カードベース暗号の形式検証の再考

○ 藤田 和弘† 米山 一樹† 品川 和雅†‡

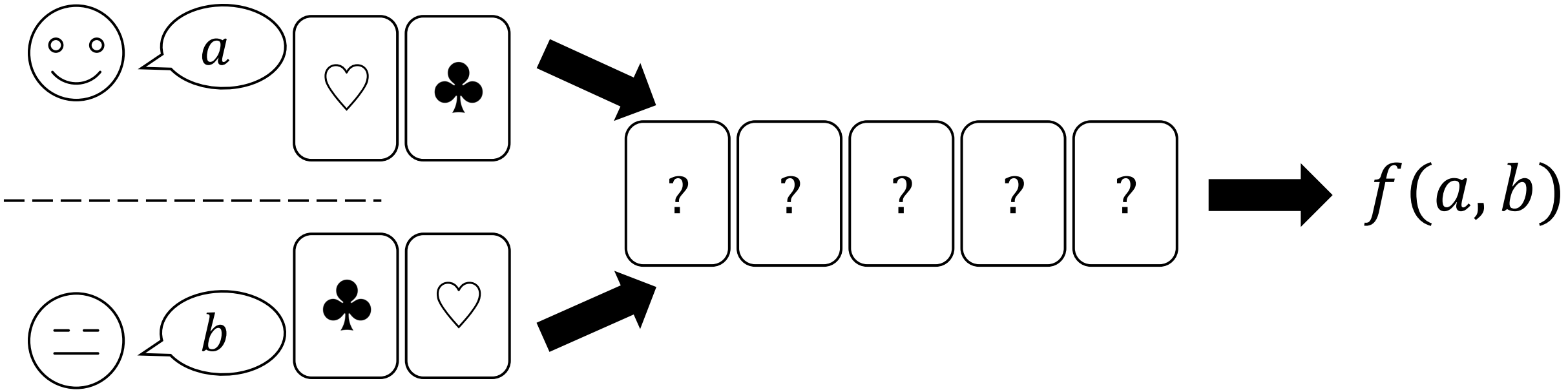
†茨城大学
‡産業技術総合研究所

概要

- 秘密計算の一種、カードベース暗号
- カード枚数/ステップ数の下界証明が重要課題
- 計算機で全探索する検証方法（既存研究）
  - 形式手法（Bounded Model Checking）を活用
- 証明に使用されたプログラムの
  バグを指摘、修正（本発表）

カードベース暗号とは

- 秘密計算の一種
- トランプなどの物理カードを用いて実装



カードベース暗号の計算量の下界

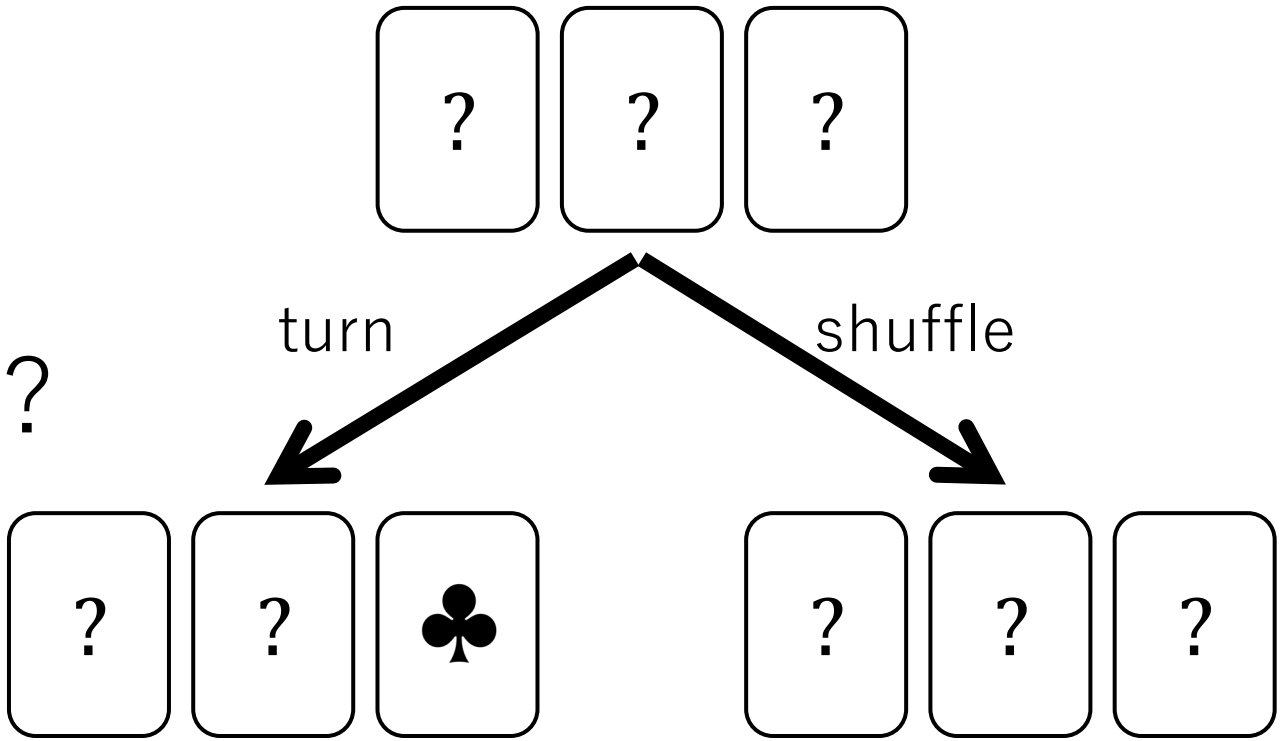
- カード枚数/ステップ数の下界は？
  - 空間計算量/時間計算量への関心

- ある関数をn枚lステップで計算する
  プロトコルは構成不可能であることを示す
  - 紙とペンによる証明
  - 全探索

全探索による不可能性の検証

- 初期状態からステップごとに遷移
- 最終状態で結果を出力
- 有向木で表せる

- 安全性を保った遷移か？
- 出力は正しいか？



既存研究[KSK21]

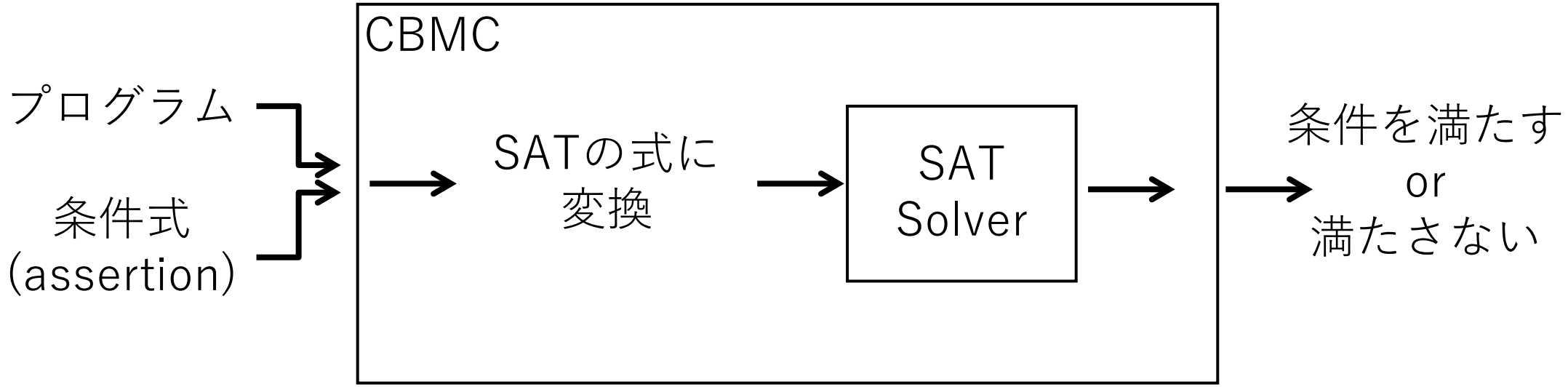
- 任意の(n, l)について
  n枚lステップのANDプロトコルを探索

- n枚のカード列にl回操作を行うプログラムを作成
  - security/correctnessに関する条件式も記述
- プログラムが条件式を満たすかどうか検証

[KSK21] Koch, A., Schrempp, M. and Kirsten, M., Card-based cryptography meets formal verification, New Generation Computing, vol.39 (2021), 115-158.

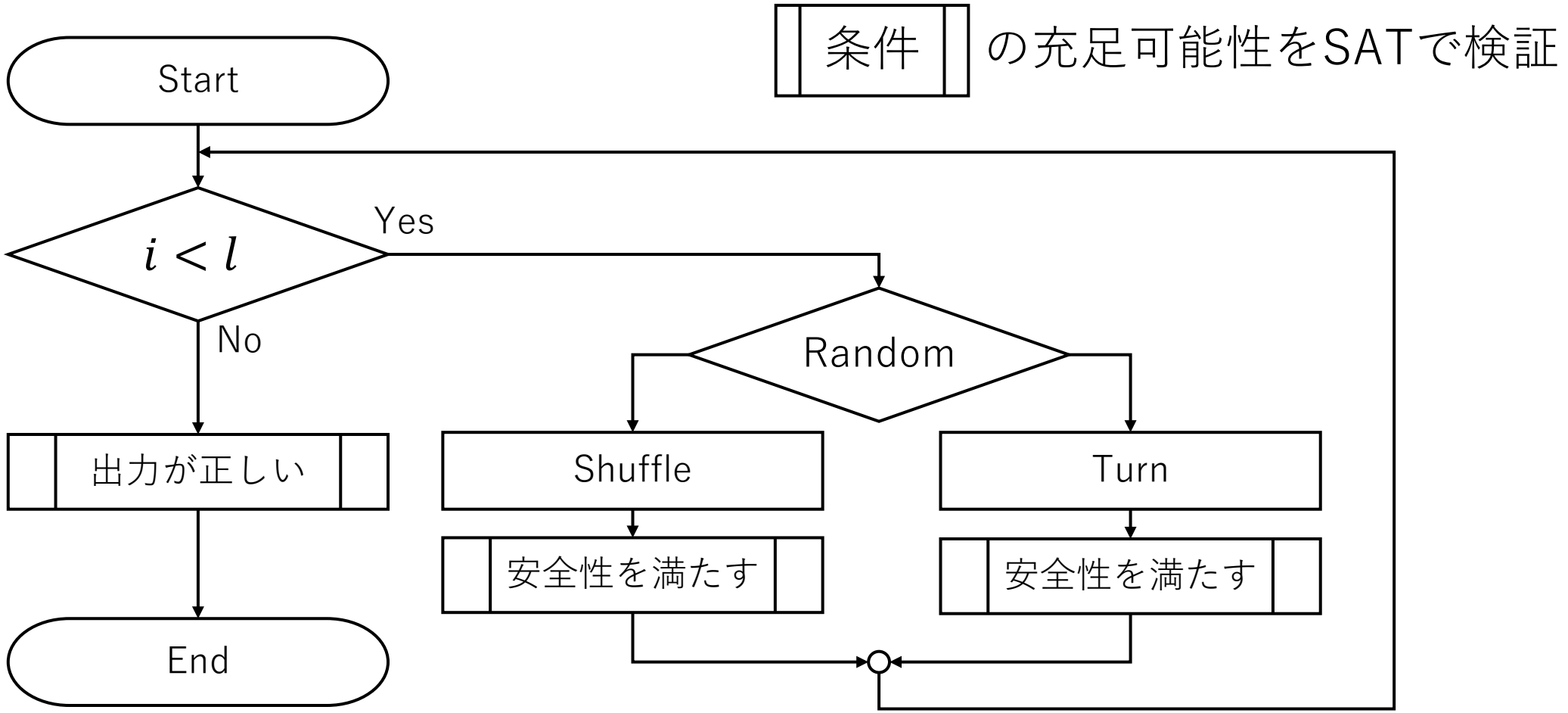
[KSK21]の構成イメージ

- CBMC[CKL04]で条件式をSATに帰着



[CKL04] Clarke, E., Kroening, D., Lerda, F. (2004). A Tool for Checking ANSI-C Programs. In: Jensen, K., Podelski, A. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2004. Lecture Notes in Computer Science, vol 2988. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24730-2_15

[KSK21]検証プログラムのフロー図



本研究の貢献

- [KSK21]の検証プログラムの誤りを
  具体的な反例とともにモジュールごとに指摘
- それぞれの誤りが検証結果に影響を与える
  検証条件を整理
- 適切な修正方法の提案

誤りがある検証条件

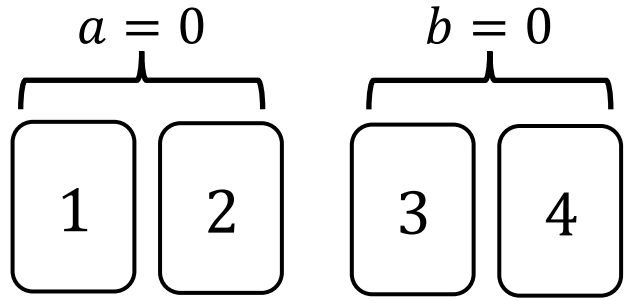
検証する安全性 / カードの種類	Probabilistic Security	Input-possibilistic Security	Output-possibilistic Security
2色カード	×		
数字カード	×	×	

- ×: 誤りが検証結果に影響を及ぼす

プロトコルの表現：state

- KWH tree: 有向木によるプロトコルの図式表現
- KWH treeのnodeをstateと呼ぶ
  - カード列と確率の組の集合
  - 入力が$00,01,10,11$である確率は$X_{00}, X_{01}, X_{10}, X_{11}$

1234	X_{00}
1243	X_{01}
2134	X_{10}
2143	X_{11}



プロトコルの表現：shuffleとturn

- shuffle: 並び替える

1234	$X_{00} + X_{01}$
2134	X_{10}
2143	X_{11}

$(\text{shuffle}, \{\text{id}, (1\ 2)\})$

1234	$1/2\,(X_{00} + X_{01} + X_{10})$
2134	$1/2\,(X_{00} + X_{01} + X_{10})$
2143	$1/2\,X_{11}$
1243	$1/2\,X_{11}$

- turn: めくる

1234	X_{00}
1243	X_{01}
2134	X_{10}
2143	X_{11}

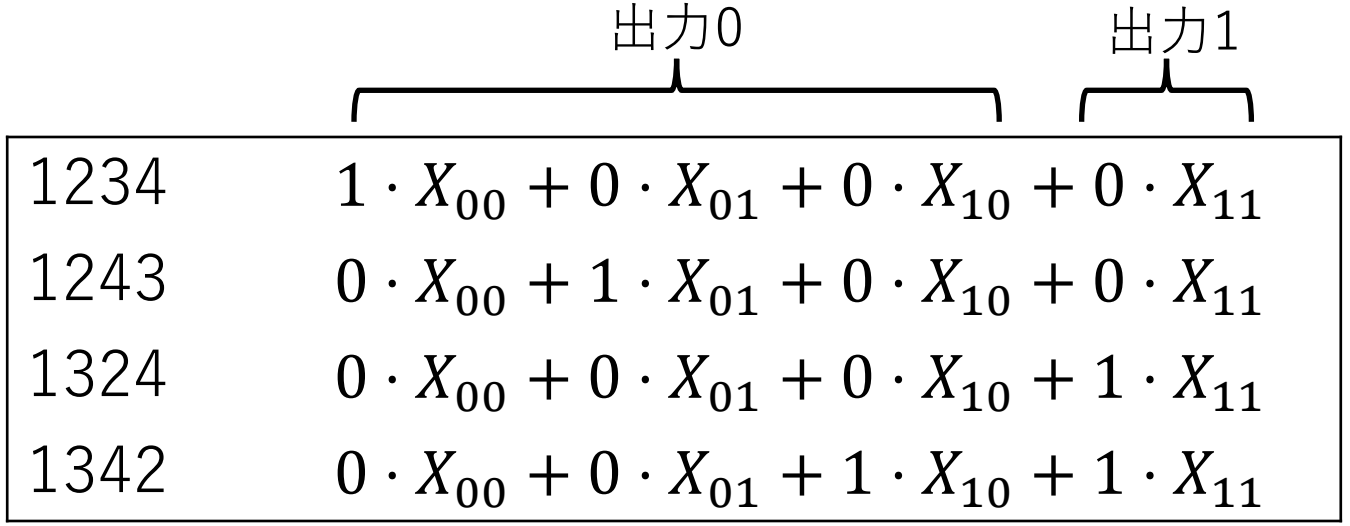
$(\text{turn}, \{1\})$

1234	X_{00}
1243	X_{01}

2134	X_{10}
2143	X_{11}

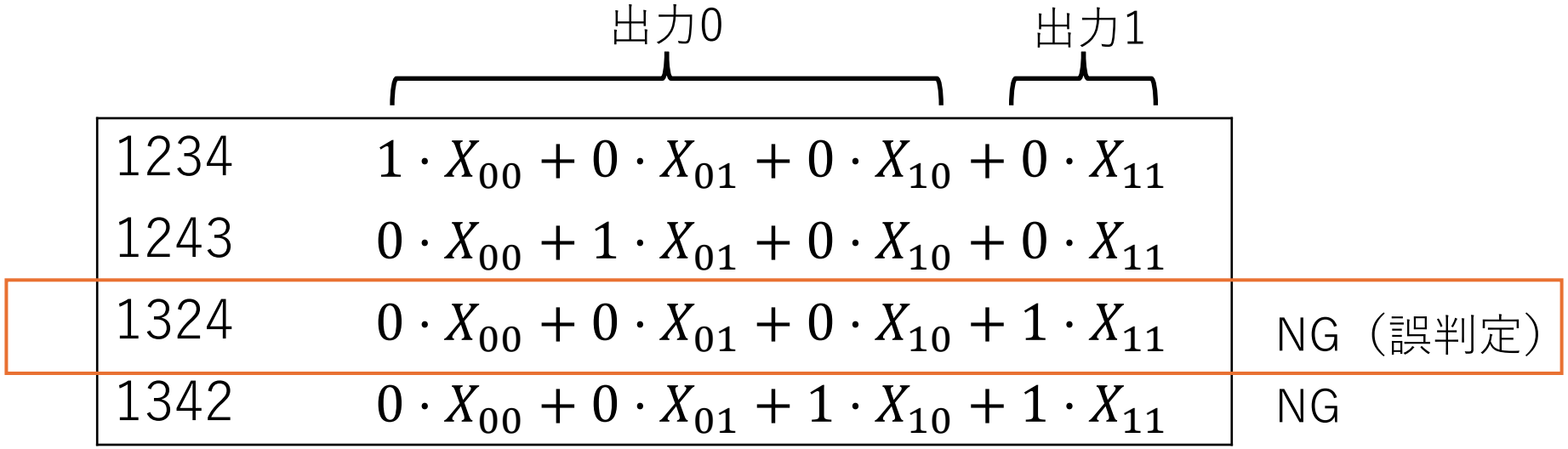
shuffle後のcorrectness検証

- 全てのカード列が出力$\mathbf{0,1}$のどちらか一方だけに対応すること
- AND関数の場合、入力$\mathbf{00,01,10}$が出力$\mathbf{0}$に対応



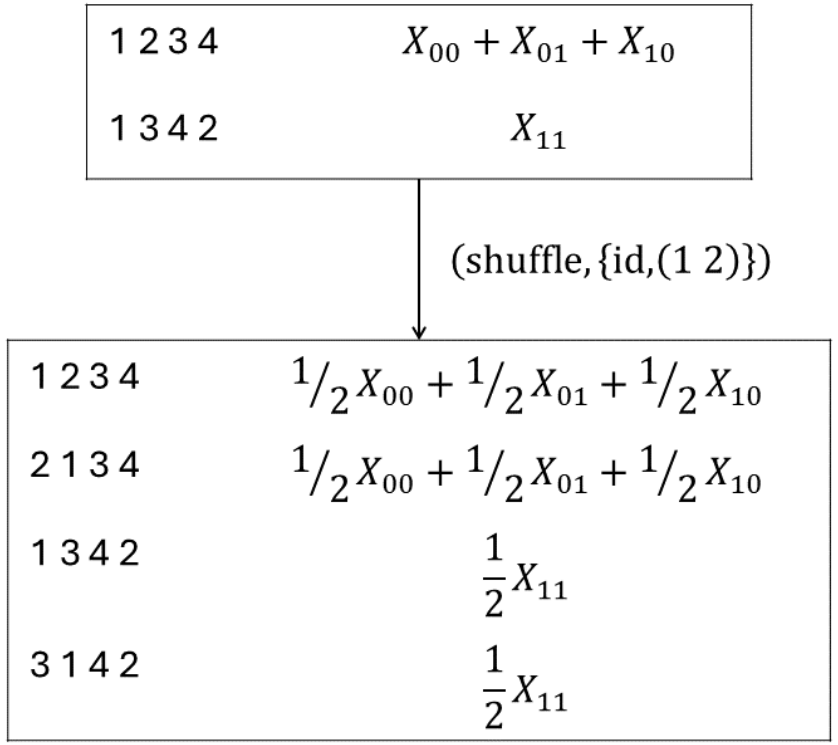
shuffle後のcorrectness検証の誤り

- [KSK21]は「出力1に対応するかどうか」を判定
- 出力1のみに対応するカード列について、correctnessを満たさないと誤判定（偽陽性）



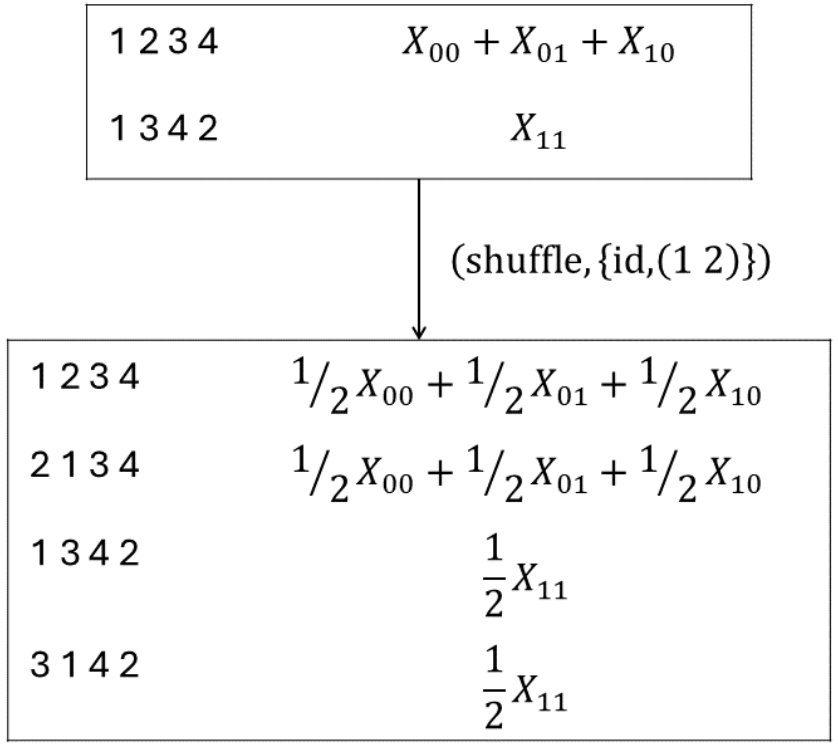
shuffle後の確率計算

- shuffleは置換集合Πから置換πを選択しカード列に適用する操作
- カード列S_nの確率$\mu(S_n)$
  shuffle後の確率は
  $\mu(S_n) := \frac{1}{|\Pi|}\sum \mu\left(\pi^{-1}(S_n)\right)$
- shuffle前の確率を割って足し込んでいけばよい



shuffle後の確率計算の誤り

- [KSK21]は、shuffle前の確率に$\frac{1}{|\Pi|}$を足したものを足し込む
- 右図の例では
  係数が全て$\frac{3}{2}$になってしまった

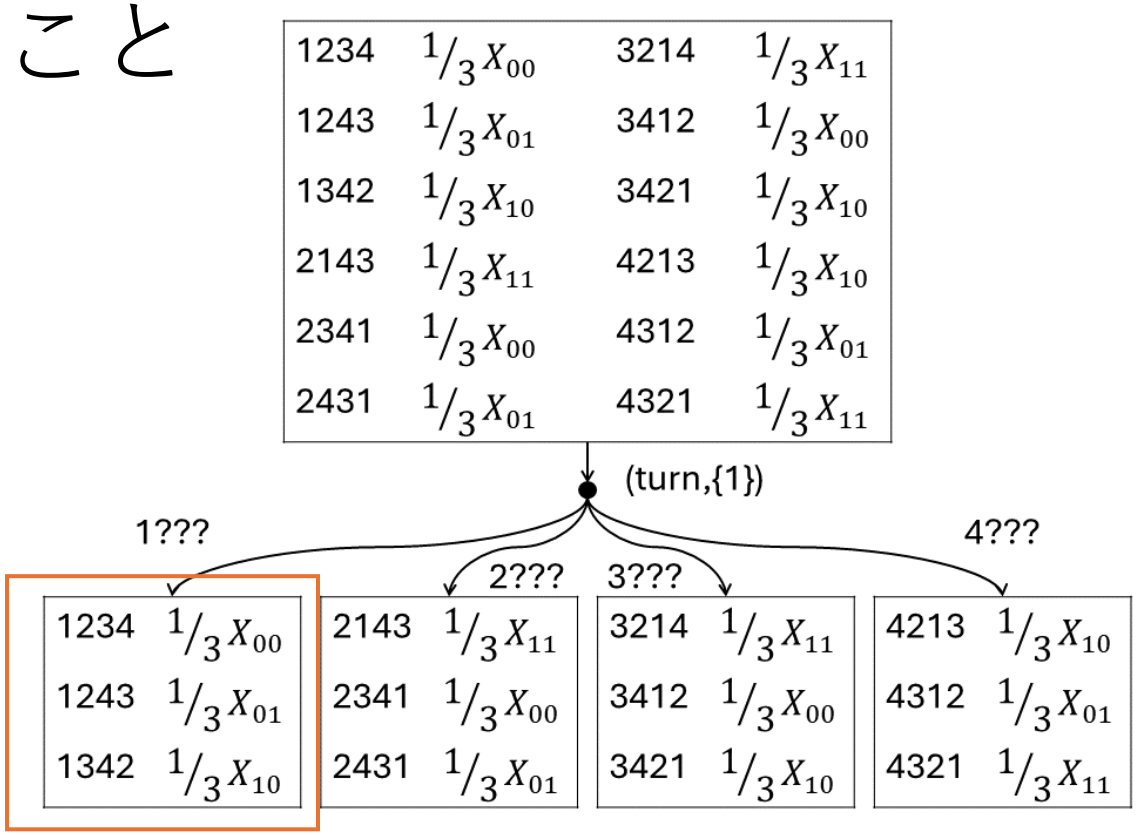


turn後のsecurity検証

- turn後のsecurityの条件は
  各分岐の確率が定数になること
  （変数X_{ab}が消えること）
- 直感：分岐しても
  入力が絞り込まれない

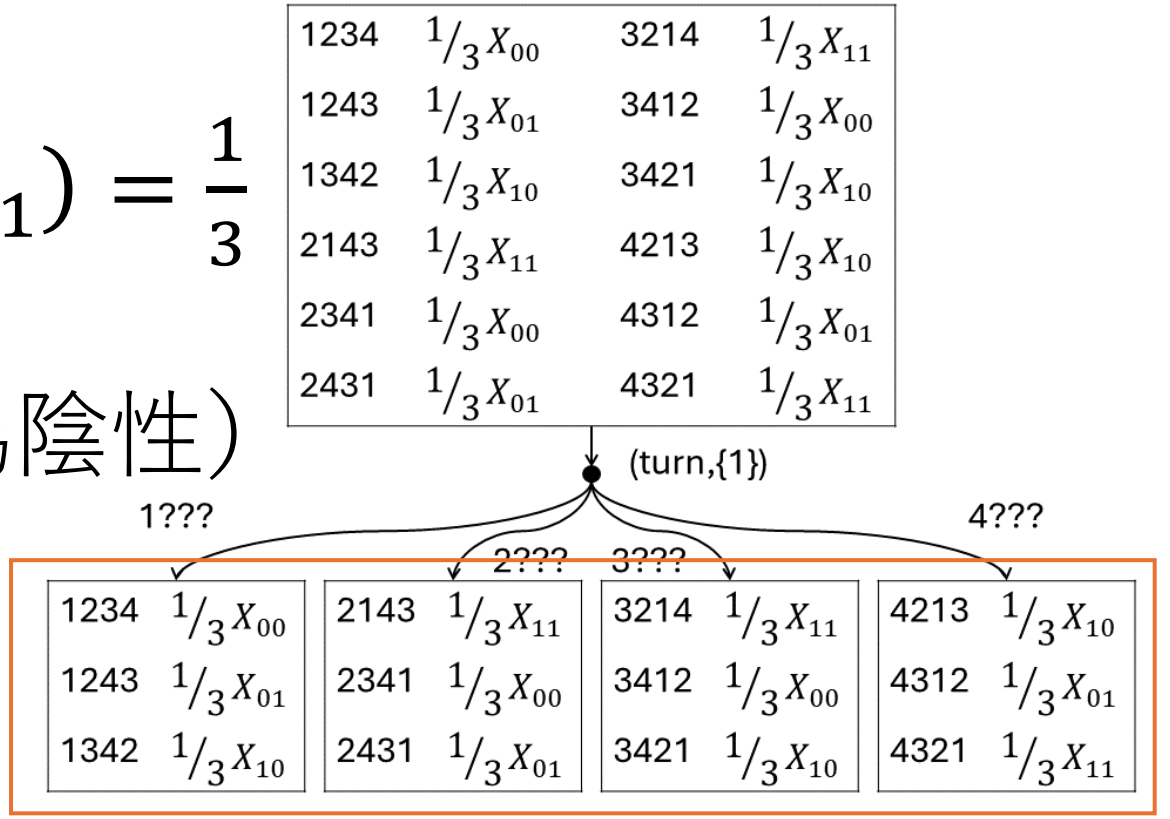
めくって1が出たら
入力1,1はありえないと
分かってしまう

$$\rho = \frac{1}{3}X_{00} + \frac{1}{3}X_{01} + \frac{1}{3}X_{10}$$



turn後のsecurity検証の誤り

- [KSK21]は全分岐で総和をとる
- 右図の例では
  $\rho = \frac{1}{3}(X_{00} + X_{01} + X_{10} + X_{11}) = \frac{1}{3}$
- 定数になるため
  secureであると誤判定（偽陰性）



まとめ：誤りがある検証条件(再掲)

検証する安全性 / カードの種類	Probabilistic Security	Input-possibilistic Security	Output-possibilistic Security
2色カード	確率計算 Security検証		
数字カード	Correctness検証 確率計算 Security検証	Correctness検証	