入力の非二進符号化に基づく効率的な大小比較カードプロトコル

縫田　光司（NUIDA, Koji）

九州大学 マス・フォア・インダストリ研究所
／産業技術総合研究所

SCIS 2023 ＠小倉/オンライン　2023年1月26日

概要

2値カード／番号カードを用いた（2者間）
大小比較カードプロトコルの提案

- 「綱引きの原理」に基づく新しい構成法
- 利点１：入力値のq進符号化（$q \geq 2$）への拡張が容易
  - $q = 3$を用いれば**カード枚数・シャッフル回数を既存方式より漸近的に削減**（2値カードの場合）
- 利点２：番号カードへの拡張が容易
  - （カード枚数は増えるが）**シャッフル回数を削減**

目次

- 背景：カードベース暗号
- 大小比較カードプロトコルの既存方式
- 提案方式（2値カード）
- 提案方式（番号カード）

目次

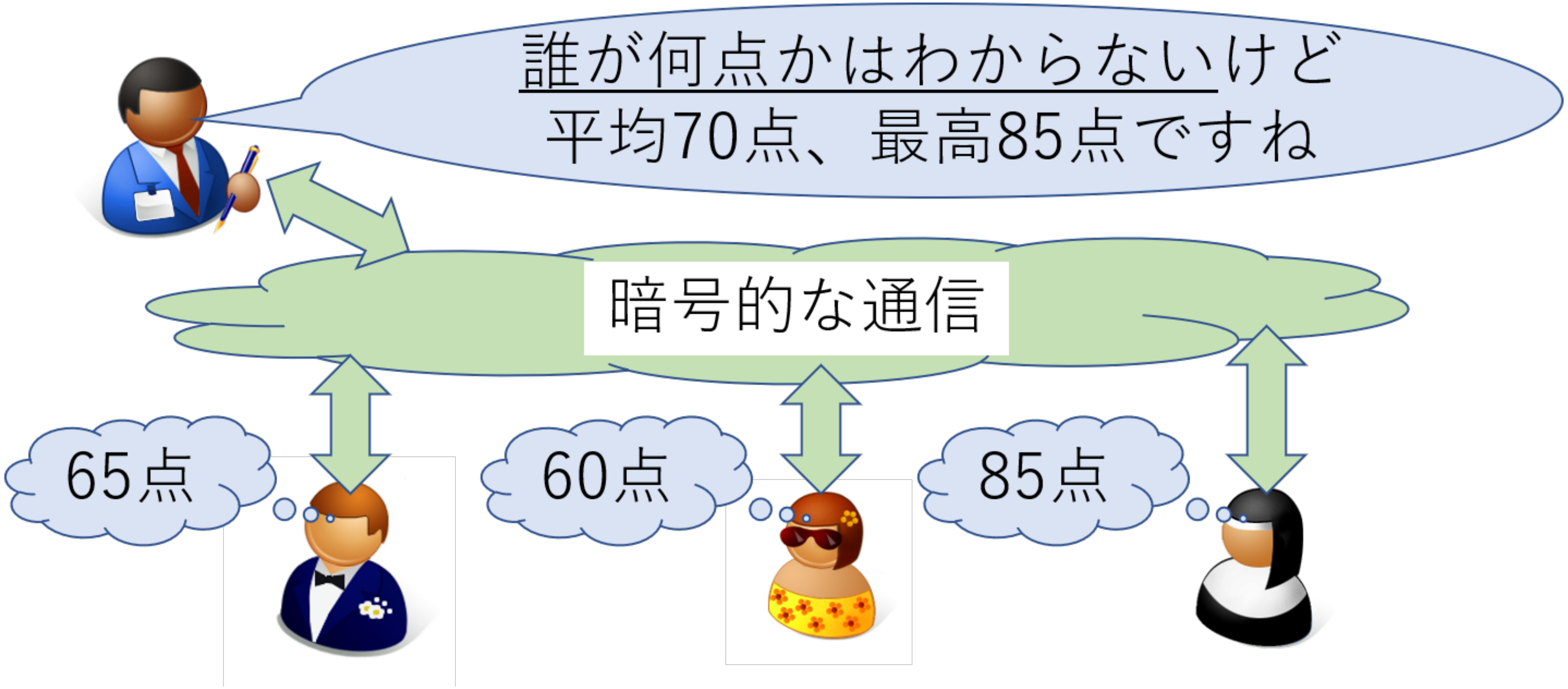
- **<u>背景：カードベース暗号</u>**
- 大小比較カードプロトコルの既存方式
- 提案方式（2値カード）
- 提案方式（番号カード）

秘密計算とは

互いの入力を隠したまま、所望の出力のみを得る技術

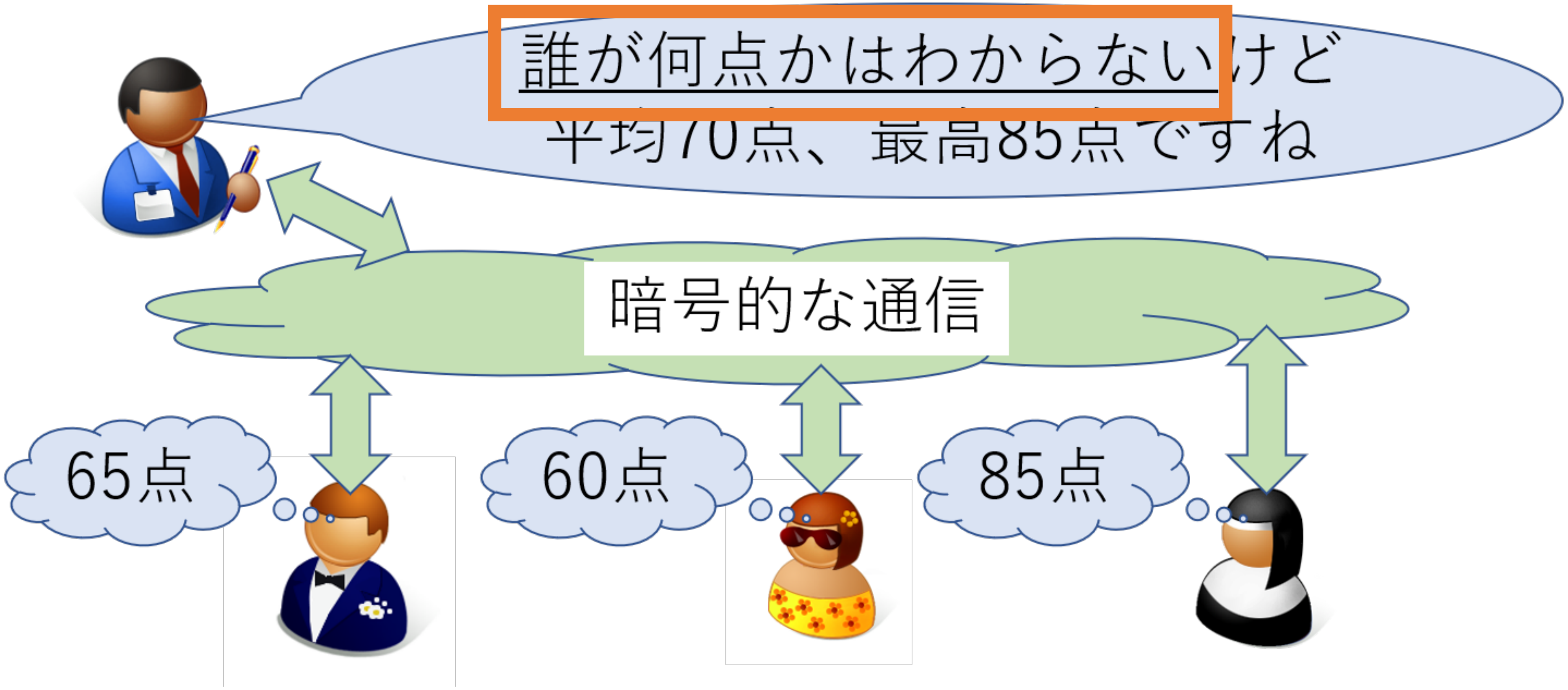
秘密計算とは

互いの入力を隠したまま、所望の出力のみを得る技術



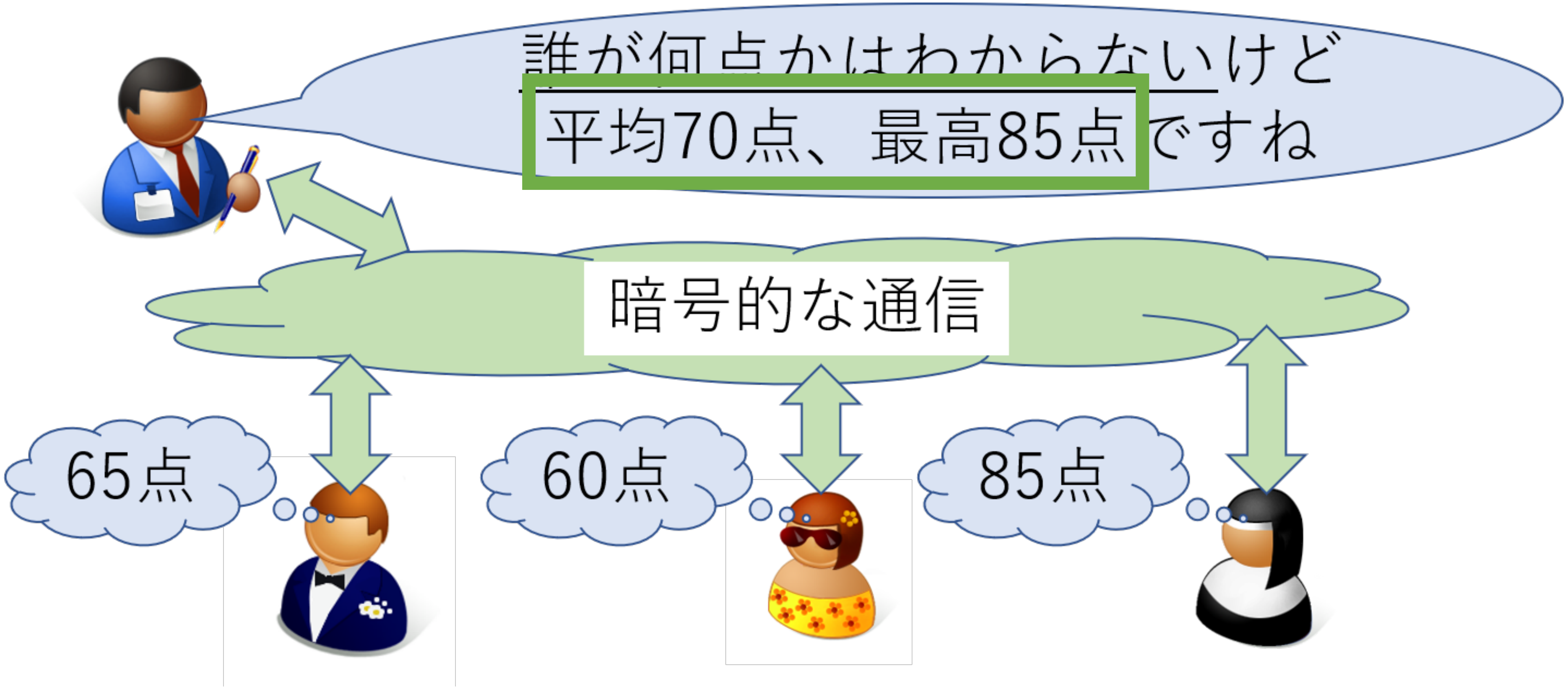
秘密計算とは

互いの入力を隠したまま、所望の出力のみを得る技術



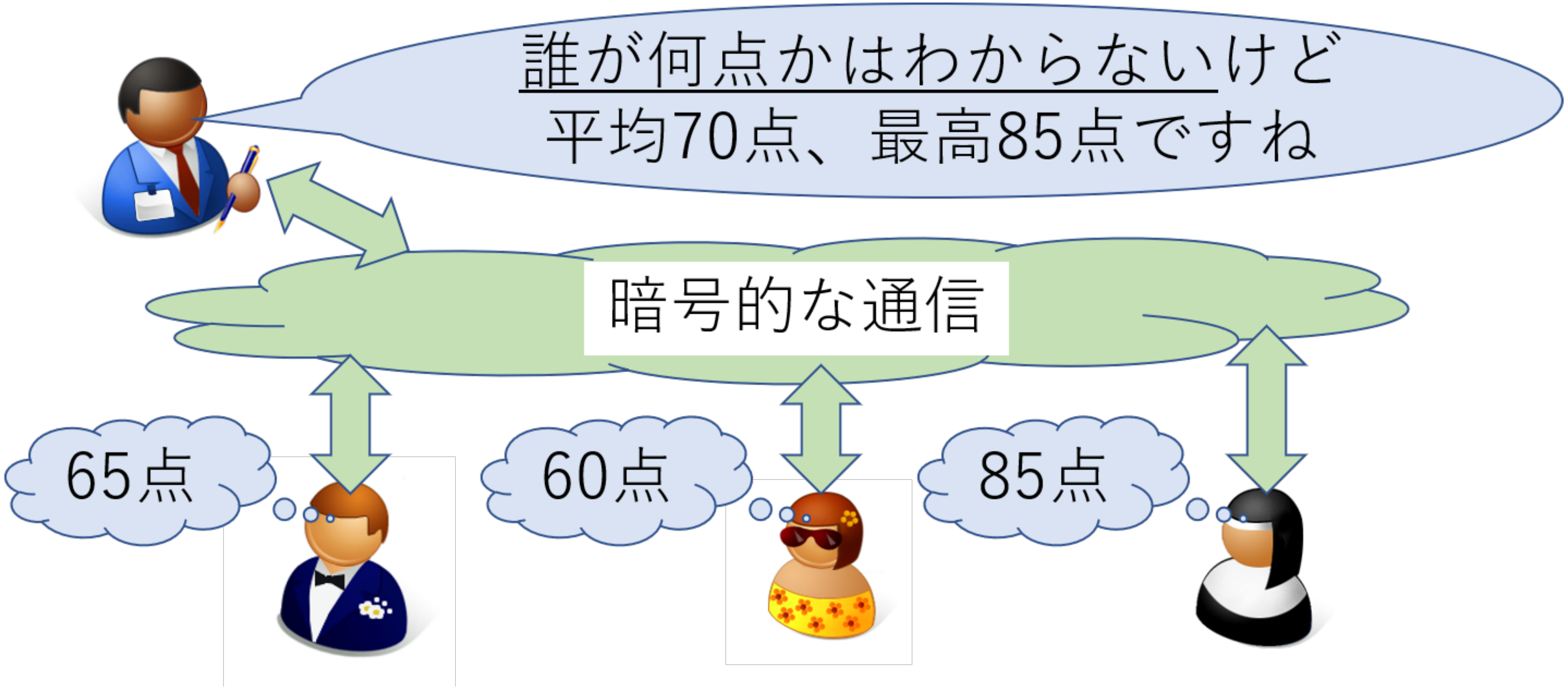
秘密計算とは

互いの入力を隠したまま、所望の出力のみを得る技術



秘密計算とは

互いの入力を隠したまま、所望の出力のみを得る技術



カードベース暗号（カードプロトコル）

- 物理的（非電子的）なカードを用いた秘密計算（[den Boer '89]、…）
- 裏側の模様が同一（識別不可能）なカード列を用いる
  - **2値カード**：表側の模様が2種類（♥、♣）
  - **番号カード**：表側に異なる数字が書かれている
- 複数のカードの**シャッフル**によりランダム性を実現
  - シャッフルを含め、カードはすべて公開の場で操作（本発表では秘匿置換は考えない）

[B. den Boer, "More Efficient Match-Making and Satisfiability: The Five Card Trick", EUROCRYPT 1989]

カードベース暗号（カードプロトコル）

- 物理的（非電子的）なカードを用いた秘密計算（[den Boer '89]、…）
- 裏側の模様が同一（識別不可能）なカード列を用いる
  - **2値カード**：表側の模様が2種類（♥、♣）
  - **番号カード**：表側に異なる数字が書かれている
- 複数のカードの**シャッフル**によりランダム性を実現
  - シャッフルを含め、カードはすべて公開の場で操作（本発表では秘匿置換は考えない）

[B. den Boer, "More Efficient Match-Making and Satisfiability: The Five Card Trick", EUROCRYPT 1989]

カードベース暗号（カードプロトコル）

- 物理的（非電子的）なカードを用いた秘密計算
  ([den Boer '89]、…)
- 裏側の模様が同一（識別不可能）なカード列を用いる
  - **2値カード**：表側の模様が2種類（♥、♣）
  - **番号カード**：表側に異なる数字が書かれている
- 複数のカードの**シャッフル**によりランダム性を実現
  - シャッフルを含め、カードはすべて公開の場で操作
    （本発表では秘匿置換は考えない）

[B. den Boer, "More Efficient Match-Making and Satisfiability: The Five Card Trick", EUROCRYPT 1989]

カードベース暗号（カードプロトコル）

- 物理的（非電子的）なカードを用いた秘密計算
  ([den Boer '89]、…)
- 裏側の模様が同一（識別不可能）なカード列を用いる
  - **2値カード**：表側の模様が2種類（♥、♣）
  - **番号カード**：表側に異なる数字が書かれている
- 複数のカードの**シャッフル**によりランダム性を実現
  - シャッフルを含め、カードはすべて公開の場で操作
    （本発表では秘匿置換は考えない）

[B. den Boer, “More Efficient Match-Making and Satisfiability: The Five Card Trick”, EUROCRYPT 1989]

（本発表で扱う）シャッフルの種類

ランダムカット（RC）

1 2 3

→
- 1 2 3 or
- 2 3 1 or
- 3 1 2

パイルシフティングシャッフル（PSS）

1 2 a b A B

→
- 1 2 a b A B or
- a b A B 1 2 or
- A B 1 2 a b

完全シャッフル（CS）

1 2 3

→
- 1 2 3 or
- 1 3 2 or
- ⋮
- 3 2 1

＊実際はすべて裏向きにして行う

（本発表で扱う）シャッフルの種類

ランダムカット（RC）

1 2 3

1 2 3 or

2 3 1 or

3 1 2

パイルシフティングシャッフル（PSS）

1 2 a b A B

1 2 a b A B or

a b A B 1 2 or

A B 1 2 a b

完全シャッフル（CS）

1 2 3

1 2 3 or

1 3 2 or

⋮

3 2 1

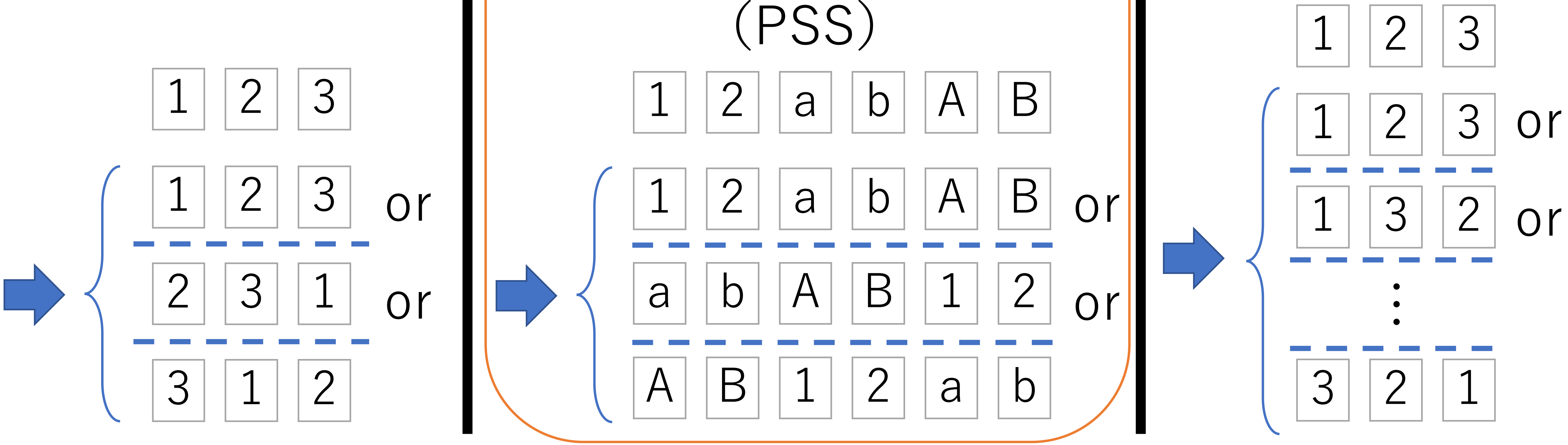
＊実際はすべて裏向きにして行う

（本発表で扱う）シャッフルの種類

ランダムカット (RC)

パイルシフティングシャッフル (PSS)

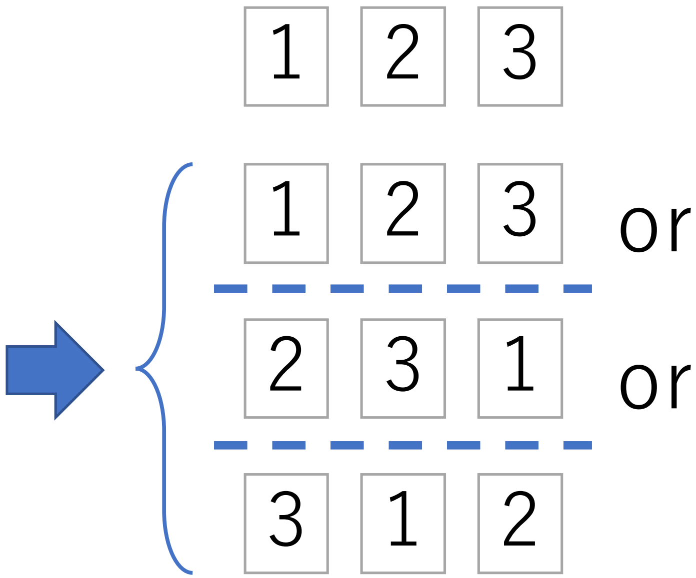
完全シャッフル (CS)



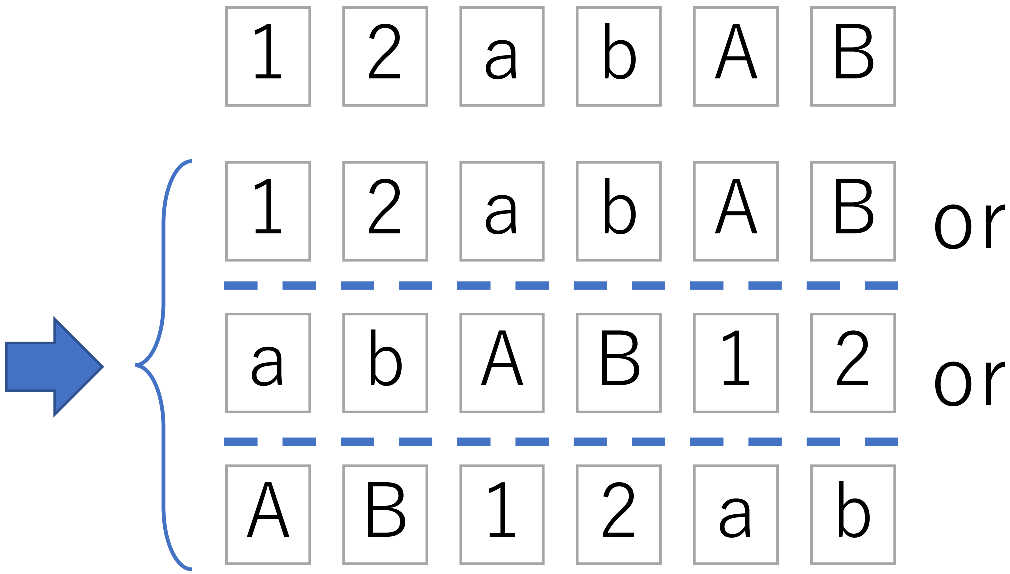
＊実際はすべて裏向きにして行う

（本発表で扱う）シャッフルの種類

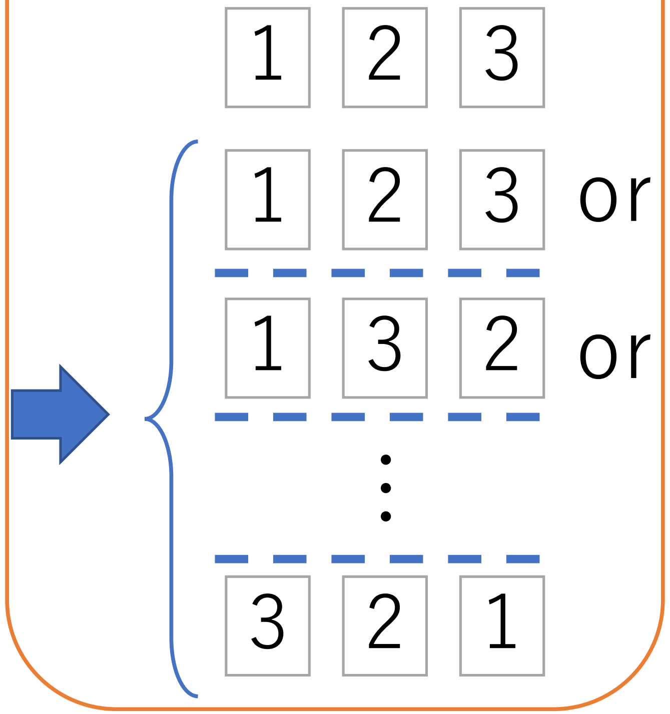
ランダムカット（RC）



パイルシフティングシャッフル（PSS）



完全シャッフル（CS）



＊実際はすべて裏向きにして行う

目次

- 背景：カードベース暗号
- **<u>大小比較カードプロトコルの既存方式</u>**
- 提案方式（2値カード）
- 提案方式（番号カード）

大小比較の秘密計算

- 2者間で（ある範囲内の）整数値入力を大小比較する
- 通常の秘密計算でも代表的な問題の一つ [Yao ’82]
- カードベース暗号でも同様に重要
  （[Nakai+ ’16][Ono-Manabe ’18][Miyahara+ ’20]…）

[A. C.-C. Yao, “Protocols for Secure Computations”, FOCS 1982]

[T. Nakai et al., “Efficient Card-Based Cryptographic Protocols for Millionaires’ Problem Utilizing Private Permutation”, CANS 2016]

[H. Ono, Y. Manabe, “Efficient Card-Based Cryptographic Protocol for the Millionaires’ Problem Using Private Input Operations”, AsiaJCIS 2018]

[D. Miyahara et al., “Practical Card-Based Implementations of Yao’s Millionaire Protocol”, Theoretical Computer Science (2020)]

[Nakai+ ’16]のプロトコル

（をシャッフルベースに変換＋説明しやすくしたもの）

＊[Yao ’82]の大小比較プロトコルのカードベース版

[Nakai+ '16]のプロトコル

（をシャッフルベースに変換＋説明しやすくしたもの）

＊[Yao '82]の大小比較プロトコルのカードベース版

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

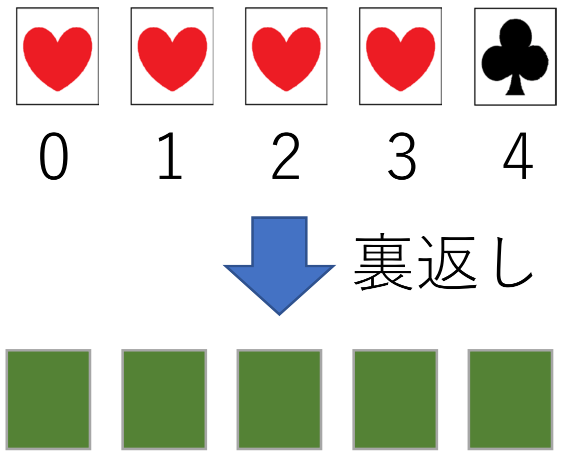
[Nakai+ '16]のプロトコル

（をシャッフルベースに変換＋説明しやすくしたもの）

＊[Yao '82]の大小比較プロトコルのカードベース版

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

1 （Alice入力）



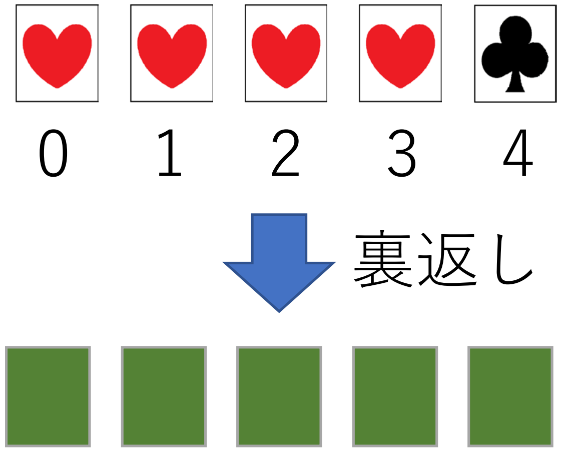
[Nakai+ '16]のプロトコル

（をシャッフルベースに変換＋説明しやすくしたもの）

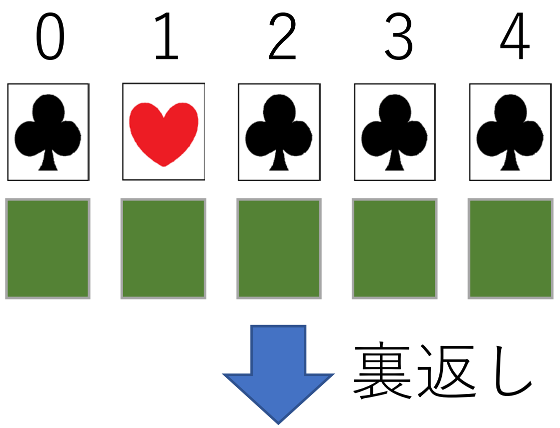
＊[Yao '82]の大小比較プロトコルのカードベース版

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

1（Alice入力）



2（Bob入力）



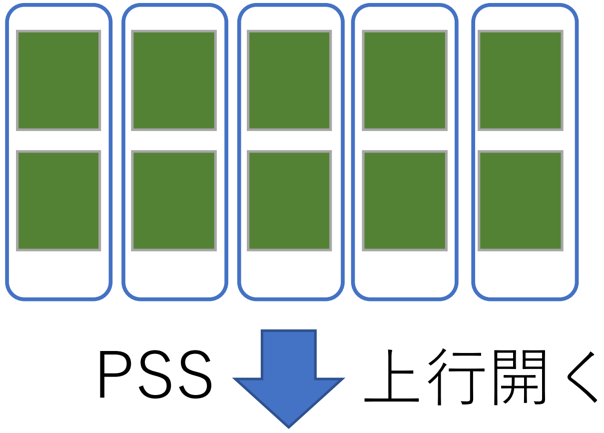
[Nakai+ ’16]のプロトコル

（をシャッフルベースに変換＋説明しやすくしたもの）

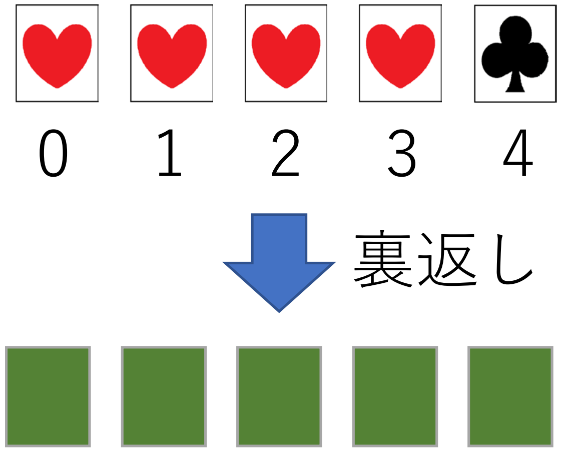
＊[Yao ’82]の大小比較プロトコルのカードベース版

例）入力範囲$[0,4]$、入力$\alpha = 3, \beta = 1$

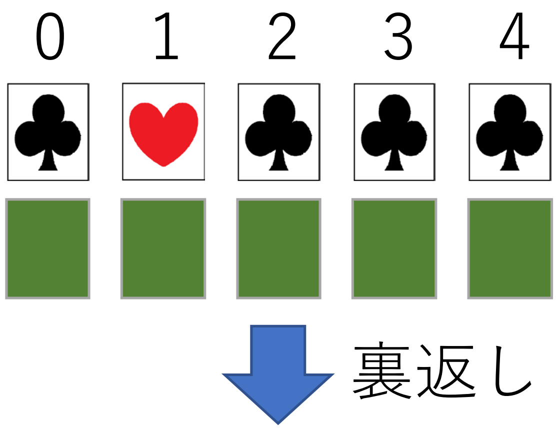
③



① (Alice入力)



② (Bob入力)

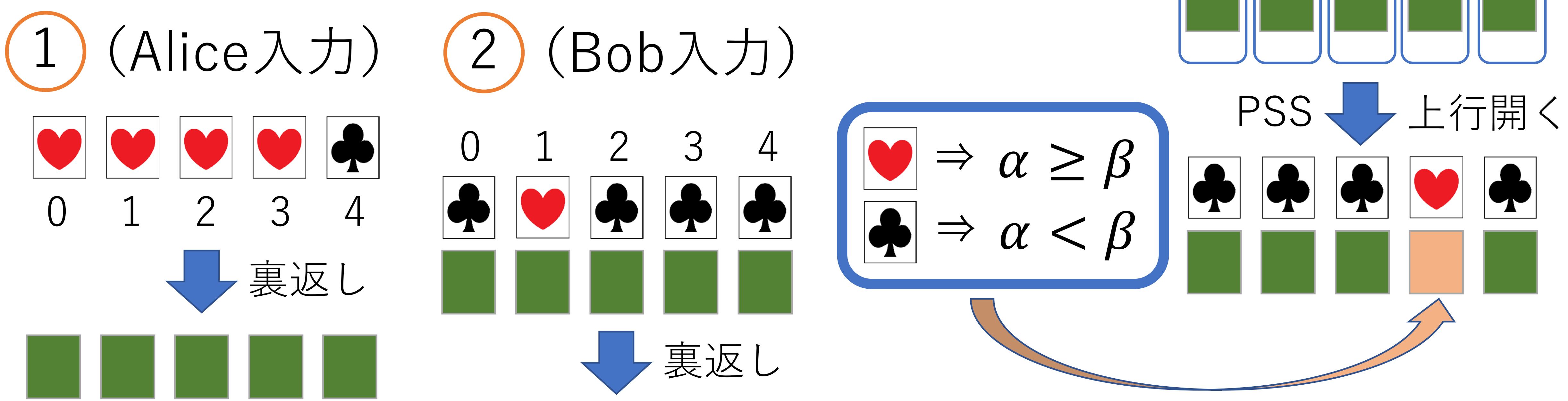


[Nakai+ '16]のプロトコル

（をシャッフルベースに変換＋説明しやすくしたもの）

＊[Yao '82]の大小比較プロトコルのカードベース版

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

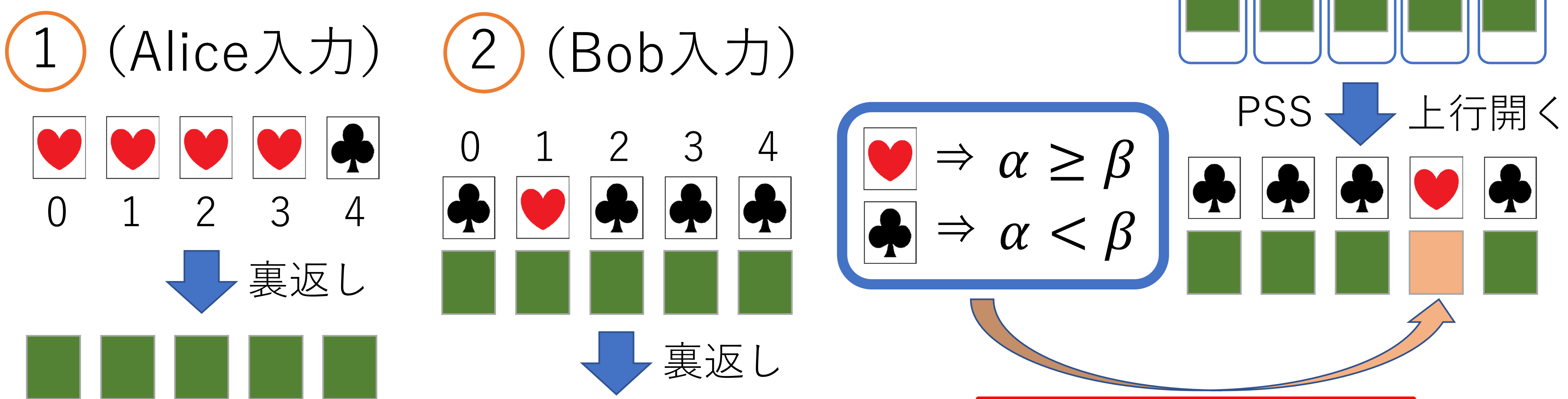


[Nakai+ ’16]のプロトコル

（をシャッフルベースに変換＋説明しやすくしたもの）

＊[Yao ’82]の大小比較プロトコルのカードベース版

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$



☹ カードが多い

[Miyahara+ '20]のプロトコル

- 通常の大小比較：上位ビットから順番に比較
  - 決着がつけば終了、引き分けなら次のビットを比較
- ↑を既存のビット演算カードプロトコルで実現
- 2値カードの場合：カード$4\ell+2$枚、RC $2\ell-1$回（ℓ：入力値の最大ビット長）
- 番号カードの場合：カード$4\ell+4$枚、**PSS $6\ell-2$回**
  - ランダムバイセクションカット（PSSの一種）使用

[Miyahara+ '20]のプロトコル

- 通常の大小比較：上位ビットから順番に比較
  - 決着がつけば終了、引き分けなら次のビットを比較
- ↑を既存のビット演算カードプロトコルで実現
- 2値カードの場合：カード$4\ell + 2$枚、RC $2\ell - 1$回
  （ℓ：入力値の最大ビット長）
- 番号カードの場合：カード$4\ell + 4$枚、**PSS $6\ell - 2$回**
  - ランダムバイセクションカット（PSSの一種）使用

[Miyahara+ '20]のプロトコル

- 通常の大小比較：上位ビットから順番に比較
  - 決着がつけば終了、引き分けなら次のビットを比較
- ↑を既存のビット演算カードプロトコルで実現
- 2値カードの場合：カード$4\ell + 2$枚、RC $2\ell - 1$回
  （ℓ：入力値の最大ビット長）
- 番号カードの場合：カード$4\ell + 4$枚、**PSS $6\ell - 2$回**
  - ランダムバイセクションカット（PSSの一種）使用

効率化に向けた考察

- [Miyahara+ '20]のカード・シャッフル数は入力値のビット長（＝再帰の深さ）の線形オーダー
  - q進符号化（$q > 2$）を使えば再帰の深さを減らせる
  - ↑各桁の比較部分をカードでどう実現する？
- [Nakai+ '16]を各桁の比較に使えないか？
  - ☺ 値の範囲 q が増えても**シャッフル回数は増えない**
  - ☹ **引き分けを区別できない**ので再帰的に使えない
- → [Nakai+ '16]の**利点を保ったまま再帰可能にしたい**

効率化に向けた考察

- [Miyahara+ '20]のカード・シャッフル数は入力値のビット長（＝再帰の深さ）の線形オーダー
  - **q進符号化（$q > 2$）を使えば再帰の深さを減らせる**
  - **↑各桁の比較部分をカードでどう実現する？**
- [Nakai+ '16]を各桁の比較に使えないか？
  - ☺ 値の範囲 q が増えても**シャッフル回数は増えない**
  - ☹ **引き分けを区別できない**ので再帰的に使えない
- → [Nakai+ '16]の**利点を保ったまま再帰可能にしたい**

効率化に向けた考察

- [Miyahara+ '20]のカード・シャッフル数は入力値のビット長（＝再帰の深さ）の線形オーダー
  - **q進符号化（$q > 2$）を使えば再帰の深さを減らせる**
  - **↑各桁の比較部分をカードでどう実現する？**
- [Nakai+ '16]を各桁の比較に使えないか？
  - ☺ 値の範囲 q が増えても**シャッフル回数は増えない**
  - ☹ **引き分けを区別できない**ので再帰的に使えない
- → [Nakai+ '16]の**利点を保ったまま再帰可能にしたい**

効率化に向けた考察

- [Miyahara+ '20]のカード・シャッフル数は入力値のビット長（＝再帰の深さ）の線形オーダー
  - **q進符号化（$q > 2$）を使えば再帰の深さを減らせる**
  - **↑各桁の比較部分をカードでどう実現する？**
- [Nakai+ '16]を各桁の比較に使えないか？
  - ☺ 値の範囲 q が増えても**シャッフル回数は増えない**
  - ☹ **引き分けを区別できない**ので再帰的に使えない
- → [Nakai+ '16]の**利点を保ったまま再帰可能にしたい**

目次

- 背景：カードベース暗号
- 大小比較カードプロトコルの既存方式
- **提案方式（2値カード）**
- 提案方式（番号カード）

アイデア：「綱引きの原理」

- 綱の中央に印を付けておく
- まず、Aliceは長さ α だけ綱を自分の側に引っ張る
- 次に、Bobは長さ β だけ綱を自分の側に引っ張る
- 印がAlice（, Bob）の側にあれば$\alpha > \beta$（, $\alpha < \beta$）、印が中央に残れば$\alpha = \beta$
  - [Nakai+ '16]とは異なり、**引き分けも区別できる**
  - ＊**これ自体は安全ではない**（が、出力は正しい）

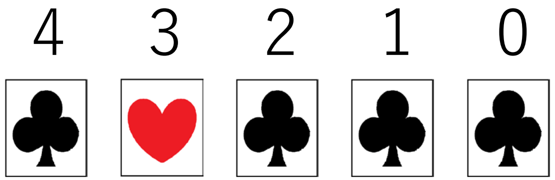
アイデア：「綱引きの原理」

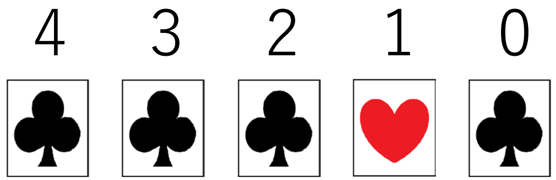
- 綱の中央に印を付けておく
- まず、Aliceは長さ α だけ綱を自分の側に引っ張る
- 次に、Bobは長さ β だけ綱を自分の側に引っ張る
- 印がAlice（, Bob）の側にあれば$\alpha > \beta$（, $\alpha < \beta$）、印が中央に残れば$\alpha = \beta$
  - [Nakai+ '16]とは異なり、**引き分けも区別できる**
  - ＊**これ自体は安全ではない**（が、出力は正しい）

提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

$\alpha =$ 

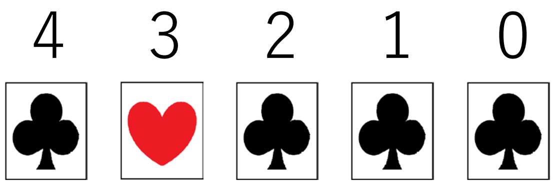
$\beta =$ 

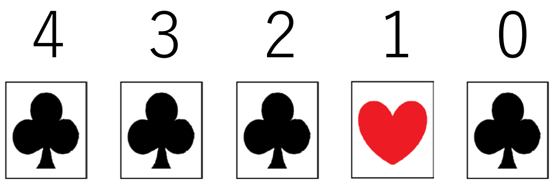
1

提案方式のサブプロトコル

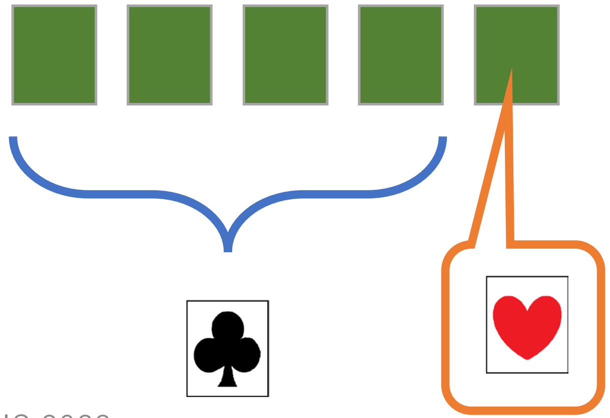
例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

$\alpha =$ 

$\beta =$ 

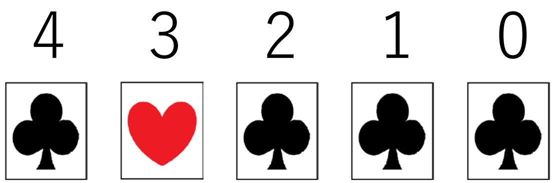
1

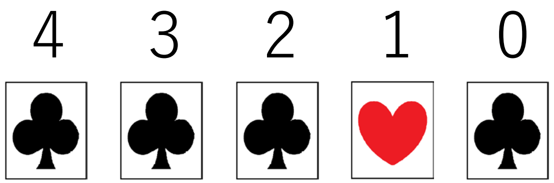


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

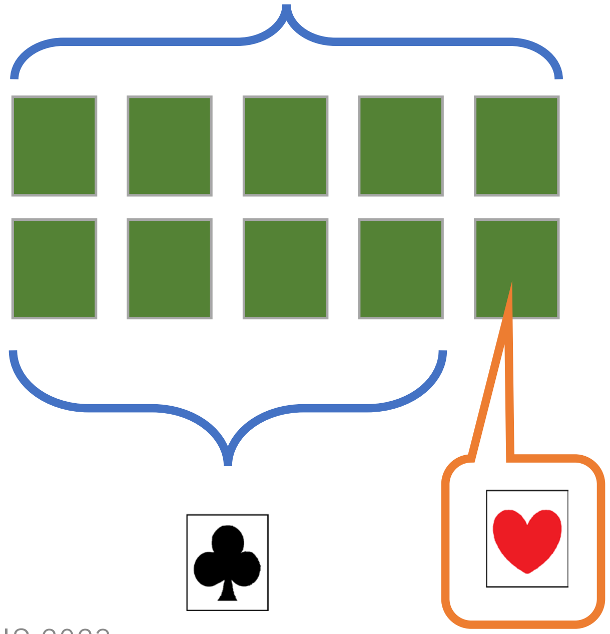
＊入力をunary符号化

$\alpha =$ 

$\beta =$ 

αの逆順

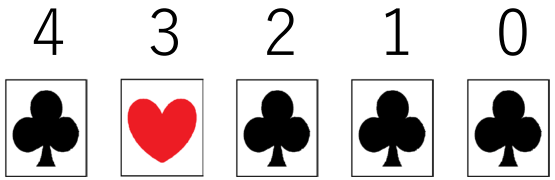
1

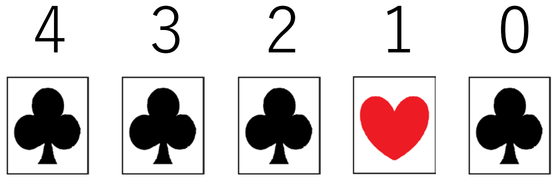


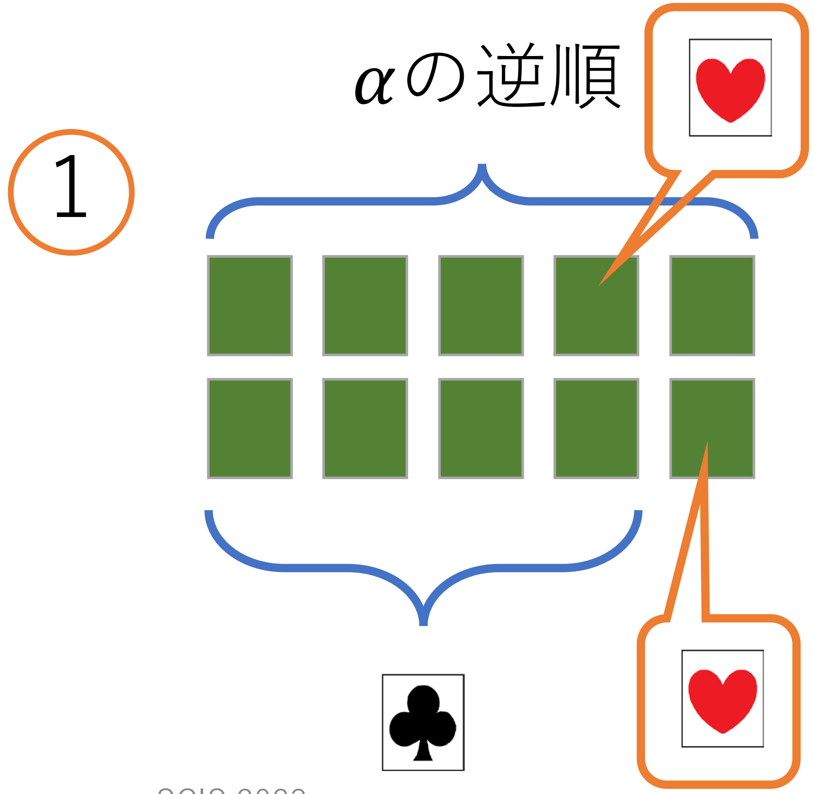
提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

$\alpha =$ 

$\beta =$ 



提案方式のサブプロトコル

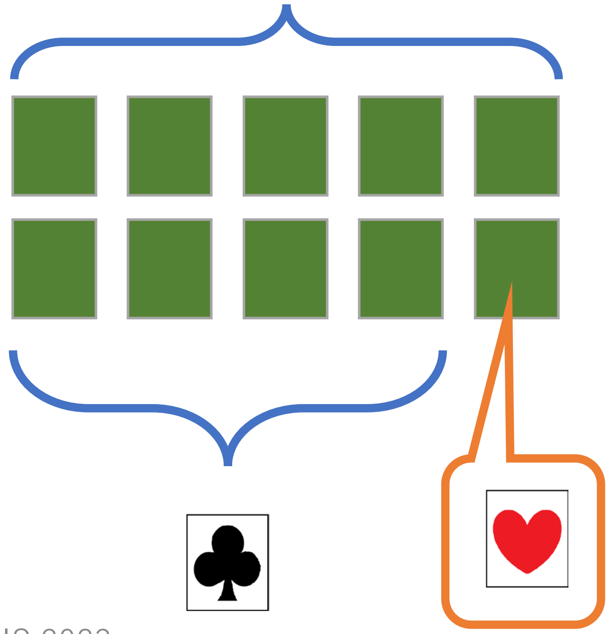
例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

	4	3	2	1	0
$\alpha =$	♣	♥	♣	♣	♣
$\beta =$	♣	♣	♣	♥	♣

αの逆順

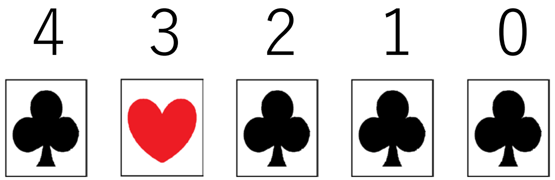
1

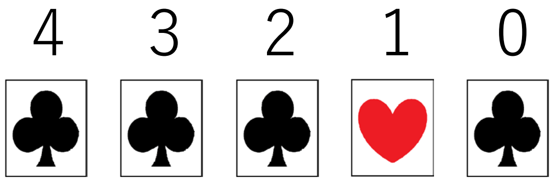


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

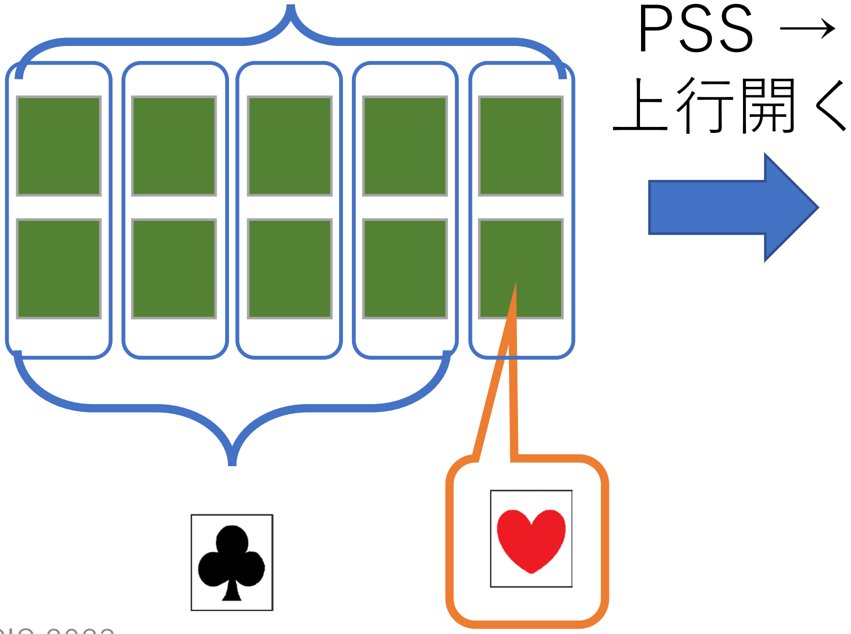
＊入力をunary符号化

$\alpha =$ 

$\beta =$ 

αの逆順

1

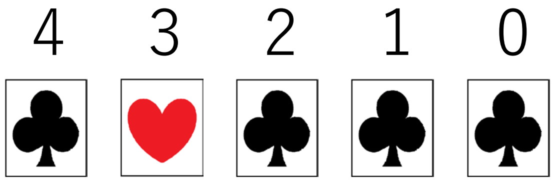


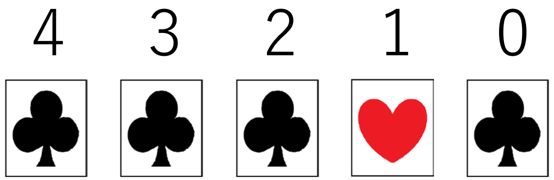
PSS →
上行開く

提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

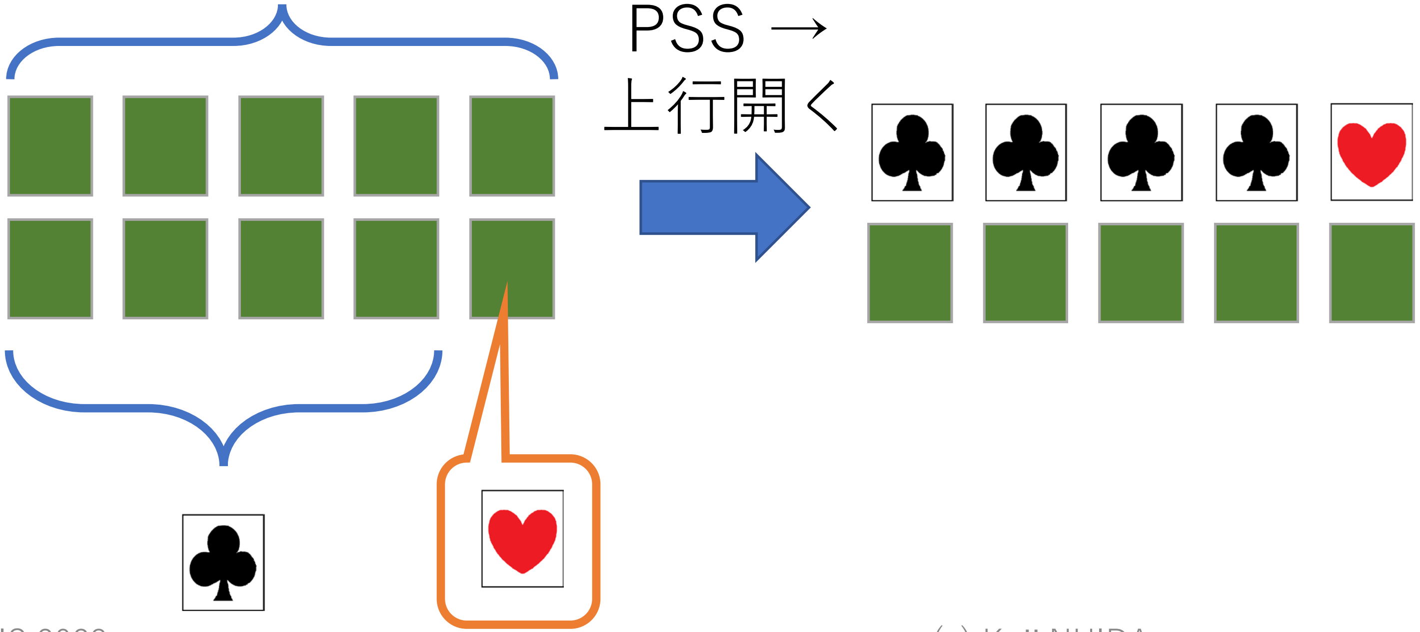
＊入力をunary符号化

$\alpha =$ 

$\beta =$ 

αの逆順

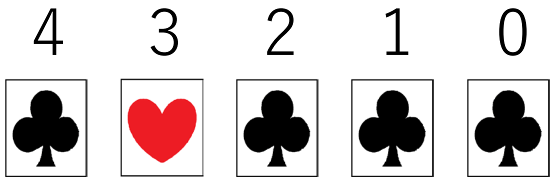
1

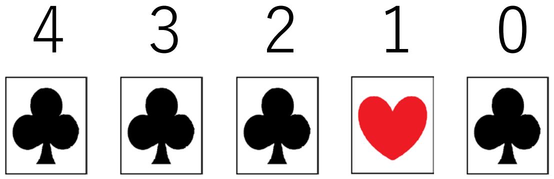


提案方式のサブプロトコル

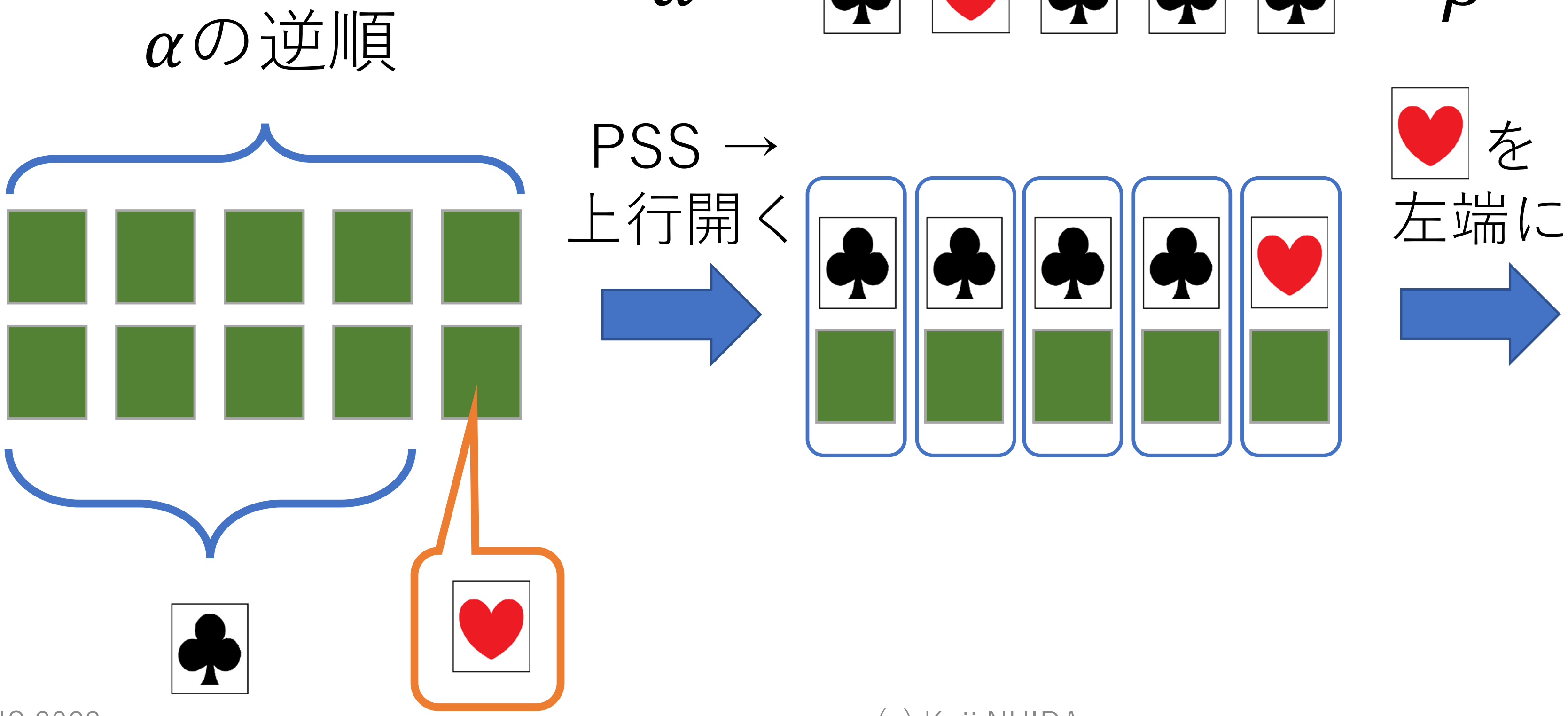
例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

$\alpha =$ 

$\beta =$ 

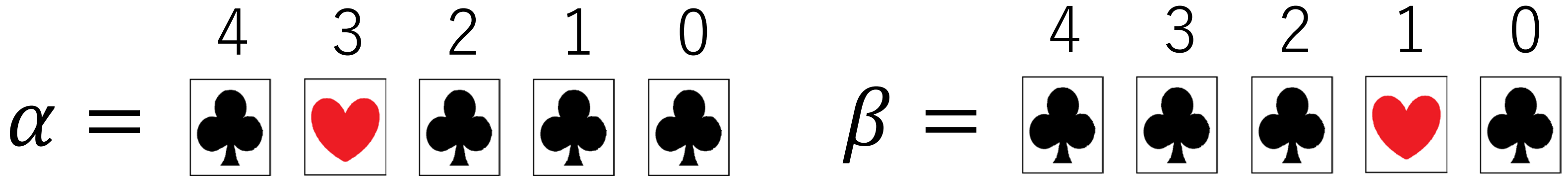
1



提案方式のサブプロトコル

例）入力範囲$[0,4]$、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化



提案方式のサブプロトコル

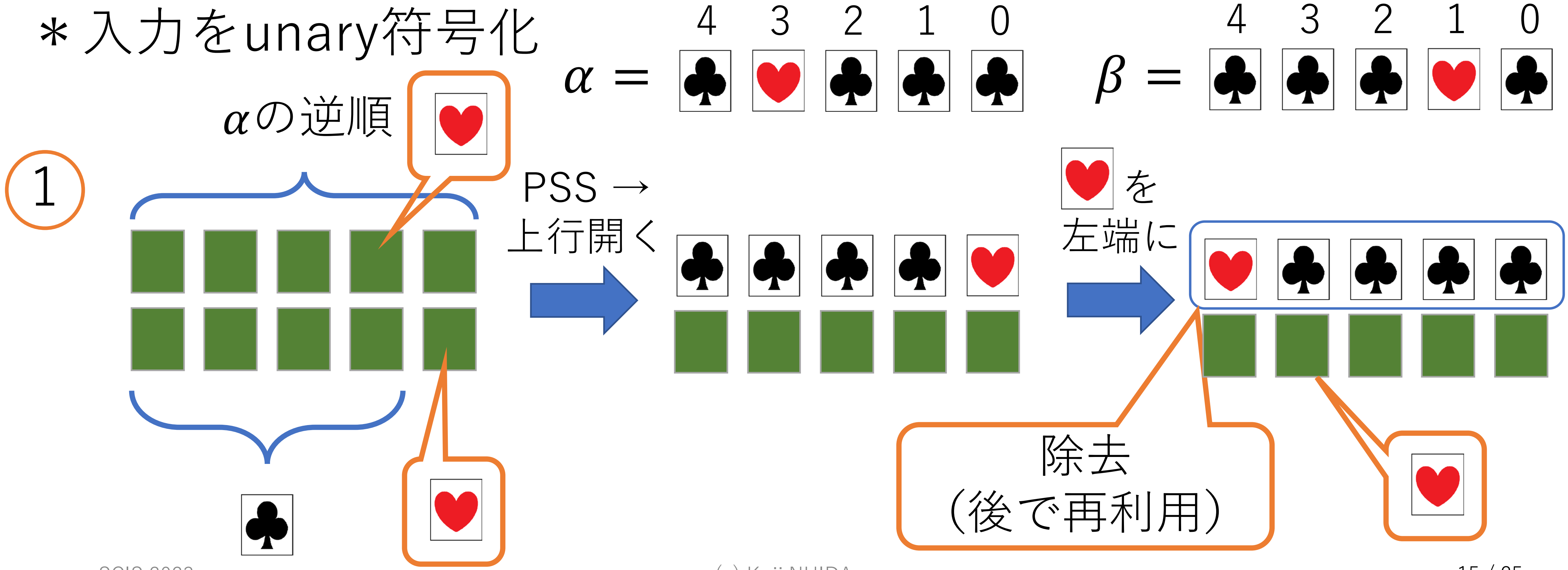
例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

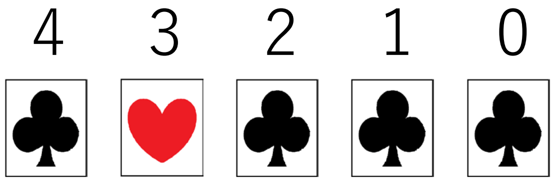
＊入力をunary符号化

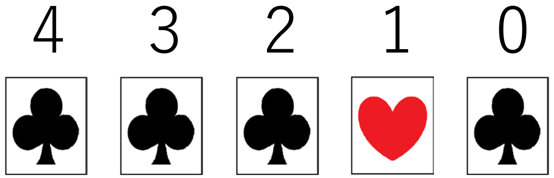


提案方式のサブプロトコル

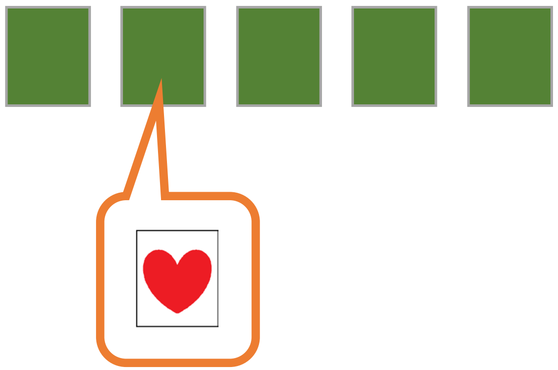
例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

$\alpha =$ 

$\beta =$ 

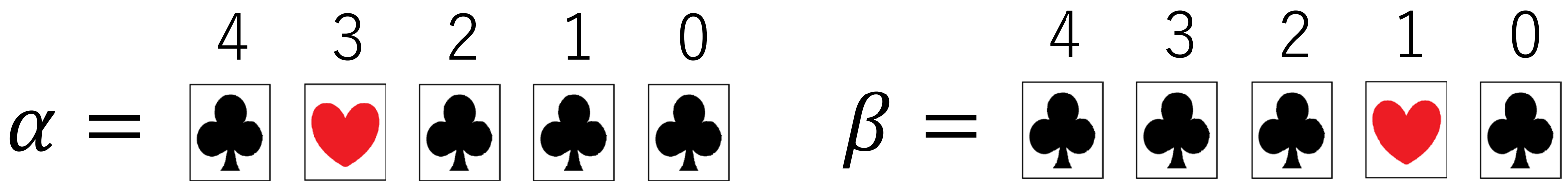
2



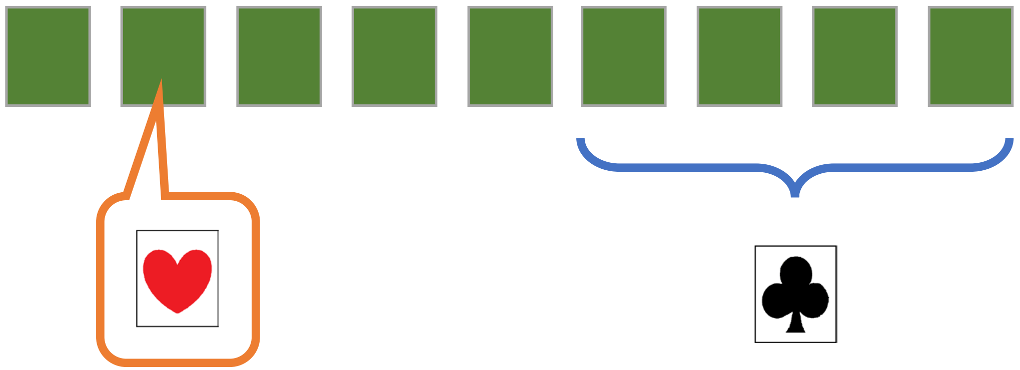
提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化



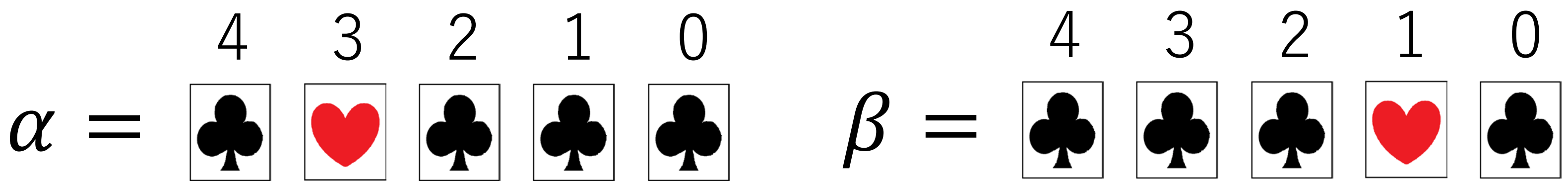
②

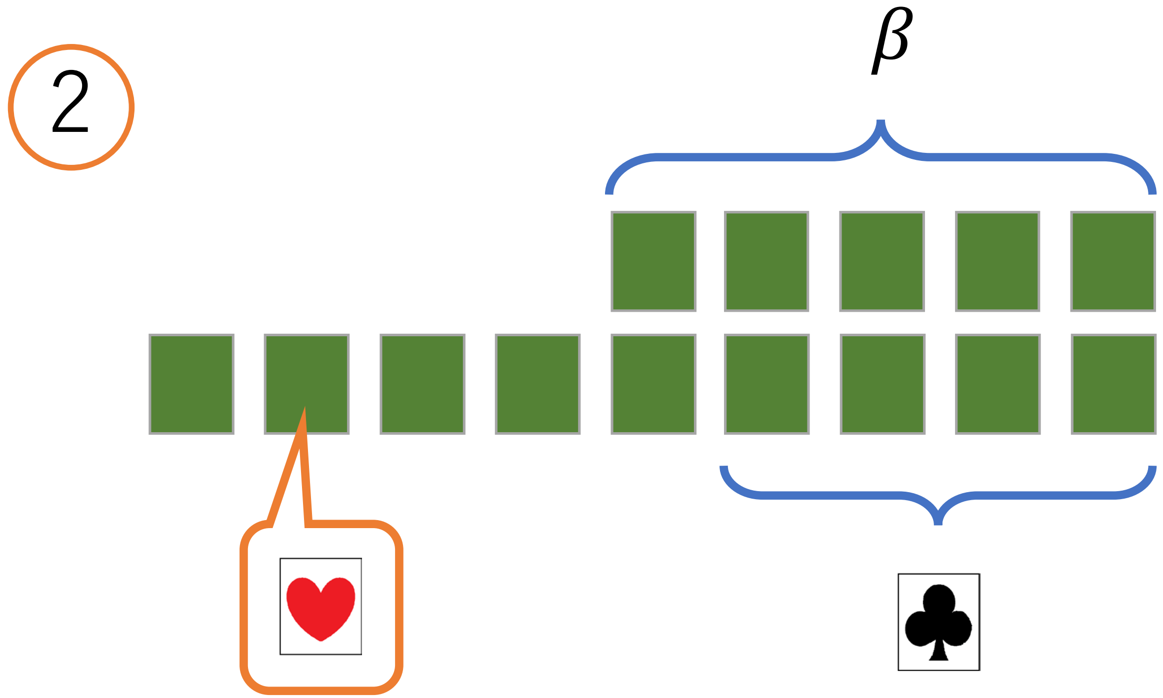


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

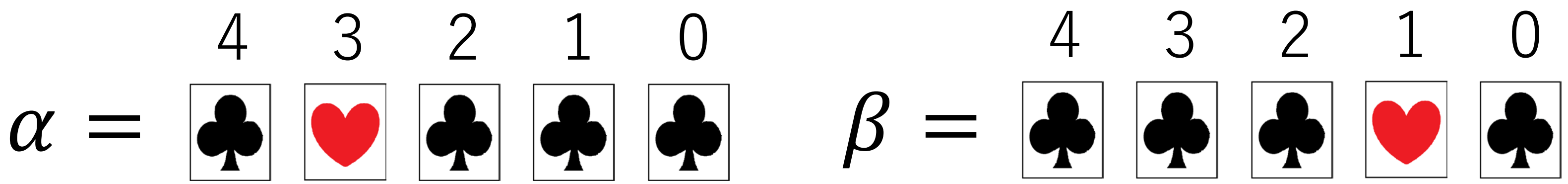


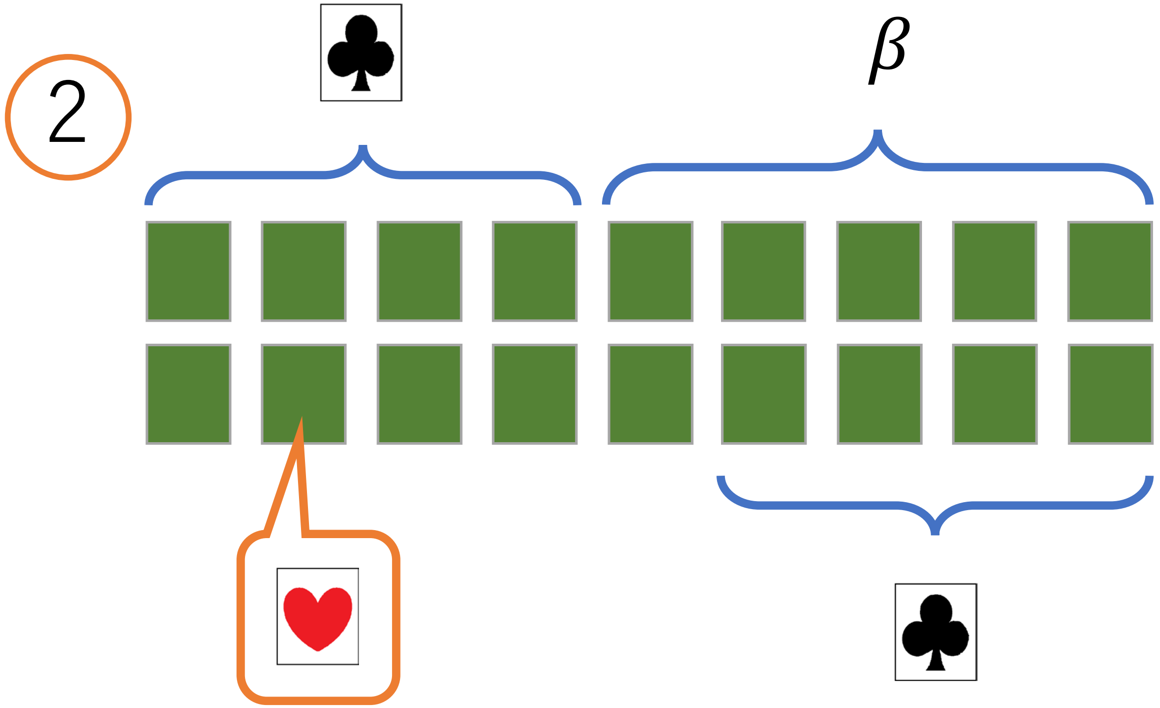


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

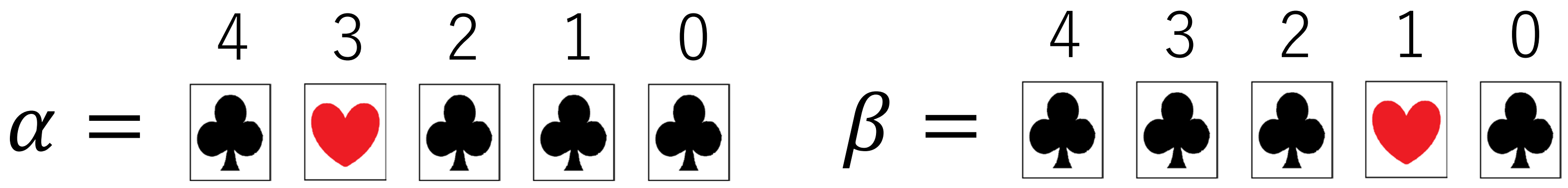


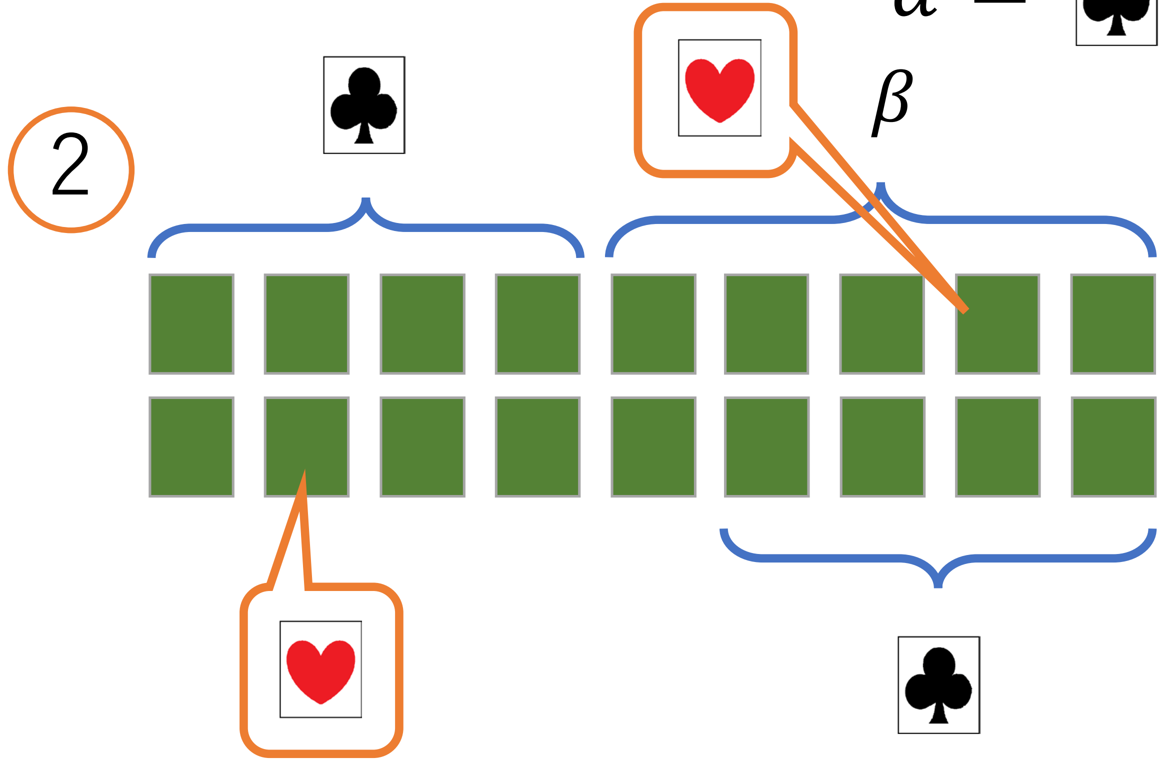


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

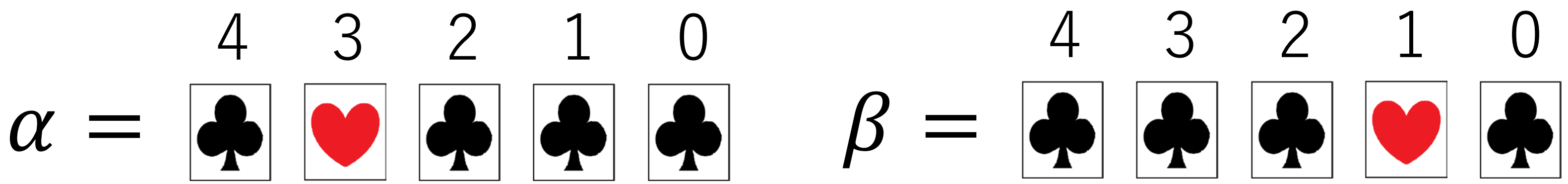


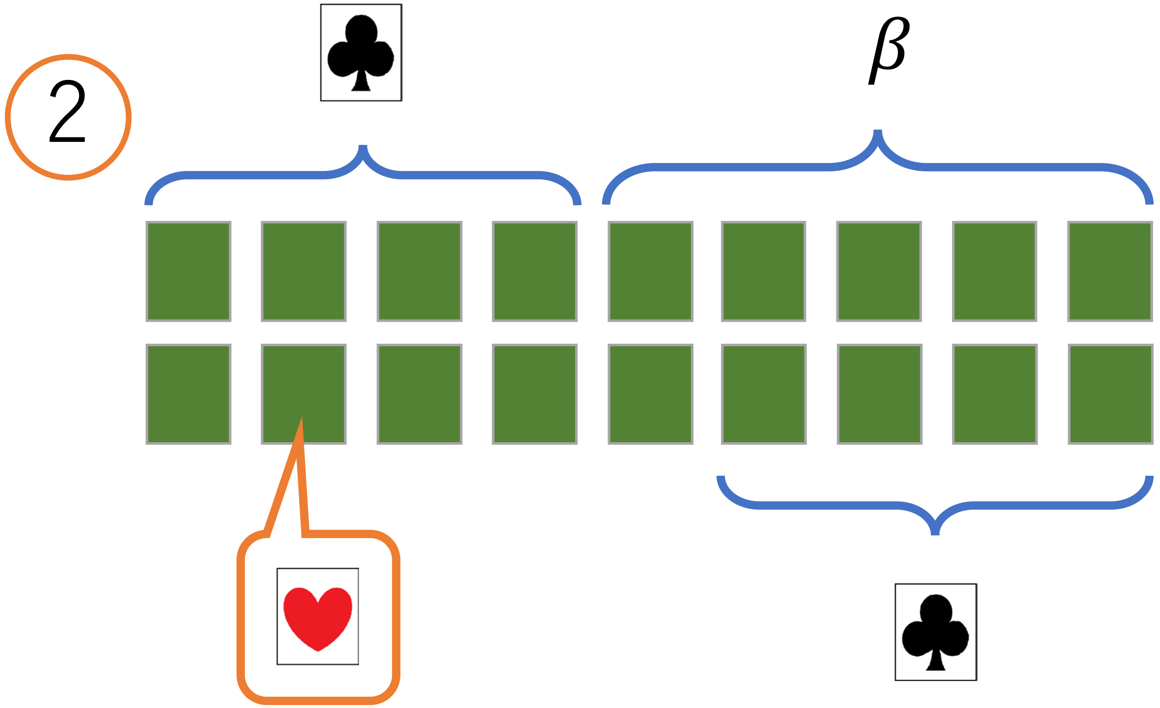


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

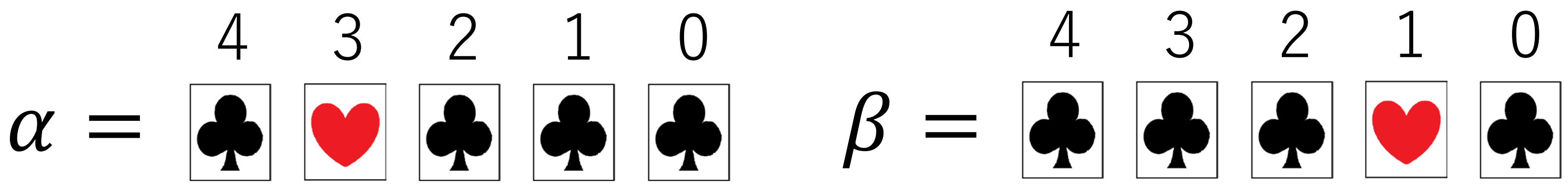


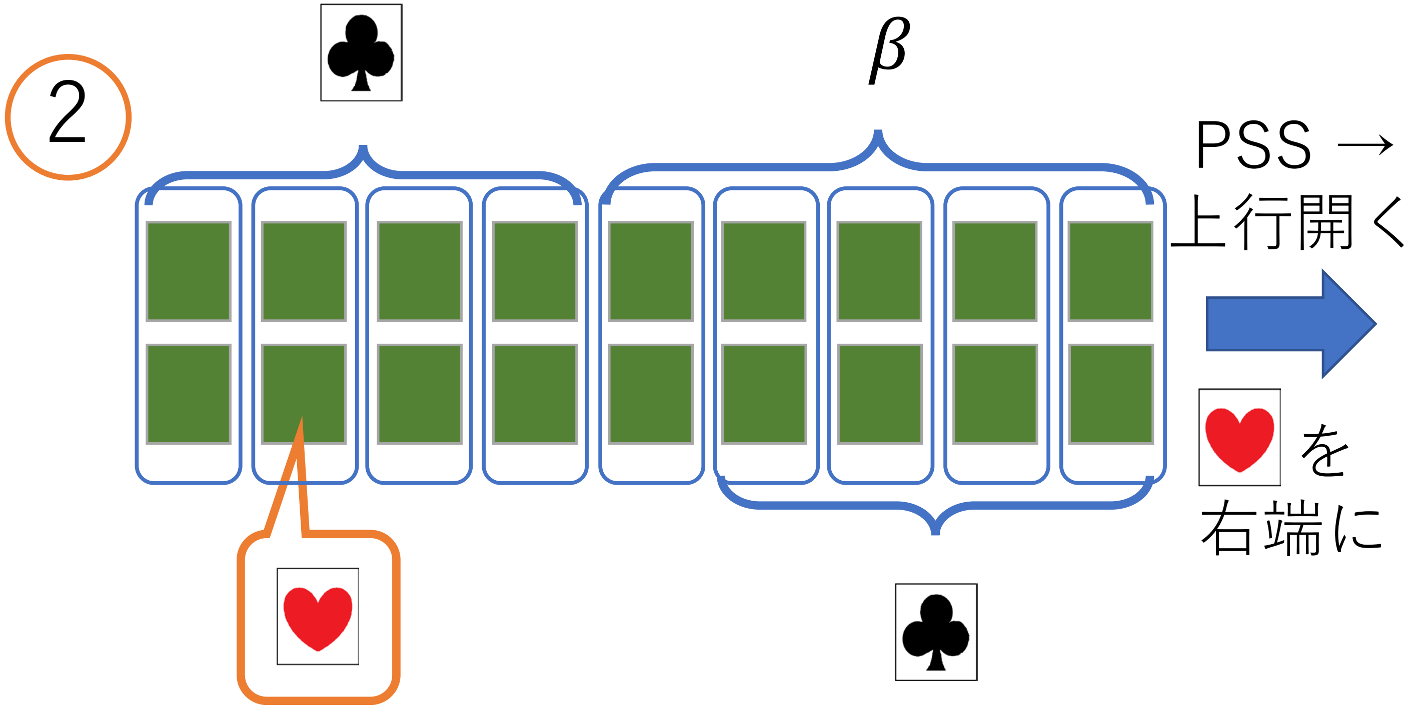


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

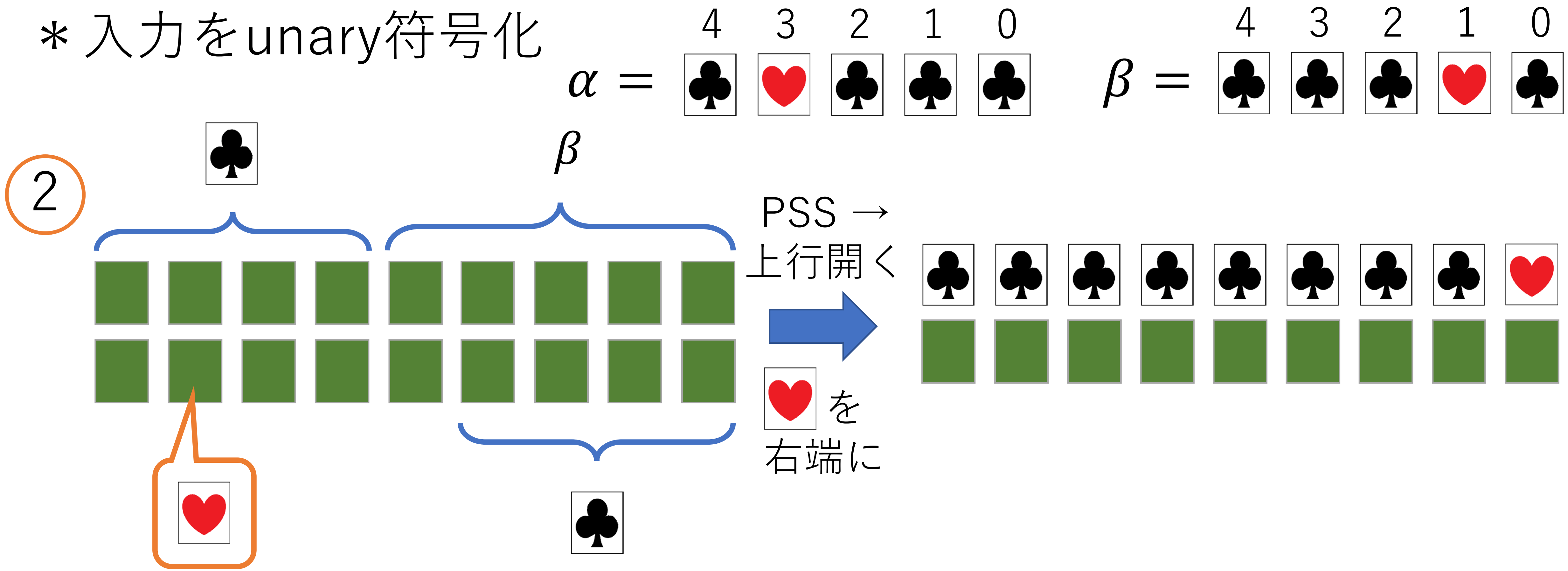




提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

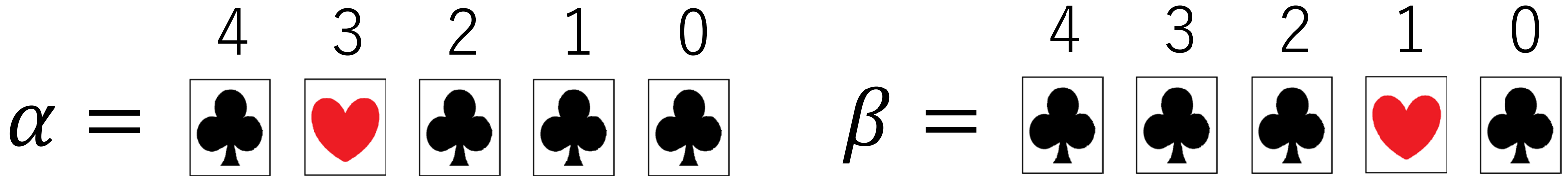
＊入力をunary符号化

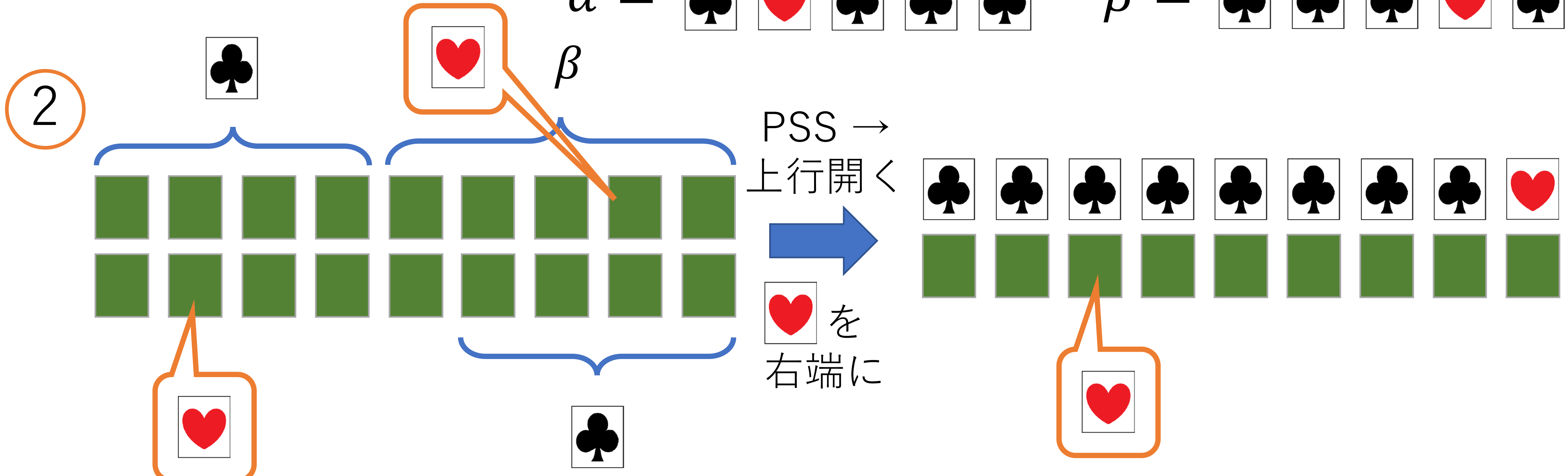


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

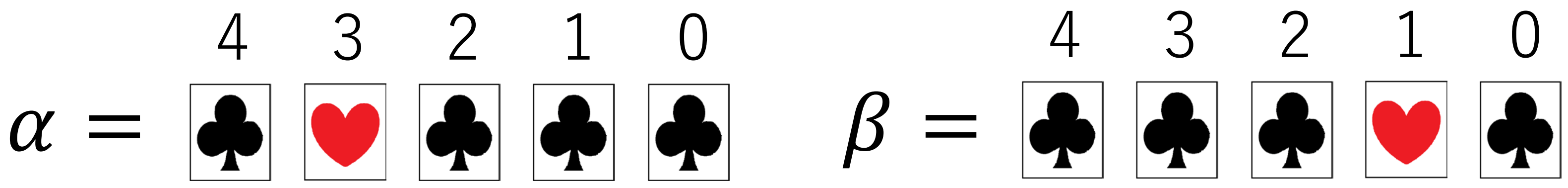


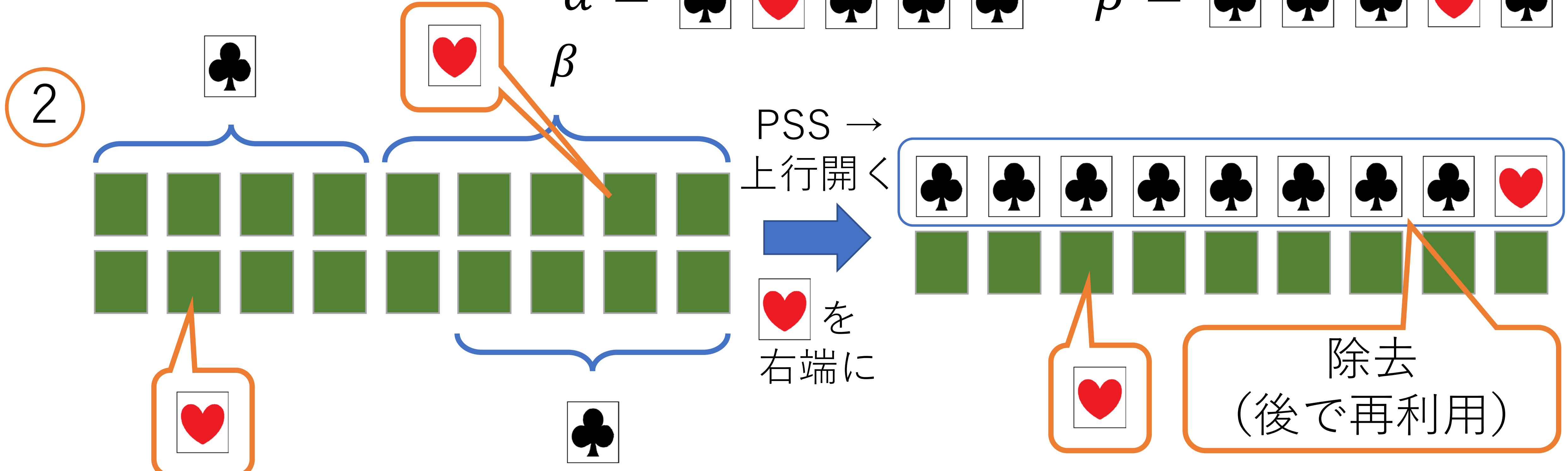


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化





提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

	4	3	2	1	0
$\alpha =$	♣	♥	♣	♣	♣

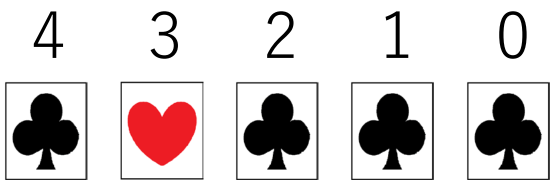
	4	3	2	1	0
$\beta =$	♣	♣	♣	♥	♣

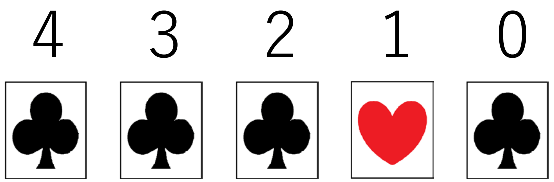
3

提案方式のサブプロトコル

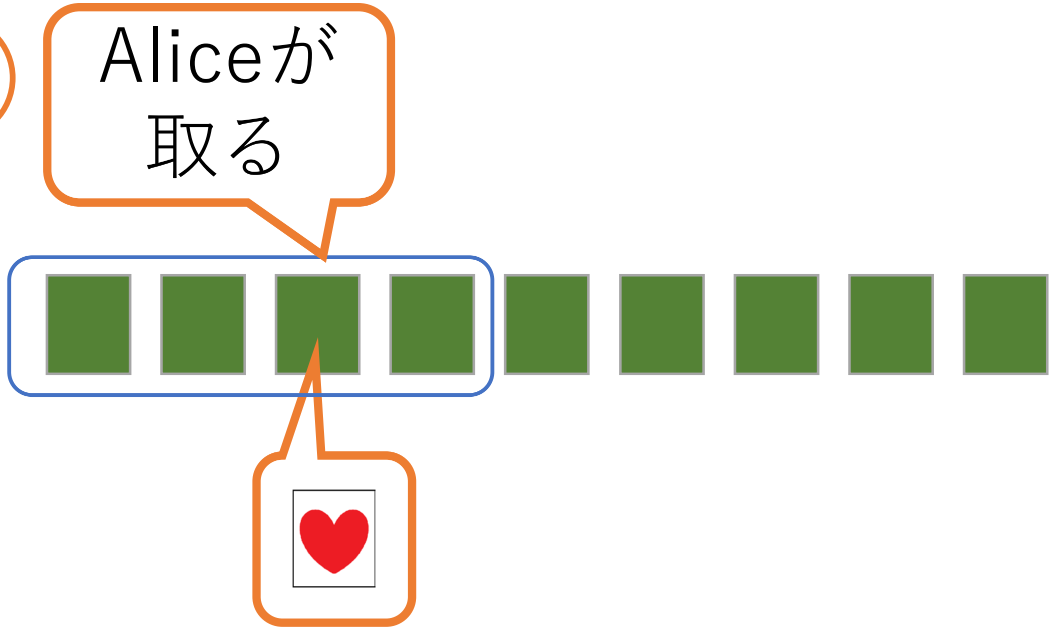
例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

$\alpha =$ 

$\beta =$ 

3

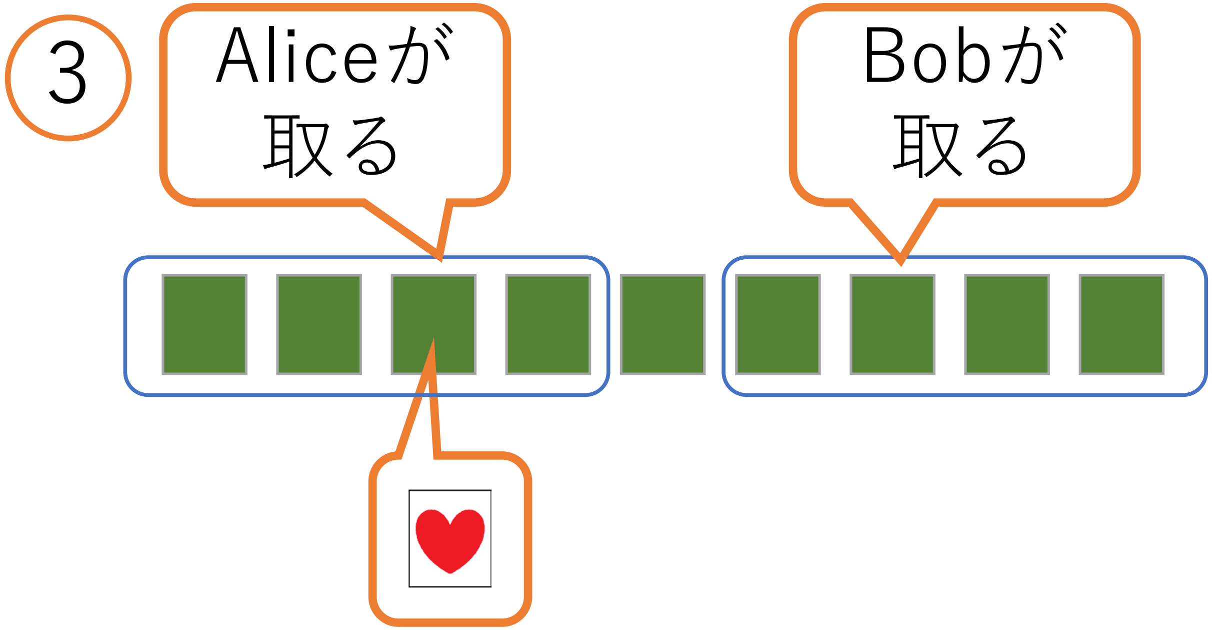


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

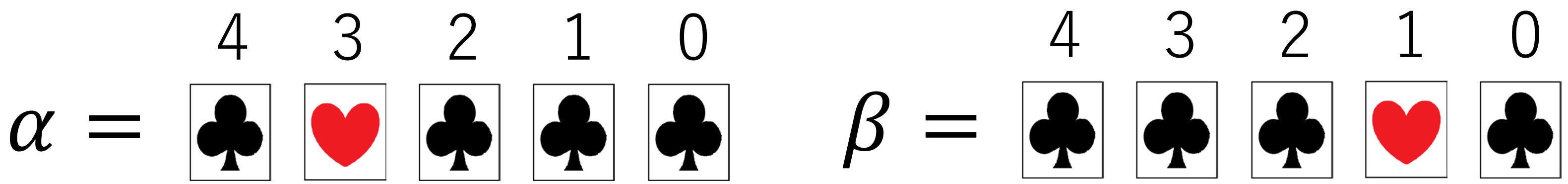
	4	3	2	1	0
$\alpha =$	♣	♥	♣	♣	♣
$\beta =$	♣	♣	♣	♥	♣

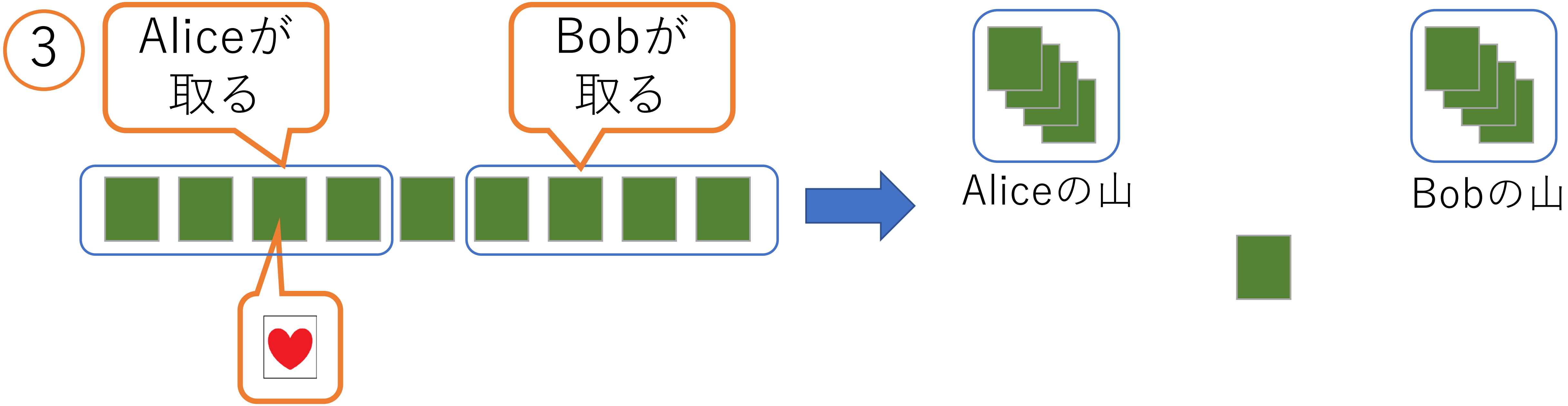


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

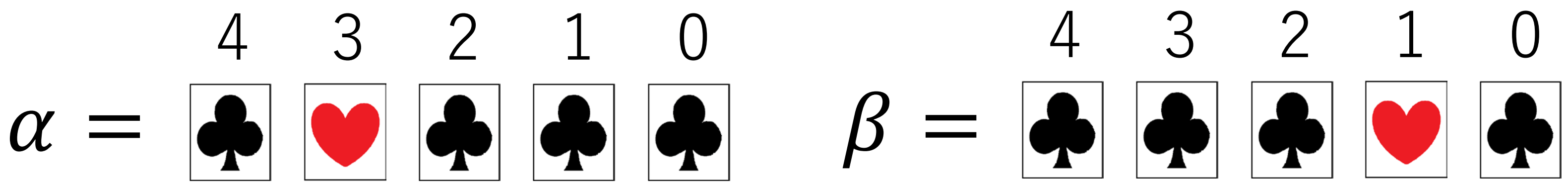


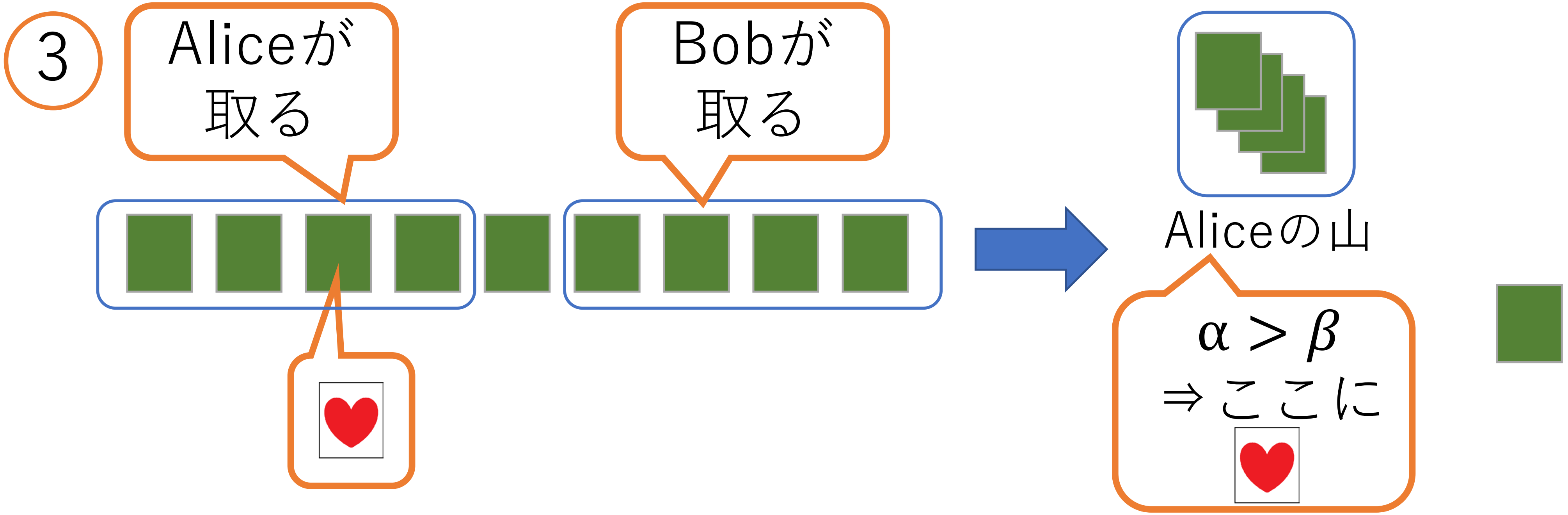


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

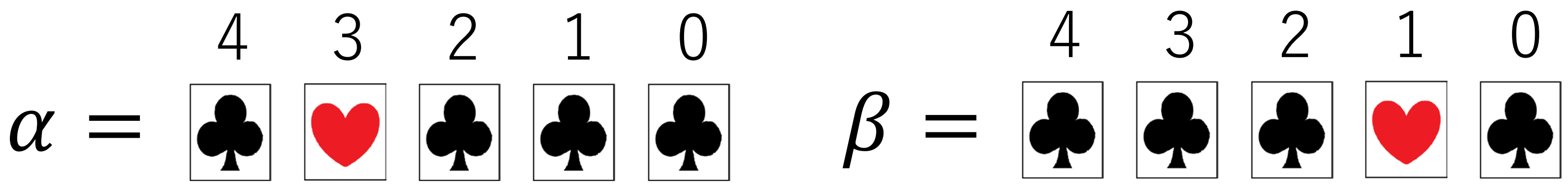


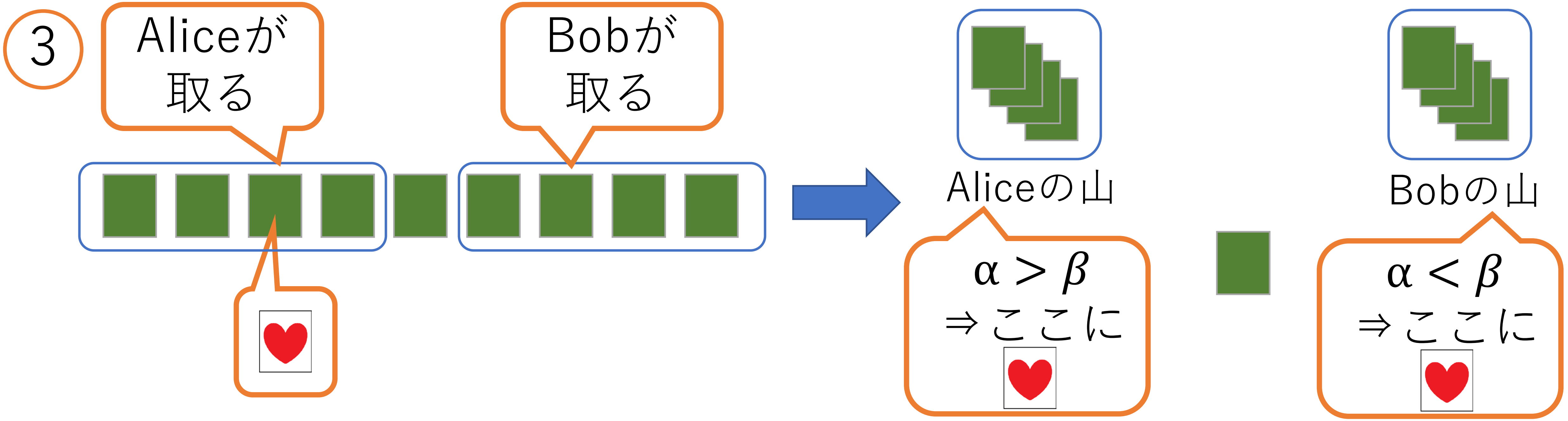


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

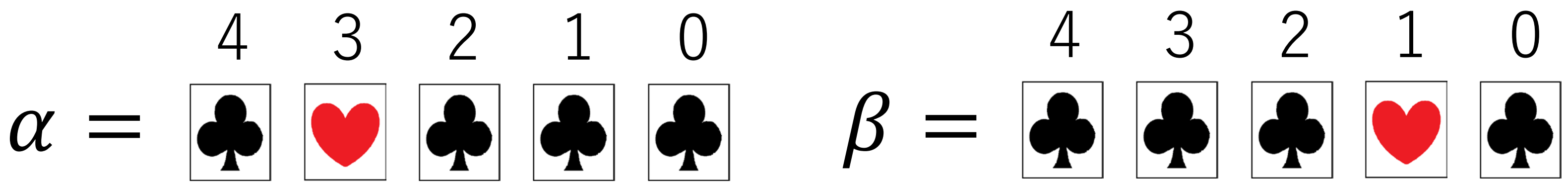


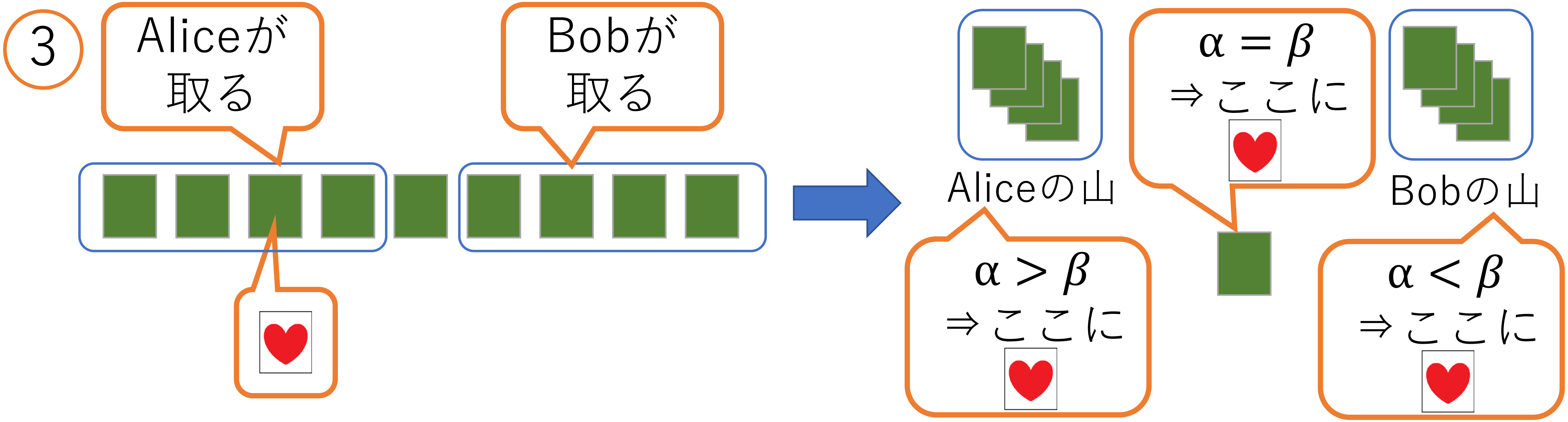


提案方式のサブプロトコル

例）入力範囲[0,4]、入力$\alpha = 3, \beta = 1$

＊入力をunary符号化

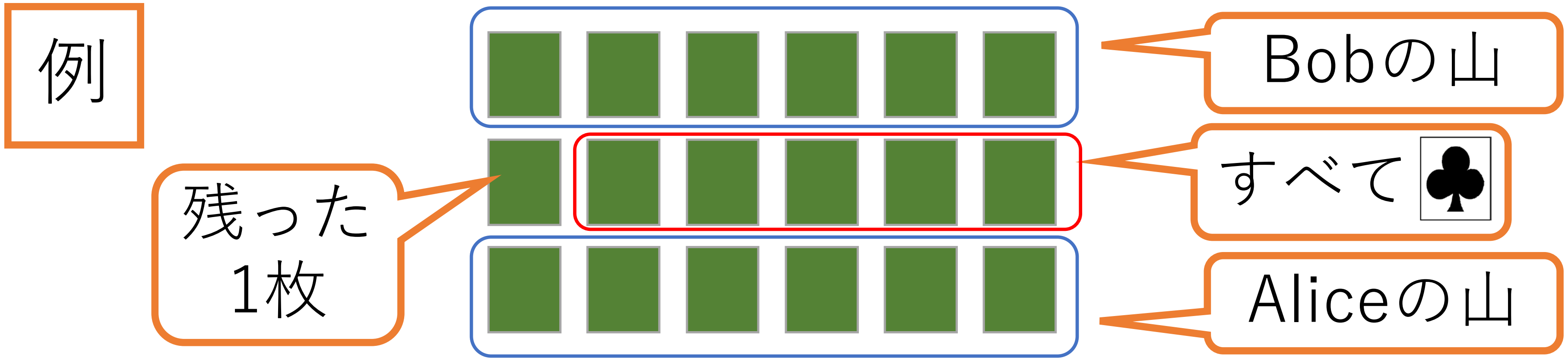




提案方式（前半）

- 各々の入力をq進数表示
- 上位桁から順にサブプロトコルを実行
  - 現在の桁の手順で残った1枚を次の桁で使う
  - 決着がついた後の桁は、最初の1枚が♣になり、AliceもBobも♣だけを取ることになる
- $\alpha > \beta$（, $\alpha < \beta$）であれば、Alice（, Bob）の山に♥
- $\alpha = \beta$であれば、最後に残る1枚が♥

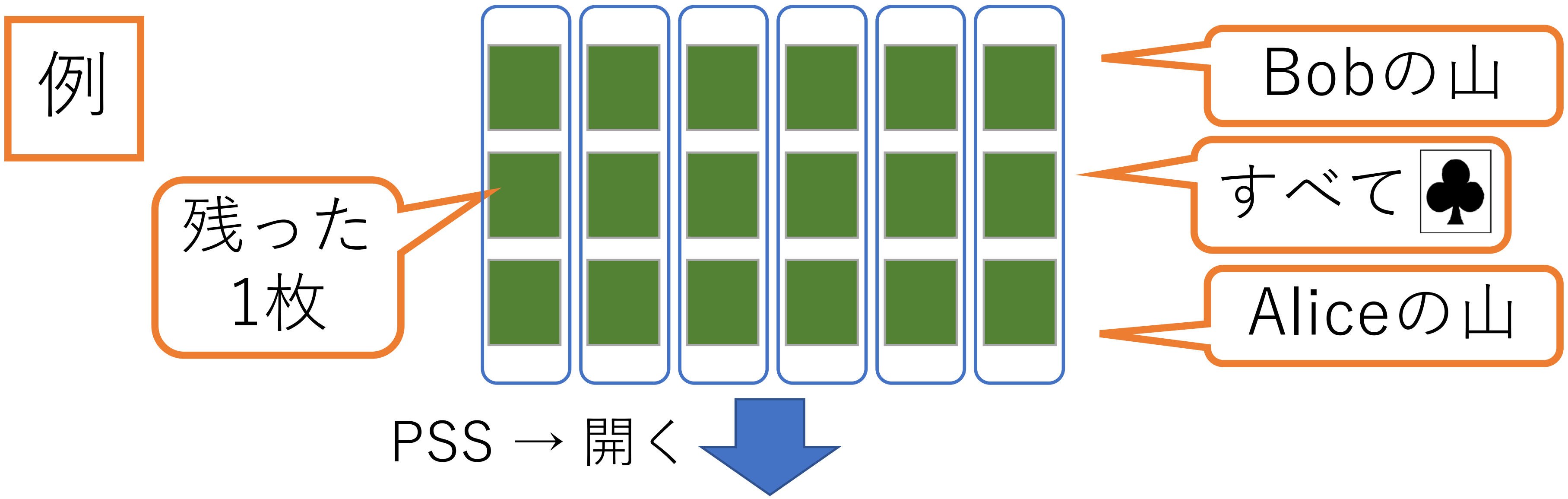
提案方式（後半）＊非コミット型の場合



提案方式（後半）＊非コミット型の場合

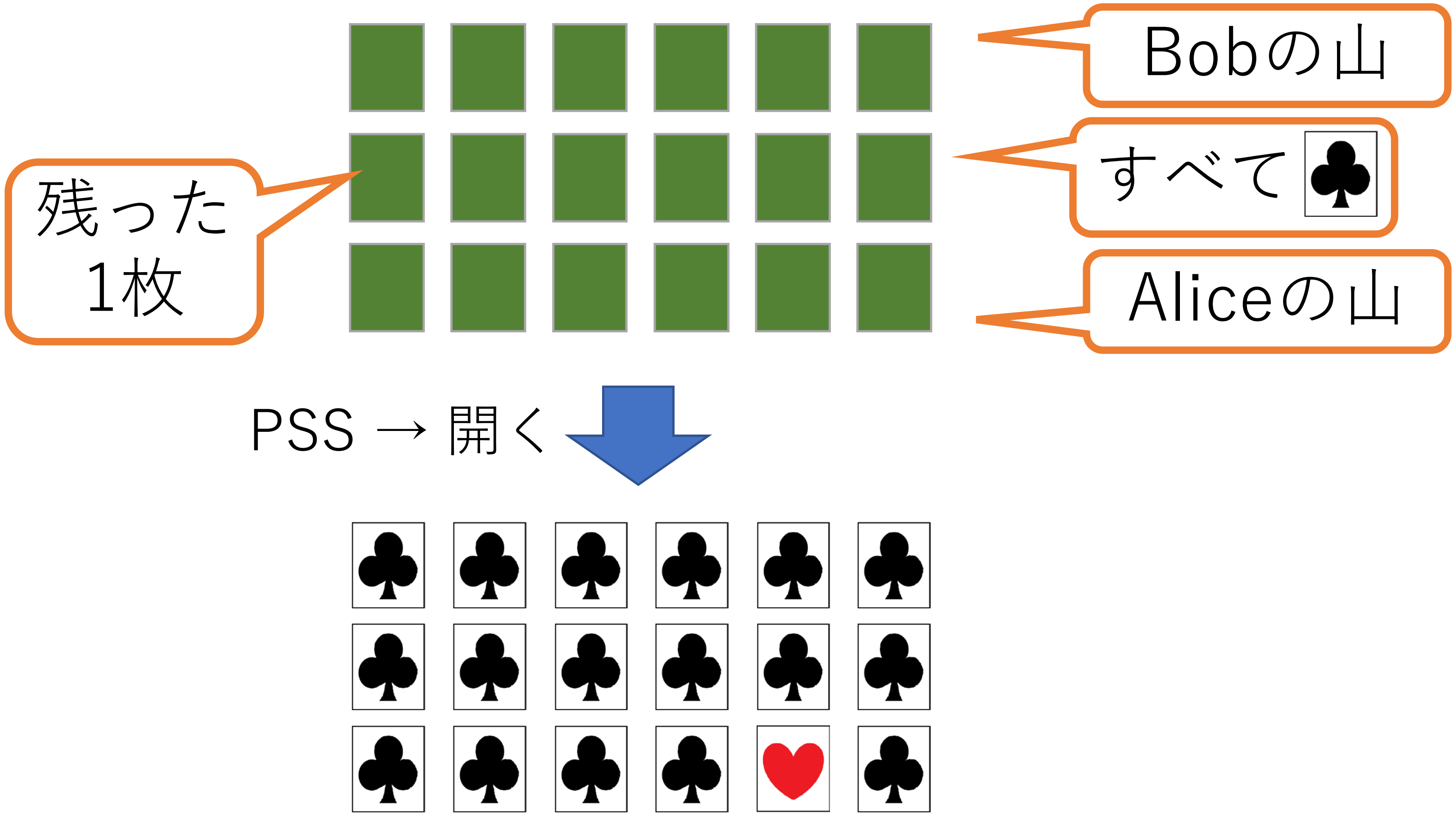
例

提案方式（後半）＊非コミット型の場合



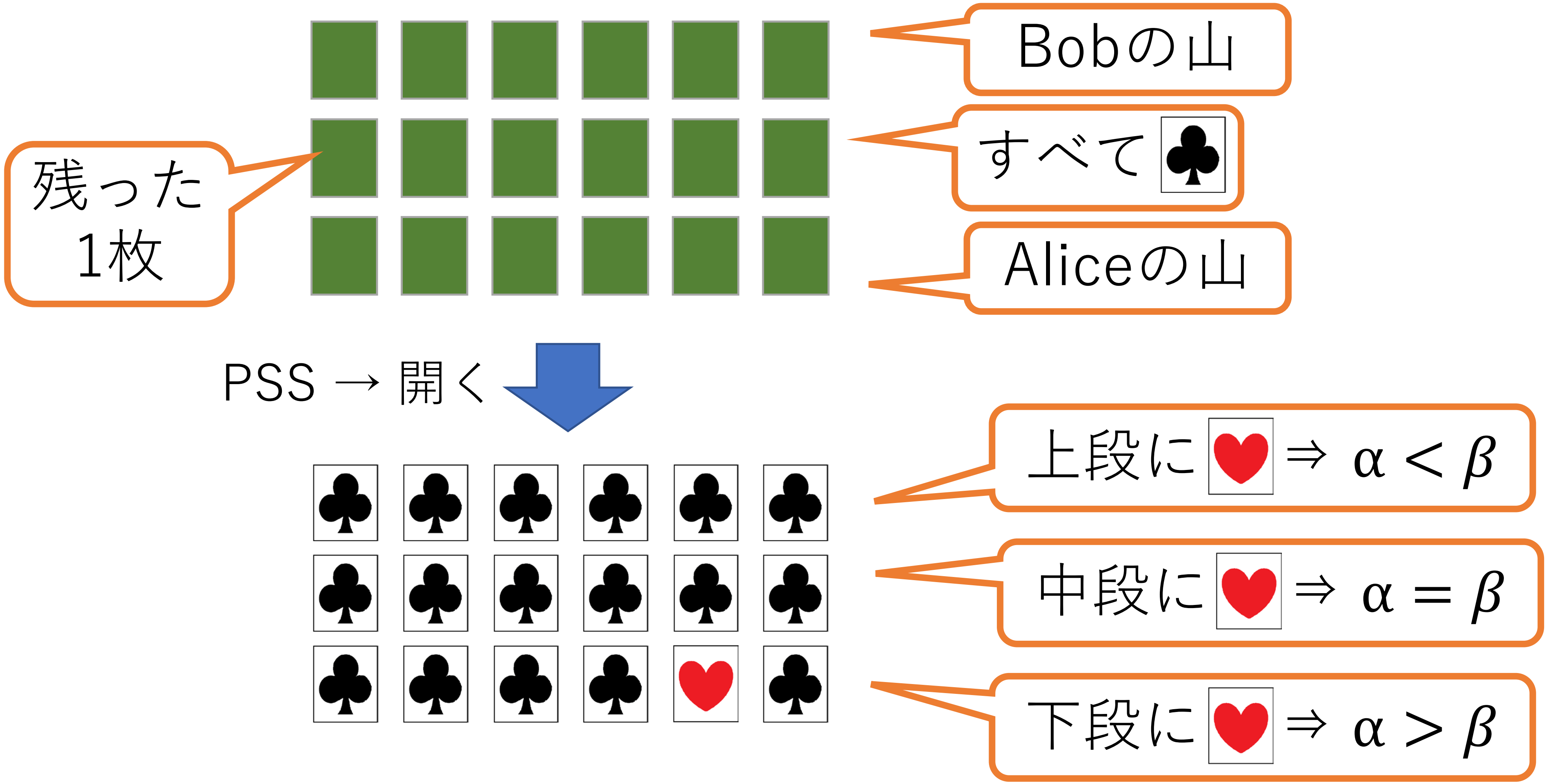
提案方式（後半）＊非コミット型の場合

例



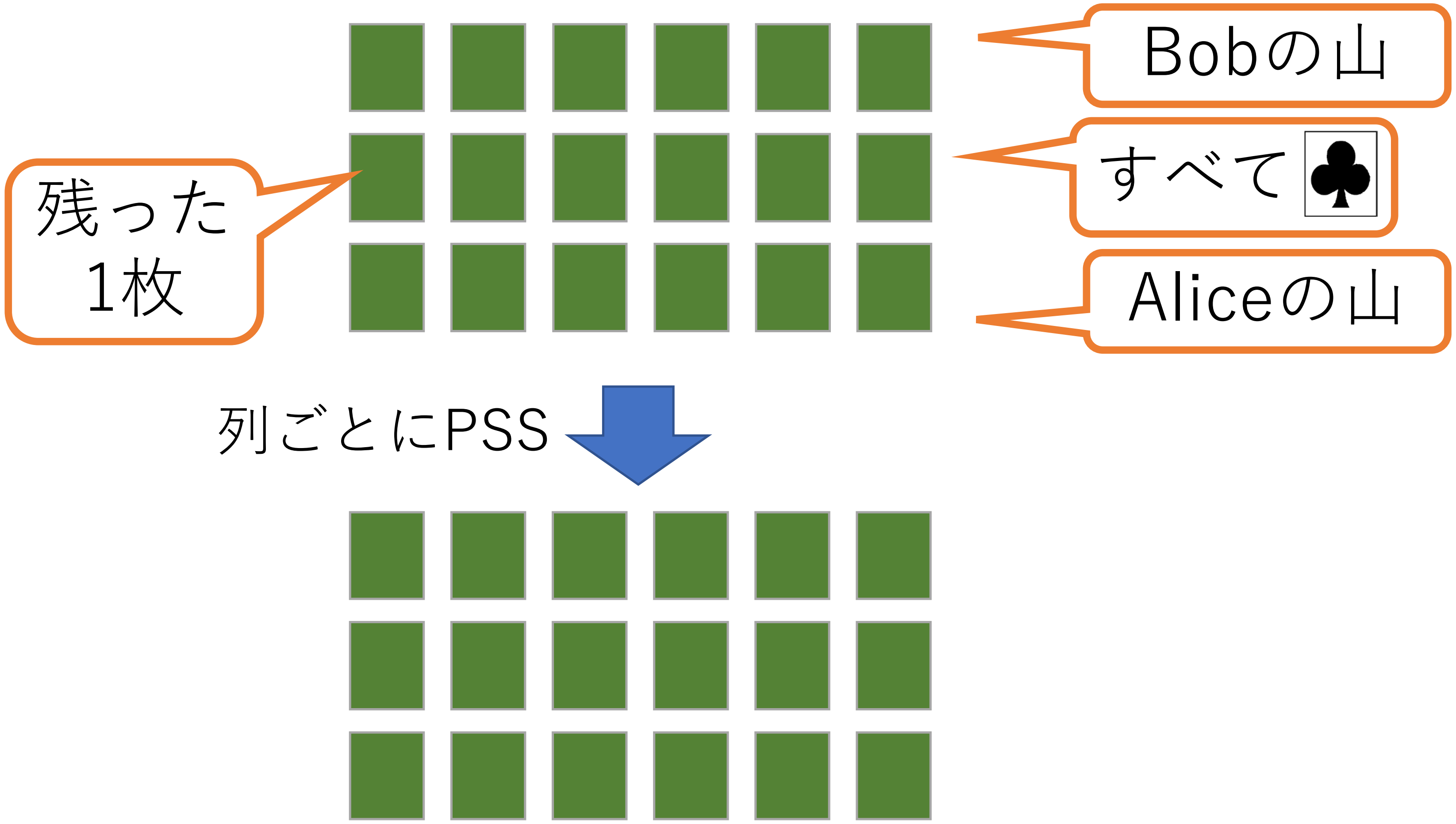
提案方式（後半）＊非コミット型の場合

例

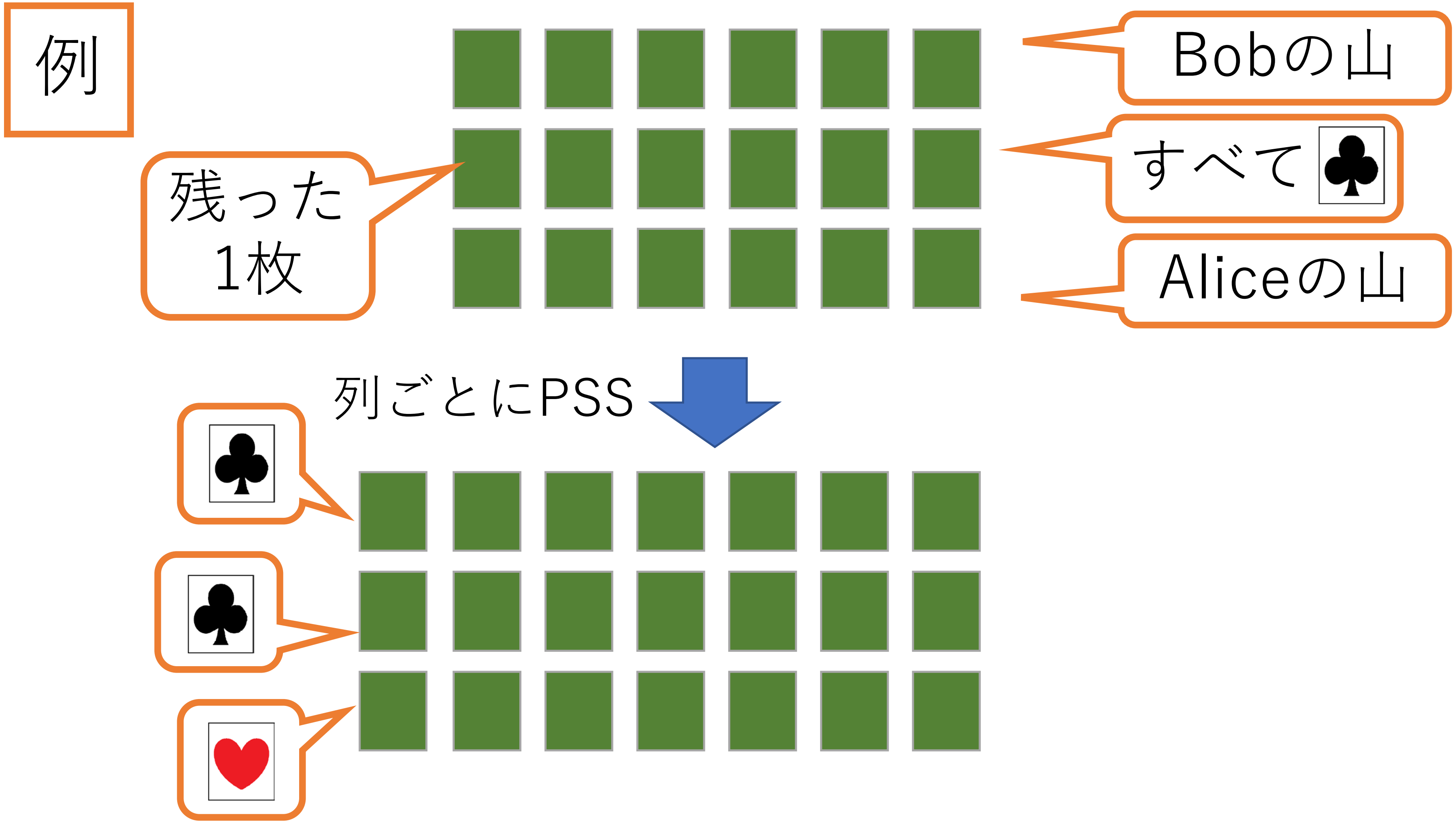


提案方式（後半）＊コミット型の場合

例

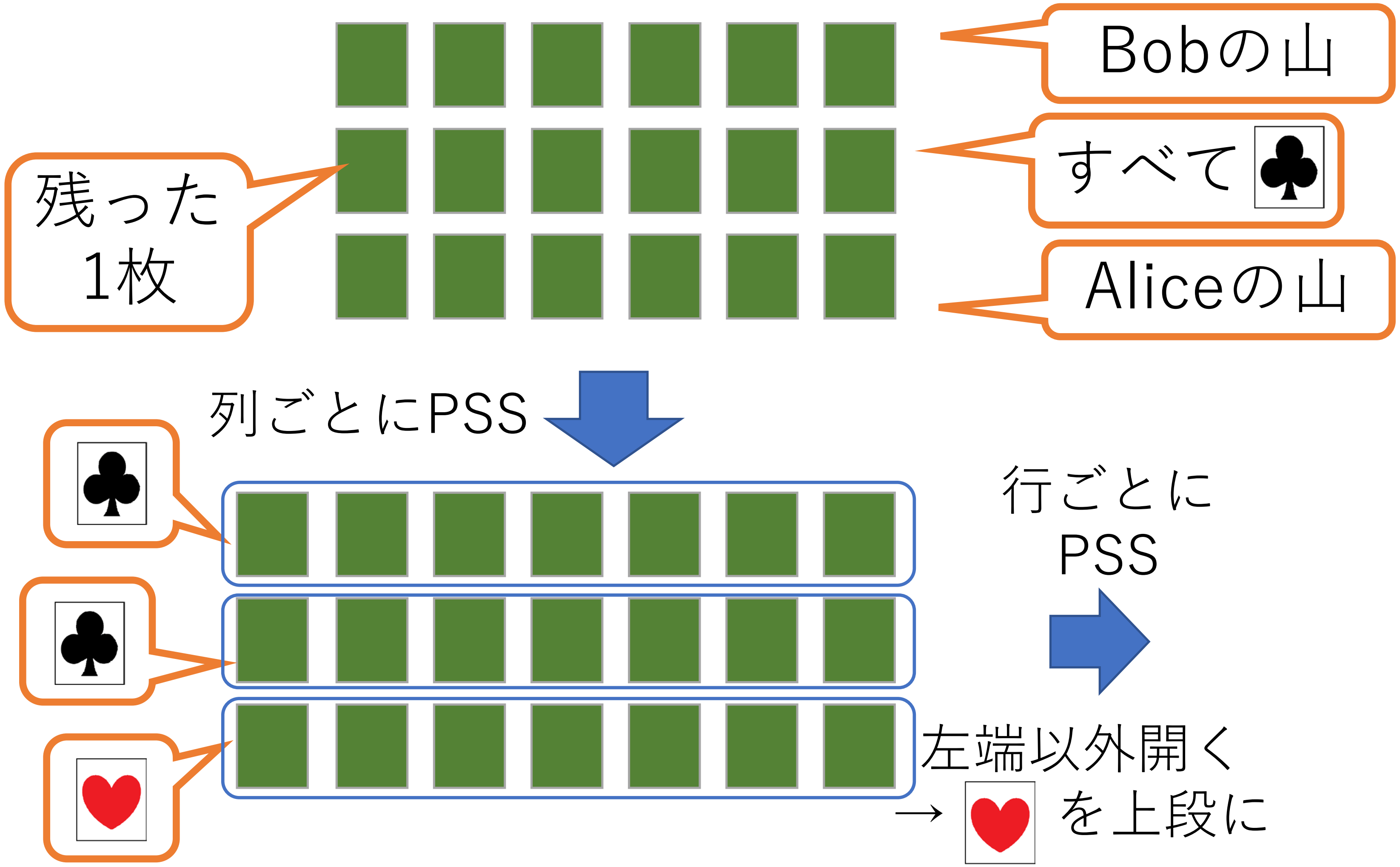


提案方式（後半）＊コミット型の場合

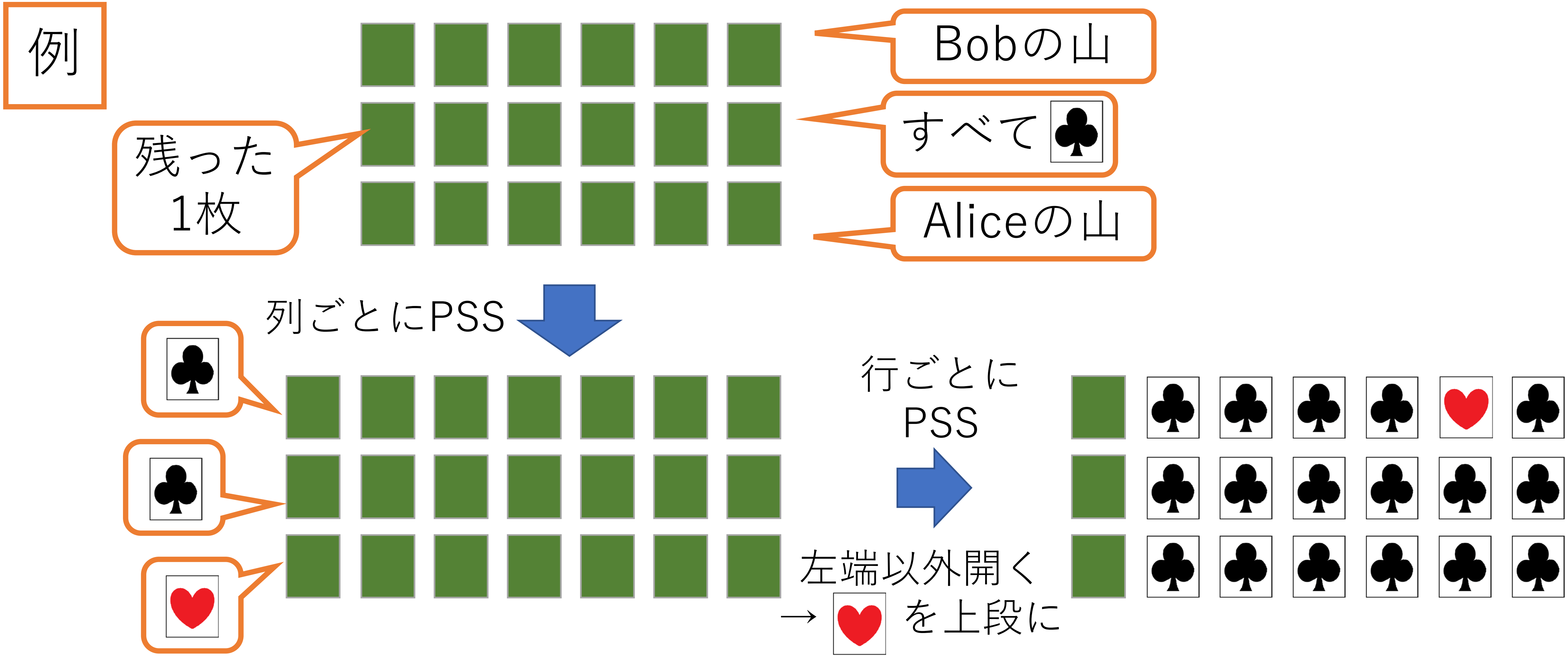


提案方式（後半）＊コミット型の場合

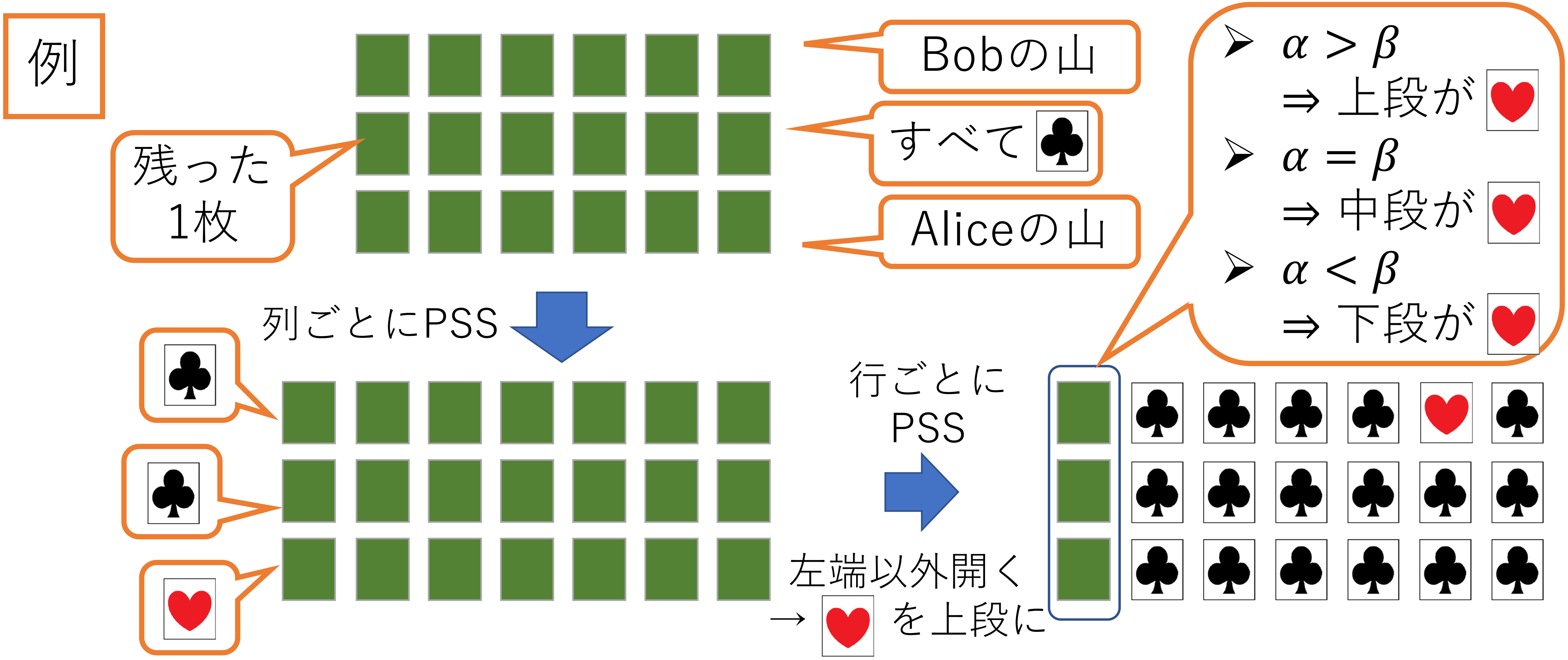
例



提案方式（後半）＊コミット型の場合



提案方式（後半）＊コミット型の場合



既存方式との比較（2値カードの場合）

方式 （入力$[0, m-1]$）	[Miyahara+ '20]	[Shinagawa-Nuida '21] (*)	**本研究 (**)** **（コミット型）**
カード枚数	$4\lceil\log_2 m\rceil + 2$	$98\lceil\log_2 m\rceil - 72$	$\mathbf{6\lceil\log_3 m\rceil + 5}$
漸近的係数 （$\times \log_2 m$）	4	98	$\mathbf{3.7855\ldots}$
シャッフル	$2\lceil\log_2 m\rceil - 1$ RC	1（複雑）	$\mathbf{2\lceil\log_3 m\rceil + 2}$ **PSS**
漸近的係数 （$\times \log_2 m$）	2	-	$\mathbf{1.2618\ldots}$

(*) [Miyahara+ '20]と同じ大小比較回路をgarbled circuit化した場合

(**) $q = 3$とし、出力を$\alpha \geq \beta$と$\alpha < \beta$の二択にしてカード枚数を減らしたもの

[K. Shinagawa, K. Nuida, "A Single Shuffle Is Enough for Secure Card-Based Computation of Any Boolean Circuit", Discrete Applied Mathematics (2021)]

目次

- 背景：カードベース暗号
- 大小比較カードプロトコルの既存方式
- 提案方式（2値カード）
- **提案方式（番号カード）**

提案方式の番号カードへの拡張

- 基本的な構成は2値カードの場合と同様
- 変更点：
  - 番号カードを ♥ 用、1行目 ♣ 用、2行目 ♣ 用に分類
    - 入力の符号化前に後者2種各々を完全シャッフル
  - 「どのループで取られたカードか」を隠すため、
  カードの再利用が（ほぼ）できない
  → カード枚数の増加

既存方式との比較（番号カードの場合）

方式（入力$[0, m-1]$）	[Miyahara+ '20]	**本研究 (*)（コミット型）**
カード枚数	$4\lceil \log_2 m \rceil + 4$	$\mathbf{7\lceil \log_2 m \rceil + 2}$
漸近的係数（$\times \log_2 m$）	4	$\mathbf{7}$
（2値カードの場合）	(4)	$(3.7855\ldots)$
シャッフル	$6\lceil \log_2 m \rceil - 2$ PSS	$\mathbf{2\lceil \log_2 m \rceil + 2}$ **PSS, 2 CS**
漸近的係数（$\times \log_2 m$）	6	$\mathbf{2}$
（2値カードの場合）	(2)	$(1.2618\ldots)$

(*) $q = 2$とし、出力を$\alpha \geq \beta$と$\alpha < \beta$の二択にしてカード枚数を減らしたもの

注）提案方式について、q を大きくすると
シャッフル回数が減るがカード枚数が増える

まとめと補足

（2者間）**大小比較カードプロトコル**の提案

- 既存方式 [Miyahara+ '20]との漸近的な比較：
  - 2値カード：枚数**約94.6%**、シャッフル回数**約63.1%**
  - 番号カード：枚数175%、シャッフル回数**約33.3%**

補足：番号カードでの提案方式

- 入力のコミットメントが予め与えられた場合にも、PSS各1回で「ランダムなカードを用いた符号化」に変換可能なため、提案方式を適用可能