Computer Security Symposium 2017
23 - 25 October 2017

実行時間に基づいたカードベースプロトコルの評価手法*

上田 格[1]　林 優一[2]　水木 敬明[3]　曽根 秀昭[3]

概要：電子機器を用いずに人間が簡単に秘密計算を実行する手段として，カードベースプロトコルが存在する．この研究分野での主な目標の一つに，効率よく秘密計算ができるプロトコルを作り出すことが挙げられる．その性能評価の指標として，既存研究では用いるカードの枚数やカードの種類数，ループが発生する場合を考慮した平均試行回数等が用いられてきた．しかし，これらの性能評価ではプロトコルにおける各操作の実行時間や手数の多さは考慮していなかった．そこで本稿では，新たな評価指標としてプロトコル中の操作数や実行時間を取り上げ，それらを用いてプロトコルの評価を行う手法を提案し，実際に既存のプロトコルの実行時間を評価する．

キーワード：暗号，カードベース暗号，秘密計算

An Evaluation Method on Card-based Protocols Based on Their Execution Time

Itaru Ueda[1]　Yu-ichi Hayashi[2]　Takaaki Mizuki[3]　Hideaki Sone[3]

1. はじめに

トランプのようなカード組を用いて，人間が実際に秘密計算を行う手法として，カードベースプロトコルが存在する．カードベースプロトコルでは，黒 ♣ と赤 ♡ のカードをそれぞれ 1 枚ずつ使用することで，次のようにビット値を表現する．

$$\boxed{\clubsuit}\boxed{\heartsuit} = 0,\ \boxed{\heartsuit}\boxed{\clubsuit} = 1$$

Alice と Bob は，このエンコーディングに従い，各々の秘密のビット a，b の値を秘匿したまま，テーブルの上に置くことができる．

$$\underbrace{\boxed{?}\boxed{?}}_{a}\quad\underbrace{\boxed{?}\boxed{?}}_{b}$$

ただし，使用するカードの裏面 ? は全て同一で見分けがつかないものとし，カードの表面 ♡ あるいは ♣ も同一色であれば見分けがつかないものとしている．このような裏に置かれた 2 枚のカードを（対応するビットの）**コミットメント**という．

カードベースプロトコルは，このような 2 つのコミットメントを入力とし，各々のビット値を秘匿したまま，その論理演算値のみを出力することができる．例えば，最も有用なプロトコルの一つであるコミット型 AND プロトコルを用いると，$a \wedge b$ のコミットメントを得ることができる．

$$\underbrace{\boxed{?}\boxed{?}}_{a}\ \underbrace{\boxed{?}\boxed{?}}_{b}\ \rightarrow\ \ldots\ \rightarrow\ \underbrace{\boxed{?}\boxed{?}}_{a\wedge b}$$

このように，コミットメントを出力するプロトコルを**コミット型**と呼んでいる．コミット型のプロトコルは，出力結果を次の計算にも使用できるため，実用上利用価値が高い．表 1 にこれまでに知られているいくつかのコミット型 AND プロトコルを示す．この表からもわかるように，カードベースプロトコルの研究分野の目標の一つに，効率的なプロトコルを開発することがある．その “効率性” の評価は，各プロトコル中で使用するカードの色数や枚数，実行

*本稿は，2017 年電子情報通信学会ソサイエティ大会での発表を発展させたものである．

1 東北大学 情報科学研究科
Graduate School of Information Sciences, Tohoku University
2 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
3 東北大学 サイバーサイエンスセンター
Cyberscience Center, Tohoku University

時にループが発生する場合を考慮した平均試行回数などで行われてきた．しかし，実際に現実の場面でプロトコルを実行する際，これらの既存の評価指標では，プロトコルの終了までに必要な操作数や実行時間を適切に見積もることは難しい．従って，本稿では，プロトコル中で生じるすべての操作を考慮し，それを基に実行時間を与える性能指標を検討する．また，表 1 に示した既存のコミット型 AND プロトコルを対象に，これらの操作数を算出して，実行時間を評価する．

表 1 コミット型 AND プロトコル

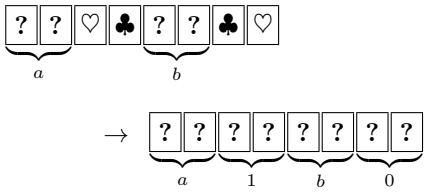
	色数	枚数	平均試行回数
Crépeau & Kilian [1]	4	10	6
Niemi & Renvall [2]	2	12	2.5
Stiglic [3]	2	8	2
Mizuki & Sone [4]	2	6	1

2. カードベースプロトコルの例

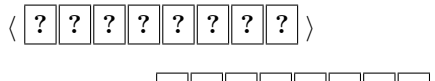
コミット型 AND プロトコルの一例として，ここでは Stiglic のプロトコル [3] を紹介する．

表 1 に示したように，このプロトコルは 2 色 8 枚のカードを使い，2 回の平均試行を要する．具体的には次のとおりである．Alice と Bob は，a と b のコミットメントをそれぞれ入力し，4 枚の追加カード ♣ ♣ ♡ ♡ を用いて次を実行する．

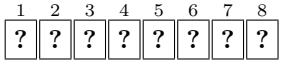
(1) 次のようにカードを並べる．



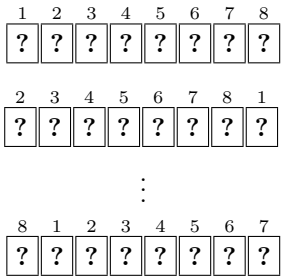
(2) ランダムカットを適用する．



ランダムカットは，巡回的なシャッフルを意味し，8 枚の裏になったカード

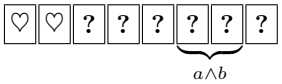


が与えられた場合，1/8 の確率で次の 8 通りのいずれかの状態になる．

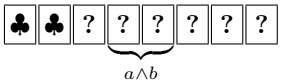


(3) 左端 2 枚をめくる．

(a) ♡ ♡ なら

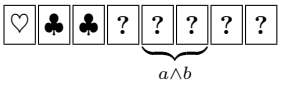


(b) ♣ ♣ なら

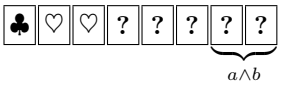


(c) ♣ ♡ あるいは ♡ ♣ なら左から 3 枚目をめくる

(i) ♡ ♣ ♣ なら



(ii) ♣ ♡ ♡ なら



(iii) ♣ ♡ ♣ あるいは ♡ ♣ ♡ なら
表になっているカードを裏返し（2）へ戻る

このプロトコルを $\mathcal{P}_{\mathrm{Sti}}$ と書くことにする．$\mathcal{P}_{\mathrm{Sti}}$ のステップ（2）において，ランダムカットというシャッフル操作を使用している．ステップ（3）(c)（iii）が起きて（2）へ戻る確率は $\frac{1}{2}$ であるので，平均試行回数は 2 回となる．なお，ランダムカットは，安全性を確保したまま人間の手で容易に実装することが可能であることが知られている [6]．

このプロトコルからもわかるように，カードベースプロトコルで使用される操作は，次のように定義されている [5]．

(1)「1 枚めくる操作」
d 枚のカード列 $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ 中の i 番目のカード α_i をめくる操作：

$$(\mathsf{turn}, i)$$

(2)「置換操作」
d 枚のカード列 $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ に対し，置換 $\pi \in S_d$（S_d は d 次の対称群を示す）を適用し，カード列 $(\alpha_{\pi^{-1}(1)}, \alpha_{\pi^{-1}(2)}, \ldots, \alpha_{\pi^{-1}(d)})$ を得る操作：

$$(\mathsf{perm},\ \pi)$$

(3)「シャッフル操作」

d 枚のカード列 $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ に対し，置換集合 $\Pi \subseteq S_d$ から確率分布 $\mathcal{F}$ に従い得られる置換 $\pi \in \Pi$ を適用する操作：

$$(\mathsf{shuffle},\ \Pi,\ \mathcal{F})$$

3. 実行時間を与える性能指標

前節で見たように，カードベースプロトコルでは turn，perm，shuffle という操作が用いられる．プロトコルの実行時間を評価するためには，これらをすべて考慮する必要がある．しかし，表 1 のような性能評価では情報が欠けており，全体の実行時間を見積もることは難しいことがすぐに分かるであろう．

本節では，表 1 のそれぞれのプロトコルにおいて各操作がどれくらい発生するか詳細に調べる．

3.1 プロトコルで使用される操作

2 節で紹介した turn，perm，shuffle という 3 つの操作に加え，本稿では 1 枚のカードを表面を上にして追加するという操作を考え，これを place と表現することにする．従って，カードベースプロトコルは全 4 操作で構成されることになる．

以降ではプロトコルを詳細に解析していくが，その際 KWH-tree と呼ばれる状態遷移図を使用することで，各操作による状態遷移及びその遷移確率を容易に把握することができる．実際に 3.2 節でその解析を行う．

3.2 各プロトコルにおける操作回数の解析

ここではまず，2 節で紹介したプロトコル $\mathcal{P}_{\mathrm{Sti}}$ を詳しく解析する．$\mathcal{P}_{\mathrm{Sti}}$ の KWH-tree は図 1 のように書ける．これに基づき，$\mathcal{P}_{\mathrm{Sti}}$ の各操作をカウントすると以下のようになる．

(1) $\mathcal{P}_{\mathrm{Sti}}$ におけるカードを追加する回数（place）

4 枚の追加カードを使用しているため，place 操作の回数は 4 となる．

(2) $\mathcal{P}_{\mathrm{Sti}}$ におけるめくる操作の回数（turn）

（1）で追加した 4 枚のカードの表面を確認してから裏返すため，turn を 4 回操作する．また，初めのランダムカット後の $(\mathsf{turn}, \{1,2\})$ で 2 回の turn を行う．ここで ♣♣ か ♡♡ が出てプロトコルが終了する確率は $\frac{1}{8} \times 2$ である．一方，ここでプロトコルが終了せずに $(\mathsf{turn}, \{3\})$ のフェーズでプロトコルが終了する確率は $\frac{3}{8} \times \frac{1}{3} \times 2$ である．$(\mathsf{turn}, \{3\})$ でもプロトコルが終了しない場合，表にした 3 枚のカードを再度裏返し，ランダムカットを実行して再び $(\mathsf{turn}, \{1,2\})$ が行われる．

これらを考慮し，turn 操作の回数の期待値を求めると，

$$4 + \sum_{n=1}^{\infty}(12n-7) \times \frac{1}{4} \times \left(\frac{1}{2}\right)^{n-1} = 12.5$$

となる．

(3) $\mathcal{P}_{\mathrm{Sti}}$ における置換操作の回数（perm）

$\mathcal{P}_{\mathrm{Sti}}$ においては置換操作が行われないため，perm 操作の回数は 0 となる．

(4) $\mathcal{P}_{\mathrm{Sti}}$ におけるシャッフル操作の回数（shuffle）

（2）と同様に $(\mathsf{turn}, \{1,2\})$ でプロトコルが終了する場合の確率は $\frac{1}{4}$，$(\mathsf{turn}, \{3\})$ でプロトコルが終了する場合の確率は $\frac{1}{4}$，$(\mathsf{turn}, \{3\})$ でもプロトコルが終了せずにループが発生する場合の確率が $\frac{1}{2}$ である．従って，上記全ての場合を考慮して shuffle 操作の回数の期待値を求めると以下のようになる．

$$\sum_{n=1}^{\infty} n \times \frac{1}{2} \times \left(\frac{1}{2}\right)^{n-1} = 2$$

以上をまとめると，$\mathcal{P}_{\mathrm{Sti}}$ における各操作の回数は表 2 の下から 2 行目の通りとなる．

表 2 コミット型 AND プロトコルにおける各操作数

	place 数	turn 数	perm 数	shuffle 数
$\mathcal{P}_{\mathrm{CK}}$ [1]	6	21	1	8
$\mathcal{P}_{\mathrm{NR}}$ [2]	8	28	4.5	7.5
$\mathcal{P}_{\mathrm{Sti}}$ [3]	4	12.5	0	2
$\mathcal{P}_{\mathrm{MS}}$ [4]	2	4	2	1

$\mathcal{P}_{\mathrm{MS}}$ については，過去の文献（例えば [7]）で KWH-tree が記述されているので，ここでは $\mathcal{P}_{\mathrm{CK}}$ と $\mathcal{P}_{\mathrm{NR}}$ の KWH-tree を図 2 と図 3 に記す．これらの KWH-tree を用いると，同様にして $\mathcal{P}_{\mathrm{CK}}$，$\mathcal{P}_{\mathrm{NR}}$，$\mathcal{P}_{\mathrm{MS}}$ の各操作をカウントすることができ，その結果は表 2 に示す通りである．

3.3 プロトコルの実行時間

ここでは，各プロトコルの実行時間の記述方法を与える．

まず，カードベースプロトコルを構成する 4 操作 $\mathsf{turn}, \mathsf{perm}, \mathsf{shuffle}, \mathsf{place}$ それぞれの単体の所要時間を t_{turn}，t_{perm}，t_{shuf}，t_{place} というパラメータで表すことにする．このとき，プロトコル $\mathcal{P}$ の全体の実行時間を $\mathrm{Time}(\mathcal{P})$ と書くことにすると，表 2 のデータを基に下記のようになる．

(1) Crépeau & Kilian のプロトコル $\mathcal{P}_{\mathrm{CK}}$

$\mathrm{Time}(\mathcal{P}_{\mathrm{CK}}) = 6t_{\mathrm{place}} + 21t_{\mathrm{turn}} + t_{\mathrm{perm}} + 8t_{\mathrm{shuf}}$

(2) Niemi & Renvall のプロトコル $\mathcal{P}_{\mathrm{NR}}$

$\mathrm{Time}(\mathcal{P}_{\mathrm{NR}}) = 8t_{\mathrm{place}} + 28t_{\mathrm{turn}} + 4.5t_{\mathrm{perm}} + 7.5t_{\mathrm{shuf}}$

(3) Stiglic のプロトコル $\mathcal{P}_{\mathrm{Sti}}$

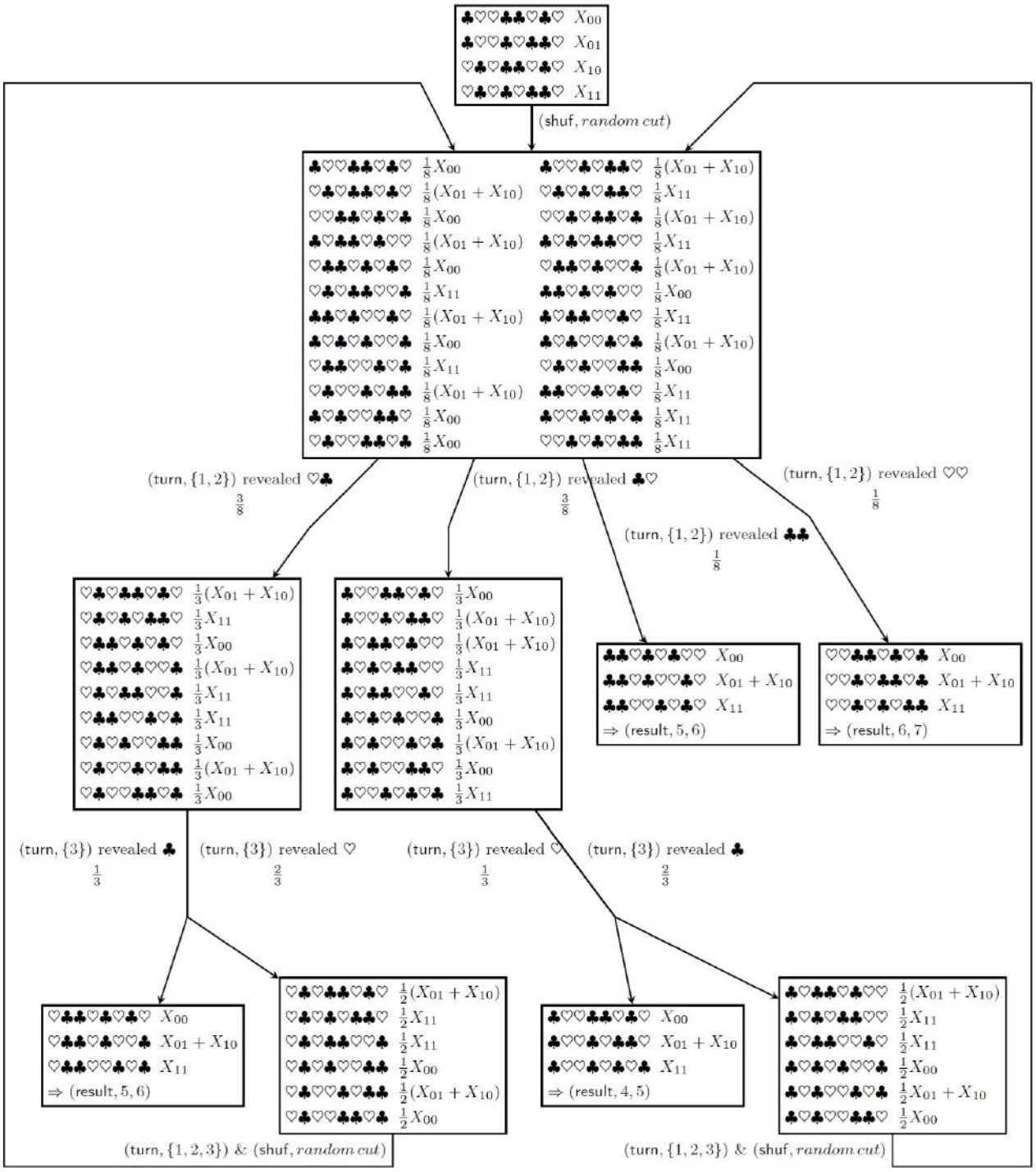


図 1 $\mathcal{P}_{\mathrm{Sti}}$ の KWH-tree

$\mathrm{Time}(\mathcal{P}_{\mathrm{Sti}}) = 4t_{\mathrm{place}} + 12.5t_{\mathrm{turn}} + 2t_{\mathrm{shuf}}$

(4) Mizuki & Sone のプロトコル $\mathcal{P}_{\mathrm{MS}}$

$\mathrm{Time}(\mathcal{P}_{\mathrm{MS}}) = 2t_{\mathrm{place}} + 4t_{\mathrm{turn}} + 2t_{\mathrm{perm}} + t_{\mathrm{shuf}}$

4. 提案指標に基づく性能評価

本節では，表 2 あるいは 3.3 節の結果に基づき，表 1 のプロトコルの性能比較を行う．

4.1 実行時間を基にしたプロトコルの性能比較

ここでは，各プロトコルの実行時間の大小を比較する．

まず，3.3 節で示した実行時間を与える式の各係数を比較すると，

$$\mathrm{Time}(\mathcal{P}_{\mathrm{Sti}}) < \mathrm{Time}(\mathcal{P}_{\mathrm{CK}})$$

$$\mathrm{Time}(\mathcal{P}_{\mathrm{Sti}}) < \mathrm{Time}(\mathcal{P}_{\mathrm{NR}})$$

は無条件に成立することがすぐに分かる．

従って，$\mathrm{Time}(\mathcal{P}_{\mathrm{Sti}})$ と $\mathrm{Time}(\mathcal{P}_{\mathrm{MS}})$ の大小関係を比較す

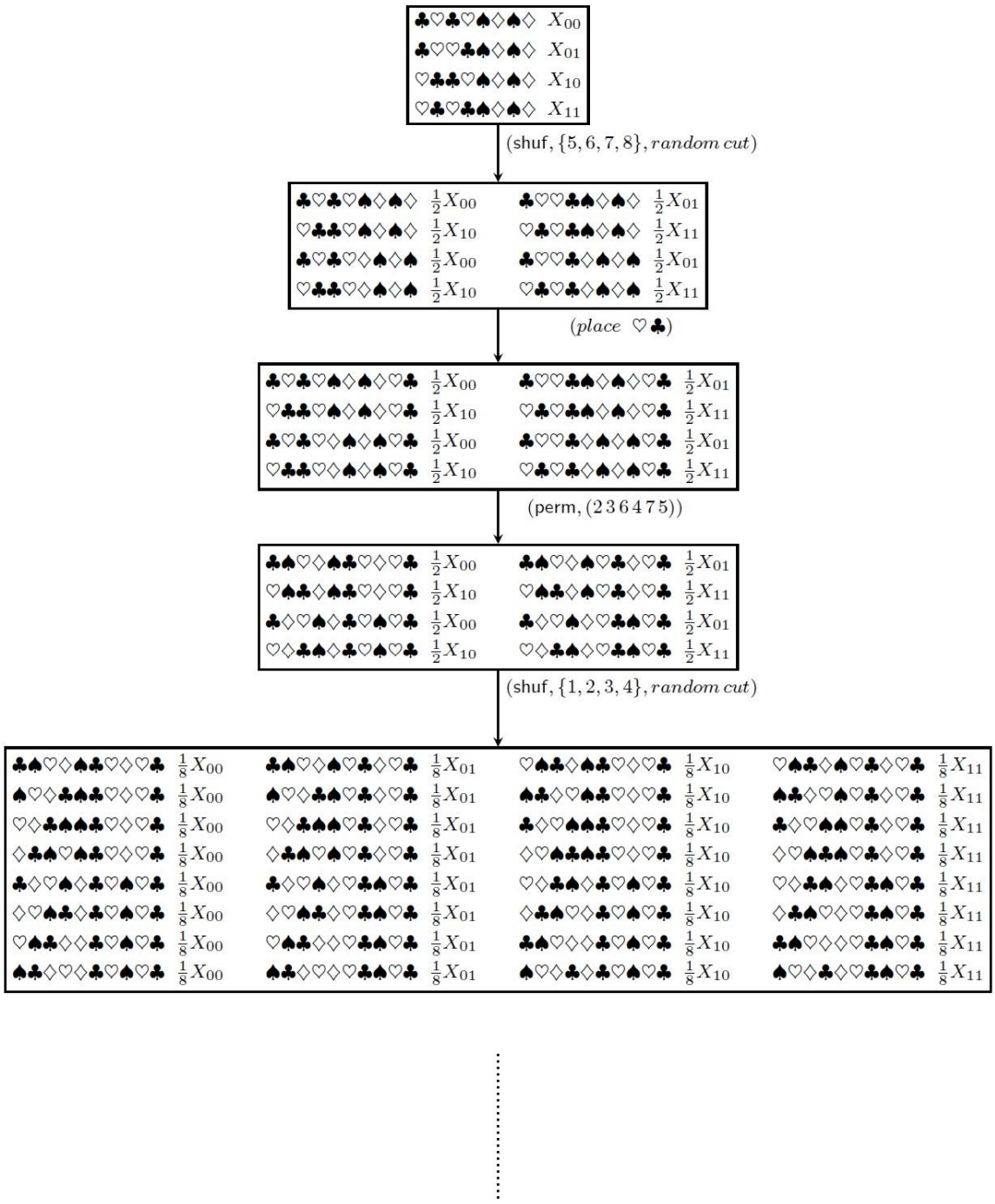


図 2 $\mathcal{P}_{\mathrm{CK}}$ の KWH-tree の一部

ることで，表 1 のプロトコルで最も実行時間が小さいと見積もられるものを決定したい．

$t_{\mathrm{place}}, t_{\mathrm{turn}}, t_{\mathrm{perm}}$ に関して，著者によって実際にカード組を操作して，所要時間を計測したところ，大要 $t_{\mathrm{place}} = t_{\mathrm{turn}}$ かつ $0.1t_{\mathrm{perm}} < t_{\mathrm{turn}}$ という関係が得られた．また，シャッフル操作は置換操作よりも所要時間が大きいと自然に考えられるため，$t_{\mathrm{perm}} < t_{\mathrm{shuf}}$ と表せる．以上の関係式から $\mathrm{Time}(\mathcal{P}_{\mathrm{Sti}})$ と $\mathrm{Time}(\mathcal{P}_{\mathrm{MS}})$ を比較すると，

$$\begin{aligned}\mathrm{Time}(\mathcal{P}_{\mathrm{MS}}) &= 2t_{\mathrm{place}} + 4t_{\mathrm{turn}} + 2t_{\mathrm{perm}} + t_{\mathrm{shuf}} \\ &< 2t_{\mathrm{place}} + 14t_{\mathrm{turn}} + t_{\mathrm{perm}} + t_{\mathrm{shuf}} \\ &< 4t_{\mathrm{place}} + 12.5t_{\mathrm{turn}} + 2t_{\mathrm{shuf}} = \mathrm{Time}(\mathcal{P}_{\mathrm{Sti}})\end{aligned}$$

となり，$\mathrm{Time}(\mathcal{P}_{\mathrm{MS}}) < \mathrm{Time}(\mathcal{P}_{\mathrm{Sti}})$ となることが確認できる．

従って，$\mathcal{P}_{\mathrm{MS}}$ が最も実行時間の小さいプロトコルであると言える．

4.2 シャッフルの実行時間による評価

4.1 節において $t_{\mathrm{perm}} < t_{\mathrm{shuf}}$ と仮定していたが，ここで

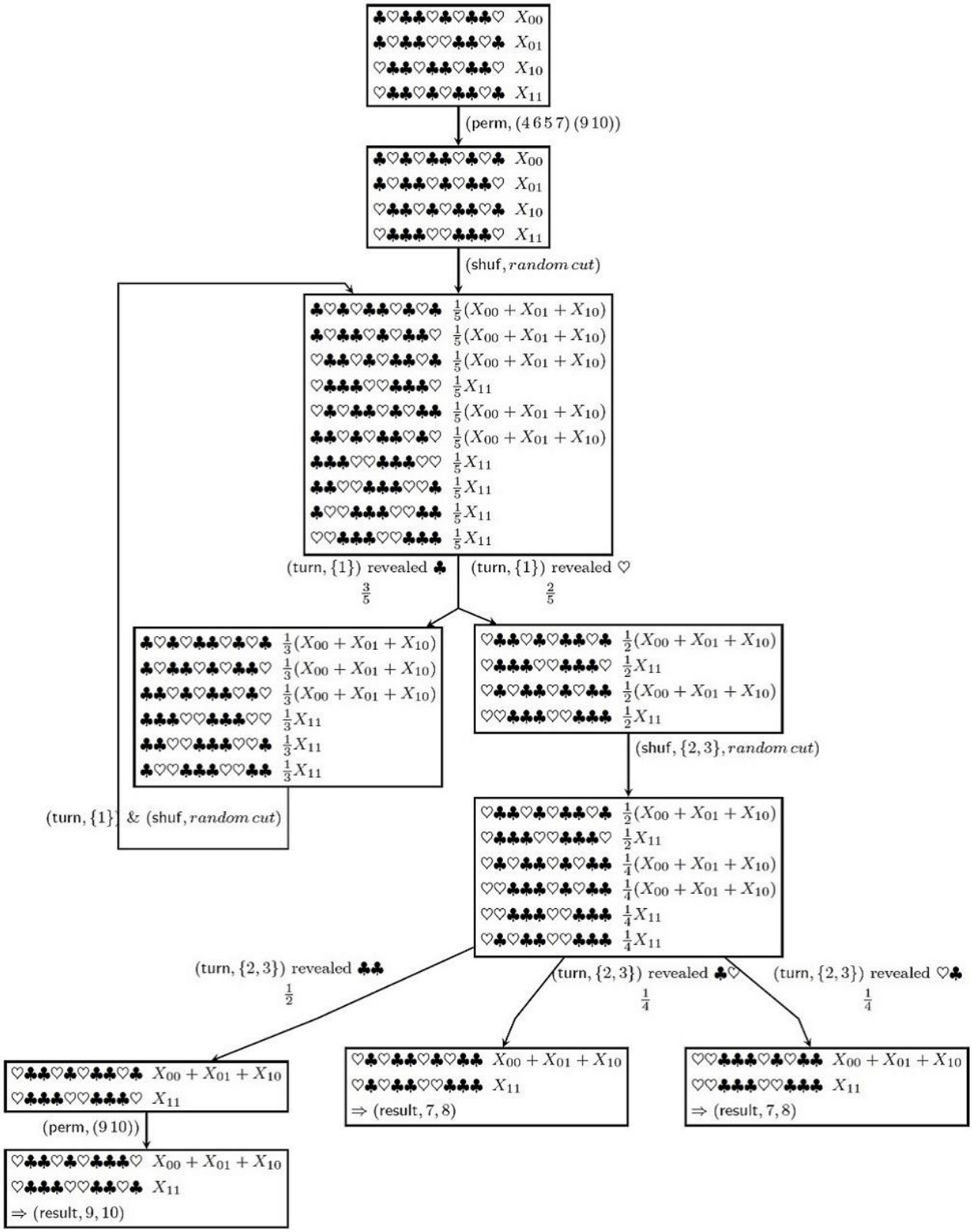


図 3 $\mathcal{P}_{\mathrm{NR}}$ の KWH-tree

はシャッフルの操作時間 t_{shuf} を変化させ，各プロトコルの実行時間の変化を可視化する．t_{shuf} 以外のパラメータは，実際の所要時間を基に

$$t_{\mathrm{place}} = t_{\mathrm{turn}} = 0.8, \quad t_{\mathrm{perm}} = 7t_{\mathrm{turn}}$$

とし，t_{shuf} を 3 秒から 60 秒まで変化させると，図 4 のようになる．

この図から，$\mathcal{P}_{\mathrm{CK}}, \mathcal{P}_{\mathrm{NR}}$ と $\mathcal{P}_{\mathrm{Sti}}, \mathcal{P}_{\mathrm{MS}}$ を比較すると，明らかに $\mathcal{P}_{\mathrm{Sti}}$，$\mathcal{P}_{\mathrm{MS}}$ 方が効率が良いことが見て取れる．

また，シャッフルの実行時間が 3 秒以下，すなわち 4.1

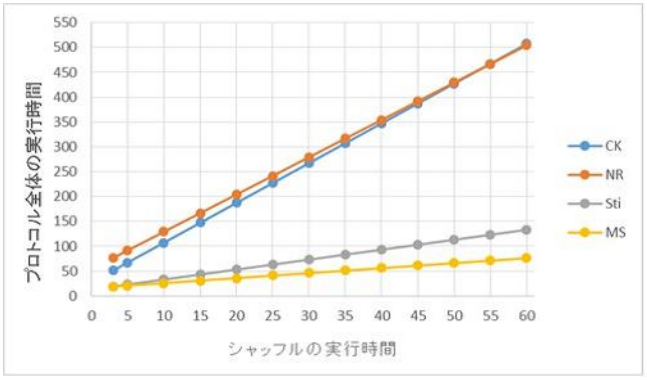


図 4　シャッフル時間を変化させた際の各プロトコルにおける実行時間の比較

節の $t_{\mathrm{perm}} < t_{\mathrm{shuf}}$ という前提を満たさない場合，$\mathcal{P}_{\mathrm{Sti}}$ の方が効率の良いプロトコルと言える．しかしながら，通常の人間はせいぜい約 1 秒にカードデッキを 1 回カットする程度が限界である．つまり，3 秒間ではカットの回数は 2，3 回程度しか行うことができないことになり，十分にカードがランダムになっているか疑問である．更に，2，3 回程度のカットであればその操作を見ているプレイヤーあるいは第三者はシャッフルを見破ることができる可能性がある．従って，$t_{\mathrm{shuf}} < t_{\mathrm{perm}}$ となる場合はシャッフルにおける安全性に疑問が残る結果となり，$\mathrm{Time}(\mathcal{P}_{\mathrm{MS}}) < \mathrm{Time}(\mathcal{P}_{\mathrm{Sti}})$ としてよいと言える．

加えて，$\mathcal{P}_{\mathrm{CK}}$ と $\mathcal{P}_{\mathrm{NR}}$ を見てみると，約 60 秒後にその実行時間の大小が入れ替わる．しかし，こちらも現実でプロトコルを実行することを考慮した場合，操作者が約 60 秒もシャッフルを実行することはあまり現実的とは言えない．従って，$\mathrm{Time}(\mathcal{P}_{\mathrm{CK}}) < \mathrm{Time}(\mathcal{P}_{\mathrm{NR}})$ としてよいと言える．

シャッフルの実行時間と安全性の関係については，付録 A.1 でもう少し考察を加える．

5. おわりに

カードベースプロトコルの既存の性能評価パラメータでは，プロトコルの操作数が考慮されていなかったり，実際の実行時間を適切に見積もることが困難であった．そこで本稿ではその操作数と操作時間を算出し，その新たなパラメータを基にプロトコルの性能評価を行う手法を提案した．加えて，提案手法を用いて既存のコミット型 AND プロトコルのいくつかに対して実際に評価を行った．

なお，本稿では入力が a，b の 2 ビットの場合を想定していたが，n ビット入力論理積を秘密計算したい場合，AND プロトコルを並列に実行することで，合計 $\lceil \log_2 n \rceil$ 回分の実行時間で n 入力の論理積のコミットメントを得ることができると言える．例えば，$\mathcal{P}_{\mathrm{MS}}$ を 7 ビット入力 $(a, b, c, d, e, f, g \in \{0, 1\})$ で実行することを例に挙げると，1 段階目は $a \wedge b$，$c \wedge d$，$e \wedge f$ を並列実行する．続いて，2 段階目で $(a \wedge b) \wedge (c \wedge d)$，$(e \wedge f) \wedge g$ を並列実行．最後に 3 段階目で $(a \wedge b \wedge c \wedge d) \wedge (e \wedge f \wedge g)$ を実行する．このようにして $\mathcal{P}_{\mathrm{MS}}$ による 7 ビット全体の実行時間を算出すると，$3\,\mathrm{Time}(\mathcal{P}_{\mathrm{MS}})$ となる．

今後の課題として，他の既存プロトコルへの適用や，各操作（特にシャッフル操作）の所要時間パラメータの詳細化が挙げられる．

参考文献

[1] Crépeau and Kilian, “Discreet solitary games,” CRYPTO ’93, LNCS, vol.773, pp.319330, 1994.
[2] V. Niemi and A. Renvall, “Secure multiparty computations without computers,” Theoretical Computer Science, vol.191, no.12, pp.173183, 1998.
[3] A. Stiglic, “Computations with a deck of cards,” Theoretical Computer Science, vol.259, no.12, pp.671678, 2001.
[4] Mizuki and Sone, “Six-card secure AND and four-card secure XOR,” FAW2009, LNCS, vol.5598, pp.358369, 2009.
[5] Mizuki and Shizuya, “A formalization of card-based crypto- graphic protocols via abstract machine,” International Journal of Information Security, vol.13, no.1, pp.1523, 2014.
[6] I.ueda, et al. “How to implement a random bisection cut,” TPNC 2016, Springer International Publishing, Proceedings 5, pp.58-69, 2016.
[7] T.Mizuki, and H.Shizuya. “Computational Model of Card-Based Cryptographic Protocols and Its Applications,” IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences 100.1 (2017): 3-11.

付　　録

A.1 ランダムカットの安全性評価に対するアイディア

これまで本稿で述べてきた，カードベースプロトコルにおいて使用される 4 操作の内，既存のカードベースプロトコルの研究の中で最も関心を集める事柄の一つとして，シャッフル操作の安全性がある．すなわち，相手プレイヤーあるいは第三者にその結果が見破られることなく実行できるシャッフルの種類や実装方法はあるのかという問題である．

その問題に対し，既存研究 [6] では，ヒンズーカットという実装を用いたランダムカットが安全性を確保したまま人間が容易に実行できるシャッフルであるということが示されている．その際，ヒンズーカットを用いたランダムカットの安全性に対する実験的評価が行われているが，ここではヒンズーカットのモデル化を行い，より詳細かつ定量的に評価を行うアイディアを示す．

ヒンズーカットは整えたカードデッキのカードを下から上に移動させるカットであり，図 A·1〜図 A·3 のように 3 つのフェーズに分解することができる．このそれぞれの

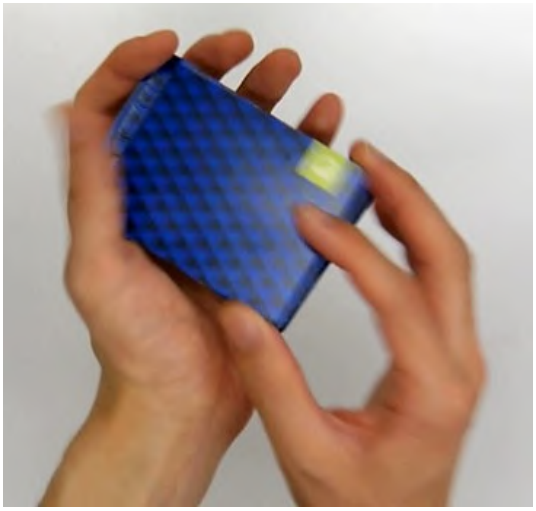


図 A·1 “つかむ” フェーズ

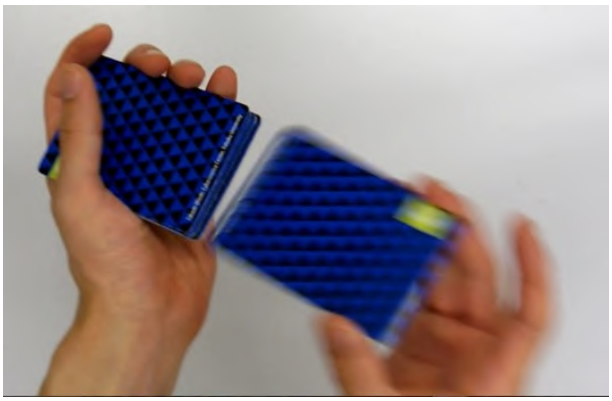


図 A·2 “引く” フェーズ

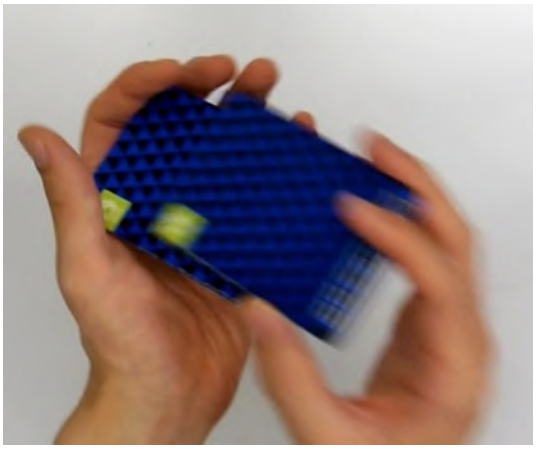


図 A·3 “のせる” フェーズ

フェーズを “つかむ”，“引く”，“のせる” フェーズと呼ぶことにする．

A.1.1 “つかむ” フェーズ

このフェーズにおいては，以下の安全性パラメータが考えられる．

(1) デッキの中でつかむカードの位置と枚数
n 枚中上から $x \in \{2, \ldots, n\}$ 枚目をつかむ．この時 n が偶数でかつ $x = \frac{n}{2}$ の時はランダム二等分割カット [4] であると言える．

(2) 使用しているカードの厚さ
カードが厚ければ厚いほどカット時につかみやすいと考えられるが，その場合相手プレイヤー及び第三者が横から見た際にシャッフルを見破りやすくなってしまうと考えられる．

A.1.2 “引く” フェーズ

このフェーズにおいては，以下の安全性パラメータが考えられる．

(1) 見えるカードの面積
カードの面積が大きければ大きいほどそのカードを引くときに時間がかかり，シャッフルが見破られやすくなると考えられる．

(2) カードをデッキから引くときのカードのばらつき度
カードデッキが整っていない状態で無造作にカットを行って，一目でカットしたカードの枚数が認識できればシャッフルが見破られやすくなると考えられる．また，ヒンズーカットはこのばらつき度が低いため安全性が高いと考えられる．

(3) カードのデザイン
カードのデザインが錯視を利用して，カット時の枚数が見づらかったり，カードの色がまわりの環境と同化するような色の場合はシャッフルを見破られにくくなると考えられる．

A.1.3 “のせる” フェーズ

このフェーズにおいては，以下の安全性パラメータが考えられる．

(1) “引く” フェーズから “のせる” フェーズが終了するまでの時間
カードを移動させる時間が長ければ長いほど，カットされている枚数が見やすくなると考えられる．

(2) デッキにカードをのせる際のばらつき度
デッキにカードをのせる際に，カードがばらついていれば一目でカットしたカードの枚数がわかり，シャッフルが見破られやすくなると考えられる．

今後の課題として，これらのパラメーター一つ一つを検証し，どのパラメータがどの程度安全性に影響するのか調査することが挙げられる．