Vol.2022-DPS-190 No.34
Vol.2022-CSEC-96 No.34
2022/3/11

対称論理関数に対する最小枚数プロトコルの改良

四方　隼人[1,a)]　豊田　航大[1]　宮原　大輝[2,3]　水木　敬明[2,3]

概要：カードベース暗号の研究分野において最も重要な問題の一つに「最小枚数のカードによるプロトコルの構成」がある．すなわち，1 ビットを 2 枚のカードで符号化し，n 個のビットを示す $2n$ 枚のカード列だけが入力として与えられたときに，どのような論理関数が秘密計算できるか，という問題である．著者らは SCIS 2022 において，n 変数の対称論理関数に対する最小枚数のプロトコルを与えたが，n が 14 以上である必要があり，かつ特殊なケースの使用を必要とするという，制約の大きいものであった．そこで本稿では，これらの制約を緩和するプロトコルの構築を試みる．具体的には，本稿で提案するプロトコルは特殊なケースの使用を必要とせず，n は 8 以上であればよい．

1. はじめに

入力値を秘匿したまま出力のみを得る計算を**秘密計算** [14] と呼ぶ．専門的な知識がなくても正当性や安全性が容易に理解可能であることから，物理的なカード組を用いて秘密計算を行う**カードベース暗号**が発展してきた（サーベイの例 [2, 6, 12]）．カードベース暗号において，使用カード枚数を減らすことは重要な問題であり，著者ら [15] は SCIS 2022 において対称論理関数に対して最小の枚数で秘密計算できるプロトコルを考案した．しかし，そのプロトコルには 2 つの制約があり，本研究ではそれらの制約を取り除くことを目的とする．

1.1 カードベース暗号とは

カードベース暗号では，入力値を $\boxed{\clubsuit}$ と $\boxed{\heartsuit}$ のカード組を使って，

$$\boxed{\clubsuit}\boxed{\heartsuit} = 0, \quad \boxed{\heartsuit}\boxed{\clubsuit} = 1 \qquad (1)$$

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．

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

コミットメントを入力として秘密計算を行う一連の手順を**カードベース暗号プロトコル**という．

[1] 東北大学
Tohoku Uniersity
[2] 電気通信大学
The University of Electro-Communications
[3] 産業技術総合研究所
Institute of Advanced Industrial Science and Technology
[a)] hayato.shikata.r3@dc.tohoku.ac.jp

1.2 最小枚数のプロトコル

カードベース暗号の研究分野において最も重要なテーマの一つは「最小枚数のカードを用いたプロトコルの構成」である．符号化ルール (1) の通り，1 ビットは 2 枚のカードで符号化されるため，n 変数の論理関数 $f : \{0,1\}^n \to \{0,1\}$ に対する最小枚数のプロトコルは，$2n$ 枚のカードを用いたものを意味する．すなわち，n 入力 $x_1, x_2, \ldots, x_n \in \{0,1\}$ に対応する n 個のコミットメント

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n}$$

のみを入力として，カードをめくる操作やシャッフル等を適用したのち，目的とする関数 $f(x_1, x_2, \ldots, x_n)$ の値だけを出力するものが，最小枚数のプロトコルである．

1.3 既知の結果と本稿の貢献

著者ら [15] は n 入力の対称論理関数に対して，最小枚数，すなわち $2n$ 枚で秘密計算するプロトコルを考案した．ここで，n 入力の論理関数 $f : \{0,1\}^n \to \{0,1\}$ が**対称**であるとは，任意の i, j（$1 \le i, j \le n$）に対して

$$\begin{aligned} &f(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) \\ =&f(x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n) \end{aligned}$$

が成立することである．

しかし，著者らが考案したプロトコルには最低入力数が 14 であるという制限と特殊なカードケースを使う必要があるという制約があり，これらの解決が課題であった [15]．本稿では，この既存のプロトコルの課題を改善した，対称論理関数に対する新しいプロトコルを示す．すなわち，必要入力数を 8 入力まで削減し，特殊なカードケースの使用

Vol.2022-DPS-190 No.34
Vol.2022-CSEC-96 No.34
2022/3/11

を無くせることを示す．したがって，提案プロトコルは，$n \geq 8$ なる任意の n 変数対称論理関数を $2n$ 枚のカードで秘密計算する．

2. 準備と既存研究

本節では，対称関数の性質や，既存のプロトコルについて説明する．以下では，対象論理関数 $f : \{0,1\}^n \to \{0,1\}$ を単に対称関数と呼ぶことがある．

2.1 対称関数の計算

$f : \{0,1\}^n \to \{0,1\}$ を対称関数とする．このとき

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

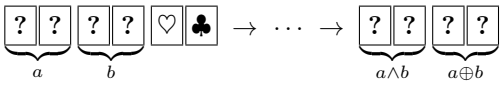
の値は合計値 $\sum_{i=1}^{n} x_i$ のみに依存するという性質がある．すなわち，ある関数 $g : \{0,1,\ldots,n\} \to \{0,1\}$ が存在して，

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

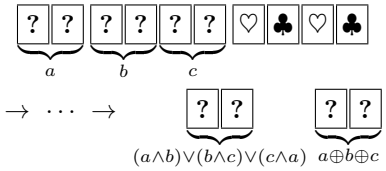
となる．このことから，対称関数 f を秘密計算したい場合には合計値 $\sum_{i=1}^{n} x_i$ を計算すれば良いことがわかる．

2.2 半加算器のプロトコルと全加算器のプロトコル

2.1 節で述べた合計値を計算するのに役に立つのが半加算器プロトコルである．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 [4]，その後 Nishida ら [8] 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．



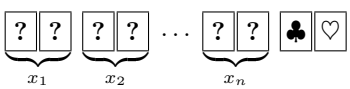
また，全加算器は 4 枚の追加カードを用いたものが 2013 年に与えられている [4]．



著者らの SCIS 2022 のプロトコル [15] ではこれらの半加算器と全加算器のプロトコルをサブプロトコルとして用いているが，本研究のプロトコルでもこれらを活用する．

2.3 追加カード 2 枚を用いた対称関数に対するプロトコル

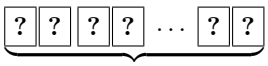
$f : \{0,1\}^n \to \{0,1\}$ を対称関数とする．Nishida ら [8] は，n 個のコミットメントと 2 枚の追加カード



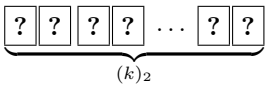
が与えられたとき，2.2 節で紹介した半加算器のプロトコルを用いて合計値

$$\sum_{i=1}^{n} x_i$$

の二進数表現を表すコミットメントの列

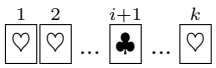


を作り，これから式 (2) のような $g(\sum_{i=1}^{n} x_i)$ を求めることで，任意の対称関数が 2 枚の追加カードで秘密計算できることを示した．ここで本稿において，非負整数 k に対して $(k)_2$ は k の二進数表現を意味し，それを $\lceil \log_2(k+1) \rceil$ 個のコミットメントで表現したものを

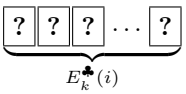


のように書いている．

上のプロトコルは対称関数の値域が $\{0,1\}$ であるが，Ruangwises と Itoh [10, 11] は任意の値域 R を持つ対称関数 $f : \{0,1\}^n \to R$ に対するプロトコルを追加カード 2 枚で構成した．彼らのプロトコルでは，合計値を二進数表現する代わりに，次のような ♣-scheme 表示や ♡-scheme 表示を用いた．♣-scheme 表示では，$k \geq 2$ とし，1 枚の ♣ カードと $k-1$ 枚の ♡ カードを用い，$i+1$ 番目に ♣ を置くことで整数 i $(0 \leq i \leq k-1)$ を表現する．



以降ではこのようなカード列が裏になっているものを $E_k^{\clubsuit}(i)$ と表す．

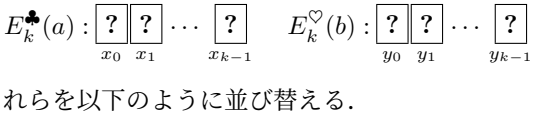


♡-scheme 表示や $E_k^{\heartsuit}(i)$ は色を逆転して同様に定義する．

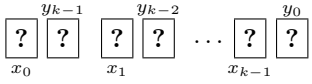
2.4 scheme 表示どうしの加算

Ruangwises と Itoh [10, 11] は scheme 表示された 2 つの整数の加算を行う手法を次のように提案した（この手法のベースは Shinagawa らのアイデア [13] となっている）．

(1) scheme 表示された二つの整数 a, b を表すカード列 $E_k^{\clubsuit}(a)$ と $E_k^{\heartsuit}(b)$ がある．便宜的に各カードに次のような名称を付ける．



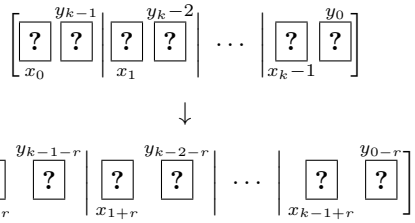
(2) これらを以下のように並び替える．



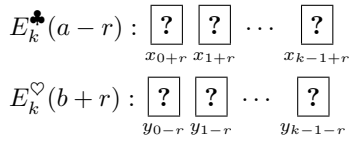
(3) このカード列に**ランダム 2-section カット**（Pile-Shifting シャッフル [9] とも呼ばれる）を適用する．

Vol.2022-DPS-190 No.34
Vol.2022-CSEC-96 No.34
2022/3/11

ここで，ランダム 2-section カットとは，2 枚のカードを 1 つの束とし，束の単位で巡回的にシャッフルする方法である．従って，カード列は r を乱数として次のように遷移する．

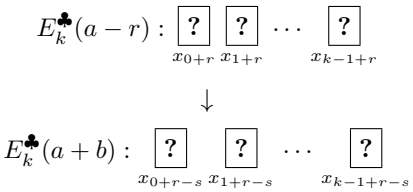


(4) これらを元のように並べ替え直す．



ここで，a にはランダムな値 r が減算され，b には r が加算されている．

(5) $E_k^{\heartsuit}(b+r)$ のカード列をめくり，その数値分 ($s=b+r$) だけ $E_k^{\clubsuit}(a-r)$ のカード組を右に巡回的にシフトする ($a-r$ に s を加算する)．$E_k^{\heartsuit}(b+r)$ のカード列をめくったとき，b にはランダムな値 r が加算されているため，b の値は漏れない．

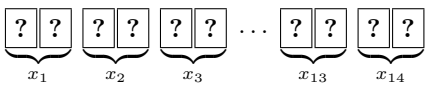


これにより，a と b の値を漏らすことなく $(a-r)+(b+r)=a+b$ を秘密計算できる．すなわち，$E_k^{\clubsuit}(a+b)$ が得られる．

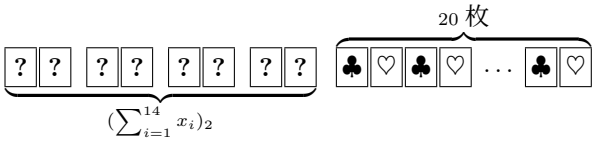
今の説明では，♣-scheme 表示と ♡-scheme 表示どうしの加算を行ったが，他の組み合わせでも問題ない．

2.5 著者らが過去に提案した対称関数計算プロトコル

著者ら [15] は任意の n 入力の対称関数 $f:\{0,1\}^n \to \{0,1\}$ に対して，最小枚数のプロトコルを構成した．ただし，そのプロトコルは $n \geq 14$ である必要があり，特殊なカードケースの使用を必要とする．すなわち，$x_1, x_2, ..., x_{14} \in \{0,1\}$ のコミットメント



を入力として，これらと特殊なカードケースを用いて，$(\sum_{i=1}^{14} x_i)_2$ のコミットメント列



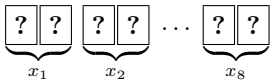
を出力するプロトコルを構築した．

次節では，このプロトコルの改良を行い，$n \geq 8$ という条件まで適用範囲を拡げ，かつ特殊ケースを使わないで新しいプロトコルを構築する．

3. 提案プロトコル

本節では，著者らが過去に提案した対称関数の最小枚数計算プロトコル [15]（以下では「SCIS 2022 のプロトコル」と呼ぶ）の課題を解決するため，プロトコルの改良を行う．この改良したプロトコルは，特殊なカードケースの使用を含まず，必要な入力数が 8 入力となっている．

以下では，説明を簡単にするために $n=8$ とする（$n \geq 9$ の場合にも容易に拡張できる）．従って，プロトコルへの入力は次のような 16 枚のカード列である．

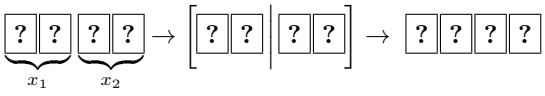


プロトコル設計の基本的な方針としては，これらの 8 つのコミットメントのビット値を加算していく．

3.1 $x_1 + x_2$ の計算

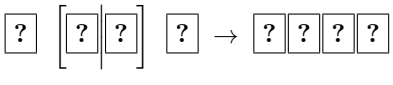
まず，先頭の 2 つのコミットメントから $x_1 + x_2$ の加算を行う．この手順は SCIS 2022 プロトコル [15] と同じである（そのベースは 2012 年に開発された 2 入力 AND プロトコル [5] である）．具体的には次のようにして，$x_1 + x_2$ の値を示す $E_3^{\clubsuit}(x_1+x_2)$ あるいは $E_3^{\heartsuit}(x_1+x_2)$ を作る．

(1) x_1 と x_2 のコミットメントに**ランダム二等分割カット** [7] を適用する．

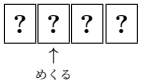


このランダム二等分割カットは，前半と後半の 2 枚ずつをそれぞれ 1 つの束として，それらをシャッフルするものである．

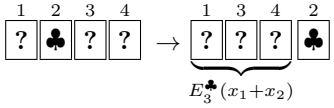
(2) 中央の 2 枚をシャッフルする．



(3) 左から 2 枚目のカードをめくり，そのカードが ♣ カードか ♡ カードかによって以下の 2 パターンの内のいずれかが 1/2 の確率で起きる．



(a) ♣ の場合，次のように 3 枚のカードを集めることで，♣-scheme 表示の加算結果を得る．



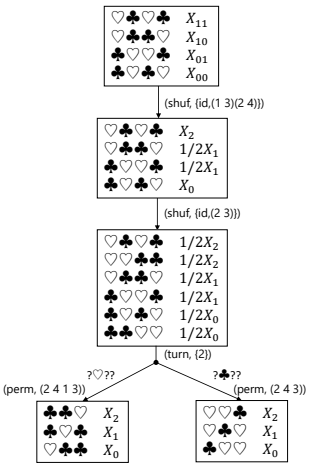
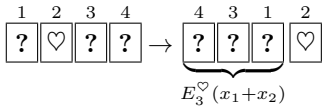


図 1: 3 枚エンコード加算の KWH 木．ただし $X_0 = X_{00}, X_1 = X_{01} + X_{10}, X_2 = X_{11}$ としている．

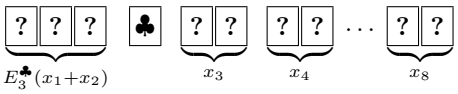
(b) ♡ の場合, 次のように並び替えることで, ♡-scheme 表示の加算結果を得る．



このプロトコルを 3 **枚エンコード加算**と呼ぶことにする．その正当性と安全性は，いわゆる KWH 木 [1,3] を描くことで証明することができる．実際にその KWH 木を図 1 に示す．

以上のように, x_1 と x_2 のコミットメントから $E_3^{\clubsuit}(x_1+x_2)$ あるいは $E_3^{\heartsuit}(x_1+x_2)$（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．

以降では説明のため，一般性を失うことなく，加算結果は ♣-scheme で得られた，すなわち $E_3^{\clubsuit}(x_1+x_2)$ が得られたとする．このとき，フリーカードは ♣ である．まとめると，一連の操作の後の状態は以下のようになっている．

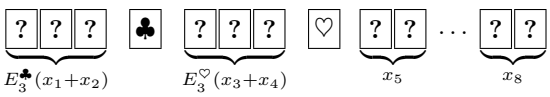


3.2 $x_3 + x_4$ の計算

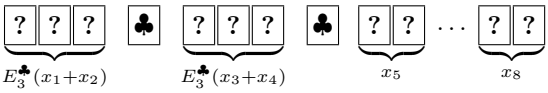
SCIS 2022 のプロトコル [15] では，$x_1 + x_2$ の加算で得られたフリーカードと異なるスートのフリーカードを得るために特殊なカードケースを使用して x_3 と x_4 の加算を行った．本プロトコルでは，特殊なカードケースの使用をせずに以下のように加算を行う．

3.1 節で説明した 3 枚エンコード加算を x_3 と x_4 のコミットメントに対して同様に適用する．3 枚エンコード加算を適用した後，次の 2 通りのうちの 1 つに遷移する．

(1) 出現するフリーカードが ♡ の場合，次のカード列が得られる．



(2) 出現するフリーカードが ♣ の場合，次のカード列が得られる．

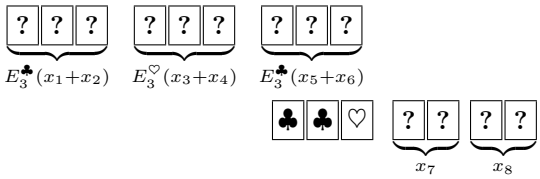


SCIS 2022 プロトコル [15] では（特殊なケースにより）この段階で既に 2 種類のフリーカード ♣ ♡ を得ているため，2.2 節で説明した半加算器のプロトコルを実行して $x_5 + x_6$ の加算を行うことができる．しかし，本稿のプロトコルでは，上記の (2) の場合のように，この段階で 1 種類のフリーカードを 2 枚得ていることがあり得る．すなわち，(1) の場合は良いが，(2) の場合は半加算器プロトコルに進めない．そこで，2.4 節で説明した加算を利用して，次のように 2 種類のフリーカードを確実に得ることを考える．

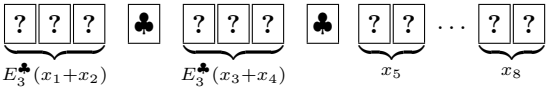
3.3 $x_5 + x_6$ の計算

3.2 節までにおいて，2 種類のフリーカードが得られている (1) の場合と，♣ のフリーカードしかなく，♡ のフリーカードがない (2) の場合の 2 つの可能性があり，それぞれ次のように加算を進める．

(1) 既に 2 種類のフリーカードが ♣ ♡ と揃っているので，$x_5 + x_6$ に対しても 3 枚エンコードの加算を行う．(♣ カードが出現するか ♡ カードが出現するかは確率的に決まるが，一般性を失うことなく ♣ が出たとする．)



(2) フリーカードが ♣ ♣ しかないため，$x_5 + x_6$ の加算を経て ♡ を作り出したい．これを実現するため 2.4 節で説明した加算プロトコルを利用し，♡ のカード生み出しつつ $x_5 + x_6$ を加算する．今の状態は以下のようになっている．



x_5 と x_6 のコミットメントは

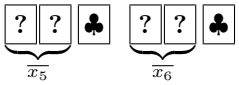
$$\clubsuit\ \heartsuit = 0, \quad \heartsuit\ \clubsuit = 1$$

で符号化されていることを思い出してほしい．これらは ♣ のカードの位置で整数値を表していると見ることができ，$E_2^{\clubsuit}(x_5)$ や $E_2^{\clubsuit}(x_6)$ と見なせる．これらのカード組の反転を考えると，これらは $\overline{x_5}$ と $\overline{x_6}$ のコミットメントであり，♡ のカードの位置で値を表しているため，$E_2^{\heartsuit}(x_5)$ や $E_2^{\heartsuit}(x_6)$ と見なせる．また，こ

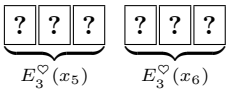
Vol.2022-DPS-190 No.34
Vol.2022-CSEC-96 No.34
2022/3/11

のとき右端に ♣ を追加しても表している値は変化せず，$E_3^{\heartsuit}(x_5)$ や $E_3^{\heartsuit}(x_6)$ となる．このことを利用して加算を行う[i]．

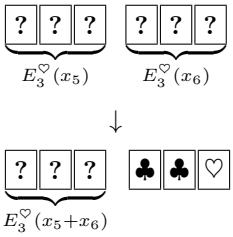
(a) x_5 と x_6 の反転の右端にフリーカードの ♣♣ を 1 枚ずつ配置する．



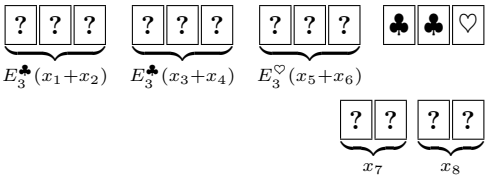
これらはそれぞれ $E_3^{\heartsuit}(x_5)$ と $E_3^{\heartsuit}(x_6)$ となる．



(b) $E_3^{\heartsuit}(x_5)$ と $E_3^{\heartsuit}(x_6)$ に対して 2.4 節で説明した加算を適用する．



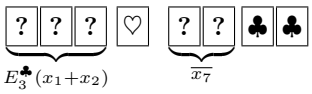
(c) 最終的に以下のようなカード列が残る．



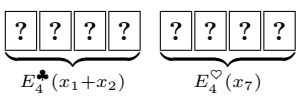
3.4 x_1+x_2 と x_7 の加算

3.3 節で説明した通り，いまフリーカードは ♣♣♡ である（一般にはどちらかの種類のカードが 2 枚，他の種類のカードが 1 枚である）．次に，$E_3^{\clubsuit}(x_1+x_2)$ と x_7 のコミットメントを用いてこれらを加算する．

(1) $E_3^{\clubsuit}(x_1+x_2)$ の右端に ♡ を，$\overline{x_7}$ の右端に ♣♣ を裏向きで配置する．

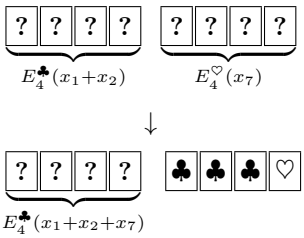


これらは x_1+x_2 の整数値が ♣ の位置で表され，x_7 の値が ♡ の位置で表されている．すなわち，それぞれ 4 枚エンコードで整数値が表されている．

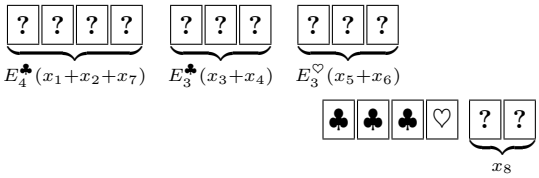


(2) 2.4 節で説明した加算をこれらに対して適用する．

[i] 同色のフリーカード 2 枚を使って加算するというアイデアは，品川和雅氏の助言による．

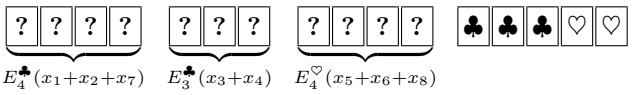


(3) 最終的に以下のようなカード列が残る．



3.5 x_5+x_6 と x_8 の加算

いま，フリーカードは ♣♣♣♡ である．これらと $E_3^{\heartsuit}(x_5+x_6)$ と x_8 のコミットメントを使って，3.4 節と同様に加算すると，以下のような状態になる．

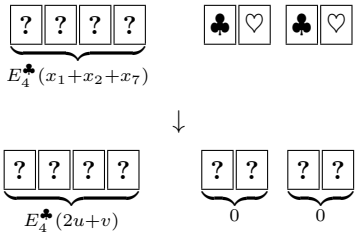


3.6 4 枚エンコードされた整数値の二進数化

SCIS 2022 のプロトコル [15] では，scheme 表示された整数値を二進数化（コミットメント化）し，加算器を適用できるようにした．ただしその際，フリーカードを 12 枚も使用し，その結果，SCIS 2022 のプロトコル [15] では多くの入力（14 入力）を必要とした．本稿では，二進数化を効率化して，必要なフリーカードの枚数を 4 枚に減らす（このことが必要入力枚数の削減に大きく貢献している）．

3.5 節までの手順で，異なるスートのフリーカードがそれぞれ 2 枚以上得られている．これらを用いて 4 枚エンコードされた $E_4^{\clubsuit}(x_1+x_2+x_7)$ と $E_4^{\heartsuit}(x_5+x_6+x_8)$ を二進数化する．まずは，$E_4^{\clubsuit}(x_1+x_2+x_7)$ を二進数化する．

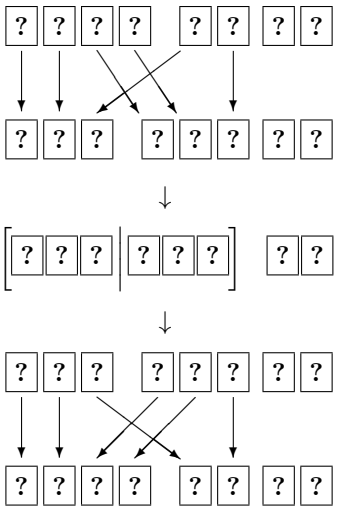
(1) カードを次のように並べ，表向きのカードを裏向きにする（2 つの 0 のコミットメントになる）．



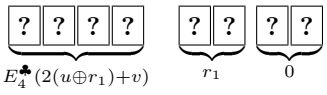
ただし，$x_1+x_2+x_7=2u+v$ なる $u,v\in\{0,1\}$ を導入している．

(2) $E_4^{\heartsuit}(2u+v)$ と中央の 0 のコミットメントを同期させてシャッフルする．

Vol.2022-DPS-190 No.34
Vol.2022-CSEC-96 No.34
2022/3/11

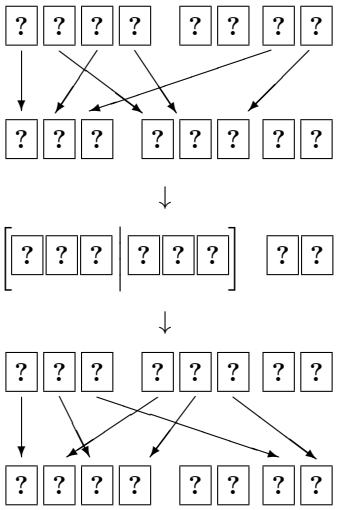


このとき，ランダムなビット $r_1 \in \{0,1\}$ が存在し，

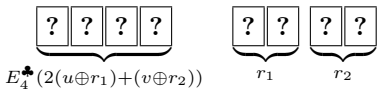


となっている．

(3) 次に，$E_4^{\clubsuit}(2(u \oplus r_1) + v)$ と右側の 0 のコミットメントを同期させてシャッフルする．

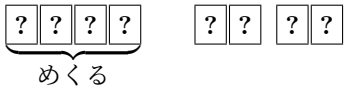


このとき，ランダムなビット $r_2 \in \{0,1\}$ が存在し，

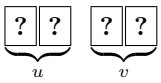


となっている．

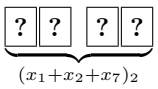
(4) $E_4^{\clubsuit}(2(u \oplus r_1) + (v \oplus r_2))$ のカードをめくる．



(5) めくったカードの整数値の二進数表現を考える．二進数表現で $\boxed{\heartsuit}\boxed{\clubsuit} = 1$ ならば対応するコミットメントを入れ替え，$\boxed{\clubsuit}\boxed{\heartsuit} = 0$ ならばコミットメントは入れ替えない．列挙すると表 1 のようになる．このようにカードを並べ替えることにより



が得られ，これは

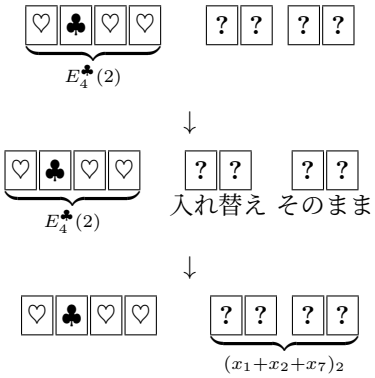


である．

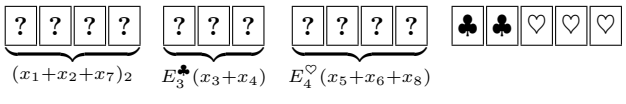
表 1: 二進数化の変換表（ステップ (5)）

めくったカード	二進数表現	入れ替え操作
$\heartsuit\heartsuit\heartsuit\clubsuit$ $E_4^{\clubsuit}(0)$	$\clubsuit\heartsuit$ 0 $\clubsuit\heartsuit$ 0	?? そのまま ?? そのまま
$\heartsuit\heartsuit\clubsuit\heartsuit$ $E_4^{\clubsuit}(1)$	$\clubsuit\heartsuit$ 0 $\heartsuit\clubsuit$ 1	?? そのまま ?? 入れ替え
$\heartsuit\clubsuit\heartsuit\heartsuit$ $E_4^{\clubsuit}(2)$	$\heartsuit\clubsuit$ 1 $\clubsuit\heartsuit$ 0	?? 入れ替え ?? そのまま
$\clubsuit\heartsuit\heartsuit\heartsuit$ $E_4^{\clubsuit}(3)$	$\heartsuit\clubsuit$ 1 $\heartsuit\clubsuit$ 1	?? 入れ替え ?? 入れ替え

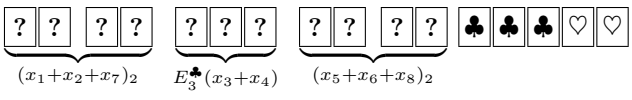
例えば，カードをめくって $E_4^{\clubsuit}(2)$ が出現した場合は，



以上の操作で $E_4^{\clubsuit}(x_1 + x_2 + x_7)$ の二進数化（コミットメント化）ができる．操作終了後，カードの状態は，



となっている．フリーカードは $\boxed{\clubsuit}$ と $\boxed{\heartsuit}$ がともに 2 枚ずつあるため，$E_4^{\heartsuit}(x_5 + x_6 + x_8)$ も二進数化することができ，実行すると



となる．

以上の操作は，図 2 の KWH 木を参照すればその正当性が確認できる．

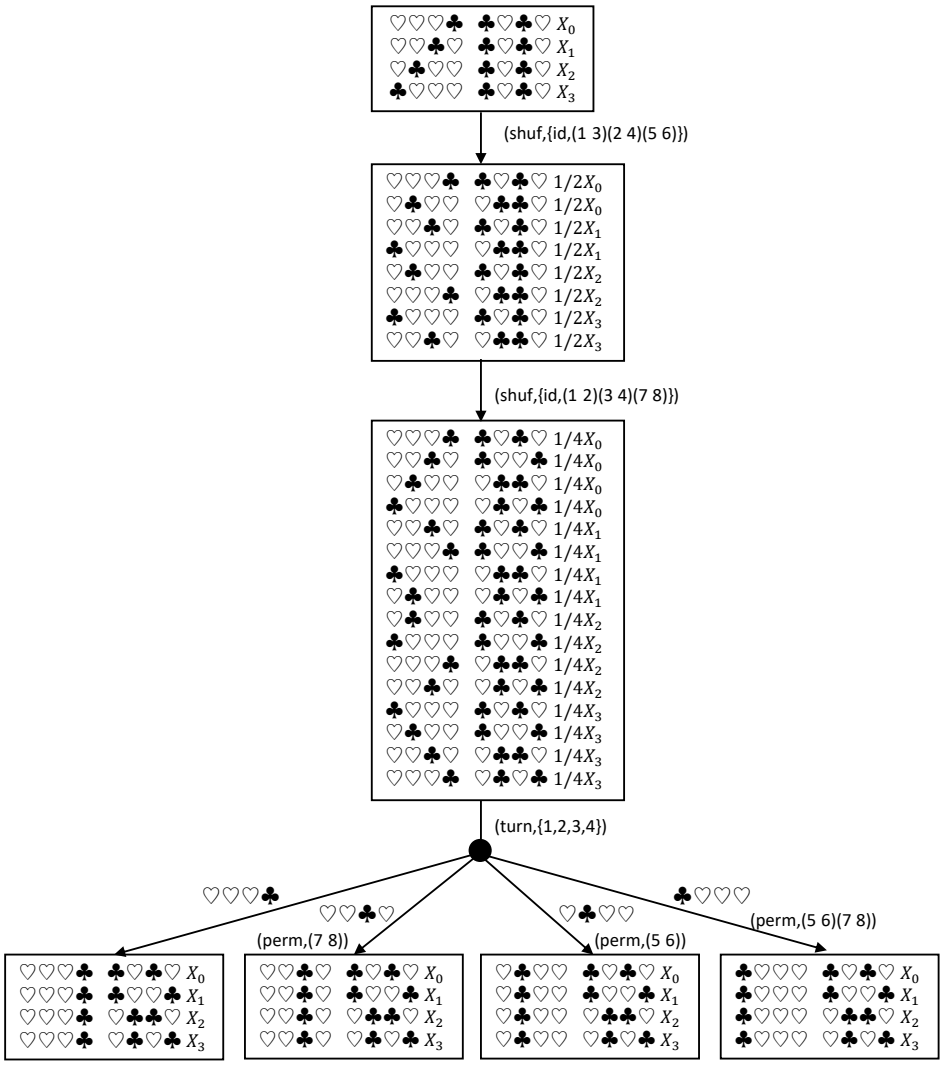
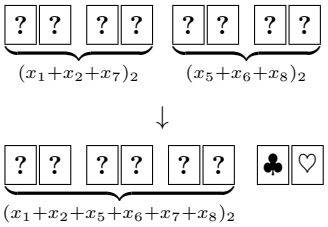


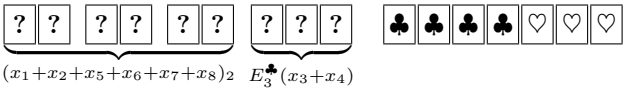
図 2: 二進数化の KWH 木．ただし，$E_4^{\clubsuit}(i)$ の確率を X_i としている．

3.7 $(x_1+x_2+x_7)_2$ と $(x_5+x_6+x_8)_2$ の全加算

二進数化できた $(x_1+x_2+x_7)_2$ と $(x_5+x_6+x_8)_2$ をフリーカードの ♣ ♣ ♡ ♡ を用いて 2.2 節で説明した全加算機のプロトコルで加算する．

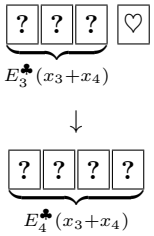


この操作を実行後，全体の状態は以下のようになる．

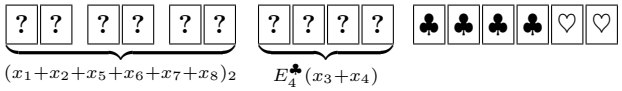


3.8 $E_3^{\clubsuit}(x_3+x_4)$ の二進数化と全体の加算

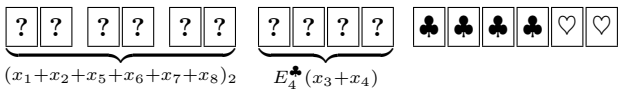
$E_3^{\clubsuit}(x_3+x_4)$ の右端に ♡ を追加して 4 枚エンコードにしても整数値は変化しない．すなわち，

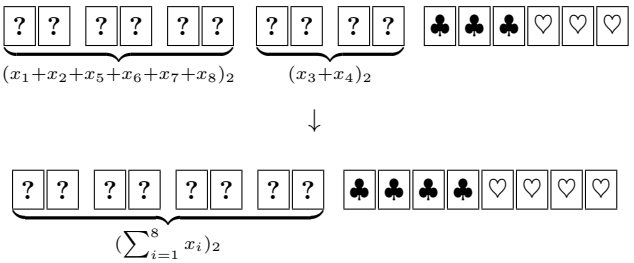


とすることができる．状況をまとめると，



となっている．これらを 3.6 節と同じ手法で二進数化し，フリーカードを用いて $(x_1+x_2+x_5+x_6+x_7+x_8)_2$ に 2.2 節の全加算器のプロトコルで加算する．





以上で入力の合計値 $\sum_{i=1}^{8} x_i$ の二進数表現を作り出すことができたので，あとは式 (2) のような $g(\sum_{i=1}^{8} x_i)$ を計算すればよい．

なお，$n \geq 9$ のときには，x_9 以降のコミットメントを半加算器のプロトコルを用いてさらに加算すればよい．

4. おわりに

本稿では，著者らが SCIS 2022 で提案した最小枚数，すなわち $2n$ 枚での対称関数の秘密計算について，最低必要入力数を 14 入力から 8 入力に減らし，特殊なカードケースの使用を不要にした．提案プロトコルは通常のカードベース暗号の計算モデルで動く最小枚数のプロトコルである．

本稿のプロトコルは必要な入力数が $n \geq 8$ となっており，これからさらに削減できるかどうかが今後の課題である．現状，図 3 のグラフのように，$3 \leq n \leq 7$ のときに対称関数に対する最小枚数のプロトコルが構成できていない．

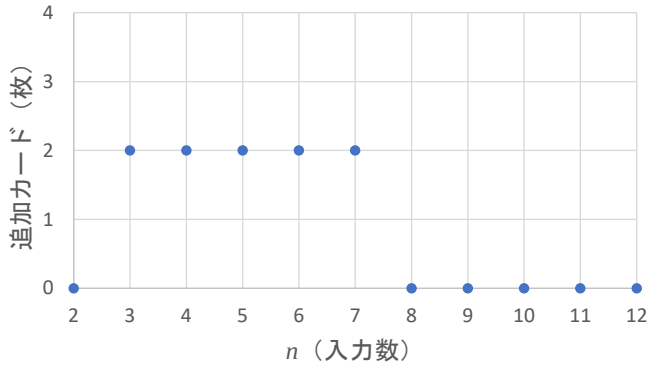


図 3: 対称関数の計算に必要な追加カード枚数

謝辞 3.3 節の加算のやり方について茨城大学の品川和雅氏にご助言いただいたことに深謝する．本研究の一部は JSPS 科研費 JP19J21153 と JP21K11881 の助成を受けている．

参考文献

[1] Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T. and Sone, H.: The Minimum Number of Cards in Practical Card-Based Protocols, *Advances in Cryptology – ASIACRYPT 2017* (Takagi, T. and Peyrin, T., eds.), LNCS, Vol. 10626, Cham, Springer, pp. 126–155 (2017).

[2] Koch, A.: Cryptographic Protocols from Physical Assumptions, PhD Thesis, Karlsruhe Institute of Technology (2019).

[3] Koch, A., Walzer, S. and Härtel, K.: Card-Based Cryptographic Protocols Using a Minimal Number of Cards, *Advances in Cryptology – ASIACRYPT 2015* (Iwata, T. and Cheon, J. H., eds.), LNCS, Vol. 9452, Berlin, Heidelberg, Springer, pp. 783–807 (2015).

[4] Mizuki, T., Asiedu, I. K. and Sone, H.: Voting with a Logarithmic Number of Cards, *Unconventional Computation and Natural Computation* (Mauri, G., Dennunzio, A., Manzoni, L. and Porreca, A. E., eds.), LNCS, Vol. 7956, Berlin, Heidelberg, Springer, pp. 162–173 (2013).

[5] Mizuki, T., Kumamoto, M. and Sone, H.: The Five-Card Trick Can Be Done with Four Cards, *Advances in Cryptology – ASIACRYPT 2012* (Wang, X. and Sako, K., eds.), LNCS, Vol. 7658, Berlin, Heidelberg, Springer, pp. 598–606 (2012).

[6] Mizuki, T. and Shizuya, H.: Computational Model of Card-Based Cryptographic Protocols and Its Applications, *IEICE Trans. Fundamentals*, Vol. E100.A, No. 1, pp. 3–11 (2017).

[7] Mizuki, T. and Sone, H.: Six-Card Secure AND and Four-Card Secure XOR, *Frontiers in Algorithmics* (Deng, X., Hopcroft, J. E. and Xue, J., eds.), LNCS, Vol. 5598, Berlin, Heidelberg, Springer, pp. 358–369 (2009).

[8] Nishida, T., Hayashi, Y., Mizuki, T. and Sone, H.: Card-Based Protocols for Any Boolean Function, *Theory and Applications of Models of Computation* (Jain, R., Jain, S. and Stephan, F., eds.), LNCS, Vol. 9076, Cham, Springer, pp. 110–121 (2015).

[9] Nishimura, A., Hayashi, Y., Mizuki, T. and Sone, H.: Pile-shifting scramble for card-based protocols, *IEICE Trans. Fundamentals*, Vol. 101, No. 9, pp. 1494–1502 (2018).

[10] Ruangwises, S. and Itoh, T.: Securely Computing the n-Variable Equality Function with $2n$ Cards, *Theory and Applications of Models of Computation* (Chen, J., Feng, Q. and Xu, J., eds.), LNCS, Vol. 12337, Springer, pp. 25–36 (2020).

[11] Ruangwises, S. and Itoh, T.: Securely computing the n-variable equality function with $2n$ cards, *Theoretical Computer Science*, Vol. 887, pp. 99–110 (2021).

[12] Shinagawa, K.: On the Construction of Easy to Perform Card-Based Protocols, PhD Thesis, Tokyo Institute of Technology (2020).

[13] Shinagawa, K., Mizuki, T., Schuldt, J. C. N., Nuida, K., Kanayama, N., Nishide, T., Hanaoka, G. and Okamoto, E.: Multi-party Computation with Small Shuffle Complexity Using Regular Polygon Cards, *Provable Security* (Au, M.-H. and Miyaji, A., eds.), LNCS, Vol. 9451, Cham, Springer, pp. 127–146 (2015).

[14] Yao, A. C.: Protocols for Secure Computations, *Annual Symposium on Foundations of Computer Science*, FOCS'82, Washington, IEEE, pp. 160–164 (1982).

[15] 四方隼人，豊田航大，宮原大輝，水木敬明：最小のカード枚数による対称関数の秘密計算について，2022 年暗号と情報セキュリティシンポジウム（SCIS 2022），No. 1F4-3, pp. 1–7 (2022).