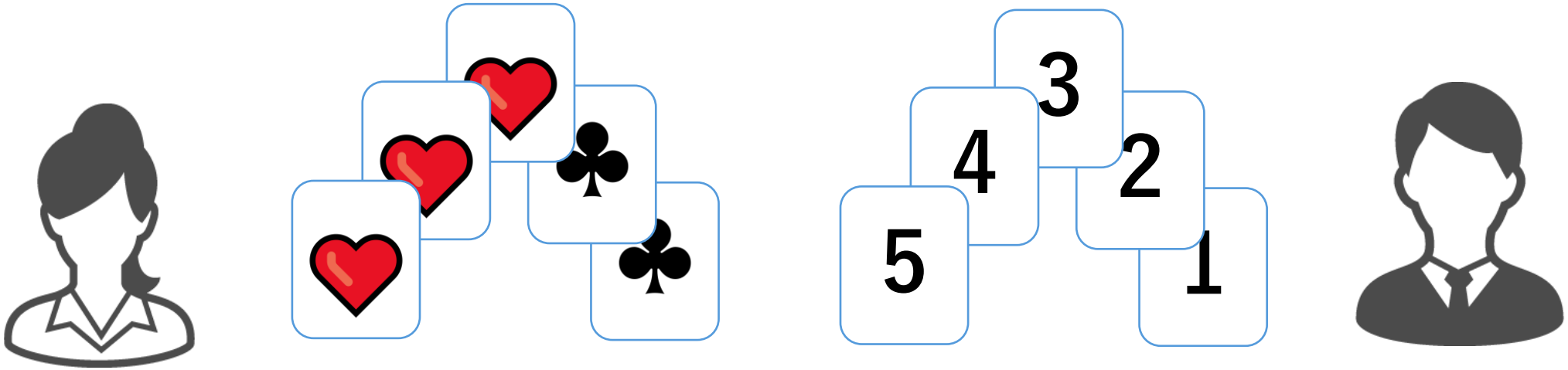
ゲートあたり6枚で実行できる カードベースガーブルド回路

電気通信大学
小野 知樹

カードベース暗号[Boer89]

- 計算機ではなく，物理的なカードを用いた計算
  - 秘密計算，ゼロ知識証明
- 全ての操作は公開で行う



[Boer89] B. den Boer, “More efficient match-making and satisfiability: The Five Card Trick,” EUROCRYPT, 1989

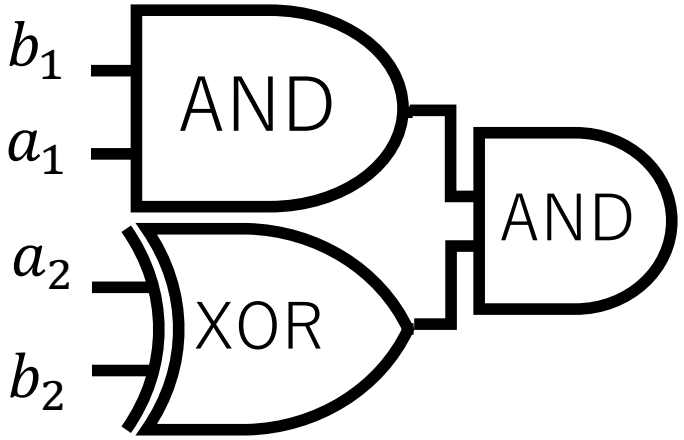
秘密計算

- データを暗号化したまま計算できる技術
  - 例：2者のうちどちらがお金持ちであるかを比較したいが
    自身の資産は知られたくない
- 今回の目標は任意の論理回路（関数）に対する秘密計算

任意の論理回路に対する秘密計算

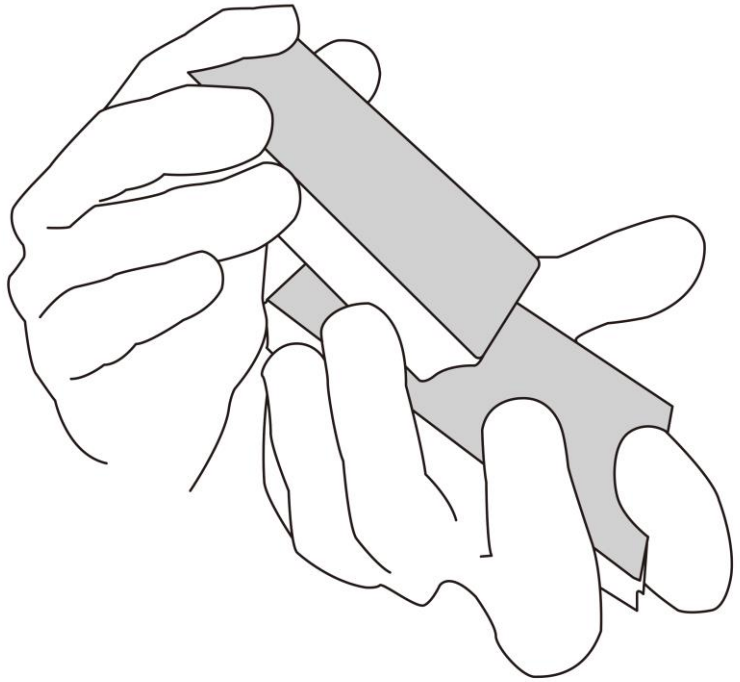
- 任意の論理回路：任意の計算機上で計算できる関数（公開）
- 互いの入力と計算途中の値（中間値）を秘匿しつつ出力を得る
  - AND，XOR，NOTを計算することができればよい
    - ✓ 任意の論理回路はAND，XOR，NOTの組み合わせ
  - 出力は秘匿しつつ次の入力として用いることができる必要がある

例：$(b_1 \wedge a_1) \wedge (a_2 \oplus b_2)$を計算する論理回路



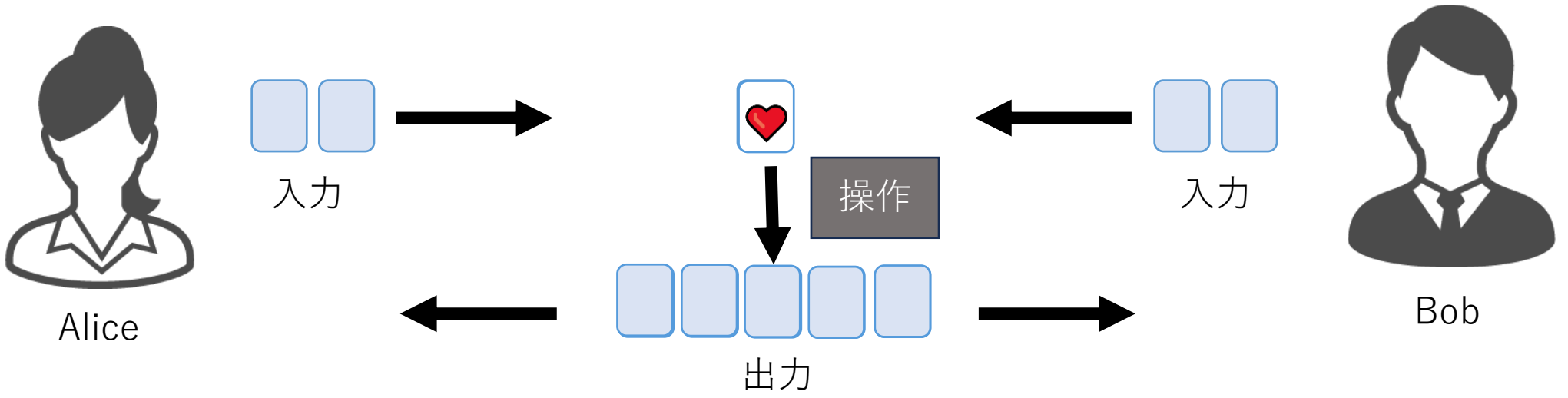
シャッフル操作

- 実行したプレイヤを含めプレイヤ全員が乱数を知らない並び替え
  - 公開の場で行うため不正はできない
- ランダマイズのために必要
- 実行に課題



今回のプロトコルで用いるカード

- 2種カード(表：♥ ♣　裏 □　)
- 同じ模様のカードは見分けられず，裏面同士も見分けられない
  - 裏面からは表面が一切わからない
  - 裏返してシャッフルすると全てのプレイヤに
    カードの表面のマークがわからなくなる



カードベース暗号の効率指標と目標

- カードベース暗号の効率指標は以下の二つ
  - シャッフル回数：少ないほど良い
  - カード枚数：少ないほど良い
- 目標：シャッフル回数を最小化（１回に）しつつ、
  カード枚数が少ないプロトコルの構成
  - 実行を容易にするため
  - シャッフル回数は０回にはできない [SN21]
    - ✓ 入力が漏れないようにするため

[SN21] K. Shinagawa, and K. Nuida. “A single shuffle is enough for secure card-based computation of any Boolean circuit.,” Discrete Applied Mathematics, 2021.

ガーブルド回路アプローチ

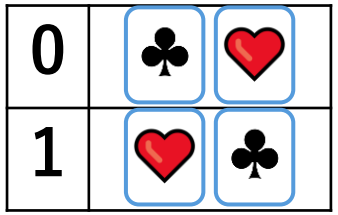
任意の論理回路に対するカードベース暗号プロトコル

	AND, XOR, NOTプロトコルを 組み合わせるアプローチ（主流）	ガーブルド回路アプローチ [SN21][TMM23], [KS23], [OSN+23]
構成方法	プロトコルの出力を 次のプロトコルの入力とする	ガーブルド回路を再現
シャッフル回数	多い (ゲート数に応じて増える)	少ない (1回)
カード枚数	少ない	多い

[SN21] K. Shinagawa, and K. Nuida. “A single shuffle is enough for secure card-based computation of any Boolean circuit.,” Discrete Applied Mathematics, 2021.
[TMM23] K. Tozawa et al. “Single-shuffle card-based protocol with eight cards per gate.,” UCNC, 2023.
[KS23] Y. Manabe, K. Shinagawa: Free-XOR in Card-Based Garbled Circuits. CANS 2023: 232-248
[OSN+23] T. Ono et al. “Single-Shuffle Card-Based Protocols with Six Cards per Gate.,” ICISC, 2023.

コミット型，非コミット型

- コミット型
  - 広義には入力と出力の符号化ルールが同じもの
  - 狭義には入力と出力の符号化ルールが次のようなもの



  - プロトコルを組み合わせやすい

- 非コミット型
  - 入出力が上のような符号化ルールではないもの
  - 効率が良くなりやすい

シャッフル1回のプロトコル

- 任意の論理回路に対するプロトコルのカード枚数を削減

  q は計算対象の論理回路のゲート数, n, mは入出力ゲート数

プロトコル	カード枚数	コミット型	シャッフルの種類
Shinagawa-Nuida [SN21]	$24q + 2n$	✔	一様閉
Tozawaら [TMM23]	$8q + 2n$	✔	一様閉
Onoら[OSN+23]	$6q + 2n$		一様
Onoら[OSN+23]	$6q + 2n + 2m$	✔	一様

他にManabe-Shinagwa[KS23]らのプロトコルが存在

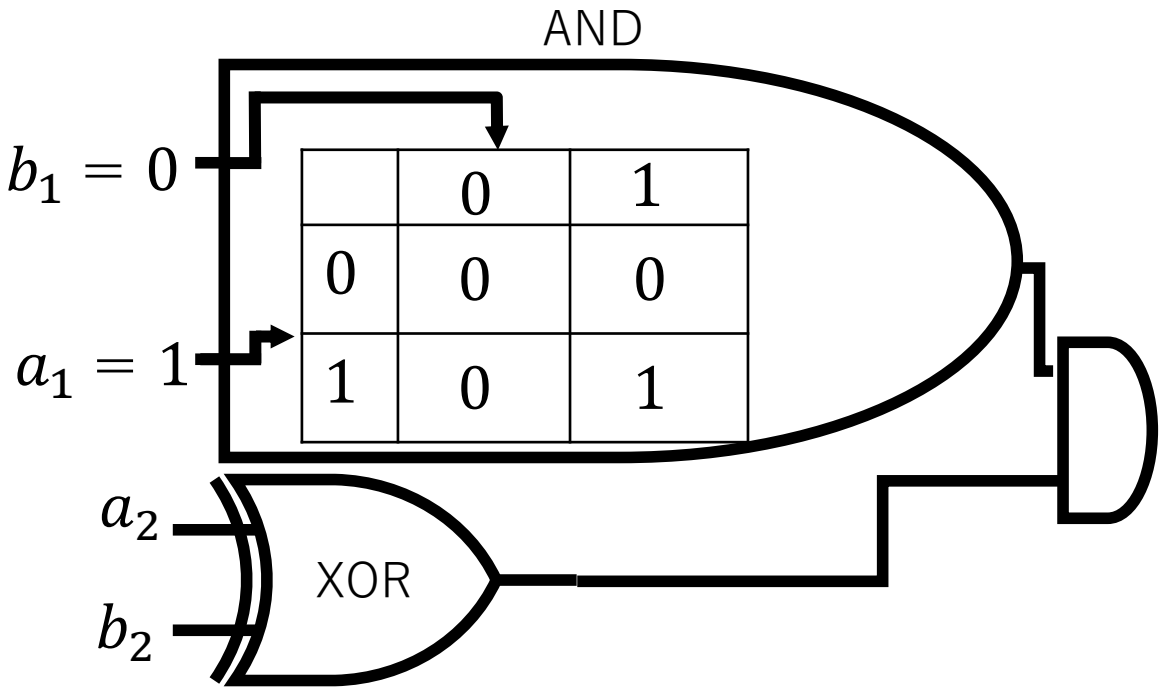
ゲートの種類がわかっている条件でXORのカード枚数を減らす

1ゲート8枚のプロトコル-準備

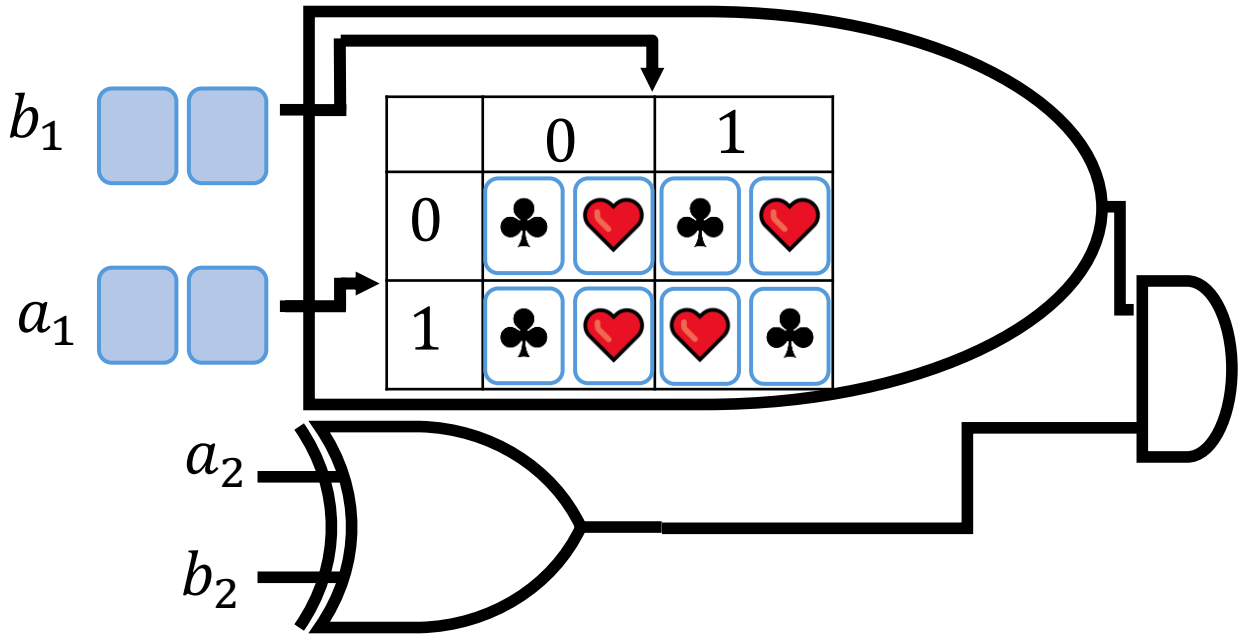
- 符号化に従って，入力やゲートをカードで表す
  - 入力は秘密裏に作成し，裏向き
- NOTは左右のカードを入れ替えることで用意しておく

符号化

0	♣ ♥
1	♥ ♣



	0	1
0	0	0
1	0	1



	0	1
0	♣ ♥	♣ ♥
1	♣ ♥	♥ ♣

プロトコルの方針

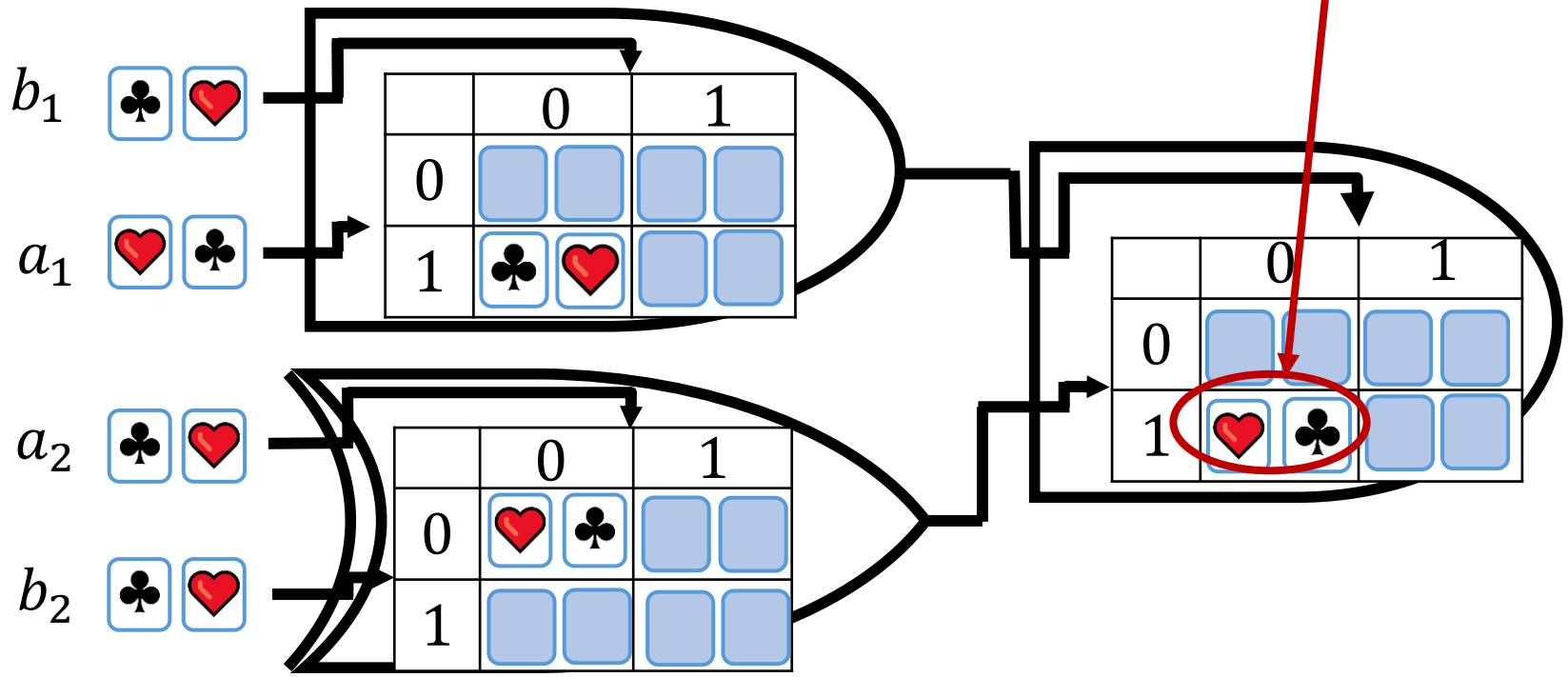
- 目的：入力と中間値を秘匿しつつ出力を得る
  - 準備したカードを裏返して**上手にシャッフル**してから**上手にめくる**

出力のみ分かる

入力と中間値を秘匿し，かつ出力が変わらないシャッフル

符号化

	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥



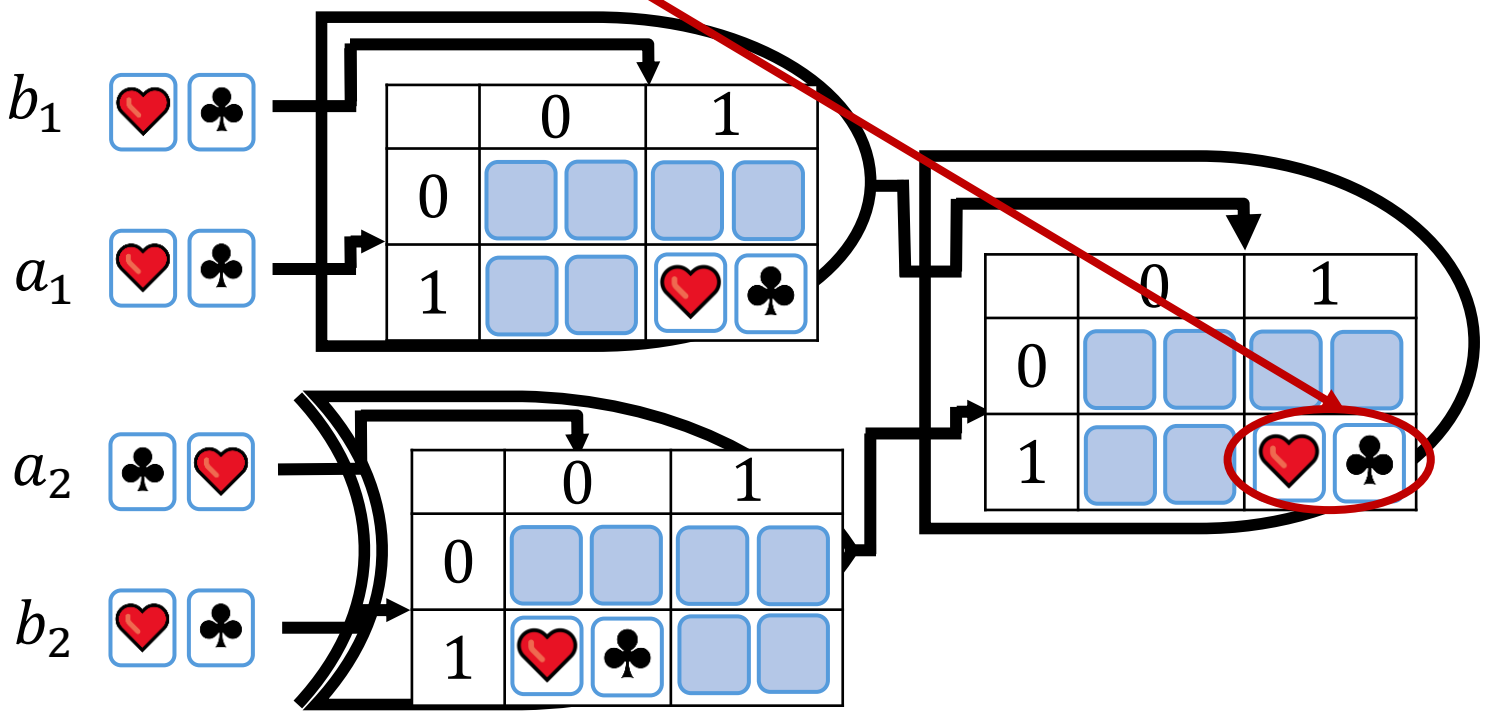
プロトコルが出力を得る方法

- 入力に対応する場所のカードを表にすることを繰り返す
  - 最後のゲートのカードが表す値が出力
  - 上手いシャッフルと組み合わせることで出力のみ明らかにする
- 例：対象の論理回路

  $(b_1 \wedge a_1) \wedge (a_2 \oplus b_2)$
  - $(a_1, a_2, b_1, b_2) = (1,0,1,1)$

符号化

	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥

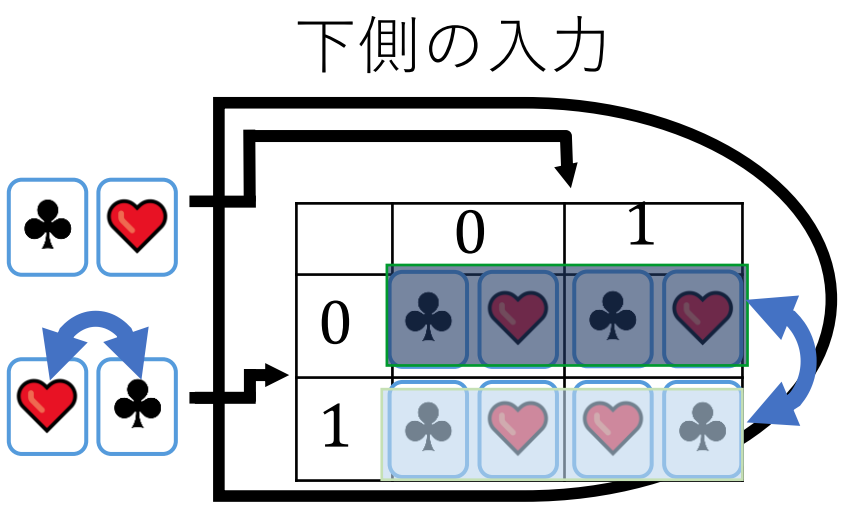


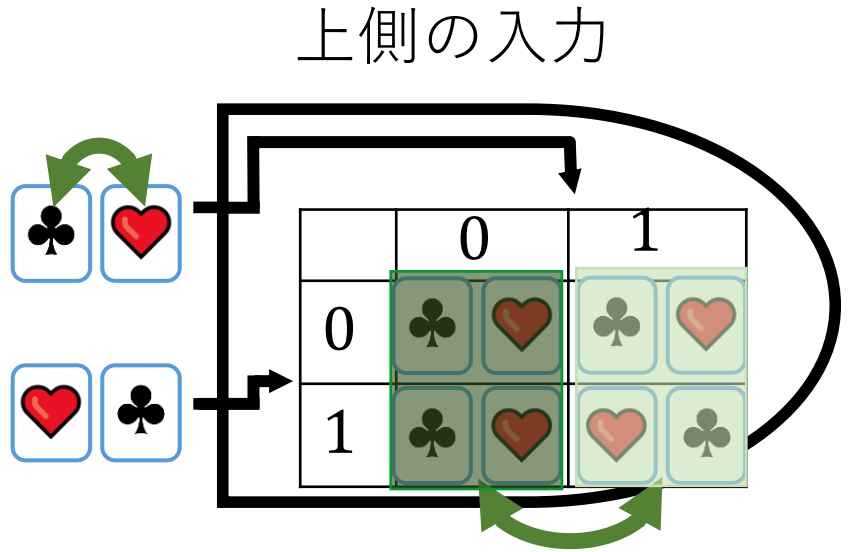
入力の秘匿（ビット反転）と出力の維持

- 入力の秘匿（カードは裏返しておく）
  - 入力を表す2枚のカードをシャッフル
- 出力を維持（カードは裏返しておく）
  - 入力の秘匿に合わせてゲートのカードをシャッフル

符号化

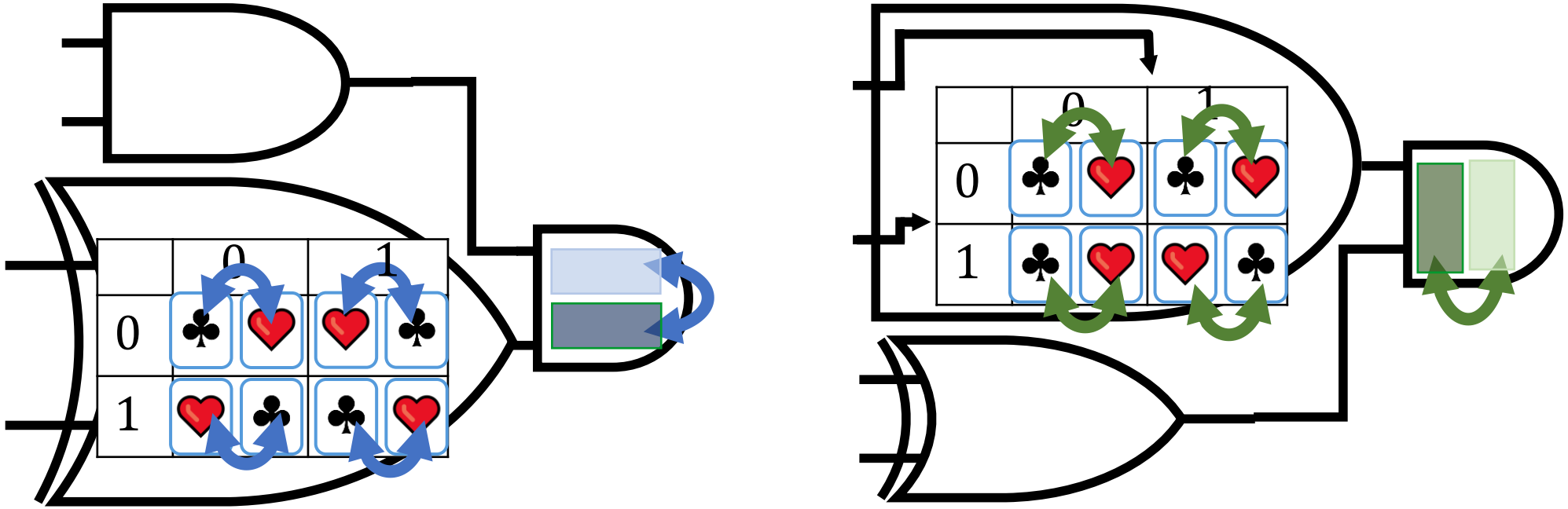
	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥





中間値の秘匿（ビット反転）と出力の維持

- 中間値の秘匿（カードは裏返しておく）
  - 中間値を表すカードを対応させてシャッフル
- 出力を維持（カードは裏返しておく）
  - 中間値の秘匿に合わせてゲートのカードをシャッフル

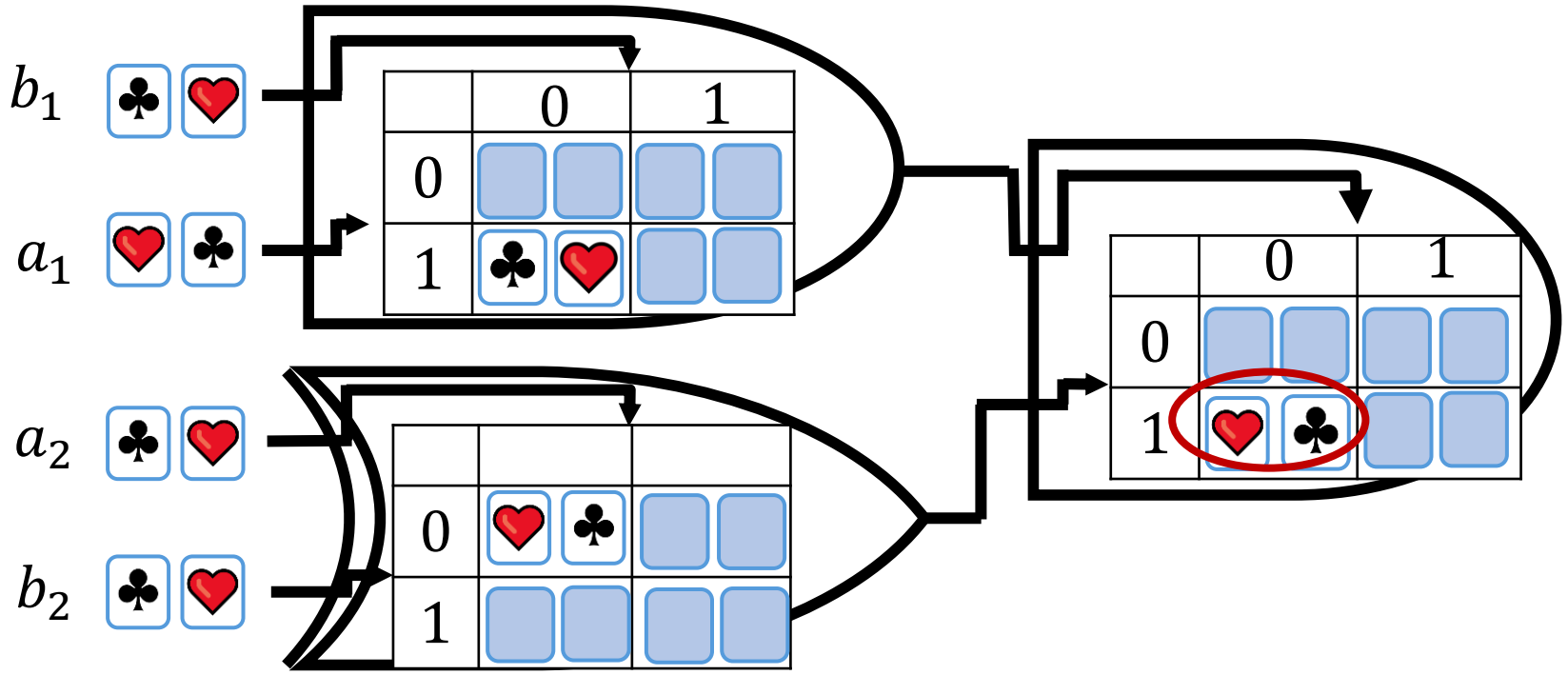


プロトコルの出力

- ゲートの入力に対応するカードを表向きにする
- 入力の値に対応するゲートのカードを表向きにする
  - 表向きの入力や中間値のカードはシャッフルされたので値は秘匿

符号化

	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥



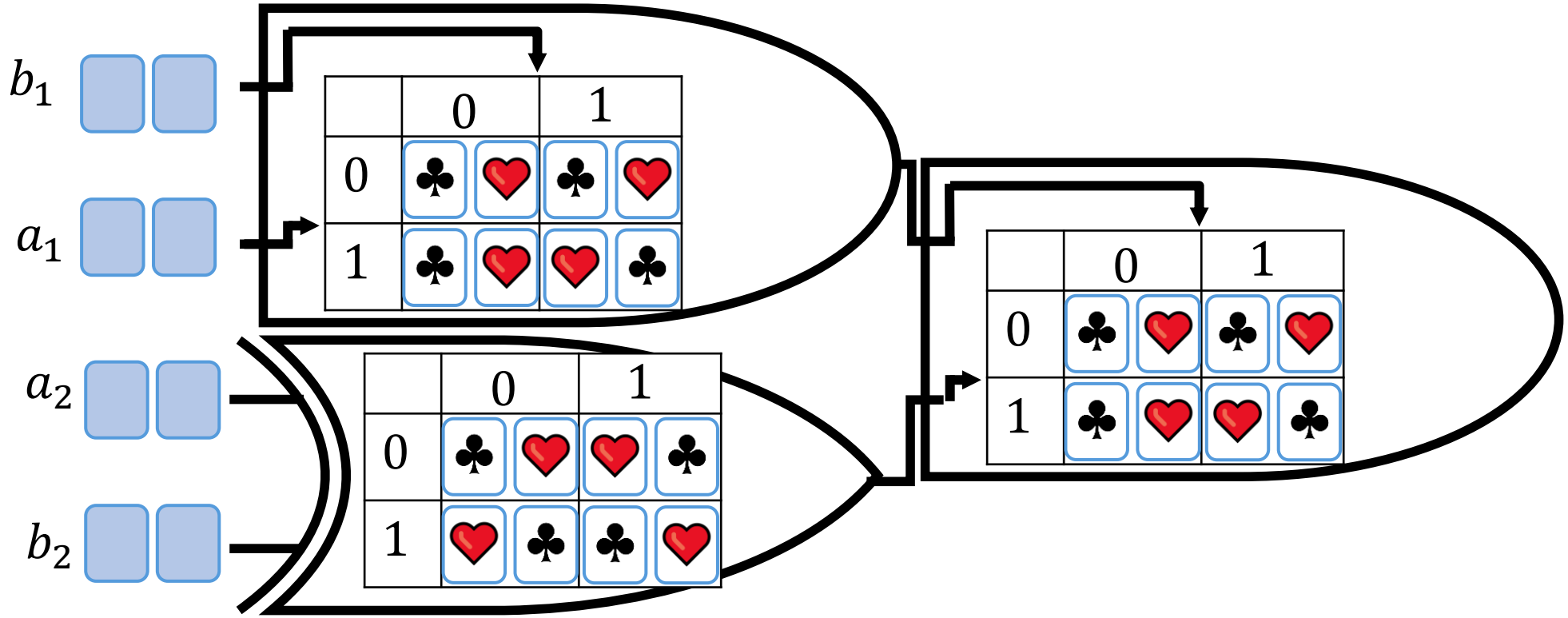
連続するシャッフルはまとめられる[MS14]

- シャッフル間でカードを表にしなければ，それらのシャッフルは１回にまとめられる
  - シャッフルは置換の集合とそれ上の置換の分布からなる（ともに任意）
  - 置換は合成でき，分布を選ぶことができる
- 1ゲート8枚のプロトコルのシャッフルは連続なのでまとめられる

[MS14]T. Mizuki and H. Shizuya, “A formalization of card-based cryptographic protocols via abstract machine,” Int. J. Inf. Sec., 2014

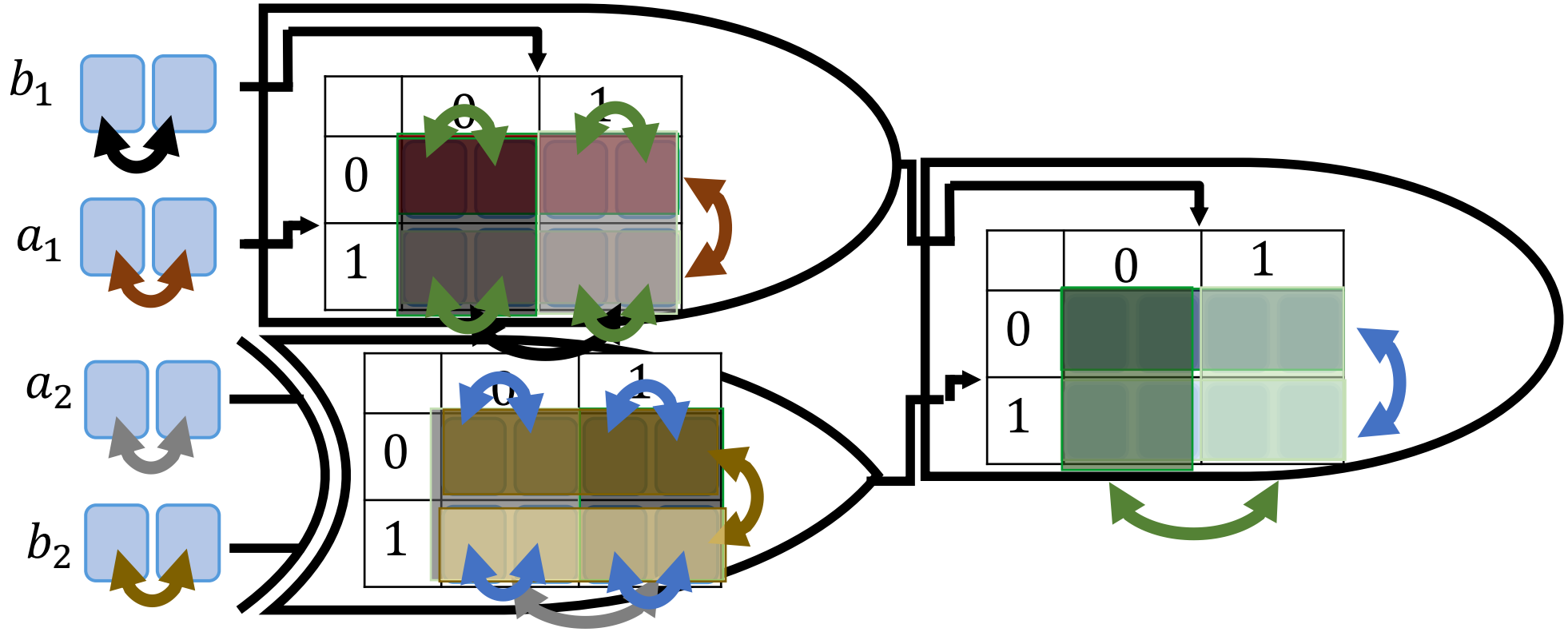
1ゲート8枚のプロトコルの具体例-準備

- 対象の論理回路 $(b_1 \wedge a_1) \wedge (a_2 \oplus b_2)$
- $(a_1, a_2, b_1, b_2) = (1,0,1,1)$とするが本人のみ分かる．裏向きでおく



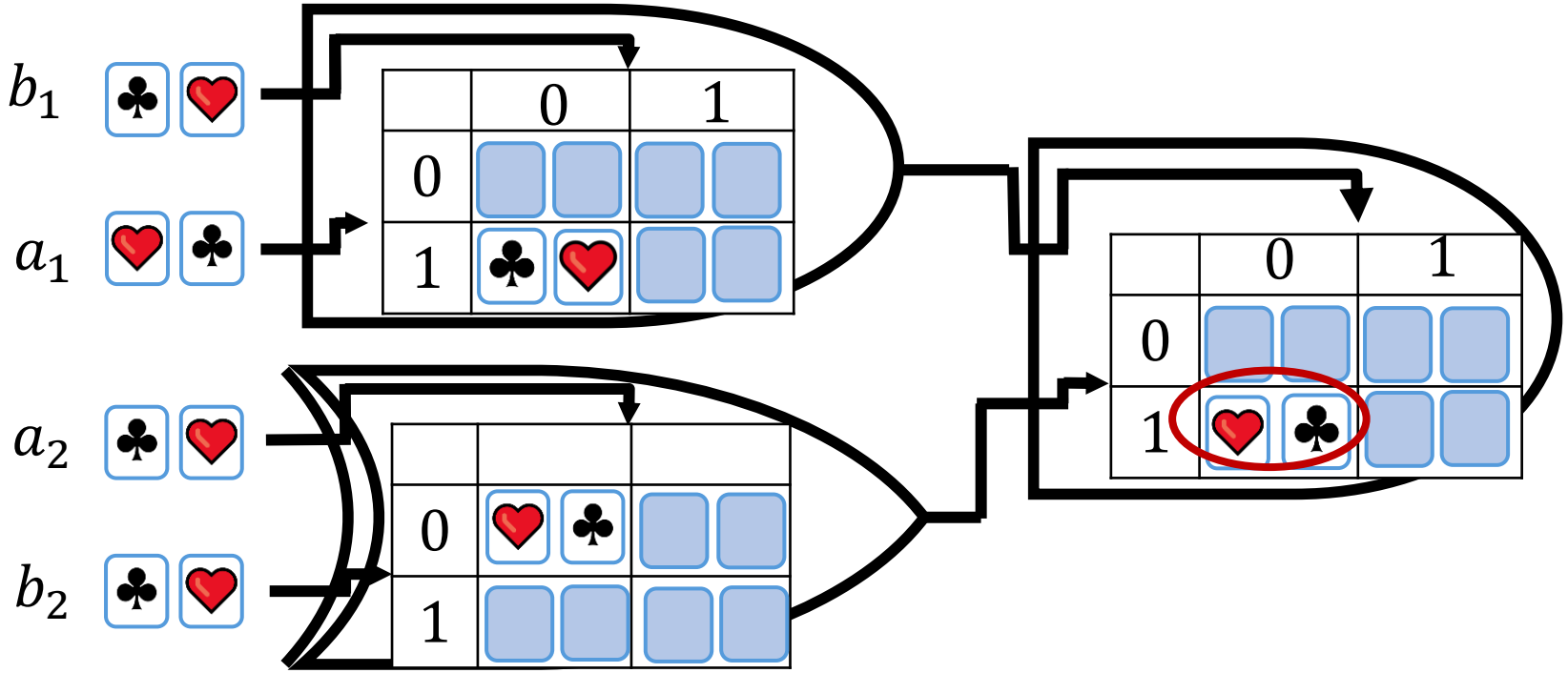
1ゲート8枚のプロトコル-シャッフル

- 出力以外はランダムにビット反転される
- 連続しているシャッフルのなのでまとめると理論上１回で行える



プロトコルの出力

- ゲートの入力に対応するカードを表向きにする



1ゲートあたり8枚のプロトコルを紹介した

- 任意の論理回路に対するシャッフル１回のプロトコル

  q は計算対象の論理回路のゲート数, n, mは入出力ゲート数

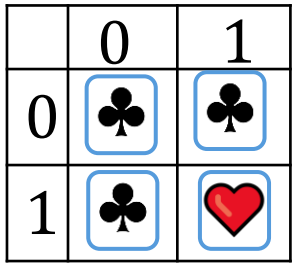
プロトコル	カード枚数	コミット型	シャッフルの種類
Shinagawa-Nuida [SN21]	$24q + 2n$	✔	一様閉
Tozawaら [TMM23]	$8q + 2n$	✔	一様閉
Onoら[OSN+23]	$6q + 2n$		一様
Onoら[OSN+23]	$6q + 2n + 2m$	✔	一様

他にManabe-Shinagwa[KS23]らのプロトコルが存在
　ゲートの種類がわかっている条件でXORのカード枚数を減らす

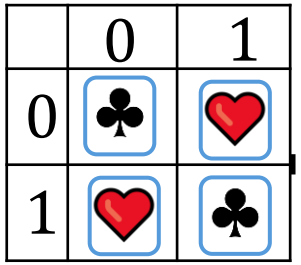
1ゲートあたり6枚のプロトコル-アイデア

- 1ビットを1枚で符号化

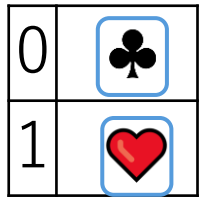
AND



XOR

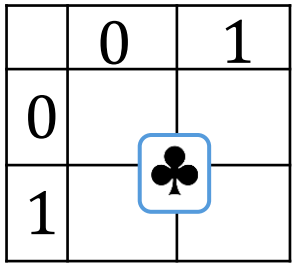


符号化

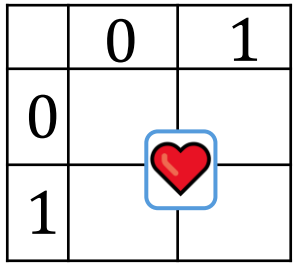


- 次の二つの２入力1出力のゲートは特別な方法で符号化する

常に0を出力



常に1を出力

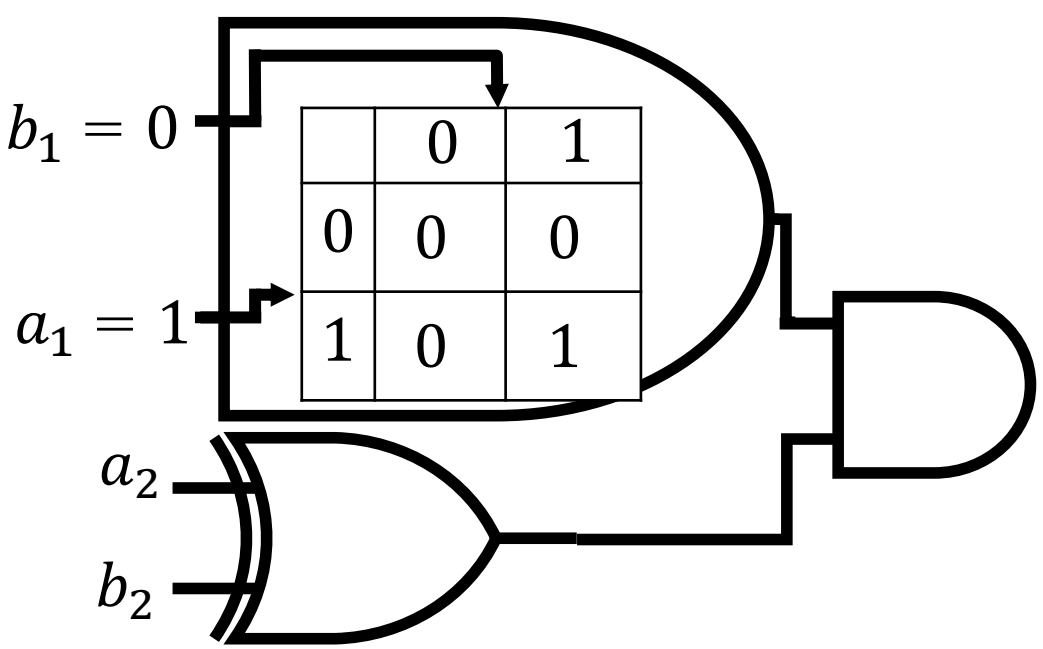


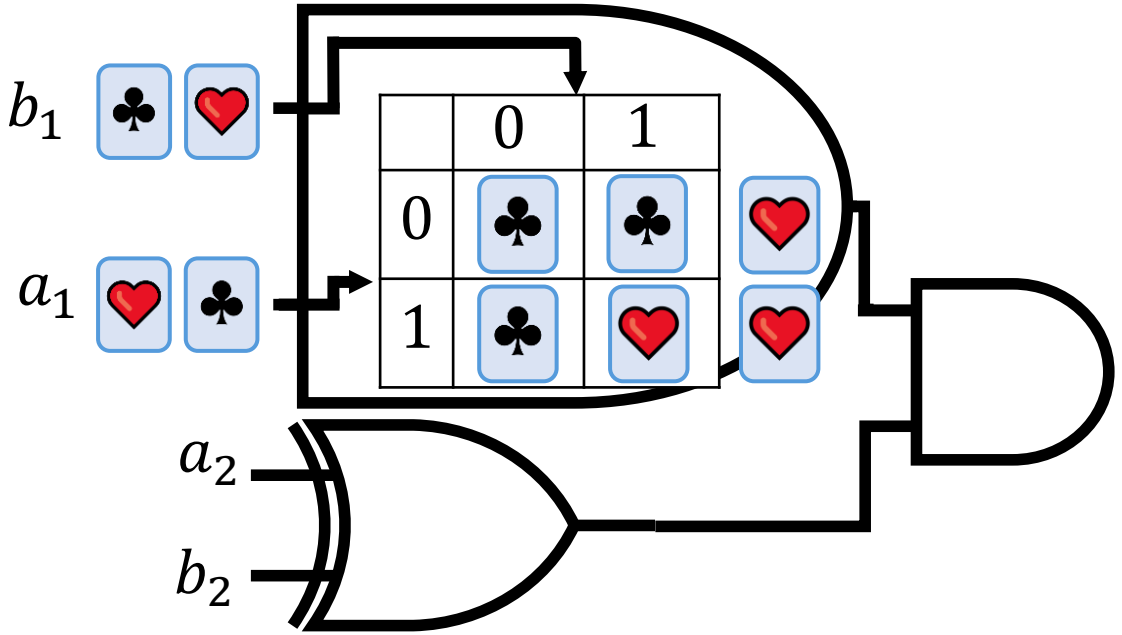
1ゲートあたり6枚のプロトコル-準備

- 符号化に従って，入力やゲートをカードで表す
- ゲートの種類は公開
  - 1ゲートあたり6枚（**任意の2入力１出力ゲートを作れる**）
  - 1入力あたり2枚

符号化

	入力		ゲート
0	♣	♥	♣
1	♥	♣	♥





プロトコルの方針

- 目的：入力と中間値を秘匿しつつ出力を得る

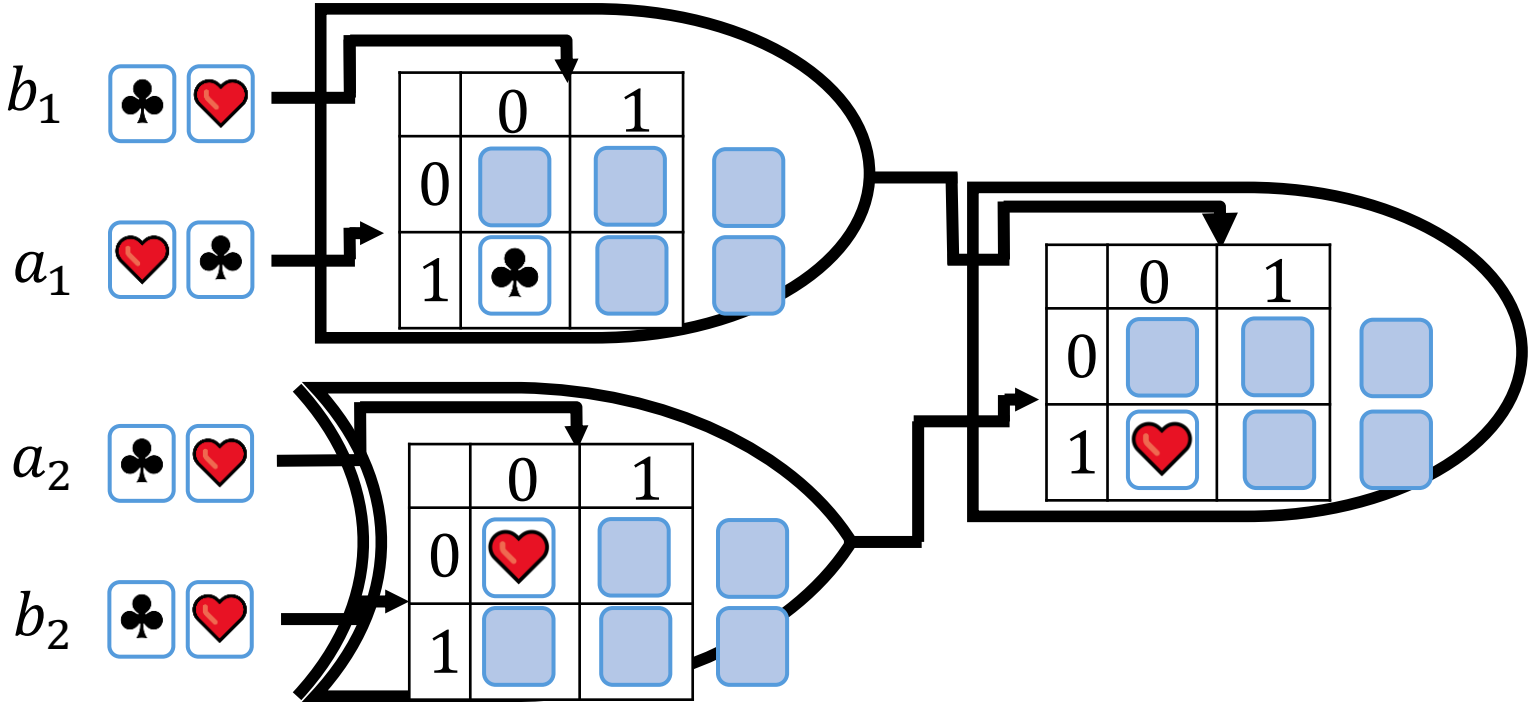
出力のみ分かる

  - 準備したカードを裏返して**上手にシャッフル**してから**上手にめくる**

入力と中間値を秘匿し，かつ出力が変わらないシャッフル

符号化

	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥



プロトコルが出力を得る方法

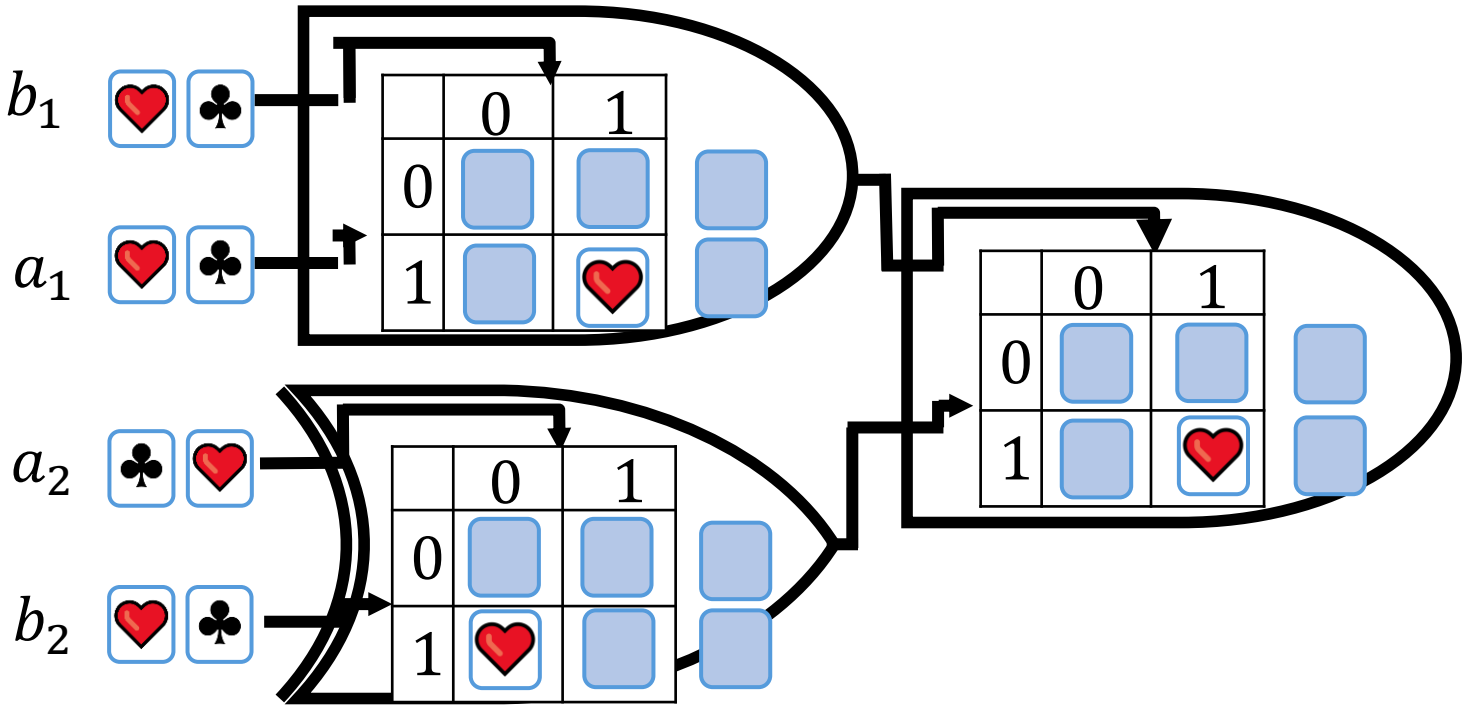
- 入力に対応する場所のカードを表にすることを繰り返す
  - 最後のゲートのカードが表す値が出力
- 上手いシャッフルと組み合わせることで出力のみわかる
- 例：対象の論理回路

$(b_1 \wedge a_1) \wedge (a_2 \oplus b_2)$

$a_1 = 1, a_2 = 0, b_1 = 1, b_2 = 1$

符号化

	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥

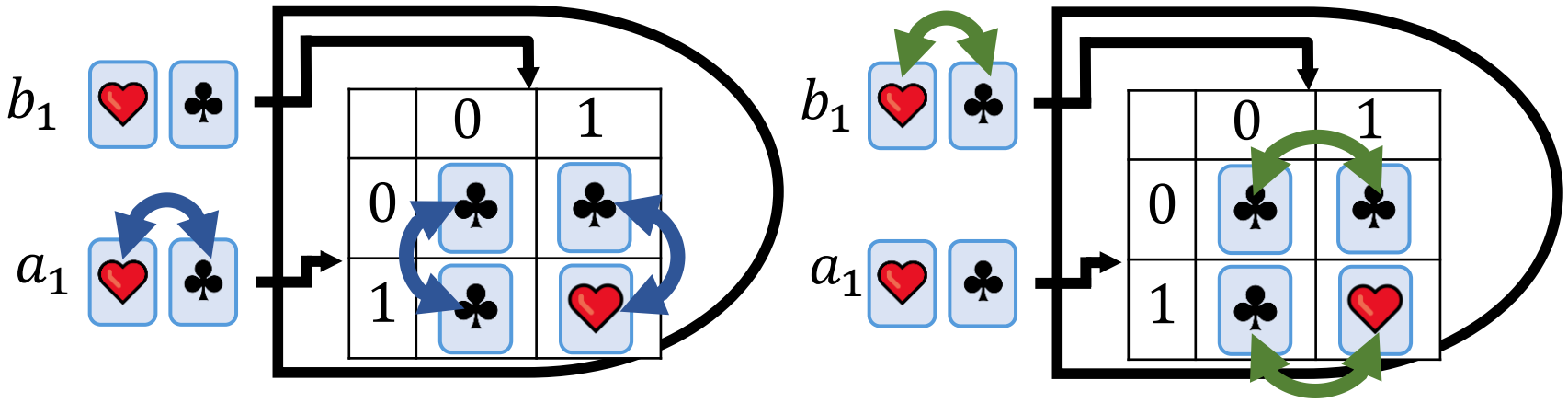


入力の秘匿と出力の維持

- 入力の秘匿（カードは裏返しておく）
  - 入力を表す2枚のカードをシャッフル
- 出力を維持（カードは裏返しておく）
  - 入力の秘匿に合わせてゲートのカードをシャッフル

符号化

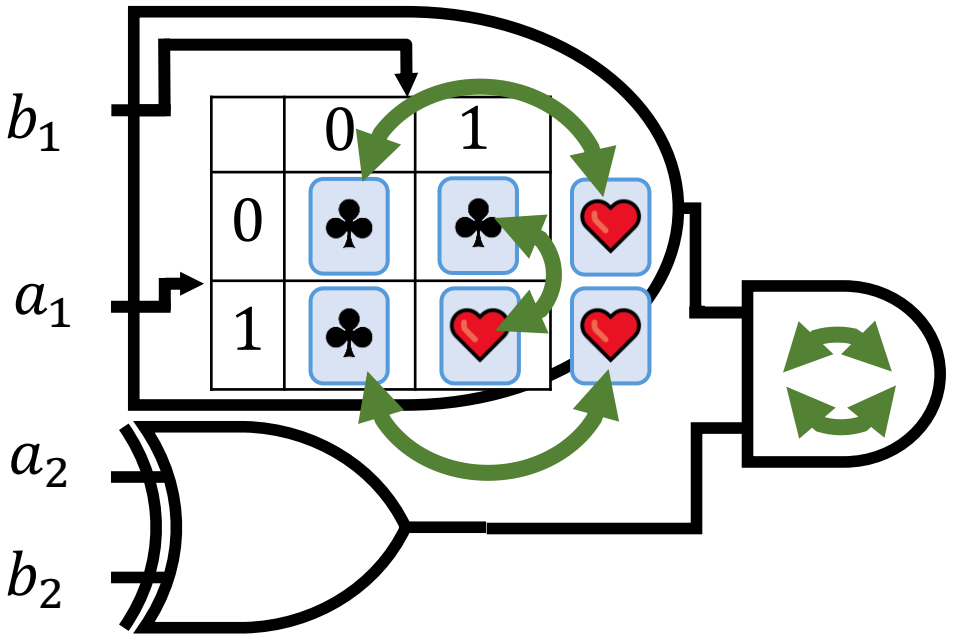
	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥

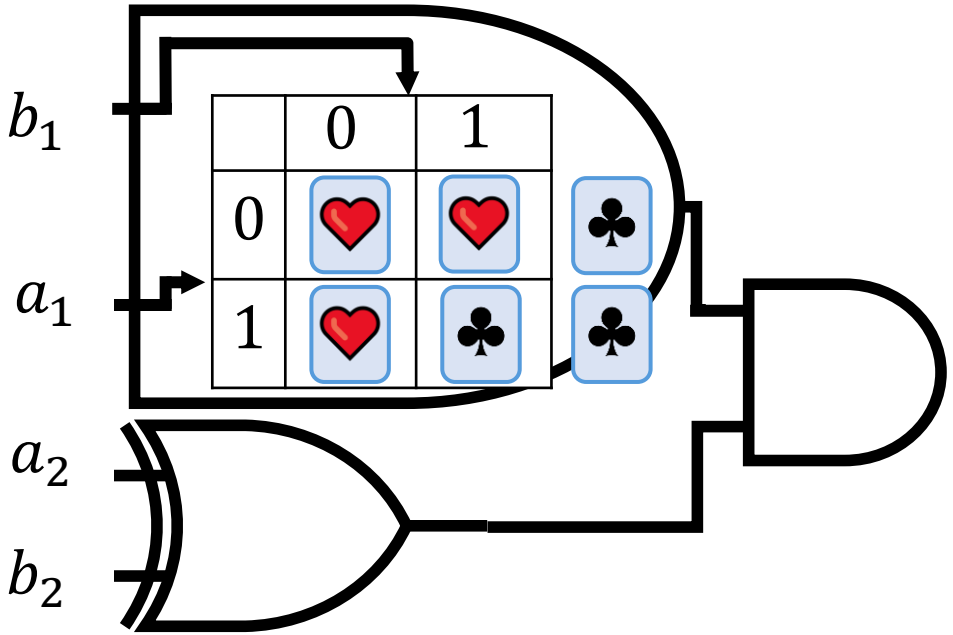


中間値の秘匿（ビット反転）と出力の維持

- 中間値の秘匿（カードは裏返しておく）
  - 中間値を表すカードをゲートの種類に合わせてシャッフル
- 出力を維持（カードは裏返しておく）
  - 中間値の秘匿に合わせてゲートのカードをシャッフル

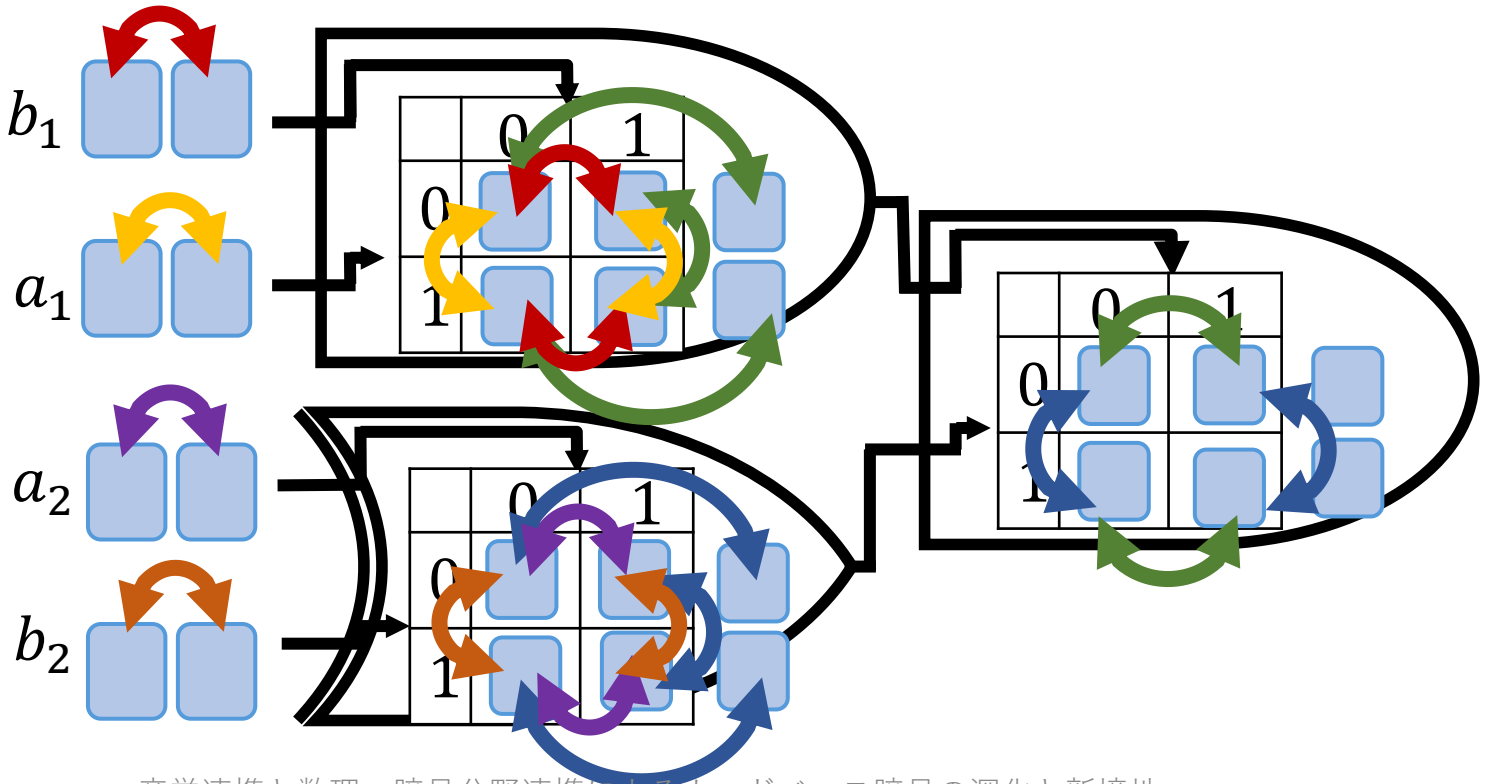
余りのカードを使える！





シャッフルの順番

- 出力側のシャッフルを先に行う
  - 例：青緑のシャッフルを先に行う
- 閉じていない

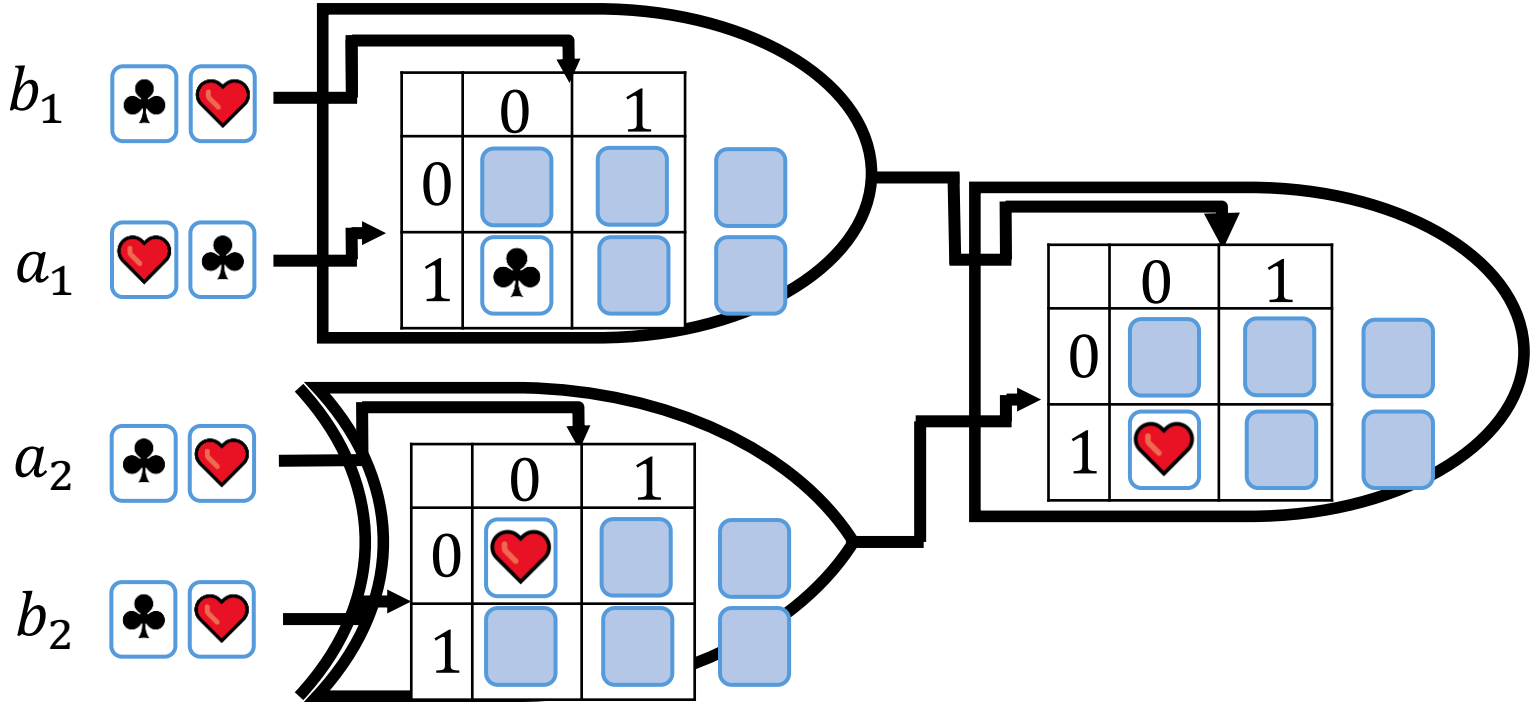


プロトコルの出力

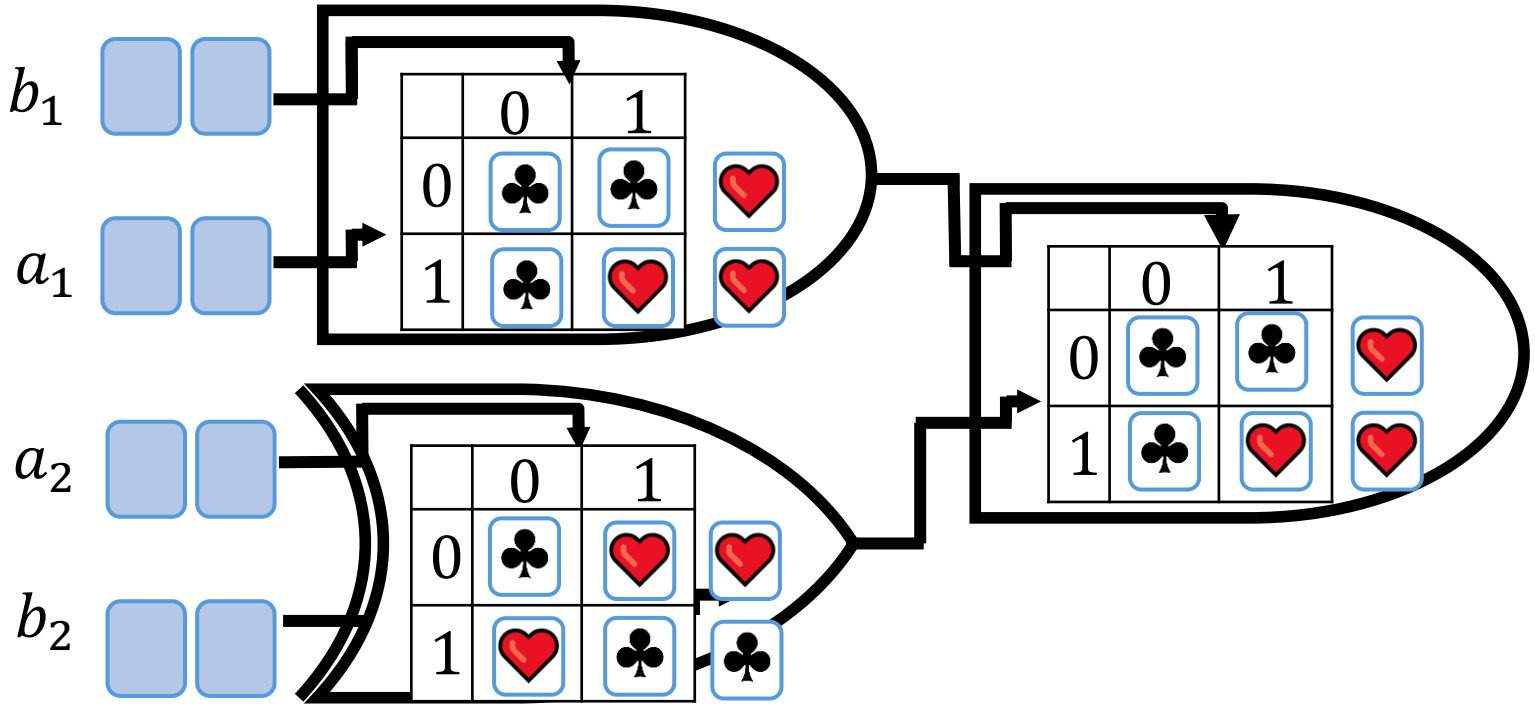
- 入力に対応するカードを表向きにする
- 入力の値に対応するゲートのカードを表向きにする
  - 表向きの入力や中間値のカードはシャッフルされたので値は秘匿

符号化

	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥

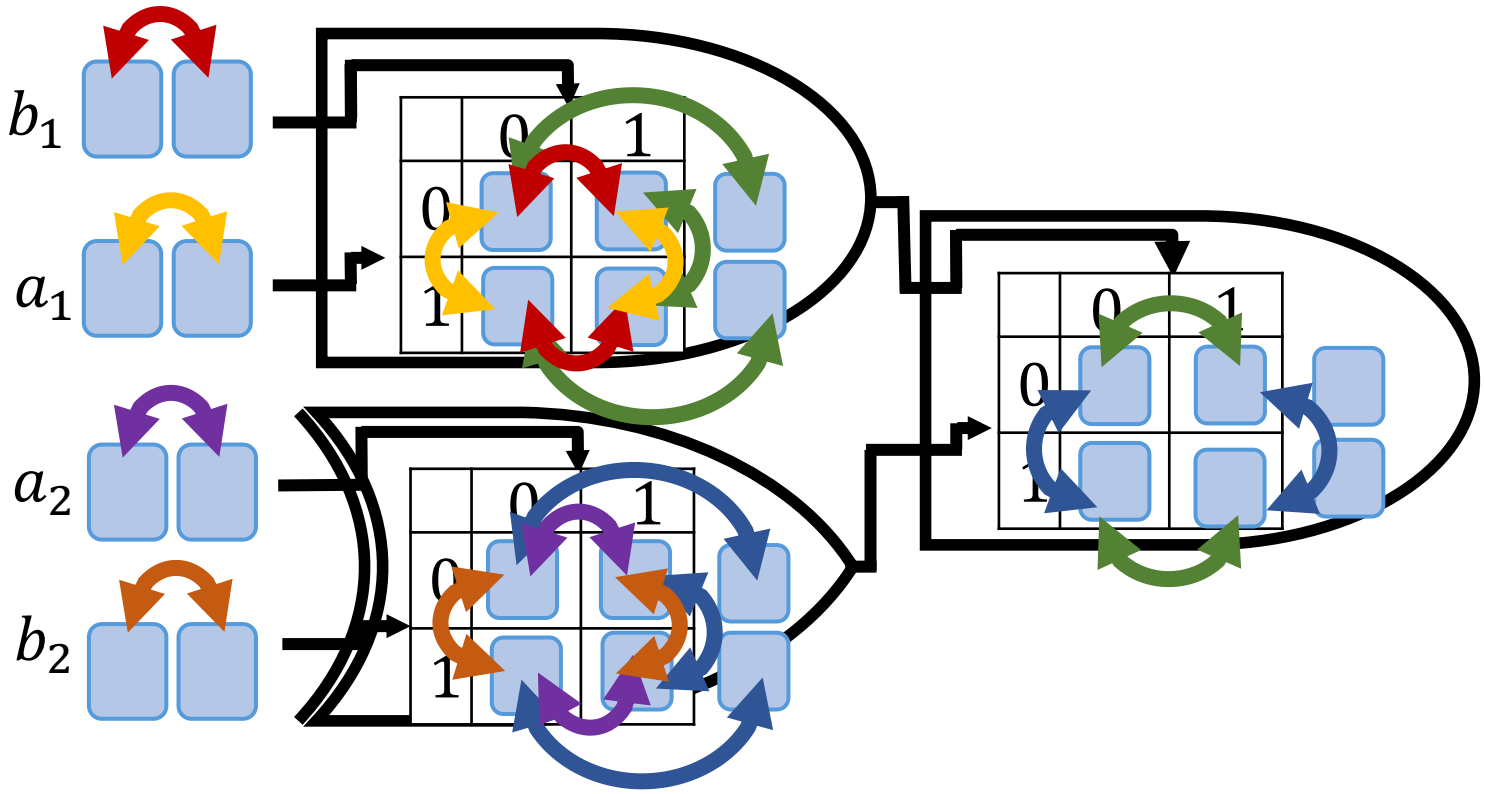


1ゲート6枚のプロトコルの具体例-準備



シャッフル

- 連続しているシャッフルなのでまとめられる

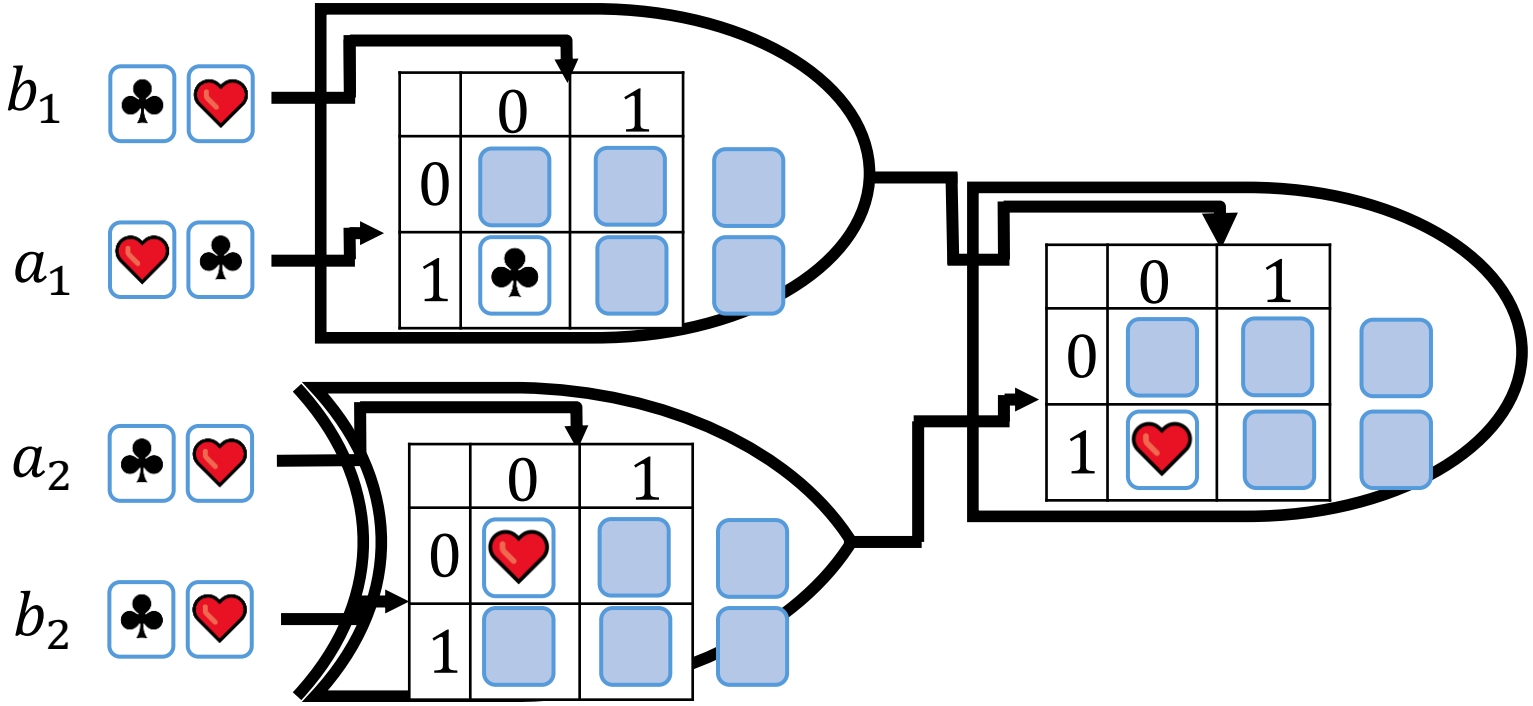


プロトコルの出力

- 入力に対応するカードを表向きにする
- 入力の値に対応するゲートのカードを表向きにする.

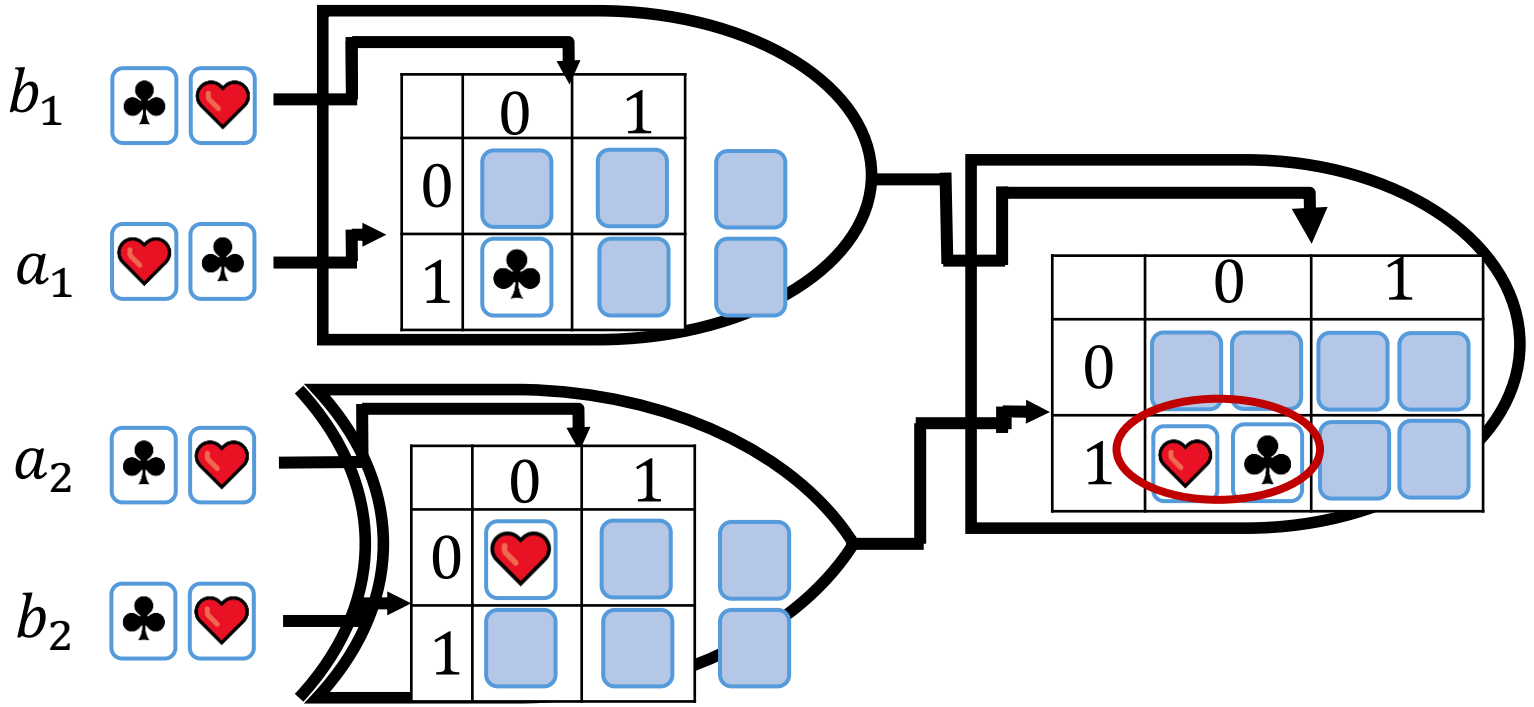
符号化

	入力	ゲート
0	♣ ♥	♣
1	♥ ♣	♥



コミット型への変形

- 最後のゲートを１ゲート8枚のプロトコルに置き換え



1ゲートあたり6枚のプロトコルを紹介した

- 任意の論理回路に対するプロトコルのカード枚数

  qは計算対象の論理回路のゲート数, n, mは入出力ゲート数

プロトコル	カード枚数	コミット型	シャッフルの種類
Shinagawa-Nuida [SN21]	$24q + 2n$	✔	一様閉
Tozawaら [TMM23]	$8q + 2n$	✔	一様閉
Onoら[OSN+23]	$6q + 2n$		一様
Onoら[OSN+23]	$6q + 2n + 2m$	✔	一様

他にManabe-Shinagwa[KS23]らのプロトコルが存在
ゲートの種類がわかっている条件でXORのカード枚数を減らす