンカードベースプロトコルの探索の手法とその結果を示す。3.1 節ではプロトコル探索アルゴリズムの概要を述べ、3.2 節ではその動作例を示す。3.3 節ではアルゴリズムの改良手法について述べる。3.4 節、3.5 節、3.6 節では探索結果について述べる。

3.1 プロトコル探索アルゴリズムの概要

プロトコル探索アルゴリズムは、プロトコル中で用いるカードの種類を規定する情報（例：“x のコミットメントを 1 個、y のコミットメントを 2 個、0 のカードを 1 枚の計 7 枚”）を入力として受け取り、そのカードのみを用いて実現可能なシングルカットフルオープンカードベースプロトコルを全て列挙するアルゴリズムである。

アルゴリズムの概略は以下の通りである。

(1) ℓ をカード枚数とする。全ての置換 $\pi \in S_\ell$ を順に生成し、以下を実行する。
  (a) カードを置換 π にしたがって並べる。
  (b) 各変数に 0/1 を割り当て、真理値表を作成する。（変数が n 個あるとき、真理値表は 2^n 行ある。）
  (c) 真理値表の各行に対して、以下のルールで番号をつける。もし以前の行と共役ならば同じ番号をつけ、そうでないならば新しい番号をつける。
  (d) 2^n 個の番号が二種類の数字であり、その真理値表が計算している関数がプロトコルリスト P（初期状態は空リスト）にまだ追加されていないとき、その関数と並び方の情報を P に追加する。
(2) プロトコルリスト P を出力する。

3.2 プロトコル探索アルゴリズムの動作例

入力として “x, y, z のコミットメントをそれぞれ 1 個ずつの計 6 枚” という情報を与えた場合の動作例を示す。

入力の形式として、$(x, \overline{x}, y, \overline{y}, z, \overline{z})$ という列をアルゴリズムに渡す。ステップ (1) において、アルゴリズムは $\pi \in S_6$ を順に生成する。ここでは、$\pi = (2\ 4\ 6)$ が生成されたとする。ステップ (a) において、カード列は $(x, \overline{z}, y, \overline{x}, z, \overline{y})$ と並べられる。ステップ (b) において、真理値表は以下の表（の右端の番号の列を無視したもの）になる。

x	$\overline{z}$	y	$\overline{x}$	z	$\overline{y}$	番号
0	1	0	1	0	1	0
0	0	0	1	1	1	1
0	1	1	1	0	0	1
0	0	1	1	1	0	1
1	1	0	0	0	1	1
1	0	0	0	1	1	1
1	1	1	0	0	0	1
1	0	1	0	1	0	0

ステップ (c) において、上の表のように番号をつける（番号の数字として 0 以上の自然数を用いたが、数字はなんでもよい）。ステップ (d) において、プロトコルリスト P に追加するかどうかの判定を行う。この例の場合、番号は二種類の数字（0 と 1）から成るので、一つ目の条件は満たされる。この関数が P に追加されていないならば、二つ目の条件も満たされ、P に $(x, \overline{x}, y, \overline{y}, z, \overline{z})$ および関数値 01111110（あるいはその 10 進数表記の 126）を追加する。

プログラムを実際に実行すると、出力結果（プロトコルリスト P）は以下の関数たちが得られる。

- 関数 00100100、カード列 $(x, y, z, \overline{x}, \overline{y}, \overline{z})$
- 関数 00011000、カード列 $(x, y, \overline{z}, \overline{x}, \overline{y}, z)$
- 関数 01111110、カード列 $(x, \overline{z}, y, \overline{x}, z, \overline{y})$
- 関数 01000010、カード列 $(x, \overline{y}, \overline{z}, \overline{x}, y, z)$

3.3 アルゴリズムの改良手法

3.1 節のアルゴリズムは、概要を示すために簡略化したものである。実際の実装では、以下の改良を行っている。

- ステップ (1) において、先頭の要素を固定した全ての置換（$(n-1)!$ 個）を探索する。巡回的に同一であるカード列（例：$(x, \overline{z}, y, \overline{x}, z, \overline{y})$ と $(\overline{z}, y, \overline{x}, z, \overline{y}, x)$）は同一の関数を計算するので、この変更を行っても出力結果に影響はない。
- ステップ (1) において置換を生成した際に、以前探索したカード列が再び得られた場合、ただちにスキップする。（$(x, x, \overline{x}, \overline{x}, y, \overline{y})$ のように同一のカード種がある場合に効果がある。この例の場合、$\pi(1) > \pi(2)$ または $\pi(3) > \pi(4)$ の場合にただちにスキップすればよい。）

3.4 3 変数プロトコルの探索

表 1 に 3 変数のシングルカットフルオープンカードベースプロトコルの探索結果を示す。得られたプロトコルの詳細については付録を参照されたい。ただし、変数 xyz は変数リスト $(x, \overline{x}, y, \overline{y}, z, \overline{z})$ を表しているなど、各変数はその否定と同数出現させている。また、関数クラスの表記として以下を用いている。

- $\mathcal{F}_1 := \{(x = y = z)?, (\overline{x} = y = z)?, (x = \overline{y} = z)?, (x = y = \overline{z})?\}$

表 1 3 変数のプロトコル探索結果

変数	カード枚数	計算できる関数
xyz	6	$\mathcal{F}_1$
$xxyz$	8	$\mathcal{F}_1 \cup \mathcal{F}_2 \cup \{x \oplus y \oplus z\}$
$xxxyz$	10	$\mathcal{F}_1$
$xxxxyz$	12	$\mathcal{F}_1 \cup \{x \oplus y \oplus z, x\}$
$xxyyz$	10	$\mathcal{F}_1 \cup \mathcal{F}_2 \cup \{y \oplus z\}$
$xxxyyz$	12	$\mathcal{F}_1 \cup \mathcal{F}_2 \cup \{y \oplus z\}$
$xxyyzz$	12	$\mathcal{F}_1 \cup \{x \oplus y \oplus z\}$
$xyz1$	7	–
$xxyz1$	9	–
$xxxyz1$	11	–
$xxyyz1$	11	–
$xyz11$	8	–
$xxyz11$	10	–
$xxxyz11$	12	–
$xxyyz11$	12	–
$xyz111$	9	–
$xxyz111$	11	–
$xyz1111$	10	–
$xxyz1111$	12	$\{x \oplus y \oplus z\}$
$xyz11111$	11	–
$xyz111111$	12	$\mathcal{F}_1$
$xyz10$	8	$\mathcal{F}_3$
$xxyz10$	10	–
$xxxyz10$	12	–
$xxyyz10$	12	–
$xyz110$	9	–
$xxyz110$	11	–
$xyz1110$	10	–
$xxyz1110$	12	–
$xyz11110$	11	–
$xyz111110$	12	–
$xyz1100$	10	–
$xxyz1100$	12	–
$xyz11100$	11	–
$xyz111000$	12	–

表 2 4 変数のプロトコル探索結果

変数	カード枚数	計算できる関数
$xyzw$	8	12 種類
$xxyzw$	10	–
$xxxyzw$	12	–
$xxyyzw$	12	–
$xyzw1$	9	–
$xxyzw1$	11	–
$xyzw11$	10	–
$xxyzw11$	12	–
$xyzw10$	10	–
$xxyzw10$	12	–
$xyzw110$	11	–
$xyzw1100$	12	–

- $\mathcal{F}_2 := \{x \oplus y, x \oplus z\}$
- $\mathcal{F}_3 := \{f(y,\overline{x},z), f(y,\overline{z},x), f(y,z,x), f(y,x,z), f(x,y,z), f(x,\overline{y},z), f(x,z,y), f(x,\overline{z},y), f(z,\overline{y},x), f(z,y,x), f(z,\overline{x},y), f(z,x,y)\}$、ただし $f(\alpha,\beta,\gamma)$ は $\alpha = 0$ ならば β を、$\alpha = 1$ ならば γ を出力する。

特筆すべきこととして、$x \oplus y \oplus z$ のプロトコルが発見されたことがある。(2 ビットの XOR は $(x = y)$? と等価なので、すでに知られていたことに注意されたい。) このプロトコルは $x\,y\,\overline{x}\,z\,x\,\overline{y}\,\overline{x}\,\overline{z}$ という並べ方によって実現される。真理値表は以下の通りである。

x	y	$\overline{x}$	z	x	$\overline{y}$	$\overline{x}$	$\overline{z}$	$x \oplus y \oplus z$
0	0	1	0	0	1	1	1	0
0	0	1	1	0	1	1	0	1
0	1	1	0	0	0	1	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	1	1	0	1	1
1	0	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	0	1

もう一つ特筆すべき現象として、変数 $xxxyz$ の 12 枚のカードで関数 x が計算できていることが挙げられる。もちろん、関数 x はシャッフルなしでも計算できる自明な関数であるが、y と z の情報を完全に消しているという点で興味深い。これは $x\,\overline{x}\,y\,x\,\overline{x}\,z\,x\,\overline{x}\,\overline{y}\,x\,\overline{x}\,\overline{z}$ という並べ方によって実現される。真理値表は以下の通りである。

x	$\overline{x}$	y	x	$\overline{x}$	z	x	$\overline{x}$	$\overline{y}$	x	$\overline{x}$	$\overline{z}$	x
0	1	0	0	1	0	0	1	1	0	1	1	0
0	1	0	0	1	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1	0	0	1	0	0
1	0	0	1	0	0	1	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1	1	0	0	1
1	0	1	1	0	0	1	0	0	1	0	1	1
1	0	1	1	0	1	1	0	0	1	0	0	1

3.5 4 変数プロトコルの探索

表 2 に 4 変数の単純プロトコルの探索結果を示す。得られたプロトコルの詳細については付録を参照されたい。変数 $xyzw$ では 12 種類の関数を発見できたが、それ以外の変数ではプロトコルを発見できなかった。なお、12 種類の全ての関数において、出力値が 0 となる入力の個数と 1 となる入力の個数が等しい性質が成り立っていた。すなわち、$N_0(f) = N_1(f)$ が成り立っていた（N_b の定義については 2.3 節を参照)。

3.6 5 変数以上のプロトコルの探索

5 変数と 6 変数において、それぞれの変数がちょうど一

度ずつ出現する場合（変数 $xyzwu$ および $xyzwuv$）では、プロトコルは存在しなかった。

4. プロトコル実装のアイディア

本章では、プロトコル実装のアイディア、特にランダムカットの実装のアイディアを紹介する。4.1 節では、よく知られた手法として、ヒンドゥーカットを用いた実装を紹介する。4.2 節では、コマを用いた実装として、磁石型コマとケース型コマを紹介する。

4.1 ヒンドゥーカットを用いた実装

シングルカットフルオープンカードベースプロトコルの物理的な実装をする上で、最も重要な部分がランダムカットの実装である。ランダムカットの実装方法として、ヒンドゥーカットによる実装が知られている。

ヒンドゥーカットとは、カード列を束にまとめて片手で持ち、「束を適当に二分割してその上下を入れ替える」という操作を十分な回数行うシャッフルである。この操作は巡回操作を適当な回数行ったものと等価であることに注意されたい。束を適当に二分割するところがポイントであり、上下の束の枚数が毎回確率的に異なるため、巡回操作が何回行われたかを数えることが格段に難しくなる。ヒンドゥーカットはトランプゲームなどで広く用いられているシャッフル操作であり、そのことはヒンドゥーカットの安全性が広く信じられていることを意味する。

4.2 コマを用いた実装

ヒンドゥーカットによる実装は、カード以外の道具を用いないため手軽に実行できるという利点があるが、一方で操作に慣れていないと上手に実行するのが難しいという問題もある。そこで、別の実装方法として、コマによる実装を紹介する。これは、コマの盤面にカードを並べて固定し、それを回転することによってランダムな回数の巡回操作を実現するという方法である。

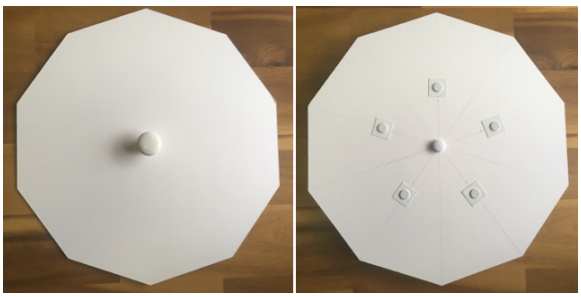

図 1 磁石型コマ：表（左）と裏（右）

図 1 は 5 枚用の磁石型コマの写真である。厚さ 1mm の厚紙をカッターで正 10 角形に加工し、中心に穴を開け、取っ手とネジを装着している。磁石型コマの特徴として、

図 2 磁石型コマ：磁石つきカード（左）とカードセット済み（右）

裏面に強力マグネットを（5 箇所に）貼っている（図 1 右）。同じく強力マグネットを貼ったカード（図 2 左）を図 2 右のようにセットすれば準備完了である。コマを回して見ると、非常に速く回転するため、肉眼で追うことは不可能である。回し終わった後は、再びカードを外し、オープンすればよい。

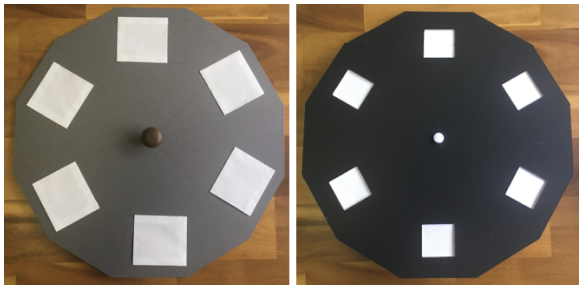

図 3 ケース型コマ：表（左）と裏（右）

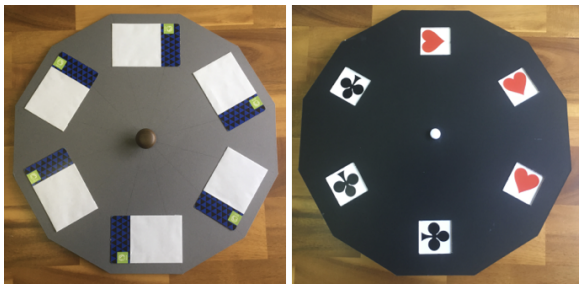

図 4 ケース型コマ：カードセット済み（左）と裏（右）

図 3 は 6 枚用のケース型コマの写真である。磁石型コマと同じように、厚さ 1mm の厚紙をカッターで正 12 角形に加工し、中心に穴を開け、取っ手とネジを装着している。ケース型コマの特徴として、各カードのセット位置に折り紙で作ったポケットが貼り付けられている（図 3 左の白い四角）。また、コマの台紙のポケットに相当する部分に穴を開けてある（図 3 右の白い四角）。この穴のおかげで、カードをセットし（図 4 左）、最後オープンしたとき、ポケットからカードを取り出さなくとも、コマを裏返すだけでオープンできる（図 4 右）。なお、穴を開けることによるオープンの簡略化は、磁石型コマにも適用できる。

付　　録

A.1　3 変数プロトコルのリスト

$\mathcal{F}_1 = \{(x = y = z)?, (\overline{x} = y = z)?, (x = \overline{y} = z)?, (x = y = \overline{z})?\}$ を計算する最小枚数のプロトコルを列挙する。

- $x\,\overline{y}\,z\,\overline{x}\,y\,\overline{z}$ — $(x = y = z)?$
- $x\,y\,\overline{z}\,\overline{x}\,\overline{y}\,z$ — $(\overline{x} = y = z)?$
- $x\,y\,z\,\overline{x}\,\overline{y}\,\overline{z}$ — $(x = \overline{y} = z)?$
- $x\,\overline{y}\,\overline{z}\,\overline{x}\,y\,z$ — $(x = y = \overline{z})?$

XOR 計算 $\{x \oplus y \oplus z, x \oplus y, y \oplus z, z \oplus x\}$ を計算する最小枚数のプロトコルを列挙する。

- $x\,y\,\overline{x}\,z\,x\,\overline{y}\,\overline{x}\,\overline{z}$ — $x \oplus y \oplus z$
- $x\,y\,x\,z\,\overline{x}\,\overline{y}\,\overline{x}\,\overline{z}$ — $x \oplus y$
- $y\,z\,y\,x\,\overline{y}\,\overline{z}\,\overline{y}\,\overline{x}$ — $y \oplus z$
- $x\,y\,\overline{x}\,z\,\overline{x}\,\overline{y}\,x\,\overline{z}$ — $z \oplus x$

$\mathcal{F}_3 = \{f(y,\overline{x},z), f(y,\overline{z},x), f(y,z,x), f(y,x,z), f(x,y,z), f(x,\overline{y},z), f(x,z,y), f(x,\overline{z},y), f(z,\overline{y},x), f(z,y,x), f(z,\overline{x},y), f(z,x,y)\}$ を計算する最小枚数のプロトコルを列挙する。

- $x\,y\,z\,0\,\overline{x}\,\overline{y}\,\overline{z}\,1$ — $f(y,\overline{x},z)$
- $x\,y\,z\,1\,\overline{x}\,\overline{y}\,\overline{z}\,0$ — $f(y,\overline{z},x)$
- $x\,y\,\overline{z}\,1\,\overline{x}\,\overline{y}\,z\,0$ — $f(y,z,x)$
- $x\,y\,\overline{z}\,0\,\overline{x}\,\overline{y}\,z\,1$ — $f(y,x,z)$
- $x\,y\,1\,z\,\overline{x}\,\overline{y}\,0\,\overline{z}$ — $f(x,y,z)$
- $x\,y\,1\,\overline{z}\,\overline{x}\,\overline{y}\,0\,z$ — $f(x,\overline{y},z)$
- $x\,y\,0\,z\,\overline{x}\,\overline{y}\,1\,\overline{z}$ — $f(x,z,y)$
- $x\,y\,0\,\overline{z}\,\overline{x}\,\overline{y}\,1\,z$ — $f(x,\overline{z},y)$
- $x\,z\,y\,1\,\overline{x}\,\overline{z}\,\overline{y}\,0$ — $f(z,\overline{y},x)$
- $x\,z\,\overline{y}\,1\,\overline{x}\,\overline{z}\,y\,0$ — $f(z,y,x)$
- $x\,z\,y\,0\,\overline{x}\,\overline{z}\,\overline{y}\,1$ — $f(z,\overline{x},y)$
- $x\,z\,\overline{y}\,0\,\overline{x}\,\overline{z}\,y\,1$ — $f(z,x,y)$

A.2　4 変数プロトコル

変数 $xyzw$（8 枚）の 4 変数プロトコルを列挙する。

- $x\,y\,z\,w\,\overline{x}\,\overline{y}\,\overline{z}\,\overline{w}$
- $x\,y\,z\,\overline{w}\,\overline{x}\,\overline{y}\,\overline{z}\,w$
- $x\,y\,\overline{z}\,w\,\overline{x}\,\overline{y}\,z\,\overline{w}$
- $x\,y\,\overline{z}\,\overline{w}\,\overline{x}\,\overline{y}\,z\,w$
- $x\,y\,w\,z\,\overline{x}\,\overline{y}\,\overline{w}\,\overline{z}$
- $x\,y\,w\overline{z}\,\overline{x}\,\overline{y}\,\overline{w}\,z$
- $x\,y\,\overline{w}\,z\,\overline{x}\,\overline{y}\,w\,\overline{z}$
- $x\,y\,\overline{w}\,\overline{z}\,\overline{x}\,\overline{y}\,w\,z$
- $x\,z\,y\,w\,\overline{x}\,\overline{z}\,\overline{y}\,\overline{w}$
- $x\,z\,y\,\overline{w}\,\overline{x}\,\overline{z}\,\overline{y}\,w$
- $x\,z\,\overline{y}\,w\,\overline{x}\,\overline{z}\,y\,\overline{w}$
- $x\,z\,\overline{y}\,\overline{w}\,\overline{x}\,\overline{z}\,y\,w$

参考文献

[1] E. Cheung, C. Hawthorne, and P. Lee. Cs 758 project: Secure computation with playing cards, 2013. `https://csclub.uwaterloo.ca/~cdchawth/files/papers/secure_playing_cards.pdf`.

[2] R. Ishikawa, E. Chida, and T. Mizuki. Efficient Card-Based Protocols for Generating a Hidden Random Permutation Without Fixed Points. In *Unconventional Computation and Natural Computation - 14th International Conference, UCNC 2015, Auckland, New Zealand, August 30 - September 3, 2015, Proceedings*, pages 215–226. Springer, 2015.

[3] C. Crépeau and J. Kilian. Discreet solitary games. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 319–330. Springer, 1993.

[4] B. den Boer. More efficient match-making and satisfiability: *The Five Card Trick*. In *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, pages 208–217. Springer, 1989.

[5] J. Kastner, A. Koch, S. Walzer, D. Miyahara, Y. Hayashi, T. Mizuki, and H. Sone. The minimum number of cards in practical card-based protocols. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, pages 126–155. Springer, 2017.

[6] A. Koch, S. Walzer, and K. Härtel. Card-based cryptographic protocols using a minimal number of cards. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, pages 783–807. Springer, 2015.

[7] M. Lothaire. Combinatrics on Words. *Cambridge Mathematical Library, 1997*.

[8] T. Mizuki and H. Sone. Six-card secure AND and four-card secure XOR. In *Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 20-23, 2009. Proceedings*, pages 358–369. Springer, 2009.

[9] V. Niemi and A. Renvall. Secure multiparty computations without computers. *Theor. Comput. Sci.*, 191(1-2):173–183, 1998.

[10] K. Shinagawa and T. Mizuki. The Six-Card Trick: Secure Computation of Three-Input Equality. In *21th International Conference on Information Security and Cryptology, ICISC 2018, November 28-30, 2018, Seoul, South Korea, Proceedings*, pages 123–131. Springer, 2018.

[11] K. Shinagawa and K. Nuida. A Single Shuffle Is Enough for Secure Card-Based Computation of Any Circuit. In IACR Cryptology ePrint Archive, `https://eprint.iacr.org/2019/380`, 2019.

[12] A. Stiglic. Computations with a deck of cards. *Theor. Comput. Sci.*, 259(1-2):671–678, 2001.

[13] 水木 敬明. カードベース暗号の教育への応用. 信学技報, vol. 116, no. 289, ISEC2016-53, pp. 13-17, 2016.