Computer Security Symposium 2020
26 - 29 October 2020

一様ランダムな不動点のない置換を生成する新しいプロトコル

村田 総馬[1,a)]　宮原 大輝[1,2]　水木 敬明[3]　曽根 秀昭[3]

概要：プレゼント交換を n 人のプレイヤーで行う場面を考えよう．各プレイヤーは（自分以外の）誰にプレゼントを贈れば良いかだけを事前に知りたい．すなわち，一様ランダムな不動点を持たない置換を秘匿した状態で生成したい．この問題に対する物理的なカード組を用いた解決方法として，最も実用的なプロトコルは 2015 年に Ishikawa らが提案したものである．そのプロトコルでは，各プレイヤーに対応した n 個のカード束を用意し，それらの束をシャッフルし，不動点の有無を確認した後，不動点が存在しなければそのまま n 個のカード束を出力とし，存在すればシャッフル操作からやり直す．本稿では，この既存手法を基に，シャッフル操作をやり直す確率を低くする手法を考える．具体的には，プレイヤー数 n より大きい数のカード束を用意することで，不動点が存在してもその不動点を除去することでシャッフルのやり直しを防ぐアイデアを導入する．このようにして得られる提案プロトコルは，不動点を持たない置換を一様ランダムに生成し，追加するカード束の数 t の値を大きくすることでシャッフル操作をやり直す確率が改善される．

キーワード：カードベース暗号，不動点のない置換，プレゼント交換

A New Protocol for Generating a Uniformly Distributed Random Permutation without Fixed Points

Soma Murata[1,a)]　Daiki Miyahara[1,2]　Takaaki Mizuki[3]　Hideaki Sone[3]

Abstract: Consider the situation where n players exchange gifts. Each player only wants to know in advance who (other than himself/herself) he/she should give a gift. That is, they want to uniformly generate a hidden random permutation without fixed points. As a simple solution to this problem, in 2015, Ishikawa *et al.* proposed a practical protocol with a deck of physical cards. In their protocol, players prepare n piles of cards, each of which corresponds to a player, and shuffle the piles. Then, the players verify whether or not the resulting piles have fixed points, and if they have no fixed point, the players output them; otherwise, the players restart the shuffle process. In this paper, based on their protocol, we consider how to decrease the probability of restarting. That is, we introduce an idea of preparing piles of cards more than the number of n players, and removing fixed points if there are fixed points to prevent repeating the shuffle again. Our proposed protocol uniformly generates a random permutation without fixed points, and the probability of restarting can be improved by increasing the number of additional t piles of cards.

Keywords: Card-based Cryptography, Permutation without Fixed Points, Exchange of Gifts

1. はじめに

$n\,(\geq 3)$ を自然数とし，プレゼント交換を n 人のプレイヤーで行う場面を考えよう．各プレイヤーはプレゼントを

[1] 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
[2] 産業技術総合研究所（AIST）
[3] 東北大学サイバーサイエンスセンター
Cyberscience Center, Tohoku University
[a)] soma.murata.p5@dc.tohoku.ac.jp

購入するに当たって（自分以外の）誰にプレゼントを贈れば良いかだけを事前に知りたい．数学的に，プレゼント交換の割り当ては（n 次対称群の要素である）置換で表現でき，その置換は自分自身にプレゼントを贈ってしまうことを避ける**不動点のない置換**である必要がある．上の目的を達成するためには，その不動点のない置換を秘匿した状態で一様ランダムに生成するのが良い．複雑なプログラムや計算機を用意せず，身近な道具を用いてプレイヤー同士で安全性を納得しながら容易に実行できる**物理的暗号プロトコル**はこのような場面に適している．

1.1 研究背景

不動点のない置換（derangement と呼ばれ，以降本文ではそのように呼ぶ）を秘匿したまま生成する問題は，1993 年に Crépeau と Kilian [1] によって初めて考案されて以降，物理的なカード組を用いた解決方法がいくつか提案されてきた．実用的なプロトコルとして，Heather ら [2] が提案した封筒と記入式カードを用いたプロトコルや，Ibaraki ら [3] が提案したプレゼント ID とギフト ID を表す 2 つのカード列を導入したプロトコルがある．これらのプロトコルの特徴として，生成される derangement は特定のサイクルを含む置換のみとなっている．

一方で，全ての derangement を対象とし，その中から一様ランダムに 1 つを生成する実用的なプロトコルは，Crépeau と Kilian [1] が提案したプロトコルと，それを改良した Ishikawa ら [4] のプロトコルである．Ishikawa らはカード束をかき混ぜる Pile-scramble シャッフルを導入することで，Crépeau と Kilian が提案した 4 色のカードを合計 $4n^2$ 枚用いたプロトコルを改良し，合計 n^2 枚の 2 色のカードで実装した．そのプロトコルの流れは次の通りである（詳細は 2.5 節で紹介する）．

(1) 各プレイヤーに対応した n 個のカード束を用意する．
(2) それらのカード束を Pile-scramble シャッフルする．
(3) 不動点の有無のみを確認する．
(4) 不動点が存在しなければそのまま n 個のカード束を出力とし，存在すればシャッフル操作からやり直す．

また，プレイヤーに対応するカード束を 2 進数のように表現して計算することにより，カード枚数を $2n\lceil \log_2 n \rceil + 6$ 枚に削減できることも示している．これらのプロトコルは上述の通り，シャッフル操作をやり直す場合があるため，有限時間に終了する保証がない．このシャッフル操作をやり直す確率は，およそ $1 - \frac{1}{e} \approx 0.63$（$e \approx 2.7$ は自然対数の底）となっている．

2018 年に Hashimoto ら [5] は，置換の型に関する性質を巧みに用いることで，初めて有限時間で終了するプロトコルを提案した．彼らのプロトコルは革新的である一方，不均一な確率分布のシャッフル操作を必要とするため，実際に人間の手で実行可能なのかどうかは判明されていない．

1.2 貢献

本稿では，Ishikawa ら [4] が提案したプロトコルを基に，シャッフル操作をやり直す確率を低くする新しい手法を提案する．カードベース暗号では実際にプレイヤーが手でプロトコルを実行するため，シャッフル操作を何度もやり直すことはなるべく避けたい．そこで，プレイヤー数 n より大きい数のカード束を用意することで，不動点が存在してもその不動点を除去することでシャッフル操作のやり直しを防ぐアイデアを導入し，Ishikawa らのプロトコルよりもやり直す確率が低い効率的なプロトコルを提案する．提案プロトコルは Ishikawa らのプロトコルと同様に，全ての derangement の中からその 1 つを一様ランダムに生成する．シャッフル操作をやり直す確率は，追加するカード束の数 t を大きくすることで改善されていく．具体的には，追加するカード束の数 t を $t = 2$ 程度にすることで，プロトコルのやり直す確率を，0.1 〜 0.2 の確率まで改善することができる．

本稿の構成は次の通りである．まず，2 節でカードベース暗号や置換における事前知識および Ishikawa らのプロトコルを紹介する．3 節では，上述のアイデアを導入した提案プロトコルの具体的な流れについて説明する．4 節では，提案プロトコルによって生成される置換が一様ランダムな全ての derangement であることを示す．また，追加するカード束の数 t とシャッフル操作をやり直す確率の関係式を示し，具体的に t の値をどれくらいにすればプロトコルのやり直し確率を改善できるかについて可視化する．

1.3 関連研究

提案プロトコルのように物理的なカード組を用いて秘密計算などの暗号タスクを解決する手法を**カードベース暗号**と呼ぶ．den Boer [6] によって初めて 5 枚のカードを用いた AND の秘密計算を実現するプロトコルが提案されて以降，XOR [7] や 3 入力多数決 [8] の秘密計算を実現するプロトコルも提案されている．また，所持金額の大小関係のみを得る金持ち比べプロトコル [9–11] や，同じグループに所属するメンバーが誰なのかだけを把握できる秘匿グループ分けプロトコル [12]，数独のようなペンシルパズルの解答に対する効率的なゼロ知識証明プロトコル [13] も提案されている．

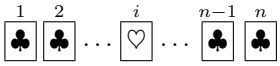
2. 準備

本節では，本稿で用いるカードの種類やシャッフル操作について説明する．さらに，Ishikawa ら [4] によって提案された置換生成プロトコルを紹介する．

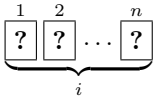
2.1 カード

本稿では，黒 $\boxed{\clubsuit}$ と赤 $\boxed{\heartsuit}$ の 2 色のカードを用いる．表面は同じ色のカードであれば区別がつかず，裏面は全て同

じ模様 $\boxed{?}$ である．自然数 $i\,(1 \le i \le n)$ を，$n-1$ 枚の黒のカードと 1 枚の赤のカードの合計 n 枚のカードを用いて，カード列の i 番目に赤のカードを，それ以外の場所に黒のカードを並べて，次のように表現する．



このルールにしたがって並べられた自然数 i を表現するカード列が，次のように裏向きで伏せられている場合，それらを i の**コミットメント**と呼ぶ．



2.2 Pile-scramble シャッフル

Pile-scramble シャッフルは Ishikawa ら [4] によって提案されたシャッフル操作である．カード枚数の等しい n 個のカード束の列を

$$(\mathsf{pile}_1, \mathsf{pile}_2, \ldots, \mathsf{pile}_n)$$

とする．それらに Pile-scramble シャッフルを適用すると，

$$(\mathsf{pile}_{\pi^{-1}(1)}, \mathsf{pile}_{\pi^{-1}(2)}, \ldots, \mathsf{pile}_{\pi^{-1}(n)})$$

のカード束の列を得る．ここで，$\pi \in S_n$ は一様ランダムな置換（S_n は n 次の対称群）である．Pile-scramble シャッフルは輪ゴムや封筒を利用することで容易に実装できる．

2.3 置換の性質

任意の置換はいくつかの互いに素な巡回置換（サイクル）の積で表現できることがよく知られている．例えば，次のような置換

$$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 5 & 6 & 4 & 2 & 7 & 1 \end{pmatrix}$$

は 3 つの互いに素な巡回置換 $\tau_1 = (4), \tau_2 = (25), \tau_3 = (1367)$ の積で表現できる．

$$\tau = \tau_1\tau_2\tau_3 = (4)(25)(1367)$$

また，巡回置換 τ_1, τ_2, τ_3 のサイクルの長さはそれぞれ $1, 2, 4$ であり，特に長さが 1 のサイクルは不動点である．置換をいくつかの互いに素な置換の積で表現したとき，この積に含まれるサイクルの長さ i の巡回置換の数を m_i とすると，置換の**型**を $\langle 1^{m_1}, 2^{m_2}, \ldots, n^{m_n} \rangle$ と表現する．例えば，上の置換 τ の型は $\langle 1^1, 2^1, 4^1 \rangle$ と表現できる．

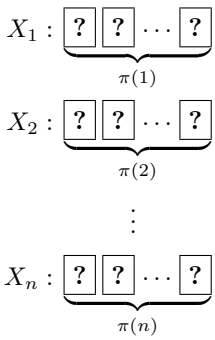
S_n における derangement の個数を d_n とすると，d_n は次の式で表すことができる．

$$d_n = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!}$$

不動点の個数がちょうど f 個ある置換の個数は ${}_n\mathrm{C}_f \cdot d_{n-f}$ である．ただし，$d_0 = 1, d_1 = 0$ とする．

2.4 カードによる置換の表現

本稿では，置換 $\pi \in S_n$ を n 個のコミットメントの列 $(X_1, X_2, \ldots, X_n)$ を用いて次のように表現する．



(1)

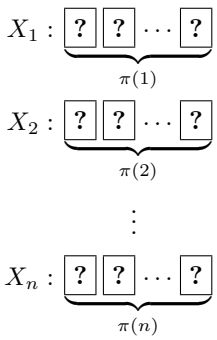
2.5 既存プロトコル

本節では，Ishikawa ら [4] が提案した derangement を一様ランダムに生成するプロトコルを紹介する．特に，Pile-scramble シャッフルを導入し，黒 $\boxed{\clubsuit}$ と赤 $\boxed{\heartsuit}$ の 2 色のカードを合計 n^2 枚用いて実行する手法を説明する．

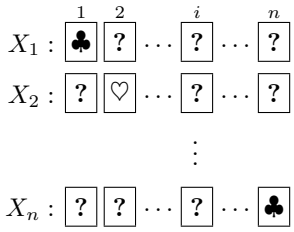
n 人のプレイヤーを $P_1, P_2, \ldots, P_n$ とする．プロトコルの流れは以下の通りである．

Step 1 プレイヤーは式 (1) にしたがって，n 次の恒等置換を表すコミットメントを並べる．すなわち，対角線上のカードはすべて赤 $\boxed{\heartsuit}$ のカードであり，他はすべて黒 $\boxed{\clubsuit}$ のカードである．

Step 2 各コミットメントをそれぞれカード束として，Pile-scramble シャッフルを行う．この操作により，一様ランダムな置換 $\pi \in S_n$ を表すカード束の列 $(X_1, X_2, \ldots, X_n)$ を得る．



Step 3 次のように対角線上にあるカード n 枚をめくり，不動点の有無を確認する．



例えば上の場合，X_2 が 2 のコミットメントであるため，$\pi(2) = 2$ が判明する．

Step 4　前ステップでめくったカードに 1 枚でも赤 $\boxed{\heartsuit}$ のカードがある場合，めくったカードを全て裏向きに伏せて Step 2 に戻り，シャッフル操作をやり直す．めくったカードが全て黒 $\boxed{\clubsuit}$ のカードの場合，π が一様ランダムな derangement である．すなわち，各プレイヤー P_i はカード列 X_i を他のプレイヤーに秘密にめくり，$\pi(i)$ を確認することで，プレイヤー $P_{\pi(i)}$ にプレゼントを贈れば良いことが分かる．

このプロトコルの Step 3 で不動点が見つかり Step 2 に戻る確率は，置換 π が derangement でない確率と等しい．n 次対称群 S_n から一様ランダムに 1 つ選んだ置換が derangement である確率は，次の式で表現できることがよく知られている．

$$\frac{d_n}{n!} = \sum_{k=0}^{n} \frac{(-1)^k}{k!}$$

さらに，

$$\lim_{n\to\infty} \sum_{k=0}^{n} \frac{(-1)^k}{k!} = \frac{1}{e}$$

であることから，このプロトコルにおけるシャッフル操作をやり直す確率はおよそ $1 - \frac{1}{e} \approx 0.63$ である．

3.　提案プロトコル

本節では，2.5 節で紹介した Ishikawa ら [4] の既存プロトコルを基に，プレイヤー数 n より大きい数のカード束を用意した新しい手法を提案する．

3.1　提案プロトコルの概要

ここでは，提案プロトコルの概要を述べる．提案プロトコルでは，まずプレイヤー数 n より大きい数 $n+t$ 個のカード束（コミットメント）を用意し，それらのカード束に Pile-scramble シャッフルを行う．すなわち，追加の t 個のカード束が，不動点が生じてもそれを吸収してくれるバッファとなる．そして，この $n+t$ 個のカード束（コミットメント列）に対して，「不動点除去操作」（3.2.1 節で説明）を行う．この操作によって判明する不動点の個数 f が $f \le t$ ならば，シャッフル操作をやり直さなくて済み，残りの $n+t-f$ 個のコミットメント列のうち，余分な $t-f$ 個のコミットメントを消去すべく，「縮退操作」（3.2.2 節で説明）を行う．

3.2 節で不動点除去操作と縮退操作について説明する．

3.2　2 つの操作の定義

$n+t$ 次の恒等置換を表すコミットメント列を用意し，2.5 節で説明した既存プロトコルの Step 1 と 2 を実行したとしよう．このとき，$n+t$ 次対称群 S_{n+t} に一様分布した置換が得られるが，このときのコミットメント列を S_{n+t} 上の**コミット型置換**と呼ぶことにする．

3.2.1　不動点除去操作

前述の S_{n+t} 上のコミット型置換に対して，2.5 節で説明した既存プロトコルの Step 3 を適用し，不動点を全て見付け，そのインデックスの集合を I_{FP} とする．すなわち，対角線上のカードを全てめくる操作を考える．この操作（**不動点除去操作**と呼ぶ）によって，I_{FP} の部分を無視することにより，残りのコミットメント列が対応する置換は derangement となっており，$S_{n+t}\backslash I_{\mathrm{FP}}$ 上に一様分布している．これを $S_{n+t}\backslash I_{\mathrm{FP}}$ 上の**コミット型** *derangement* と呼ぶ．

3.2.2　縮退操作

前述の $S_{n+t}\backslash I_{\mathrm{FP}}$ 上のコミット型 derangement を考え，$t - |I_{\mathrm{FP}}| > 0$ であると仮定し，余分なコミットメントを消去したいとしよう（いま $n+t-|I_{\mathrm{FP}}|$ 次のコミット型 derangement であるが，n 次に近づけたい）．そのため，（めくられた I_{FP} の部分を無視し）残っているコミットメントの中で一番後ろのもの，すなわち $\max(S_{n+t}\backslash I_{\mathrm{FP}})$ 番目のコミットメントをめくることにする．めくったコミットメントの値を a とすると，$\max(S_{n+t}\backslash I_{\mathrm{FP}}) \mapsto a$ のマッピングが判明し，ここでは**バイパス**と呼ぶ．元のコミット型 derangement において，いまめくった $\max(S_{n+t}\backslash I_{\mathrm{FP}})$ 番目を無視することで，$\max(S_{n+t}\backslash I_{\mathrm{FP}})$ へのマッピングは仮想的に（バイパスにより）a へのマッピングであると考えれば，1 つ次数の低いコミット型置換が得られたことになる．ただし，もし $a \mapsto \max(S_{n+t}\backslash I_{\mathrm{FP}})$ だとすると，不動点を持つことになってしまうので，a 番目のコミットメントの $\max(S_{n+t}\backslash I_{\mathrm{FP}})$ 番目のカードをめくり，赤 $\boxed{\heartsuit}$ のカードが出るかどうかで不動点チェックを行う．

(1) 赤 $\boxed{\heartsuit}$ のカードの場合．
$\max(S_{n+t}\backslash I_{\mathrm{FP}}) \mapsto a \mapsto \max(S_{n+t}\backslash I_{\mathrm{FP}})$ のサイクルが生じており，このサイクルのインデックスからなる集合を I_{cycle} とする（すなわち，$I_{\mathrm{cycle}} = \{\max(S_{n+t}\backslash I_{\mathrm{FP}}), a\}$）．このサイクル集合を無視し，残ったコミットメントに対応する置換は，$S_{n+t}\backslash(I_{\mathrm{FP}} \cup I_{\mathrm{cycle}})$ 上に一様分布するコミット型 derangement となる．このとき，（$|I_{\mathrm{cycle}}| = 2$ なので）元のコミット型 derangement より 2 つ次数が下がる．

(2) 黒 $\boxed{\clubsuit}$ のカードの場合．
バイパス $\max(S_{n+t}\backslash I_{\mathrm{FP}}) \mapsto a$ の下で，$S_{n+t}\backslash(I_{\mathrm{FP}} \cup I_{\mathrm{BP}})$ 上に一様分布する derangement となる．ただし，I_{BP} はバイパスの終点以外の点からなる集合を表す（すなわち，$I_{\mathrm{BP}} = \{\max(S_{n+t}\backslash I_{\mathrm{FP}})\}$ である）．このと

き，次数が 1 つ低いコミット型 derangement が得られたことになる．

以上の操作を**縮退操作**と呼ぶ．上では，不動点除去操作の直後のコミット型 derangement に対して縮退操作を適用したが，そのような縮退操作の後のコミット型 derangement に対してもまた縮退操作を適用することができる．すなわち，一般に，不動点集合 I_{FP} とサイクルインデックス集合 I_{cycle} とバイパス $i_1 \mapsto \cdots \mapsto i_{\ell-1} \mapsto i_\ell$ を持つ $S_{n+t}\backslash(I_{\mathrm{FP}} \cup I_{\mathrm{cycle}} \cup I_{\mathrm{BP}})$ 上のコミット型 derangement に対する縮退操作を次のように定義する．

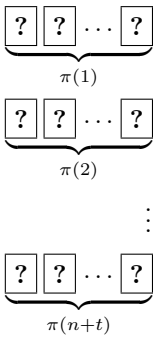
- $I_{\mathrm{BP}} = \{i_1, \ldots, i_{\ell-1}\} \neq \phi$ のとき（バイパスは $i_1 \mapsto \cdots \mapsto i_{\ell-1} \mapsto i_\ell$），$i_\ell$ 番目のコミットメントめくる．一方，$I_{\mathrm{BP}} = \phi$ のとき，残っているコミットメントの一番最後，すなわち，$\max(S_{n+t}\backslash(I_{\mathrm{FP}} \cup I_{\mathrm{cycle}}))$ 番目のコミットメントをめくる（このとき，$i_\ell = i_1 = \max(S_{n+t}\backslash(I_{\mathrm{FP}} \cup I_{\mathrm{cycle}}))$ とする）．いずれにしろ，そのコミットメントの値を $i_{\ell+1}$ とする．$i_{\ell+1}$ 番目のコミットメントの i_1 番目のカードをめくる．
  (1) 赤 $\boxed{\heartsuit}$ のカードの場合．
  $i_1 \mapsto \cdots \mapsto i_{\ell+1} \mapsto i_1$ のサイクルが生じており，このサイクルのインデックス $i_1, \ldots, i_{\ell+1}$ をサイクル集合 I_{cycle} に追加し，バイパスはなくなるので $I_{\mathrm{BP}} = \phi$ となる．このサイクル集合を無視し，残ったコミットメントに対応する置換は，$S_{n+t}\backslash(I_{\mathrm{FP}} \cup I_{\mathrm{cycle}} \cup I_{\mathrm{BP}})$ 上に一様分布するコミット型 derangement となる．このとき，($I_{\mathrm{cycle}} \cup I_{\mathrm{BP}}$ には新たに $i_\ell, i_{\ell+1}$ の 2 つ要素が追加されているので）元のコミット型 derangement より 2 つ次数が下がる．
  (2) 黒 $\boxed{\clubsuit}$ のカードの場合．
  バイパス $i_1 \mapsto \cdots \mapsto i_\ell \mapsto i_{\ell+1}$ の下で，$S_{n+t}\backslash(I_{\mathrm{FP}} \cup I_{\mathrm{cycle}} \cup I_{\mathrm{BP}})$ 上に一様分布する derangement となる．ただし，$I_{\mathrm{BP}} = \{i_1, i_2, \ldots, i_\ell\}$ である．このとき，次数が 1 つ低いコミット型 derangement が得られたことになる．

3.3 プロトコルの流れ

いま定義した 2 つの操作を用いて，提案プロトコルを次のように記述する．用意するカード束の数は，プレイヤー数 n より大きい $n+t$ 個とする．すなわち，既存プロトコル [4] より t 個多いカード束を用意する．プロトコルの流れは次の通りである．

Step 1 プレイヤーは式 (1) にしたがって $n+t$ 次の恒等置換を表すコミットメントを並べる．

Step 2 コミットメントの列に Pile-scramble シャッフルを行い，一様ランダムな置換 $\pi \in S_{n+t}$ を表すコミットメントの列を得る．



Step 3 Step 2 で得たコミット型置換に対して，3.2.1 節の不動点除去操作を行う．見つかった不動点の集合を I_{FP} とし，$f = |I_{\mathrm{FP}}|$ とする．得られた $S_{n+t}\backslash I_{\mathrm{FP}}$ 上のコミット型 derangement の次数は $n+t-f$ 次である．

- $f > t$ の場合，コミット型 derangement の次数がプレイヤー数 n に満たないので，表向きになっているカードを全て裏向きに伏せ，Step 2 に戻る．
- $f = t$ の場合，ちょうどぴったりの次数であるので，Step 5 に進む．
- $f < t$ の場合，Step 4 に進む．

Step 4 Step 3 によって得られた $S_{n+t}\backslash I_{\mathrm{FP}}$ 上のコミット型 derangement に対して，3.2.2 節の縮退操作を繰り返す．1 回の縮退操作の適用で次数は 2 つまたは 1 つ減ることを思い出そう．

- ちょうど n 次のコミット型 derangement を得られた場合，Step 5 に進む．
- n 次のコミット型 derangement が得られなかった場合（ちょうど $n-1$ 次のコミット型 derangement を得られた場合），全てのカードを裏向きに伏せ，Step 2 に戻る．

Step 5 Step 4 で得られた（$S_{n+t}\backslash(I_{\mathrm{FP}} \cup I_{\mathrm{cycle}} \cup I_{\mathrm{BP}})$ 上の）n 次のコミット型 derangement において，各コミットメントをそれぞれ n 人のプレイヤーに対応させ，プレゼント交換の割り当てとする．ただし，バイパス $i_1 \to i_2 \cdots \to i_{\ell+1}$ が存在する場合，i_1 を表すコミットメントをめくったプレイヤーは，$i_{\ell+1}$ に対応するプレイヤーにプレゼントを贈れば良いことに注意する．

3.4 プロトコルの具体例

ここでは具体的に，プレイヤーの人数を $n = 4$，追加のカード束の数を $t = 3$ として，提案プロトコルを説明する．

(1) プレイヤーは式 (1) にしたがい，7 次の恒等置換を表すコミットメントを並べる．

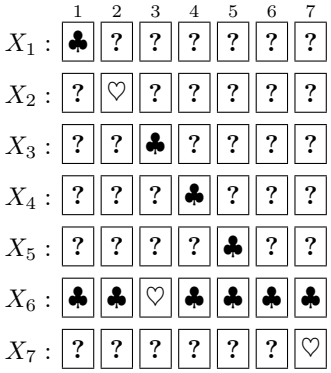
(2) 前項のコミットメントの列に Pile-scramble シャッフルを行う．このとき，コミット型置換として

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 2 & 4 & 6 & 1 & 3 & 7 \end{pmatrix}$$

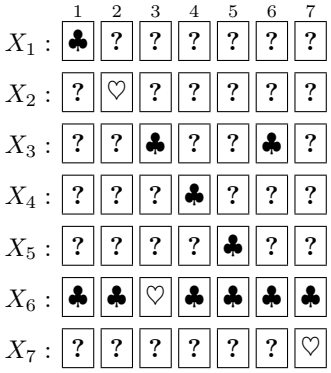
に対応したコミットメントの列 $(X_1, X_2, \ldots, X_7)$ を得たとする．

(3) 不動点除去操作を実行する．すなわち，対角線上に位置するカード 7 枚をめくり，不動点の有無を確認する．このとき，X_2 と X_7 は不動点なため，$I_{\mathrm{FP}} = \{2, 7\}$ であり，不動点の個数は $f = 2$ である．

(4) $S_7 \backslash I_{\mathrm{FP}}$ 上のコミット型 derangement に対して，残っている一番最後のコミットメント，すなわち $i_1 = \max(S_7 \backslash I_{\mathrm{FP}}) = 6$ 番目のコミットメントをめくる（縮退操作）．その値を $i_2\,(= 3)$ とする．



$i_2\,(= 3)$ 番目のコミットメントの $i_1\,(= 6)$ 番目のカードをめくる．黒 ♣ のカードが現れるので，不動点にはなっておらず，ちょうど 4 次の $S_7 \backslash (I_{\mathrm{FP}} \cup I_{\mathrm{BP}})$ 上のコミット型 derangement が得られた．ただし，$I_{\mathrm{BP}} = \{6\}$ である．



(5) 前項で得た 4 次のコミット型 derangement は，バイパス $6 \mapsto 3$ の下，$S_7 \backslash (I_{\mathrm{FP}} \cup I_{\mathrm{BP}})$ 上の次の置換を表現している．

$$\pi = \begin{pmatrix} 1 & 3 & 4 & 5 \\ 5 & 4 & 3 & 1 \end{pmatrix}$$

$\{1, 3, 4, 5\}$ を 4 人のプレイヤー P_1, P_2, P_3, P_4 と対応させることでプレゼント交換の割り当てとする．各プレイヤーは対応するコミットメントを他のプレイヤーに秘密にめくることで，その値に対応するプレイヤーにプレゼントを贈れば良いことが分かる．プレイヤー P_3 はめくるコミットメント X_4 の値が 6 となるが，バイパス $6 \mapsto 3$ より，3 に対応するプレイヤー P_2 にプレゼントを贈れば良い．

4. プロトコルの評価

本節では，提案プロトコルによって生成される置換の性質と，追加するカード束の数 t とシャッフル操作をやり直す確率に関する評価を行う．

4.1 生成する置換の性質

提案プロトコルによって生成される置換を考える．3.3 節の提案プロトコルでは最終的に Step 5 で n 次のコミット型 derangement を n 人のプレイヤーに対応させることで，プレゼント交換の割り当て行っている．Step 5 でプレイヤー $P_1, P_2, \ldots, P_n$ がそれぞれプレイヤー $P_{\rho(1)}, P_{\rho(2)}, \ldots, P_{\rho(n)}$ にプレゼントを贈るような割り当てとなったとき，提案プロトコルによって生成された derangement を ρ と表現する．例えば，3.4 節の具体例ではプレイヤー P_1, P_2, P_3, P_4 はそれぞれプレイヤー P_4, P_3, P_2, P_1 にプレゼントを贈るような割り当てとなるので，生成された ρ は次のように表現できる．

$$\rho = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

Step 5 で生成される ρ は Step 2 で適用される一様ランダムな置換 π によって一意に定まる．ここで，例として，表 1 にプレイヤー数 $n = 4$，追加のカード束の数 $t = 1$ のとき，生成される derangement の ρ と，それに対応した Step 2 における置換 π をすべて列挙する．表 1 の置換 ρ は型 $\langle 2^2 \rangle$ と型 $\langle 4 \rangle$ のいずれかの置換となっており，これらは S_4 における全ての derangement である．また，それぞれの ρ に対応する置換 π は 9 個ずつ存在し，置換 π の生成確率は $\frac{1}{5!}$ であるので，ρ のそれぞれの生成確率は $\frac{1}{5!} \times 9 = \frac{3}{40}$ となり，一様な確率分布となっていることが分かる．このように，提案プロトコルによって生成される derangement は一様ランダムである．

4.2 シャッフル操作をやり直す確率

ここでは，追加するカード束の数 $t\,(> 0)$ とシャッフル操作をやり直す確率について考える．やり直し確率は，Step 3 で不動点が t より多く見つかった場合と，Step 4 で $n - 1$ 次の derangement ができてしまった場合に発生する．

Step 3 でシャッフルをやり直す確率は Step 2 で適用される置換 $\pi \in S_{n+t}$ の不動点の個数 f が $t < f \leq n + t$ を満たすときである．$n + t$ 次対称群 S_{n+t} において，不動点を f 個持つ置換が生成される確率は $\frac{{}_{n+t}\mathrm{C}_f \cdot d_{n+t-f}}{(n+t)!}$ であるので，上の確率を $P_r[\text{Step 3 でやり直し}]$ とすると，次の式で

表 1: $n=4, t=1$ における置換 $\pi=(\pi(1),\pi(2),\pi(3),\pi(4),\pi(5))$ と生成される derangement $\rho=(\rho(1),\rho(2),\rho(3),\rho(4))$

ρ	対応する π	ρ	対応する π	ρ	対応する π	ρ	対応する π
	(1, 3, 2, 5, 4)		(1, 3, 5, 2, 4)		(1, 4, 5, 2, 3)		(1, 5, 2, 3, 4)
	(2, 1, 3, 5, 4)		(2, 4, 1, 3, 5)		(3, 4, 1, 2, 5)		(4, 1, 2, 3, 5)
	(2, 1, 4, 3, 5)		(2, 4, 1, 5, 3)		(3, 4, 1, 5, 2)		(4, 1, 2, 5, 3)
	(2, 1, 4, 5, 3)		(2, 4, 5, 3, 1)		(3, 4, 5, 2, 1)		(4, 1, 5, 3, 2)
(2, 1, 4, 3)	(2, 1, 5, 3, 4)	(2, 4, 1, 3)	(2, 5, 1, 3, 4)	(3, 4, 1, 2)	(3, 5, 1, 2, 4)	(4, 1, 2, 3)	(4, 5, 2, 3, 1)
	(2, 1, 5, 4, 3)		(2, 5, 1, 4, 3)		(3, 5, 1, 4, 2)		(5, 1, 2, 3, 4)
	(2, 5, 4, 3, 1)		(2, 5, 3, 1, 4)		(4, 2, 5, 1, 3)		(5, 1, 2, 4, 3)
	(3, 2, 1, 5, 4)		(3, 2, 5, 1, 4)		(4, 5, 3, 1, 2)		(5, 1, 3, 2, 4)
	(5, 1, 4, 3, 2)		(5, 4, 1, 3, 2)		(5, 4, 1, 2, 3)		(5, 2, 1, 3, 4)
	(1, 3, 4, 5, 2)		(1, 4, 2, 5, 3)		(1, 4, 5, 3, 2)		(1, 5, 4, 2, 3)
	(2, 3, 4, 1, 5)		(3, 1, 4, 2, 5)		(3, 4, 2, 1, 5)		(4, 3, 1, 2, 5)
	(2, 3, 4, 5, 1)		(3, 1, 4, 5, 2)		(3, 4, 2, 5, 1)		(4, 3, 1, 5, 2)
	(2, 3, 5, 1, 4)		(3, 1, 5, 2, 4)		(3, 4, 5, 1, 2)		(4, 3, 5, 2, 1)
(2, 3, 4, 1)	(2, 3, 5, 4, 1)	(3, 1, 4, 2)	(3, 1, 5, 4, 2)	(3, 4, 2, 1)	(3, 5, 2, 1, 4)	(4, 3, 1, 2)	(4, 5, 1, 2, 3)
	(2, 4, 3, 5, 1)		(3, 5, 4, 2, 1)		(3, 5, 2, 4, 1)		(5, 2, 4, 1, 3)
	(2, 5, 4, 1, 3)		(4, 1, 3, 5, 2)		(4, 2, 5, 3, 1)		(5, 3, 1, 2, 4)
	(3, 2, 4, 5, 1)		(4, 2, 1, 5, 3)		(4, 5, 3, 2, 1)		(5, 3, 1, 4, 2)
	(5, 3, 4, 1, 2)		(5, 1, 4, 2, 3)		(5, 4, 2, 1, 3)		(5, 4, 3, 1, 2)

ρ	対応する π
	(1, 5, 4, 3, 2)
	(4, 3, 2, 1, 5)
	(4, 3, 2, 5, 1)
	(4, 3, 5, 1, 2)
(4, 3, 2, 1)	(4, 5, 2, 1, 3)
	(5, 2, 4, 3, 1)
	(5, 3, 2, 1, 4)
	(5, 3, 2, 4, 1)
	(5, 4, 3, 2, 1)

表すことができる．

$$P_r[\text{Step 3 でやり直し}] = \sum_{f=t+1}^{n+t} \frac{{}_{n+t}\mathrm{C}_f \cdot d_{n+t-f}}{(n+t)!} \quad (2)$$

次に，Step 4 においてシャッフル操作をやり直す確率を考える．Step 4 において，$n+x$ 次のコミット型 derangement に対して縮退操作を行うとする．この操作を繰り返し，ちょうど n 次のコミット型 derangement が得られずシャッフル操作をやり直す確率を $\epsilon(n,x)$ とする．$n+x$ 次のコミット型 derangement に対して縮退操作を 1 回行うと，次数を 2 つ下げた $n+x-2$ 次のコミット型 derangement か，次数を 1 つ下げた $n+x-1$ 次のコミット型 derangement が得られる．前者の確率は，$n+x$ 次のコミット型 derangement が S_{n+x} の derangement のうち長さ 2 のサイクルを含むものである確率と等しく，その確率は $\frac{(n+x-1)d_{n+x-2}}{d_{n+x}}$ である．後者の確率は，$n+t$ 次のコミット型 derangement が長さ 2 のサイクルを含まない derangement である確率と等しく，その確率は $1-\frac{(n+x-1)d_{n+x-2}}{d_{n+x}}$ である．このことから，$\epsilon(n.x)$ は次のような漸化式で表現できる．

$$\begin{aligned}\epsilon(n,x) =& (1-\frac{(n+x-1)d_{n+x-2}}{d_{n+x}})\cdot\epsilon(n,x-1)\\ &+\frac{(n+x-1)d_{n+x-2}}{d_{n+x}}\cdot\epsilon(n,x-2)\end{aligned}$$

ただし，$\epsilon(n,0)=0, \epsilon(n,1)=\frac{n\cdot d_{n-1}}{d_{n+1}}$ である．Step 4 でプロトコルをやり直す確率は，Step 2 で適用される置換 $\pi\in S_{n+t}$ が不動点を $f=0$ 個から $f=t-1$ 個持つ（余分なコミットメントの個数は $x=t-f$ となる）確率とそれぞれのやり直し確率 $\epsilon(n,t-f)$ の積の合計の確率で表せることができ，次の式となる．

$$P_r[\text{Step 4 でやり直し}] = \sum_{f=0}^{t-1} \frac{{}_{n+t}\mathrm{C}_f \cdot d_{n+t-f}}{(n+t)!}\cdot\epsilon(n,t-f) \quad (3)$$

以上より，追加のカード束を t 個用意したときの提案プロトコル全体のシャッフル操作をやり直す確率 $p_{\mathrm{Fail}}(n,t)$ は式 (2) と式 (3) の和であり，プレイヤーの人数が $n=3$ から $n=20$ における $t\,(\leq 10)$ と確率 $p_{\mathrm{Fail}}(n,t)$ の関係を図 1 に示す．$t=0$ は Ishikawa らの既存プロトコルにおけるシャッフル操作をやり直す確率と同じであり，t が大きくなるにつれその確率は大きく改善され，$t=2$ ではほとんどの n で $p_{\mathrm{Fail}}(n,t)\leq 0.2$ まで改善される．追加のカード束を t 個用意するとカード枚数が $t(t+2n)$ 枚多くなるため，t をむやみに大きくすることは現実的ではない．提案プロトコルを実行するにあたって，t を 2 や 3 程度の小さい値に設定しても，既存プロトコルのシャッフル操作をやり直す確率（およそ 0.6）を 0.1 〜 0.2 の低い確率に効率化することができる．

5. おわりに

本稿では，プレイヤーの人数 n より大きい数のカード束を用意することで，シャッフル操作のやり直しを防ぐというアイデアに基づく，新しいプロトコルを提案した．この提案プロトコルは n 次の一様ランダムな derangement を生成している．また，追加するカード束の個数 t とシャッフル操作をやり直す確率の関係から t を大きくすることで

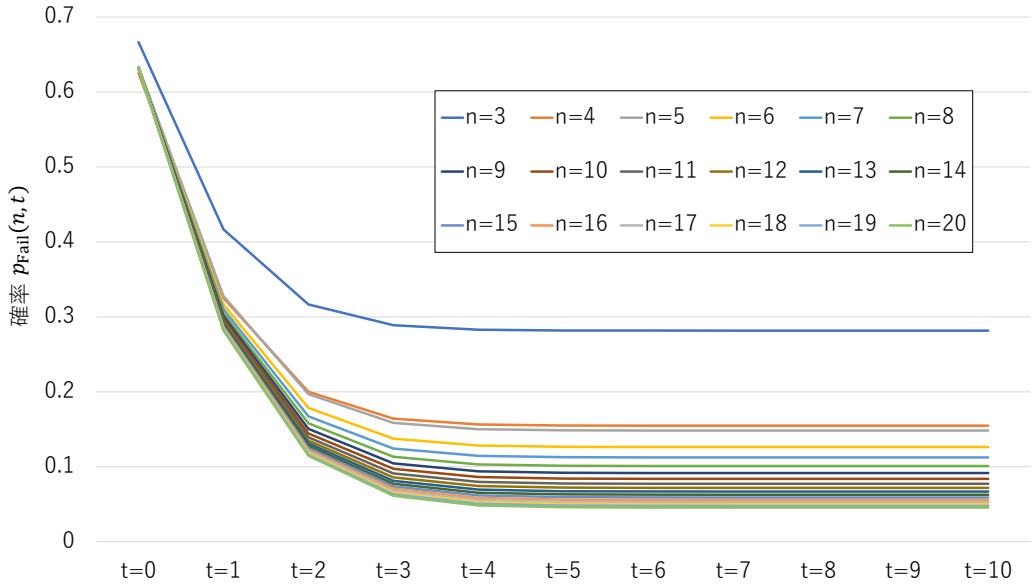


図 1: 追加するカード束の数 t とシャッフル操作をやり直す確率 $p_{\mathrm{Fail}}(n,t)$ の関係

その確率を改善できることを示した．実際に提案プロトコルを実行する場合には，必要なカード枚数とシャッフル操作をやり直す確率はプロトコル効率化においてトレードオフの関係にあるため，$t = 2, 3$ に設定するのが良いと考えられる．

また，本稿では自然数を 10 進数で表現するカード束を用いたが，それを 2 進数のように表現して計算することで，シャッフル回数は多くなるもののカード枚数の効率化を図ることができると考えられる．

参考文献

[1] C. Crépeau and J. Kilian, “Discreet solitary games,” Advances in Cryptology — CRYPTO’ 93, ed. by D.R. Stinson, vol.773, pp.319–330, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1994.

[2] J. Heather, S. Schneider, and V. Teague, “Cryptographic protocols with everyday objects,” Formal Aspects of Computing, vol.26, pp.37–62, 01 2014.

[3] T. Ibaraki and Y. Manabe, “A more efficient card-based protocol for generating a random permutation without fixed points,” 2016 Third International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), pp.252–257, Aug. 2016.

[4] R. Ishikawa, E. Chida, and T. Mizuki, “Efficient card-based protocols for generating a hidden random permutation without fixed points,” Unconventional Computation and Natural Computation, eds. by C.S. Calude and M.J. Dinneen, vol.9252, pp.215–226, Lecture Notes in Computer Science, Springer, Cham, 2015.

[5] 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 Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E101.A, pp.1503–1511, 09 2018.

[6] B. denBoer, “More efficient match-making and satisfiability the five card trick,” Advances in Cryptology — EUROCRYPT ’89, eds. by J.-J. Quisquater and J. Vandewalle, vol.434, pp.208–217, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1990.

[7] T. Mizuki, “Card-based protocols for securely computing the conjunction of multiple variables,” Theor. Comput. Sci., vol.622, no.C, pp.34–44, April 2016. https://doi.org/10.1016/j.tcs.2016.01.039

[8] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone, “Securely computing three-input functions with eight cards,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E98.A, no.6, pp.1145–1152, 2015.

[9] T. Nakai, Y. Tokushige, Y. Misawa, M. Iwamoto, and K. Ohta, “Efficient card-based cryptographic protocols for millionaires’ problem utilizing private permutations,” Cryptology and Network Security, eds. by S. Foresti and G. Persiano, vol.10052, pp.500–517, Lecture Notes in Computer Science, Springer, Cham, 2016.

[10] H. Ono and Y. Manabe, “Efficient card-based cryptographic protocols for the millionaires’ problem using private input operations,” 2018 13th Asia Joint Conference on Information Security (AsiaJCIS), pp.23–28, Aug. 2018.

[11] D. Miyahara, Y. Hayashi, T. Mizuki, and H. Sone, “Practical card-based implementations of Yao’s millionaire protocol,” Theoretical Computer Science, pp.207–221, 2019.

[12] Y. Hashimoto, K. Shinagawa, K. Nuida, M. Inamura, and G. Hanaoka, “Secure grouping protocol using a deck of cards,” Information Theoretic Security, ed. by J. Shikata, vol.10681, pp.135–152, Lecture Notes in Computer Science, Springer, Cham, 2017.

[13] T. Sasaki, T. Mizuki, and H. Sone, “Card-Based Zero-Knowledge Proof for Sudoku,” 9th International Conference on Fun with Algorithms (FUN 2018), eds. by H. Ito, S. Leonardi, L. Pagli, and G. Prencipe, vol.100, pp.29:1–29:10, Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018. http://drops.dagstuhl.de/opus/volltexte/2018/8820