最小カード枚数の7入力対称論理関数秘密計算

河又 彰吾[1,a)]　水木 敬明[1,2]

概要：カードベース暗号の中で最も重要な問題の一つに，対称論理関数に対してコミット型プロトコルを構築することが挙げられる．2015 年に Nishida らは，任意の n 入力対称論理関数に対して $2n+2$ 枚のカードでプロトコルを構成できることを示した．入力を表すのに $2n$ 枚のカードが必要であるため，追加で 2 枚のカードが必要ということになる．この追加カードを不要にできるかという問題はしばらく未解決であったが，2022 年に Shikata らは，$n \geq 8$ の場合に任意の n 入力対称論理関数に対して $2n$ 枚のプロトコルを構成した（$2n$ 枚は最小であり，従って追加カードを不要とした）．また，$n = 2$ の場合は，既存の 2 入力 AND と XOR プロトコルにより解決しているため，$3 \leq n \leq 7$ の場合に対してのこの問題が未解決である．本稿では，$n = 7$ の場合の問題を解決する．すなわち，任意の 7 入力対称論理関数に対して 14 枚（すなわち最小枚数）のプロトコルを構成する．上述の Nishida らや Shikata らのプロトコルでは必要なシャッフル回数が示されてこなかったが，本稿で提案するプロトコルではそのシャッフル回数を解析し明記する．

キーワード：カードベース暗号，秘密計算，対称論理関数

Secure Computation of Seven-Input Symmetric Boolean Functions with the Minimum Number of Cards

Shogo Kawamata[1,a)]　Takaaki Mizuki[1,2]

Abstract: One of the most important problems in card-based cryptography is the construction of committed-format protocols for symmetric Boolean functions. In 2015, Nishida et al. demonstrated that a protocol can be constructed using $2n+2$ cards for any n-input symmetric Boolean function. Since $2n$ cards are required to represent the input, two additional cards are needed. The question of whether these additional cards could be eliminated remained unsolved for several years, but in 2022, Shikata et al. constructed a protocol using $2n$ cards for any n-input symmetric Boolean function when $n \geq 8$ (where $2n$ is the minimum number of cards required, thereby eliminating the need for additional cards). Additionally, for $n = 2$, the problem is solved using existing two-input AND and XOR protocols, leaving the problem unsolved for $3 \leq n \leq 7$. In this paper, we solve the case of $n = 7$. Specifically, we construct a protocol using 14 cards (i.e., the minimum number of cards) for any seven-input symmetric Boolean function. While the numbers of required shuffles for the protocols proposed by Nishida et al. and Shikata et al. were not specified, the protocol proposed in this paper is analyzed so that we explicitly state the number of shuffles.

Keywords: Card-based cryptography, Secure computation, Symmetric Boolean functions

1. はじめに

カードベース暗号とは，物理的なカード組を用いて入力を符号化し，めくる，シャッフル，並べ替え等の操作を実行して所望の暗号機能を実現するものである．

カードベース暗号では，典型的に🂣と🂱の 2 色カード

[1] 東北大学大学院情報科学研究科，〒 980–8578 宮城県仙台市青葉区荒巻字青葉 6–3,
Graduate School of Information Sciences, Tohoku University, 6—3 Aramaki-Aza-Aoba, Aoba-ku, Sendai 980-–8578, Japan
[2] 東北大学サイバーサイエンスセンター，〒 980–8578 宮城県仙台市青葉区荒巻字青葉 6–3
Cyberscience Center, Tohoku University, 6–3 Aramaki-Aza-Aoba, Aoba-ku, Sendai 980–8578, Japan
[a)] kawamata.shogo.p3@dc.tohoku.ac.jp

を用いる[i]．前提として，裏面はすべて同一（本稿では ? と表現する）であり，同じ種類のカードはそれぞれ区別がつかないものとする．

1.1 符号化と秘密計算プロトコル

カードベース暗号ではビット値を扱うために，典型的に，

$$\clubsuit\heartsuit = 0, \quad \heartsuit\clubsuit = 1 \tag{1}$$

のように符号化する．ビット $x \in \{0,1\}$ が式 (1) に従って，2 枚のカードとして裏返しに置かれるとき，この 2 枚のカードを x の**コミットメント**と呼び，次のように表現する．

$$\underbrace{??}_{x}$$

秘密計算を実現する**カードベースプロトコル**は典型的に，エンコードルール (1) に従って入力を符号化し，めくる，シャッフル，並べ替え等の操作を実行し，所望の出力を得る．1 入力当たり 2 枚のカード組で構成されるため，n 入力の論理関数の秘密計算には少なくとも $2n$ 枚のカードを要する．カードベース暗号では，コミットメントで出力が得られるプロトコルを**コミット型**，コミットメント以外で出力が得られるプロトコルを**非コミット型**と呼ぶ．本稿ではコミット型プロトコルのみを扱う．

1.2 対称関数と既存研究

一般に秘密計算では，対称関数を対象としてプロトコルを構成することが多い．（対称関数とは入力の順番を入れ替えても出力の変わらない関数のことである．）カードベース暗号おいても，対称論理関数のクラスに属する関数が研究対象になることが多い．

$f : \{0,1\}^n \to \{0,1\}$ を n 入力対称論理関数とすると，

$$f(x_1, x_2, \ldots, x_n)$$

の値は合計値 $\sum_{i=1}^{n} x_i$ のみに依存するという性質がある．すなわち，ある関数 $g : \{0,1,\ldots,n\} \to \{0,1\}$ が存在して，

$$f(x_1, x_2, \ldots, x_n) = g\left(\sum_{i=1}^{n} x_i\right) \tag{2}$$

となる．

2015 年に Nishida ら [4] は，任意の n 入力対称論理関数 $f : \{0,1\}^n \to \{0,1\}$ に対して，n 個のコミットメントと 2 枚の追加カードから $f(x_1,\ldots,x_n)$ のコミットメントを出力する汎用的なプロトコルを構成した．

$$\underbrace{??}_{x_1}\underbrace{??}_{x_2}\cdots\underbrace{??}_{x_n}\clubsuit\heartsuit \to \cdots \to \underbrace{??}_{f}$$

[i] 数字カード [1]，トランプカード [2]，トライアングルカード [3] を用いるプロトコルなども存在する．

今述べた Nishida らのプロトコルは追加カード $\clubsuit\heartsuit$ を必要とするが，$n = 2$ の場合については，既存の（2 入力）AND プロトコル [5] および XOR プロトコル [6] により，2 枚の追加カードを不要にできる．すなわち，任意の 2 入力対称論理関数 $f : \{0,1\}^2 \to \{0,1\}$ に対して次のようなプロトコルが構成できる．

$$\underbrace{??}_{x_1}\underbrace{??}_{x_2} \to \cdots \to \underbrace{??}_{f}$$

$n \geq 3$ の場合について，追加カードを不要できるかどうかについては，しばらくの間未解決問題として残されていたが，2022 年に Shikata ら [7] は $n \geq 8$ の場合に対して肯定的に解決した．すなわち，$n \geq 8$ なる任意の n 入力対称論理関数 $f : \{0,1\}^n \to \{0,1\}$ に対して，n 個のコミットメントだけから $f(x_1,\ldots,x_n)$ のコミットメントを出力する汎用的なプロトコルを構成した．

$$\underbrace{??}_{x_1}\underbrace{??}_{x_2}\cdots\underbrace{??}_{x_n} \to \cdots \to \underbrace{??}_{f}$$

従って，いまだ解決されていないのは，$3 \leq n \leq 7$ の場合である．

1.3 本稿の貢献

本稿では，前述の未解決問題に取り組み，$n = 7$ の場合を解決する．すなわち，任意の 7 入力対称論理関数 $f : \{0,1\}^7 \to \{0,1\}$ に対して，7 個のコミットメントだけから $f(x_1, x_2, \ldots, x_7)$ のコミットメントを出力するプロトコルを構成する．

$$\underbrace{??}_{x_1}\underbrace{??}_{x_2}\cdots\underbrace{??}_{x_7} \to \cdots \to \underbrace{??}_{f}$$

提案プロトコルの基本的な流れは Shikata らの $n \geq 8$ のプロトコル [7] を踏襲する．まず，7 個のコミットメント

$$\underbrace{??}_{x_1}\underbrace{??}_{x_2}\underbrace{??}_{x_3}\underbrace{??}_{x_4}\underbrace{??}_{x_5}\underbrace{??}_{x_6}\underbrace{??}_{x_7}$$

を入力として，$\sum_{i=1}^{7} x_i$ の値に対応する 3 つのコミットメントからなる列を得る．

$$\underbrace{??????}_{(\sum_{i=1}^{7} x_i)_2}$$

（これは二進数の整数をコミットメントの列として表現していることを意味する．$(\sum_{i=1}^{7} x_i)_2$ は 3 ビットであることに注意しよう．）次に式 (2) のような $g\left(\sum_{i=1}^{n} x_i\right)$ のコミットメントを出力する．

$$\underbrace{??}_{g(\sum_{i=1}^{n} x_i)}$$

これが求めたかった $f(x_1, x_2, \ldots, x_7)$ のコミットメントその

ものとなる．

以上のように Shikata らのプロトコルと大枠は同じであるが，もちろんそのままでは $n = 7$ の場合には対応できず，本稿では，「整数エンコードの色変換 [8]」という手法を適用することで，$n = 7$ の場合に対応させる．さらに，提案プロトコルでは，$\sum_{i=1}^{7} x_i$ の値に対応する 3 つのコミットメントを得たあとの手順を関数ごとに精密に設計し，全体で必要なシャッフル回数を明示する．

なお，既存の対称論理関数に対する汎用プロトコルでは，これまで厳密なシャッフル回数が求められておらず，この提案プロトコルが（$n = 7$ という限定されたものではあるが）一般的な対称論理関数のプロトコルの厳密なシャッフル回数を与える初めての取り組みと言える．

1.4 関連研究

本稿ではコミット型プロトコルのみを扱うが，もし非コミット型を許すならば，$n = 3$ の場合は Ruangwises [9] の取り組みによって解決している．すなわち，すべての 3 入力対称論理関数に対して 6 枚のカードで非コミット型プロトコルが構成できることを示している．さらに，（非コミット型の前提で）$n = 4, 5, 6, 7$ において，部分的に解を与えている（すなわち，一部の関数に対して $2n$ 枚のプロトコルを与えている）．

対称論理関数に対する関連研究として，Ruangwises と Itoh [10] は，任意の値域を仮定した非コミット型プロトコルを 2 枚の追加カードで設計している．また，関数のクラスを限定することで，追加カードを減らせることが知られている [8, 11]．さらに，使えるカードの色を増やすことで，追加カードを減らせることが知られている [12]．

2. 準備

本節では，準備として，提案プロトコルの説明に必要な用語などを与える．

2.1 パイルスクランブルシャッフル

パイルスクランブルシャッフル [13] は，同じ枚数の複数のカード束を構成し，それらの束の順番を一様ランダムに入れ替えるシャッフルである（各束の中のカードの順番は変えない）．カード列にパイルスクランブルシャッフルを適用するとき，$[\cdot|\cdots|\cdot]$ のように表記する．

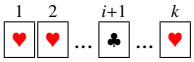
また，特に 2 分割の場合は，**ランダム二等分割カット** [6] と呼ばれる．

2.2 パイルシフティングシャッフル

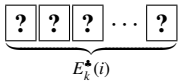
パイルシフティングシャッフル [14, 15] は，カード列をいくつかの束に分け，それらを束の単位でランダムに巡回的な並び替えを行うシャッフルである．

2.3 整数エンコーディング

$k \geq 2$ とし，カードを用いて整数 $i \in \{0, 1, \ldots, k-1\}$ を直接表現することを考えよう．Ruangwises と Itoh [10] は，1 枚の ♣ と $k-1$ 枚の ♥ を用い，$i+1$ 番目に ♣ を置くことで整数 i を表現した．



このようなカード列が裏になっているものを**整数エンコーディング**と呼び，$E_k^{\clubsuit}(i)$ と表す．



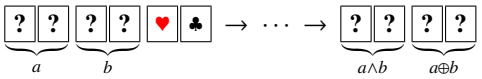
♣ と ♥ を逆にして定義されるものは $E_k^{\heartsuit}(i)$ と書く．

3. 既存のサブプロトコル

本稿で提案するプロトコルは，いくつかの既存のプロトコルをサブプロトコルとして利用するため，本節ではそれらについて説明する．

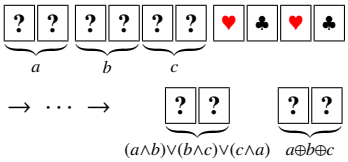
3.1 半加算器プロトコルと全加算器プロトコル

最初のカードベースの半加算器プロトコルは 2013 年に与えられ [16]，その後 Nishida ら [4] によって改良され，2 枚の追加カードを用いるものが提案された．



Nishida らの半加算器プロトコルのシャッフル回数はランダム二等分割カット 2 回で実行できる．

また，全加算器は 4 枚の追加カードを用いたものが 2013 年に与えられている [16]．



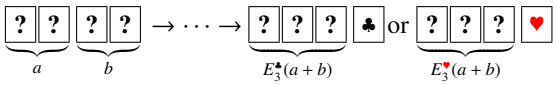
全加算器プロトコルのシャッフル回数は，ランダム二等分割カット 5 回で実行できる．

3.2 整数エンコーディングの加算

Ruangwises と Itoh [10] は，2 つの整数エンコーディング $E_k^{\clubsuit}(a)$ と $E_k^{\heartsuit}(b)$ が与えらえたとき，それらの加算を行い $E_k^{\clubsuit}(a+b)$ を得る手法を提案した．このときのシャッフル回数はパイルシフティングシャッフル 1 回のみである．

3.3 2 つのコミットメントの加算

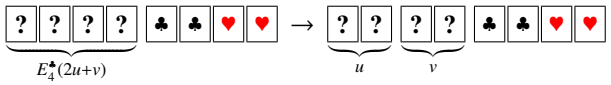
Shikata ら [7] は，$a, b \in \{0, 1\}$ のコミットメントが与えられたときに，追加カードを用いることなく加算 $E_3^{\clubsuit}(a+b)$ あるいは $E_3^{\heartsuit}(a+b)$ が得られることを示した．



すなわち，$E_3^{\clubsuit}(a+b)$ あるいは $E_3^{\heartsuit}(a+b)$（どちらになるかは 1/2 の確率）と，フリーカードを 1 枚得ることができる．このプロトコルはランダム二等分割カット 2 回を用いる．

3.4 整数エンコーディングの二進数化

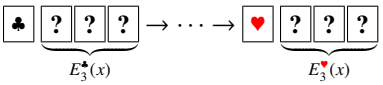
Shikata ら [7] は，4 枚の追加カードを用いて，$E_4^{\clubsuit}(2u+v)$ を二進数化する手法を提案した．



このときのシャッフル回数はランダム二等分割カット 2 回である．

3.5 整数エンコーディングの色変換プロトコル

Shikata ら [8] は，追加カード 1 枚を用いて整数エンコーディングの色を変換するプロトコルを提案した．

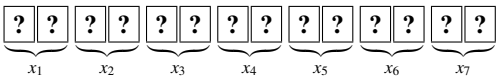


プロトコルの詳細は省略するが，2 回のランダム二等分割カットを行ったのちカードをめくり，望んだ色が出るまでステップを繰り返す．従って，このプロトコルのシャッフル回数の期待値は 4 である．

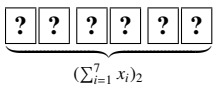
4. 提案プロトコルの前半

本節では，提案プロトコルの前半の手順を与える．

$f:\{0,1\}^7 \to \{0,1\}$ を任意の 7 入力対称論理関数とする．提案プロトコルの前半では，



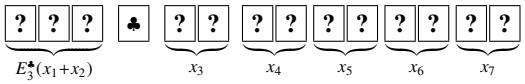
を入力として，



を出力する．このことを次の通りに実現する．

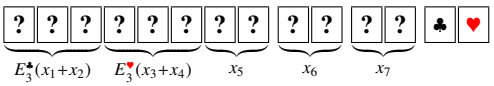
4.1 $x_1 + x_2$ の計算

まず，3.3 節で説明した「2 つのコミットメントの加算」を用いて，x_1 と x_2 のコミットメントを加算する．一般性を失うことなく，加算結果は $E_3^{\clubsuit}(x_1+x_2)$ が得られたとする．このとき，フリーカードは $\boxed{\clubsuit}$ である．まとめると，現在の状態は次のようになっている．



4.2 $x_3 + x_4$ の計算

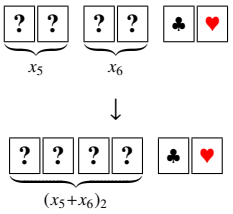
次に，同様にして，x_3 と x_4 のコミットメントを加算する．もし，$E_3^{\clubsuit}(x_3+x_4)$ が得られた場合は 3.5 節で説明した色変換を行う．これにより，いずれにしろ $E_3^{\heartsuit}(x_3+x_4)$ が得られ，$\heartsuit$ のフリーカードを得る．



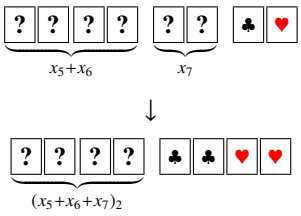
4.3 $x_5 + x_6 + x_7$ の計算

いま 2 色のフリーカードが得られているので，これらを用いて次を行う．

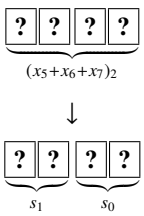
(1) 3.1 節で説明した半加算器プロトコルを x_5 と x_6 のコミットメントに適用する．



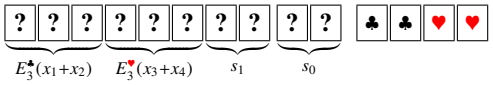
(2) 同様に $x_5 + x_6$ と x_7 のコミットメントに対して，半加算器プロトコルを適用する．



以下，$(x_5 + x_6 + x_7)$ の上位ビットを s_1, 下位ビットを s_0 と書く．



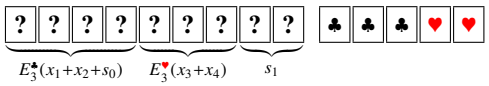
現在の状態をまとめると以下のようになっている．



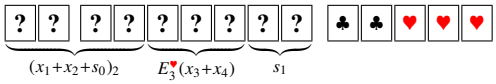
4.4 $(x_1 + x_2)$ と s_0 の加算と二進数化

s_0 のコミットメントに 1 枚のフリーカードを加え整数エ

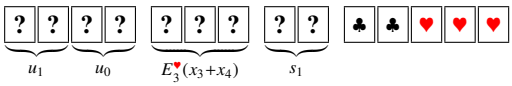
ンコーディングとしたのち，$E_3^{\clubsuit}(x_1+x_2)$ と加算する．



$E_3^{\clubsuit}(x_1+x_2+s_0)$ を 3.4 節で説明した方法で二進数化する．

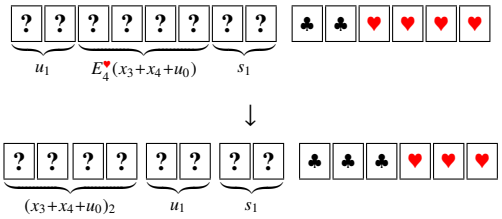


以下，$(x_1+x_2+s_0)_2$ を上位ビットを u_1 下位ビットを u_0 と書く．



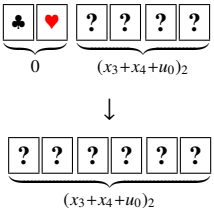
4.5 x_3+x_4 と u_0 の加算と二進数化

4.4 節と同様に x_3+x_4 と u_0 の加算と二進数化を行う．

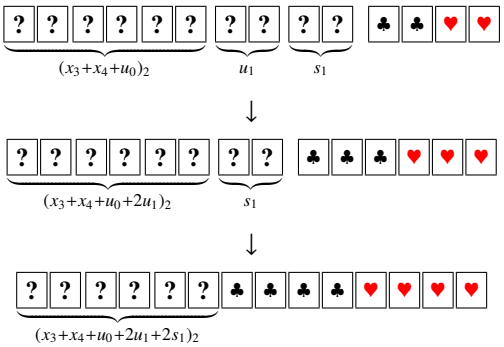


4.6 u_1 と $x_3+x_4+u_0$ と s_1 の加算

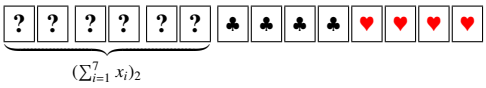
最後に u_1 と $x_3+x_4+u_0$ と s_1 の加算を次のように行う．まず，フリーカードの ♣ と ♥ を 1 枚ずつ用いて，$x_3+x_4+u_0$ の左端に 0 のコミットメントを追加する．（すなわち，2^2 のビットを追加する．）



これに対し $x_3+x_4+u_0$ に s_1 と u_1 を加算したい．ここで，s_1 と u_1 はもともと二進数表現の 2^1 のビットの数であることに注意しなければならない．これらを $x_3+x_4+u_0$ に加算するときは，$x_3+x_4+u_0$ の 2^1 のビットに 3.1 節の半加算器で加算する．



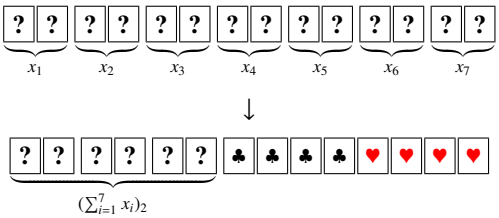
$x_3+x_4+u_0+2u_1+2s_1=\sum_{i=1}^{7}(x_i)$ であるので，



が得られた．

5. 提案プロトコルの後半

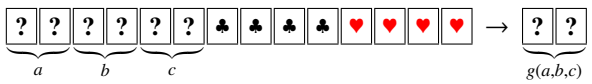
$f:\{0,1\}^7\to\{0,1\}$ を任意の 7 入力対称論理関数とし，$g:\{0,1,\ldots,7\}\to\{0,1\}$ を式 (2) を満たす関数とする．4 節で与えた提案プロトコルの前半により，



が得られる．本節では，このカード列から

を得る，提案プロトコルの後半の手順を与える．

以下では読み易さのため，g は $g:\{0,1\}^3\to\{0,1\}$ のように 3 ビット入力とみなし，記号の置き換えを行い，次を実現するプロトコルを（任意の g に対して）構成することで，提案プロトコルの後半の完成とする．



5.1 NPN 同値類

任意の 3 入力論理関数 g は，$2^{2^3}=256$ 通り存在し，これら全てを列挙するのは煩わしい．一方，ある関数に対してプロトコルが構成できるとき，入力変数の否定 (Negation of inputs)，入力変数の置換 (Permutation of inputs)，出力の否定 (Negation of outout) により得られる関数に対してもまた，プロトコルが構成できることがすぐに分かる[ii]．このような 3 つの操作により関数の集合の上の同値関係が作られ，得られる各同値類は *NPN* **同値類**と呼ばれる [17,18]．3 入力論理関数については 14 種類の同値類に分割できることが知られている．従って，そのような 14 種類の（代表元の）関数に対してプロトコルを定めることにより，任意の 3 入力論理関数に対するプロトコルを定めることができる．

プロトコルの構築を容易にする理由により，NPN 同値類の代表元の表示に次の get 演算を用いる．

$$\mathrm{get}^0(x,y)=x,\ \mathrm{get}^1(x,y)=y$$

[ii] NOT 演算は，コミットメントを構成する 2 枚を入れ替えることで簡単に実現できることに注意しよう．

14 個の同値類の代表元の名称[iii]と，それらの論理式を表 1 の 1 列目と 2 列目に記す．

5.2 プロトコルの構成：自明あるいは既知なケース

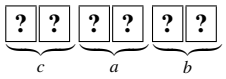
まず，Const，Val については，プロトコルは自明であり，シャッフルを要しない．

次に，Or，Xor については，既存の OR や XOR プロトコル [6] により，シャッフル 1 回で実現できる．And3，Xor3，OrAnd，XorAnd，AndXor についても，既存プロトコルの 2 回の実行により，シャッフル 2 回で実現できる．

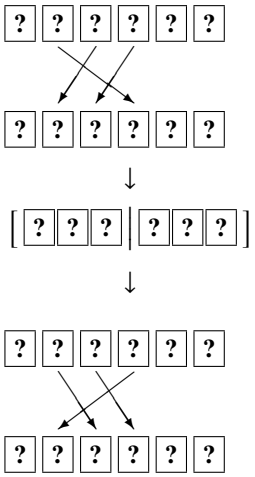
5.3 プロトコルの構成：get 演算の利用

Mux = $\mathrm{get}^c(a,b)$ については，次のプロトコルが対応する（これは [6] の AND プロトコルの一般化である）．

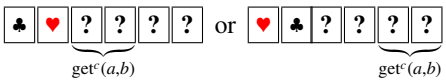
(1) カード列を次のように並べる．



(2) 次のように並び替え，ランダム二等分割カットを行う．



(3) 左から 1 枚目と 2 枚目のカードをめくる．めくられたカード列の並びに応じて次のようにして $\mathrm{get}^c(a,b)$ のコミットメントが得られる．



5.4 プロトコルの構成：入力保存 AND の利用

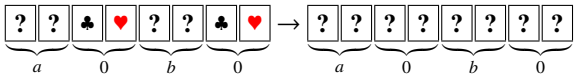
Gamble，Onehot，Majority，Dot については，以下に示す Ishikawa ら [13] によって示された入力保存 AND プロトコルおよびその応用プロトコルを活用する．

(1) 2 つの入力コミットメントと 4 枚の追加カードを次のように置き，追加カードを裏返す．このとき，追加カードは 0 のコミットメントとなる．

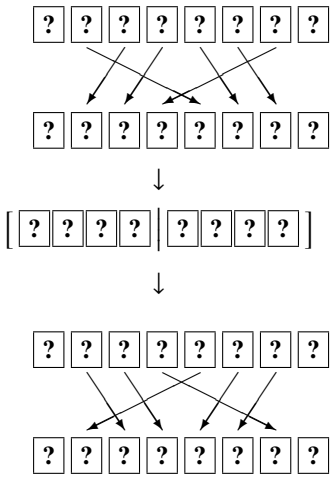
[iii] 大部分は文献 [18] で用いられている名称を使っている．

表 1: NPN 同値類の代表元とシャッフル回数

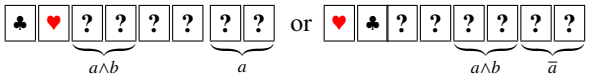
代表元名称	論理式	シャッフル回数
Const	0	0
Val	a	0
Or	$a \vee b$	1
Xor	$a \oplus b$	1
And3	$a \wedge b \wedge c$	2
Gamble	$\mathrm{get}^{a\oplus b}(\overline{a \oplus c}, 0)$	2
Onehot	$\mathrm{get}^{a\oplus b\oplus c}(0, \overline{a \wedge c})$	3
Xor3	$a \oplus b \oplus c$	2
Majority	$\mathrm{get}^{a\oplus b}(a, c)$	2
OrAnd	$a \wedge (b \vee c)$	2
Dot	$(a \vee b) \wedge (a \oplus c)$	2
Mux	$\mathrm{get}^c(a, b)$	1
XorAnd	$a \wedge (b \oplus c)$	2
AndXor	$a \oplus (b \wedge c)$	2



(2) 次のように並び替え，ランダム二等分割カットを行う．



(3) 左から 1 枚目と 2 枚目のカードをめくる．めくられたカード列の並びに応じて次のようにして $a \wedge b$ および a のコピーのコミットメントが得られる．



上記の方法により，コピーと AND を 1 回のシャッフルで並列に実行できる．また, この AND やコピーは OR や XOR に置き換えられる．

Gamble では，Ishikawa らのプロトコルを用いて，

(1) $a \oplus b$ の XOR プロトコルと $a \oplus c$ の XOR プロトコルを，Ishikawa らの手法で並列実行する．

(2) $\mathrm{get}^{a \oplus b}(\overline{a \oplus c}, 0)$ の get 演算プロトコルを実行する．

というプロセスで，シャッフル回数 2 回で実現できる．

Onehot では，Ishikawa らのプロトコルを用いて

(1) a のコピープロトコルと $a \oplus b$ の XOR プロトコルを，Ishikawa らの手法で並列実行する．

(2) $(a \oplus b) \oplus c$ の XOR プロトコルと $a \wedge c$ の AND プロトコルを，Ishikawa らの手法で並列実行する．

(3) $\mathrm{get}^{a \oplus b \oplus c}(0, \overline{a \wedge c})$ の get 演算プロトコルを実行する．

というプロセスで，シャッフル回数 3 回で実行できる．

同様にして Majority では a のコピープロトコルと $a \oplus b$ の XOR プロトコルを Ishikawa らの手法で並列実行したのち get プロトコルを行うというプロセスで，シャッフル回数 2 回で実行できる．Dot は $a \vee b$ と $a \oplus c$ を Ishikawa らの手法で並列実行したのちに AND プロトコルを実行することで，シャッフル回数 2 回で実行できる．

上記のように入力保存 AND プロトコルおよびその応用プロトコルを活用することにより，3 入力論理関数は表 1 で示したシャッフル回数で実現することができる．

6. 提案プロトコルのシャッフル回数

本節では，提案プロトコルのシャッフル回数を，各ステップごとに求める．

(1) $x_1 + x_2$ の計算

$x_1 + x_2$ の計算では，2.3 節の半加算器を 1 回実行する．そのためシャッフル回数は 2 回である．

(2) $x_3 + x_4$ の計算

$x_3 + x_4$ の計算では，2.3 節の半加算器を 1 回実行し，1/2 の確率で色変換プロトコルが実行される．半加算器のシャッフル回数は 2 回であり，色変換プロトコルのシャッフル回数の期待値は 4 回であるため，このステップのシャッフルの期待値は 4 回である．

(3) $x_5 + x_6 + x_7$ の計算

$x_5 + x_6 + x_7$ の計算では，2.3 節の半加算器を 2 回実行する．$x_5 + x_6$ の計算でのシャッフル回数は 2 回であり，$(x_5 + x_6)$ と x_7 の加算は，下位ビットの半加算器で 2 回，上位ビットと繰り上がりの OR 計算による 1 回であるため，このステップのシャッフル回数は 5 回である．

(4) $(x_1 + x_2)$ と s_0 の加算と二進数化

$(x_1 + x_2)$ と s_0 の加算と二進数化では，3.2 節の加算を 1 回実行し，その後 3.4 節に示した二進数化を 1 回実行する．3.2 節の加算のシャッフル回数は 1 回であり，二進数化のシャッフル回数は 2 回であるため，このステップのシャッフル回数は 3 回である．

(5) $x_3 + x_4$ と u_0 の加算と二進数化

$x_3 + x_4$ と u_0 の加算と二進数化では，(4) と同様に 3.2 節の加算を 1 回実行し，その後 3.4 節に示した二進数化を 1 回実行する．(4) と同様，このステップのシャッフル回数は 3 回である．

(6) u_1 と $x_3 + x_4 + u_0$ と s_1 の加算

u_1 と $x_3 + x_4 + u_0$ と s_1 の加算では，2.3 節の半加算器を 2 回実行する．半加算器のシャッフル回数は 2 回であり，その後繰り上がりによる OR 計算を 1 回行う．このシャッフル 3 回の操作を 2 回行うため，このステップのシャッフル回数は 6 回である．

(7) 3 入力論理関数の計算

3 入力論理関数の計算では，5 節で示したプロトコルに従ってシャッフルを実行する．このステップのシャッフル回数は 0 ~ 3 回である．

上記のステップによりシャッフル回数が求められる．これをまとめたのが表 2 である．

表 2: 7 入力対称論理関数のシャッフル回数

	操作	シャッフル回数
1	$x_1 + x_2$ の計算	2
2	$x_3 + x_4$ の計算	4（期待値）
3	$x_5 + x_6 + x_7$ の計算	5
4	$x_1 + x_2$ と s_0 の加算と二進数化	3
5	$x_3 + x_4$ と u_0 の加算と二進数化	3
6	u_1 と $x_3 + x_4 + u_0$ と s_1 の加算	6
7	3 入力論理関数の計算	0 ~ 3
	合計	23 ~ 26

7. おわりに

本稿では，任意の 7 入力対称論理関数 $f : \{0,1\}^7 \to \{0,1\}$ に対して，14 枚のカードでコミット型プロトコルを構成できることを示した．また，その際に必要なシャッフル回数を詳細に求めた．

この結果により，未解決問題として残されているのは，$3 \le n \le 6$ なる任意の n 入力対称論理関数 $f : \{0,1\}^n \to \{0,1\}$ に対して $2n$ 枚のコミット型プロトコルが構成できるか，となる．この範囲の関数すべてについてコミット型プロトコルが知られていないわけではなく，一部の関数は $2n$ 枚のコミット型プロトコルが構成されている（例えば，n-XOR など）．詳しくは，文献 [9] を参照されたい．

謝辞

本稿の提案プロトコルの骨子は，東北大学在籍時の四方隼人氏との議論により得られた成果を基にしている．本研究の一部は JSPS 科研費 JP23H00479 と JP24K02938 の助成を受けている．

参考文献

[1] T. Sasaki, T. Mizuki, and H. Sone, “Card-based zero-knowledge proof for Sudoku,” Fun with Algorithms, eds. by H. Ito, S. Leonardi, L. Pagli, and G. Prencipe, vol.100, pp.29:1–29:10, LIPIcs, Schloss Dagstuhl, Dagstuhl, Germany, 2018. https://doi.org/10.4230/LIPIcs.FUN.2018.29

[2] T. Mizuki, “Efficient and secure multiparty computations using a standard deck of playing cards,” Cryptology and Network Security, eds. by S. Foresti and G. Persiano, vol.10052, pp.484–499, LNCS, Springer, Cham, 2016. https://doi.org/10.1007/978-3-319-48965-0_29

[3] K. Shinagawa and T. Mizuki, “Card-based protocols using triangle cards,” Fun with Algorithms, eds. by H. Ito, S. Leonardi, L. Pagli, and G. Prencipe, vol.100, pp.31:1–31:13, LIPIcs, Schloss Dagstuhl, Dagstuhl, Germany, 2018. https://doi.org/10.4230/LIPIcs.FUN.2018.31

[4] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone, “Card-based protocols for any Boolean function,” Theory and Applications of Models of Computation, eds. by R. Jain, S. Jain, and F. Stephan, vol.9076, pp.110–121, LNCS, Springer, Cham, 2015. https://doi.org/10.1007/978-3-319-17142-5_11

[5] A. Koch, S. Walzer, and K. Härtel, “Card-based cryptographic protocols using a minimal number of cards,” Advances in Cryptology—ASIACRYPT 2015, eds. by T. Iwata and J.H. Cheon, vol.9452, pp.783–807, LNCS, Springer, Berlin, Heidelberg, 2015. https://doi.org/10.1007/978-3-662-48797-6_32

[6] T. Mizuki and H. Sone, “Six-card secure AND and four-card secure XOR,” Frontiers in Algorithmics, eds. by X. Deng, J.E. Hopcroft, and J. Xue, vol.5598, pp.358–369, LNCS, Springer, Berlin, Heidelberg, 2009. https://doi.org/10.1007/978-3-642-02270-8_36

[7] H. Shikata, K. Toyoda, D. Miyahara, and T. Mizuki, “Card-minimal protocols for symmetric Boolean functions of more than seven inputs,” Theoretical Aspects of Computing – ICTAC 2022, eds. by H. Seidl, Z. Liu, and C.S. Pasareanu, vol.13572, pp.388–406, LNCS, Springer, Cham, 2022. https://doi.org/10.1007/978-3-031-17715-6_25

[8] H. Shikata, D. Miyahara, and T. Mizuki, “Few-helping-card protocols for some wider class of symmetric Boolean functions with arbitrary ranges,” ACM ASIA Public-Key Cryptography Workshop, pp.33–41, ACM, New York, 2023. https://doi.org/10.1145/3591866.3593073

[9] S. Ruangwises, “The landscape of computing symmetric n-variable functions with 2n cards,” Theoretical Aspects of Computing – ICTAC 2023, eds. by E. Ábrahám, C. Dubslaff, and S.L.T. Tarifa, vol.14446, pp.74–82, LNCS, Springer, Cham, 2023. https://doi.org/10.1007/978-3-031-47963-2_6

[10] S. Ruangwises and T. Itoh, “Securely computing the n-variable equality function with $2n$ cards,” Theor. Comput. Sci., vol.887, pp.99–110, 2021. https://doi.org/10.1016/j.tcs.2021.07.007

[11] S. Ikeda, Y. Takahashi, K. Shinagawa, and K. Nuida, “Efficient card-based protocols for symmetric and partially doubly symmetric functions,” Advances in Information and Computer Security, pp.??–??, LNCS, Springer, Singapore, 2025.

[12] Y. Takahashi, K. Shinagawa, H. Shikata, and T. Mizuki, “Efficient card-based protocols for symmetric functions using four-colored decks,” ACM ASIA Public-Key Cryptography Workshop, pp.1–10, ACM, New York, 2024. https://doi.org/10.1145/3659467.3659902

[13] 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, LNCS, Springer, Cham, 2015. https://doi.org/10.1007/978-3-319-21819-9_16

[14] K. Shinagawa, T. Mizuki, J. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto, “Card-based protocols using regular polygon cards,” IEICE Trans. Fundam., vol.E100.A, no.9, pp.1900–1909, 2017. https://doi.org/10.1587/transfun.E100.A.1900

[15] A. Nishimura, Y. Hayashi, T. Mizuki, and H. Sone, “Pile-shifting scramble for card-based protocols,” IEICE Trans. Fundam., vol.101, no.9, pp.1494–1502, 2018. https://doi.org/10.1587/transfun.E101.A.1494

[16] T. Mizuki, I.K. Asiedu, and H. Sone, “Voting with a logarithmic number of cards,” Unconventional Computation and Natural Computation, eds. by G. Mauri, A. Dennunzio, L. Manzoni, and A.E. Porreca, vol.7956, pp.162–173, LNCS, Springer, Berlin, Heidelberg, 2013. https://doi.org/10.1007/978-3-642-39074-6_16

[17] T. Sasao, Switching theory for logic synthesis, Boston: Kluwer Academic Publishers, 1999.

[18] D.S. Marakkalage, E. Testa, H. Riener, A. Mishchenko, M. Soeken, and G. De Micheli, “Three-input gates for logic synthesis,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.40, no.10, pp.2184–2188, 2021.