数理最適化に基づくカードベース暗号の不可能性証明

池田 春之介[1]　品川 和雅[1,2,3]

概要：カードベース暗号において、ある条件下でプロトコルが存在しないこと（不可能性）の証明は重要である。特に、カード枚数が多い場合の不可能性証明は一般に極めて難しいため、多くの既存研究はカード枚数が少ない場合の不可能性の結果を報告している。本研究では、カードベース暗号に初めて数理最適化の手法を導入し、カード枚数が多い場合における不可能性証明に取り組む。対象とするプロトコルは、ランダムカットを 1 回適用した後、すべてのカードを公開するタイプのプロトコル（シングルカットフルオープンプロトコル）である。その結果、3 変数ブール関数に対しては、任意枚の ♣（あるいは任意枚の ♡）を追加カードとして用いた場合でも、既知のプロトコル以外に新たなプロトコルは存在しないことが判明した。これは、1 種類の追加カードのみを用いるという制約付きではあるものの、カード枚数を限定しない状況に対する不可能性であり、数理最適化を用いた不可能性証明の枠組みを新たに提供するものである。

キーワード：カードベース暗号、数理最適化、シングルカットフルオープン、不可能性証明

Impossibility Proofs on Card-Based Cryptography Based on Mathematical Optimization

Shunnosuke Ikeda[1]　Kazumasa Shinagawa[1,2,3]

Abstract: In card-based cryptography, demonstrating the impossibility of a protocol under certain conditions is a fundamental problem. Impossibility proofs become especially challenging when the number of cards is large, and consequently, most previous work addresses only cases with few number of cards. In this paper, we introduce mathematical optimization to card-based cryptography for the first time, and establish impossibility results for a large number of cards. We concentrate on single-cut full-open protocols, which perform exactly one random cut and then reveal all cards. We show that, for every three-variable Boolean function, no protocol other than those already known exists if we use any number of ♣ cards (resp., any number of ♡ cards) as helping cards. While the result is obtained under the constraint that only one-colored helping cards may be added, it yields an impossibility proof for any number of cards and provides a new framework for impossibility proofs via mathematical optimization.

Keywords: Card-based cryptography, Mathematical optimization, Single-cut full-open, Impossibility proofs

1. はじめに

カードを用いて秘密計算やゼロ知識証明などの暗号技術を物理的に表現できることが知られており、そのような研究分野をカードベース暗号 [1,2,10] と呼ぶ。通常の暗号技術と異なり、物理的かつ視覚的な形で計算が行われるため、直感的に理解しやすく、セキュリティへの納得感も高い。この特徴から、セキュリティ教育への応用も検討されており、小中学生や大学生の講義 [15,16] に用いられている。

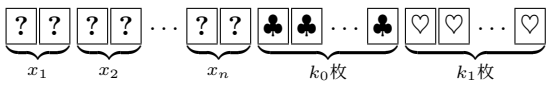
この分野における重要な研究課題のひとつが、特定の条件下でどのような関数が安全に計算可能であるかを明らかにすることである。そのためには、特定の条件下ではその関数を安全に計算できないことの証明（不可能性証明）を行う必要がある。不可能性証明の代表的手法として Koch–Walzer–Hätel による手法 [5–7,9] がある。これは、

[1] 筑波大学 University of Tsukuba
[2] 九州大学 Kyushu University
[3] 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology

プロトコルを有向グラフによって表現し、始頂点から終頂点に遷移できないことを示す手法である。この手法を用いて、AND プロトコルに対して 5 枚以下の不可能性 [5,6,9] が、コピープロトコルに対して $2k+1$ 枚以下（k は複製の個数）の不可能性 [5] が得られているが、どちらも入出力に必要なカード以外に高々 1 枚のみ用いる設定における結果である。一般に、カード枚数が多いときは組合せ爆発が起こるため、さらなる不可能性の結果を得るためには、新しい証明手法の開発が必要であるように思われる。

1.1 シングルカットフルオープンプロトコル

シングルカットフルオープン（Single-Cut Full-Open: SCFO）プロトコルとは、入力カード列にランダムカット（巡回シフトをランダムな回数行うシャッフル操作）を一度だけ適用し、その後すべてのカードを表向きにして公開するプロトコルである。特に、入力 $x_1,\dots,x_n \in \{0,1\}$ のコミットメント（2.3 節）をそれぞれ 1 個ずつ用いて、k_0 枚の ♣ と k_1 枚の ♡ を追加する SCFO プロトコルを (k_0,k_1)-SCFO プロトコルと呼ぶことにする。(k_0,k_1)-SCFO プロトコルの入力カード列は以下の並び替えである。



特に、k_0 の値を気にしないときの SCFO プロトコルを $(*,k_1)$-SCFO プロトコルと呼ぶことにする。すなわち、$(*,k_1)$-SCFO プロトコルでは任意の枚数の ♣ を追加カードとして用いてよい。同様に、k_1 の値を気にしないときの SCFO プロトコルを $(k_0,*)$-SCFO プロトコルと呼ぶ。

2 変数以上のブール関数に対する既存の SCFO プロトコルは、以下の関数に対するものが知られている。

- $x_1 \wedge x_2$（2 変数 AND 関数）[1]
- $x_1 \oplus x_2$（2 変数 XOR 関数）[13]
- $x_1 \oplus x_2 \oplus x_3$（3 変数 XOR 関数）[17]
- $(x_1 = x_2 = x_3)$?（3 変数 EQ 関数）[4,13]
- $(x_1 \wedge x_2) \vee (\overline{x_1} \wedge x_3)$（3 変数関数）[17]
- $x_1\overline{x_2}x_3 \vee x_1\overline{x_3}x_4 \vee \overline{x_1}x_2\overline{x_3} \vee x_1x_3\overline{x_4}$（4 変数関数）[17]

ただし、3 変数 XOR 関数に対する SCFO プロトコルは、x_1 のコミットメントを 2 個用いており、(k_0,k_1)-SCFO プロトコルには含まれない。他の 5 個のプロトコルはすべて (k_0,k_1)-SCFO プロトコルであり、上から順に $(k_0,k_1) = (0,1),(0,0),(0,0),(1,1),(0,0)$ である。

以上をまとめると、2 変数関数については、AND プロトコルと XOR プロトコルにより、任意関数を SCFO プロトコルによって計算可能であることが分かる。一方、3 変数以上の関数については、数少ない関数しか計算できておらず、これ以外に SCFO プロトコルによって計算可能な関数があるかどうかは未解決問題である。

1.2 本研究の貢献

本研究では、カードベース暗号における不可能性証明の新たな手法として、数理最適化に基づく手法を提案する。これは、カード分野に数理最適化を適用した初めての事例であり、本分野における数理最適化の有効性を示したことが本研究の重要な貢献である。具体的な貢献は以下である。

- $(*,0)$-SCFO プロトコルの探索を数理最適化問題として定式化し、関数を入力としてプロトコルの構成可能性を判定し、特にプロトコルが存在する場合は最小枚数の追加カードによるプロトコルを出力するアルゴリズムを構築した。
- このアルゴリズムを計算機実装し、すべての 3 変数ブール関数に対して実験を行った。結果として、3 変数 EQ 関数以外のすべての 3 変数関数に対する $(*,0)$-SCFO プロトコルの不可能性が得られた。

1.3 関連研究：不可能性証明の既存手法

カード分野における不可能性証明の手法を紹介する。提案手法はいずれにも分類されない新しい証明手法である。

- **情報量に基づく証明：**Mizuki–Shizuya [10] は 1 枚符号化に対してはパーフェクトなコピープロトコルが存在しないことを証明した。直感的には、1 枚符号化のプロトコルでは、正当性を満たすためには入力カードをめくる必要があるが、その際には情報が漏れてしまうため、正当性と安全性は両立できないことを意味する。
- **KWH 図式に基づく証明：**Koch–Walzer–Härtel [9] はプロトコルを視覚的に表現する図式（KWH 図式）を提案し、それに基づき有限時間コミット型 4 枚 AND プロトコルの不可能性を証明した。後続研究 [5,7] において多くの結果が証明されているが、この手法はカード枚数が多いときに対しては適用が難しく、既存結果はすべて 2 入力以下かつ 5 枚以下の場合である。
- **形式検証に基づく証明：**Koch–Schrempp–Kirsten [7,8] は形式検証を用いた不可能性証明手法を提案した。この手法もカード枚数が多いときに対しては適用が難しく、不可能性は 4 枚 5 ステップ以下の場合である。
- **ABC 予想に基づく証明：**Hashimoto ら [3] は不動点のない n 次の置換の一様ランダム生成には $\Omega(n\log n)$ のカードが必要であることを証明した。その証明は、不動点のない置換の個数についての素因数分解に基づいており、ABC 予想が用いられている。
- **PSM プロトコルに基づく証明：**Shinagawa–Nuida [14] はカードベースプロトコルを PSM プロトコルに変換する一般的手法を提案した。特に、n 入力 k 枚のシングルシャッフルフルオープン（SSFO）プロトコルを通信量 nk の PSM プロトコルに変換できる。これにより、PSM プロトコルの通信量下界から、SSFO プロトコルのカード枚数下界が得られることが判明した。

2. 準備

2.1 節では本研究が対象とする 3 変数ブール関数について述べる。2.2 節では、「巡回的に等しい」[18] という語の組合せ論の用語を定義する。2.3 節から 2.5 節では、本研究で必要となるカードベース暗号の基本的な定義を述べる。

本稿を通して、1 以上 n 以下の正整数の集合を $[n] := \{1, 2, \ldots, n\}$ と表記する。n 次対称群を $\mathfrak{S}_n$ と表記する。

2.1 3 変数ブール関数

n 変数ブール関数に対して、以下の 3 つの操作の組合せによって得られる同値類を NPN 同値類という。

- 一部またはすべての入力変数の否定
- n 個の入力変数の順序の並べ替え
- 出力結果の否定

関数 $f, g : \{0,1\}^n \to \{0,1\}$ が同じ NPN 同値類に属するとき、f を安全に計算するプロトコルが存在すれば、g を安全に計算するプロトコルも存在する。したがって、任意の n 変数ブール関数に対して安全な計算を行う際は、NPN 同値類の代表元に対するプロトコルを構成すればよい。

3 変数関数の NPN 同値類の代表元は以下の 14 個である。

1. 0（定数関数）
2. x_1（1 変数関数）
3. $x_1 \wedge x_2$（2 変数 AND 関数）
4. $x_1 \oplus x_2$（2 変数 XOR 関数）
5. $x_1 \wedge x_2 \wedge x_3$（3 変数 AND 関数）
6. $x_1 \oplus x_2 \oplus x_3$（3 変数 XOR 関数）
7. $(x_1 = x_2 = x_3)$?（3 変数 EQ 関数）
8. $(x_1 + x_2 + x_3 \geq 2)$?（3 変数 MAJ 関数）
9. $(x_1 + x_2 + x_3 = 1)$?
10. $x_1 \oplus (x_2 \wedge x_3)$
11. $(x_1 \wedge x_2) \vee (\overline{x_1} \wedge x_3)$
12. $x_1 \wedge (x_2 \oplus x_3)$
13. $(x_1 \vee x_2) \wedge (x_1 \oplus x_3)$
14. $x_1 \wedge (x_2 \vee x_3)$

2.2 巡回的に等しいこと

2 個の文字列 $w, w' \in \{0,1\}^n$ が巡回的に等しいとは、ある $u, v \in \{0,1\}^*$ が存在して、$w = uv$ かつ $w' = vu$（ここで uv や vu は文字列の連結を表す）が成り立つことである。巡回的に等しいことを $w \sim w'$ と表す。例えば $w = 10110$ と $w' = 11010$ に対しては、$u = 10$ と $v = 110$ と取れるため、$w \sim w'$ である。

文字列の集合 $\mathcal{W} \subseteq \{0,1\}^n$ が巡回的に等しいとは、任意の $w, w' \in \mathcal{W}$ に対して $w \sim w'$ が成り立つことである。

文字列の集合 $\mathcal{W} \subseteq \{0,1\}^n$ に対する同時挿入とは、ある共通の $u_1, \ldots, u_n \in \{0,1\}^*$（ただし u_i は空の文字列でもよい）について、各文字列 $w = w_1 w_2 \cdots w_n \in \mathcal{W}$ を文字列 $w' = w_1 u_1 w_2 u_2 \cdots w_n u_n$ へと変形する操作のことと定める。同時挿入を $\boldsymbol{u} = (u_1, \ldots, u_n)$ と表し、文字列 w が同時挿入 $\boldsymbol{u}$ によって文字列 w' に変形されることを $w' \xleftarrow{\boldsymbol{u}} w$ と表す。特に、すべての $u_1, \ldots, u_n \in \{0,1\}^*$ が 1 を用いない文字列 $u_i \in \{0\}^*$ であるときの同時挿入を同時 0-挿入と呼ぶ。同時 1-挿入も同様に定める。

2.3 カードと符号化

表が $\boxed{\clubsuit}$ と $\boxed{\heartsuit}$ の 2 種類であり、裏がどちらも $\boxed{?}$ であるようなカード組を用いる。Mizuki–Shizuya モデル [10] では、表向きのカードを ♣/? や ♡/? と表し、裏向きのカードを ?/♣ や ?/♡ と表し、例えば以下のカード列

は (?/♣, ?/♡, ♣/?, ♡/?, ?/♡) と表す。本稿ではカードの表裏が明快な状況のみを扱うため、カード列を ♣♡♣♡♡ のように表し、n 枚のカード列全体の集合を $\{\clubsuit, \heartsuit\}^n$ と表す。

本稿では ♣ $= 0$ および ♡ $= 1$ とする。すなわち、n 枚のカード列全体の集合 $\{\clubsuit, \heartsuit\}^n$ と n 文字列全体の集合 $\{0,1\}^n$ を同一視し、例えば ♣♣♡♡♡ $= 00111$ とする。

ビット値 $x \in \{0,1\}$ に対して、2 枚の裏向きのカード組 $(x, \overline{x})$ を x のコミットメントという。すなわち、$x = 0$ のときは (♣, ♡) であり、$x = 1$ のときは (♡, ♣) である。x のコミットメントを以下のように表す。

カードベース暗号では、入力としてコミットメントの列が与えられる状況を通常は考えるため、入力が n ビットのときは少なくとも $2n$ 枚用いる。コミットメントの代わりに、各ビットを 1 枚で表す符号化（1 枚符号化）[11,12] を用いると、パーフェクトな安全性を達成できないこと [10] が知られており、通常は 1 枚符号化を採用しない。

2.4 ランダムカット

ランダムカットとは、カード列に対してランダムな回数だけ巡回シフトを行うシャッフル操作である。n 枚のカード列 $w = w_1 w_2 \cdots w_n \in \{\clubsuit, \heartsuit\}^n$ に対して、巡回シフト操作を $\sigma(w) = w_2 w_3 \cdots w_n w_1$ と定める。ランダムカットとは、カード列に対して σ^r（ただし $0 \leq r \leq n-1$ は一様乱数）を適用する操作である。ここで、乱数 r は誰にも一切推測できない秘匿された一様乱数である。

言い換えると、ランダムカットを $w \in \{0,1\}^n$ に対して適用すると、w と巡回的に等しい文字列全体の集合から、1 つの文字列が一様ランダムに選ばれる。したがって、$w \sim w'$ のとき、ランダムカットを w に適用したときの結果のカード列の確率分布と、w' に適用したときの結果のカード列の確率分布は、全く等しい。

例えば、裏向きに伏せられた 5 枚のカード列 ♣♣♡♡♡ に対してランダムカットを適用すると、5 通りのカード列 ♣♣♡♡♡, ♣♡♡♡♣, ♡♡♡♣♣, ♡♡♣♣♡, ♡♣♣♡♡ がそれぞれ 1/5 の確率で生じ、実際にどのカード列が選ばれたかは誰にも推測できない。

ランダムカットは、手操作で簡単に実装できることが知られている。また、コマの上にカードを配置して、回転操作によりランダムカットを実装する方法も知られている。

2.5 シングルカットフルオープンプロトコル

シングルカットフルオープン（SCFO）プロトコルとは、入力カード列にランダムカットを一度だけ適用し、その後すべてのカードを表向きに公開するプロトコルである。

計算する関数を $f : \{0,1\}^n \to \{0,1\}$ とする。(k_0, k_1)-SCFO プロトコルでは、$x_1, \ldots, x_n \in \{0,1\}$ のコミットメントに、k_0 枚の $\boxed{\clubsuit}$ と k_1 枚の $\boxed{\heartsuit}$ を追加し、合計 $(2n+k_0+k_1)$ 枚のカードを用いる。このプロトコルは、置換 $\pi \in \mathfrak{S}_{2n}$ と $k_0 + k_1$ 枚の追加カードの挿入位置によって定まる。

プロトコルの手順は以下の通りである。

1. 入力カード列 $v(\boldsymbol{x}) := x_1\overline{x_1}\cdots x_n\overline{x_n}$ を並べる。
2. 置換 $\pi \in \mathfrak{S}_{2n}$ で並び替え、$w(\boldsymbol{x}) := \pi(v(\boldsymbol{x}))$ を得る。
3. カード列 $w(\boldsymbol{x})$ の任意の位置に k_0 枚の $\boxed{\clubsuit}$ と k_1 枚の $\boxed{\heartsuit}$ を挿入し、カード列 $w'(\boldsymbol{x})$ を得る。
4. カード列 $w'(\boldsymbol{x})$ にランダムカットを適用する。
5. すべてのカードをめくる。

出力値が $b \in \{0,1\}$ となる入力集合を $\mathcal{X}^{(b)} := \{\boldsymbol{x} \in \{0,1\}^n \mid f(\boldsymbol{x}) = b\}$ とおく。上記のプロトコルが f を安全に計算するとは、以下の 2 条件を満たすことである。

- (正当性) 任意の $\boldsymbol{x} \in \mathcal{X}_0$ と $\boldsymbol{x}' \in \mathcal{X}_1$ に対して、$w'(\boldsymbol{x})$ と $w'(\boldsymbol{x}')$ は巡回的に等しくないこと
- (安全性) 任意の $b \in \{0,1\}$ について、文字列の集合 $\{w'(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}^{(b)}\}$ は巡回的に等しいこと

3. $(*, 0)$-SCFO プロトコルの探索

本節では、数理最適化に基づいて $(*, 0)$-SCFO プロトコルを探索する手法を提案する。

3.1 探索の概要

探索対象の n 変数ブール関数を $f : \{0,1\}^n \to \{0,1\}$ とする。入力カード列全体の集合を $\mathcal{V} := \{v(\boldsymbol{x}) \mid \boldsymbol{x} \in \{0,1\}^n\}$ とする。ここで、$v(\boldsymbol{x})$ は 2.5 節の手順 1 の入力カード列を表し、$v(\boldsymbol{x}) = x_1\overline{x_1}\cdots x_n\overline{x_n}$ である。置換 $\pi \in \mathfrak{S}_{2n}$ および $\mathcal{V}$ に対する同時挿入 $\boldsymbol{u}$ を固定したとき、$\mathcal{W} := \{\pi(v) \mid v \in \mathcal{V}\}$ および $\mathcal{W}' := \{w' \mid w \in \mathcal{W}, w' \xleftarrow{\boldsymbol{u}} w\}$ と定義する。すなわち $\mathcal{V}, \mathcal{W}, \mathcal{W}'$ の要素は 2.5 節のプロトコルの手順 1, 2, 3 のカード列 $v(\boldsymbol{x}), w(\boldsymbol{x}), w'(\boldsymbol{x})$ にそれぞれ対応している。

これらの $\mathcal{V}, \mathcal{W}, \mathcal{W}'$ は出力値に応じて 2 つの集合に分割できる。出力値 b に対応する集合をそれぞれ $\mathcal{V}^{(b)}, \mathcal{W}^{(b)}, \mathcal{W}'^{(b)}$ と表す。

	x	$\bar{x}$	y	$\bar{y}$	$x \oplus y$
$w_1^{(0)}$:	0	1	0	1	0
$w_1^{(1)}$:	0	1	1	0	1
$w_2^{(1)}$:	1	0	0	1	1
$w_2^{(0)}$:	1	0	1	0	0

表 1 2 変数 XOR の真理値表

探索すべきものは、プロトコルの正当性と安全性（1.1 節）を満たす $\mathcal{W}'^{(b)}$ を得るための置換 π と同時挿入 $\boldsymbol{u}$ である。具体的には、以下の条件を満たす置換 π および同時挿入 $\boldsymbol{u}$ を探索すればよい。

- 任意の $w'^{(0)} \in \mathcal{W}'^{(0)}, w'^{(1)} \in \mathcal{W}'^{(1)}$ に対して、$w'^{(0)}$ と $w'^{(1)}$ は巡回的に等しくない
- 任意の $b \in \{0,1\}$ について $\mathcal{W}'^{(b)}$ は巡回的に等しい

1 つ目の条件は正当性、2 つ目の条件は安全性を言い換えたものである。ここで、置換 π の探索は、$\mathfrak{S}_{2n}$ 全体を探索すればよいため、少なくとも有限回の探索で済むことが保証されている。一方、同時挿入 $\boldsymbol{u}$ の探索は、探索空間が有限ではないため、素直な実装では探索が停止しない。

提案手法では、同時 0-挿入 $\boldsymbol{u}$ の探索が整数最適化問題に定式化できることに着目し、以下のような探索を行う。

1. 集合 $\mathcal{W}^{(b)}$（$b \in \{0,1\}$）を与えたとき、同時 0-挿入 $\boldsymbol{u}$ を探索する整数最適化問題を解く。目的関数は $\boldsymbol{u}$ のビット長であり、条件を満たす $\boldsymbol{u}$ が存在するときは、長さ最小のものが得られる。
2. 上記の手順をすべての置換 $\pi \in \mathfrak{S}_{2n}$ について実行し、安全性の要件を満たす π と $\boldsymbol{u}$ を探索する。
3. 所望の π と $\boldsymbol{u}$ が見つかった場合、正当性の条件を検証し、条件を満たしている場合はこれらを出力する。

上記 1 は主に 3.2 節と 3.3 節において取り組む。ただし、実際には $\mathcal{W}^{(b)}$ を与えるだけでなく、シフト量（後述）も与えた上で整数最適化問題を定式化する。上記 2 は 3.4 節において取り組む。このアルゴリズムを実行することで、3 変数ブール関数に対する $(*, 0)$-SCFO プロトコルの不可能性が得られる。また、$\boxed{\clubsuit}$ と $\boxed{\heartsuit}$ の対称性より、$(0, *)$-SCFO プロトコルの不可能性も同時に得られる。

3.2 記法

出力値が $b \in \{0,1\}$ となる入力カード列の総数を $K^{(b)}$、カード枚数（ビット長）を $I \in \mathbb{N}$ とし、出力値 b の $k \in [K^{(b)}]$ 番目の入力カード列を $w_k^{(b)} := w_{1,k}^{(b)} w_{2,k}^{(b)} \cdots w_{I,k}^{(b)}$ と表す。さらに、$w_k^{(b)}$ に含まれる 0 の枚数を $J^{(0)}$、1 の枚数を $J^{(1)}$ とすると、$I = J^{(0)} + J^{(1)}$ が成り立つ。例として、2 変数 XOR 関数の真理値表を表 1 に示す。このとき、$I = 4,\ J^{(0)} = J^{(1)} = 2,\ K^{(0)} = K^{(1)} = 2$ である。

カード列 $w_k^{(b)}$ において、巡回構造を考慮すると 1 と 1 の間に挟まれる区分は $J^{(1)}$ 個存在する。例えば、カード列 0110 の場合（$J^{(1)} = 2$）は、1 つ目の 1 までの区分（2 つ

目の 1 から 1 つ目の 1 の間）と 2 つ目の 1 までの区分（1 つ目の 1 から 2 つ目の 1 の間）の 2 つの区分 $(00,\ \phi)$ に分割できる。

ここで、$w_k^{(b)}$ における 1 の位置を昇順に

$$1 \le t_{1,k}^{(b)} < t_{2,k}^{(b)} < \cdots < t_{J^{(1)},k}^{(b)} \le I$$

と定義すると、区分 $j \in [J^{(1)}]$ に含まれる 0 の枚数 $x_{j,k}^{(b)}$ は

$$x_{j,k}^{(b)} = \begin{cases} t_{j,k}^{(b)} - t_{j-1,k}^{(b)} - 1 & (1 < j \le J^{(1)}) \\ (t_{j,k}^{(b)} - 1) + (I - t_{J^{(1)},k}^{(b)}) & (j = 1) \end{cases}$$

と表される。

また、ビット長 I の入力カード列 $w_k^{(b)}$ に対しては、$I+1$ 箇所の 0-挿入位置が考えられる。例えば、入力カード列 0110（$I = 4$）の場合、挿入可能な位置は以下の 5 箇所である。

$$\begin{array}{ccccccccc} \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ & 0 & & 1 & & 1 & & 0 & \end{array}$$

巡回構造を考慮すると、最左端と最右端の挿入位置は同一視できるため、最左端の位置を除外して計 I 箇所の挿入位置を考えれば十分である。

0-挿入位置 $i \in [I]$ が区分 j に含まれるかを示す定数

$$a_{i,j,k}^{(b)} = \begin{cases} 1, & \text{位置 } i \text{ が区分 } j \text{ に含まれる}, \\ 0, & \text{それ以外}, \end{cases}$$

$(i \in [I],\ j \in [J^{(1)}],\ k \in [K^{(b)}],\ b \in \{0,1\})$ を導入し、0-挿入位置 i への 0-挿入枚数を $y_i \in \mathbb{Z}_{\ge 0}$ $(i \in [I])$ とすると、0-挿入後の各区分の 0 の枚数は

$$x'^{(b)}_{j,k} \ = \ x_{j,k}^{(b)} \ + \ \sum_{i \in [I]} a_{i,j,k}^{(b)}\, y_i,$$

と表せる。また、$\boldsymbol{x}' \coloneqq (x'^{(b)}_{j,k})_{(j,k,b) \in [J^{(1)}] \times [K^{(b)}] \times \{0,1\}} \in \mathbb{Z}_{\ge 0}^{|[J^{(1)}] \times [K^{(b)}] \times \{0,1\}|}$ とする。

このとき、同じ出力値の 2 つの入力カード列 $w_k^{(b)}$ と $w_{k+1}^{(b)}$ が巡回的に等しくなる条件は、$w_{k+1}^{(b)}$ に対する適切な巡回シフト量 $s_{k+1}^{(b)} \in \mathbb{Z}$ $(0 \le s_{k+1}^{(b)} \le J^{(1)} - 1)$ を用いて

$$x'^{(b)}_{j,k} = x'^{(b)}_{\left((j+s_{k+1}^{(b)}-1) \bmod J^{(1)}\right)+1,k+1} \quad (j \in [J^{(1)}])$$

と表せる。例えば、$w_1^{(1)} = 0110,\ w_2^{(1)} = 1001$ の場合を考えると、0-挿入枚数が 0 枚のときは、

$$x'^{(1)}_{1,1} = x_{1,1}^{(1)} = 2, \quad x'^{(1)}_{2,1} = x_{2,1}^{(1)} = 0,$$

$$x'^{(1)}_{1,2} = x_{1,2}^{(1)} = 0, \quad x'^{(1)}_{2,2} = x_{2,2}^{(1)} = 2$$

となる。このとき、適切な巡回シフト量 $s_2^{(1)} = 1$ を与えると、

$$x'^{(1)}_{\left((1+s_2^{(1)}-1) \bmod 2\right)+1,2} = x'^{(1)}_{2,\,2} = 2 = x'^{(1)}_{1,1},$$

$$x'^{(1)}_{\left((2+s_2^{(1)}-1) \bmod 2\right)+1,2} = x'^{(1)}_{1,2} = 0 = x'^{(1)}_{2,1}$$

となり、$w_1^{(1)}$ と $w_2^{(1)}$ は巡回的に等しいことがわかる。

3.3 定式化

与えられた入力カード列集合 $\boldsymbol{w} \coloneqq (w_k^{(b)})_{k \in [K^{(b)}],\ b \in \{0,1\}}$ に対して、n 変数ブール関数の $(*,0)$-SCFO プロトコルを探索する問題は、同一の出力値をもつカード列同士が同時 0-挿入後に巡回的に等しくなるような以下 2 つの変数を決定することである。

- 0-挿入位置および挿入枚数：$\boldsymbol{y} \coloneqq (y_i)_{i \in [I]} \in \mathbb{Z}_{\ge 0}^{I}$
- 巡回シフト量：$\boldsymbol{s} \coloneqq (s_k^{(b)})_{(k,b) \in [K^{(b)}] \times \{0,1\}} \in \mathbb{Z}_{\ge 0}^{|[K^{(b)}] \times \{0,1\}|}$

ここでは、巡回シフト量 $\boldsymbol{s}$ が固定された場合に、同一出力値をもつカード列同士が同時 0-挿入後に巡回的に等しくなるという制約の下で、0-挿入の総枚数を最小化する問題を考える。この問題は、次のような整数最適化問題 $\mathrm{P}(\boldsymbol{s}, \boldsymbol{w})$ として定式化できる。

$\mathrm{P}(\boldsymbol{s}, \boldsymbol{w})$:

$$\min_{\boldsymbol{x}', \boldsymbol{y}} \quad \sum_{i \in [I]} y_i \tag{1}$$

$$\text{s.t.} \quad x'^{(b)}_{j,k} = x_{j,k}^{(b)} + \sum_{i \in [I]} a_{i,j,k}^{(b)} y_i \quad (j \in [J^{(1)}],\ k \in [K^{(b)}],\ b \in \{0,1\}) \tag{2}$$

$$x'^{(b)}_{j,1} = x'^{(b)}_{\left((j+s_{k+1}^{(b)}-1) \bmod J^{(1)}\right)+1,k+1} \quad (j \in [J^{(1)}],\ k \in [K^{(b)} - 1],\ b \in \{0,1\}) \tag{3}$$

$$x'^{(b)}_{j,k} \in \mathbb{Z}_{\ge 0} \quad (j \in [J^{(1)}],\ k \in [K^{(b)}],\ b \in \{0,1\}) \tag{4}$$

$$y_i \in \mathbb{Z}_{\ge 0} \quad (i \in [I]) \tag{5}$$

式 (1) の目的関数は、同時 0-挿入の総枚数を表す。式 (2) は各区分における 0-挿入後の 0 の枚数を定義し、式 (3) は同一出力値をもつカード列同士が同時 0-挿入後に巡回的に等しいことを表す制約である。式 (4)–(5) は、決定変数を非負整数とする制約である。

さらに、$(*,0)$-SCFO プロトコルとして成立させるためには、正当性を満たす必要がある。すなわち異なる出力値をもつカード列同士が巡回的に等しくなってはならない。したがって、最適化問題 $\mathrm{P}(\boldsymbol{s}, \boldsymbol{w})$ の最適解 $\boldsymbol{z}^* \coloneqq (\boldsymbol{x'}^*,\ \boldsymbol{y}^*)$ について、次の式が成り立つことを確認する必要がある。

$$x'^{(0)*}_{j,1} \ne x'^{(1)*}_{\left((j+s_1^{(1)}-1) \bmod J^{(1)}\right)+1,1} \quad (j \in [J^{(1)}]) \tag{6}$$

この式は、出力値が 0 となるあるカードと出力値が 1 となるあるカード列が、0-挿入後に巡回的に等しくならないことを意味している。また、最適解が複数ある場合には、

解集合 $\mathcal{Z}^*(\boldsymbol{s},\boldsymbol{w})$ の全要素に対して、式 (6) の成立を確認する。

3.4 探索アルゴリズム

3.3 節では、所与の入力カード列集合 w および巡回シフト量 $\boldsymbol{s}$ に対する n 変数ブール関数の $(*,0)$-SCFO プロトコル探索問題を定式化した。本節では、$(*,0)$-SCFO プロトコルの有無を検証するため、すべての入力カード列の全順列と巡回シフト量の全組合せに対して、最適化問題 (1)–(5) を解く全探索アルゴリズムを提示する。

まず、入力カード列の並び替えを考慮する。例えば、2 変数 XOR 関数では、表 1 の真理値表において、$(x,\bar{x},y,\bar{y})$ の順序が設定されているが、SCFO プロトコルでは変数の順序を問わない。したがって、すべての順列について 0-挿入後に巡回的に等しくなるかを確認する必要がある。ここで、ある順序で並べられた初期入力カード列集合を $\boldsymbol{w}_0$ と表す。

集合 $[I]$ 上の置換 $\pi \in \mathfrak{S}_I$ に対して、カード列 $w_k^{(b)}$ の π による並び替えを以下のように定める。

$$\pi(w_k^{(b)}) = w_{\pi^{-1}(1),k}^{(b)} w_{\pi^{-1}(2),k}^{(b)} \cdots w_{\pi^{-1}(I),k}^{(b)}$$

また、入力カード列集合 $\boldsymbol{w}$ の π による並び替えを

$$\pi(\boldsymbol{w}) = \left(\pi(\boldsymbol{w}_k^{(b)})\right)_{k\in[K^{(b)}],\, b\in\{0,1\}}$$

と定義する。

次に、巡回シフト量についても、全組合せを列挙する。入力カード列の総数を

$$K = \sum_{b\in\{0,1\}} K^{(b)}$$

とすると、1 つ目のカード列を基準として残りの $K-1$ 列のシフト量を列挙するために、巡回シフト量の全組合せ集合を

$$\mathcal{S} \coloneqq \{0,1,\ldots,J^{(1)}-1\}^{K-1} \tag{7}$$

と定義する。

対称群 $\mathfrak{S}_I$ およびシフト量集合 $\mathcal{S}$（式 (7)）を用い、Algorithm 1 に示す全探索アルゴリズムを実行することで、$(*,0)$-SCFO プロトコルの存在を検証する。本アルゴリズムは、$\mathfrak{S}_I$ と $\mathcal{S}$ の直積上で最適化問題 (1)–(5) を反復的に解くため、出力される解集合 $\mathcal{F}$ が空集合であれば、探索対象の関数に対する $(*,0)$-SCFO プロトコルが存在しないことを意味する。

Algorithm 1 $(*,0)$-SCFO プロトコル探索アルゴリズム

入力: 初期入力カード列集合 $\boldsymbol{w}_0$

$\boldsymbol{w}_0$ から、文字列中の 0 の枚数 $J^{(0)}$、1 の枚数 $J^{(1)}$、およびビット長 I を算出
式 (7) により巡回シフト量集合 $\mathcal{S}$ を生成
最適解集合 $\mathcal{F} \leftarrow \varnothing$ を初期化
for $\pi \in \mathfrak{S}_I$ **do**
  $\boldsymbol{w} \leftarrow \pi(\boldsymbol{w}_0)$
  for $\boldsymbol{s} \in \mathcal{S}$ **do**
    if 最適化問題 P$(\boldsymbol{s},\boldsymbol{w})$ が実行可能 **then**
      for $\boldsymbol{z}^* \in \mathcal{Z}^*(\boldsymbol{s},\boldsymbol{w})$ **do**
        if $\boldsymbol{z}^*$ が式 (6) を満たす **then**
          $\mathcal{F} \leftarrow \mathcal{F} \cup \{(\boldsymbol{s},\boldsymbol{w},\boldsymbol{z}^*)\}$
        end if
      end for
    end if
  end for
end for

出力: 最適解集合 $\mathcal{F}$

4. 実験結果

本節では、2.1 節で提示した 3 変数ブール関数に対する $(*,0)$-SCFO プロトコルの不可能性を、Algorithm 1 を用いて実証する。具体的には、10 個の 3 変数関数（2.1 節のリストの 5 番〜14 番）のそれぞれに対し、Algorithm 1 を実行することで、実行可能解の有無を検証した。

最適化問題 (1)–(5) の求解には、Python 言語の SciPy ライブラリに内蔵された HiGHS 1.8.0 ソルバーを用い、実験は Apple M3（8 スレッド、4.05 GHz）、メモリ 16GB を搭載した MacBook Air 上で実施した。

その結果、3 変数 EQ 関数（リストの 7 番の関数）を除くすべての代表元において実行可能解が得られなかった。したがって、これらの代表元に対して $(*,0)$-SCFO プロトコルを構成することは不可能であることが示された。また、計算時間は最短で約 3 分、最長で約 60 分を要した。

5. おわりに

本研究では、カードベース暗号における不可能性証明の新たな手法として、数理最適化に基づく手法を提案した。具体的には、$(*,0)$-SCFO および $(0,*)$-SCFO プロトコルの構成可能性を、整数最適化問題として定式化し、探索アルゴリズムを提案した。提案手法をすべての 3 変数ブール関数に適用した結果、既知のプロトコル（例えば EQ プロトコル）を除き、$(*,0)$-SCFO プロトコルが存在しないことを計算機実験により実証した。

本研究を通じて、カードベース暗号における数理最適化の有用性が示された。一方、数理最適化に基づくカードベース暗号の研究は始まったばかりであり、多くの未解決問題が残されている。例えば以下のような問題がある。

- 4 変数以上のブール関数に対する $(*,0)$-SCFO プロトコルの構成可能性は未解決問題である。提案手法の計算量は変数 n について少なくとも指数関数以上を要するため、提案手法をそのまま 4 変数関数に適用することは計算量の観点から困難である。
- 本研究の構成可能性は $(*,0)$-SCFO および $(0,*)$-SCFO プロトコルに対するものであるが、$\boxed{\clubsuit}$ と $\boxed{\heartsuit}$ の両方を

追加カードとして用いた一般の (k_0, k_1)-SCFO プロトコルの不可能性は未解決である。

- SCFO プロトコル以外のプロトコルに対して、整数最適化問題のモデル化を行い、構成可能性を調べることは未解決問題である。

謝辞 本研究は JSPS 科研費 JP23H00479、JP21K17702 と JST CREST JPMJCR22M1 と JST 経済安全保障重要技術育成プログラム JPMJKP24U2 の支援を受けている。

参考文献

[1] B. D. Boer. More efficient match-making and satisfiability the five card trick. In J.-J. Quisquater and J. Vandewalle eds., *Advances in Cryptology – EUROCRYPT' 89*, Vol. 434 of *LNCS*, pp. 208–217, Heidelberg, 1990. Springer.

[2] C. Crépeau and J. Kilian. Discreet solitary games. In D. R. Stinson ed., *Advances in Cryptology—CRYPTO' 93*, Vol. 773 of *LNCS*, pp. 319–330, Berlin, Heidelberg, 1994. Springer.

[3] Y. Hashimoto, K. Nuida, K. Shinagawa, M. Inamura, and G. Hanaoka. Toward finite-runtime card-based protocol for generating a hidden random permutation without fixed points. *IEICE Trans. Fundam.*, E101.A(9):1503–1511, 2018.

[4] J. Heather, S. Schneider, and V. Teague. Cryptographic protocols with everyday objects. *Formal Aspects Comput.*, 26(1):37–62, 2014.

[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 T. Takagi and T. Peyrin eds., *Advances in Cryptology—ASIACRYPT 2017*, Vol. 10626 of *LNCS*, pp. 126–155, Cham, 2017. Springer.

[6] A. Koch. The landscape of optimal card-based protocols. *Mathematical Cryptology*, 1(2):115–131, 2022.

[7] A. Koch, M. Schrempp, and M. Kirsten. Card-based cryptography meets formal verification. In S. D. Galbraith and S. Moriai eds., *Advances in Cryptology–ASIACRYPT 2019*, Vol. 11921 of *LNCS*, pp. 488–517, Cham, 2019. Springer.

[8] A. Koch, M. Schrempp, and M. Kirsten. Card-based cryptography meets formal verification. *New Gener. Comput.*, 39(1):115–158, 2021.

[9] A. Koch, S. Walzer, and K. Härtel. Card-based cryptographic protocols using a minimal number of cards. In T. Iwata and J. H. Cheon eds., *Advances in Cryptology—ASIACRYPT 2015*, Vol. 9452 of *LNCS*, pp. 783–807, Berlin, Heidelberg, 2015. Springer.

[10] T. Mizuki and H. Shizuya. A formalization of card-based cryptographic protocols via abstract machine. *Int. J. Inf. Secur.*, 13(1):15–23, 2014.

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

[12] K. Shinagawa. Card-based protocols with single-card encoding. In C. Anutariya and M. M. Bonsangue eds., *Theoretical Aspects of Computing*, Vol. 15373 of *LNCS*, pp. 182–194, Cham, 2025. Springer.

[13] K. Shinagawa and T. Mizuki. The six-card trick: Secure computation of three-input equality. In K. Lee ed., *Information Security and Cryptology*, Vol. 11396 of *LNCS*, pp. 123–131, Cham, 2018. Springer.

[14] K. Shinagawa and K. Nuida. Card-based protocols imply PSM protocols. In O. Beyersdorff, M. Pilipczuk, E. Pimentel, and N. K. Thàng eds., *Theoretical Aspects of Computer Science*, Vol. 327 of *LIPIcs*, pp. 72:1–72:18, Dagstuhl, 2025. Schloss Dagstuhl.

[15] 水木敬明. カードベース暗号の教育への応用. 2016 年情報セキュリティ（ISEC）研究会, 2016.

[16] 品川和雅. カードベース暗号を題材にした小中学生向け授業の報告. 2022 年暗号と情報セキュリティシンポジウム (SCIS 2022), 2022.

[17] 品川和雅, 縫田光司. シングルカットフルオープンカードベースプロトコル. コンピュータセキュリティシンポジウム 2019（CSS2019）, 2019.

[18] 品川和雅, 縫田光司. カードベース暗号に現れる語の組合せ論. 2024 年暗号と情報セキュリティシンポジウム (SCIS 2024), 2024.