

カード組を用いた任意の論理関数の安全な計算について

西田 拓也 *¹ 林 優一 *² 水木 敬明 *³ 曽根 秀昭 *³

*^{1,2} 東北大学大学院情報科学研究科

*³ 東北大学サイバーサイエンスセンター

980-8579 仙台市青葉区荒巻字青葉 6-3-09

980-8578 仙台市青葉区荒巻字青葉 6-3

*¹nisida@s.tohoku.ac.jp

あらまし 入力を秘密にしたまま出力を得られる安全な計算 (Secure Multi-Party Computation) は、情報セキュリティを支える暗号技術の一つである。特に物理的なカード組を用いるカードベース暗号プロトコルは、情報理論的に安全な計算を実現する。既存研究では、論理積や多数決関数など、特定の論理関数にターゲットをしばりテーラーメイドのプロトコルを構築し、計算に必要なカードの枚数を求めてきた。しかし、任意の論理関数を計算する為に十分な枚数について、一般的な解は未だ検討されていない。本稿では、任意の論理関数を安全に計算できるカード枚数を与えると共に、対称関数の場合は枚数を減らせることも示す。

Secure Computation for Any Boolean Function Using a Deck of Cards

Takuya Nishida*¹ Yu-ichi Hayashi*² Takaaki Mizuki*³ Hideaki Sone*³

*^{1,2} Graduate School of Information Sciences, Tohoku University

6-3-09 Aramaki-Aza-Aoba, Aoba-ku, Sendai, Miyagi 980-8579, Japan

*¹nisida@s.tohoku.ac.jp




*³ Cyberscience Center, Tohoku University

6-3 Aramaki-Aza-Aoba, Aoba-ku, Sendai, Miyagi 980-8578, Japan

Abstract

Secure Multi-Party Computation, which outputs the result of some computation while concealing the inputs, is a cryptographic technique supporting information security. In particular, card-based cryptographic protocols that use physical cards achieve information theoretic security. The existing studies designed tailor-made protocols for certain Boolean functions such as the logical AND function and the majority function, showing the numbers of cards required for the computations. However, a sufficient number of cards to compute any function has not been revealed. In this paper, we give a sufficient condition on the number of cards for any function to be securely computed, and show that the number can be decreased for symmetric functions.

1 はじめに

黒  や赤  の物理的なカード（裏面は同一の模様 ）を何枚か用いると、安全な計算 (Secure Multi-Party Computation) を実現できることが知られている。Yao の論文 [15] から始まった安全な計算の研究は、暗号プロトコル

のコンピュータおよび通信ネットワーク上での実装を目指すことが主流である ([4, 13] 参照)。それに対して、本稿が扱うカードベース暗号プロトコルなどの、コンピュータに頼らない安全な計算 (例 [1, 5, 9, 10]) は、情報理論的な安全性を実現できると共に、電力を必要としない簡単な道具だけを使うため、どのように計算が

行われ、且つ安全性が確保されているかが非専門家でも容易に理解できる。

これまで、表 1 に示す通り多くのカードベース暗号プロトコルが考案されており、AND 演算や XOR 演算のほか、加算器や 3 変数対称関数など、ある関数に特化した効率的な（必要なカードの枚数が少ない）プロトコルも考案されている。これらの既存研究の対象は特定の関数であるが、対して本稿では任意の論理関数を安全に計算するプロトコルを提案する。

本稿は、用語や記号、現在知られている中で最も効率的な AND/XOR/コピープロトコル [7] などの紹介から始める。

表 1 既存のカードベース暗号プロトコル

考案者	色数	枚数	平均試行回数
○ 非コミット型 AND 演算			
den Boer [2]	2	5	1
Mizuki-Kumamoto-Sone [6]	2	4	1
○ コミット型 AND 演算			
Crépeau-Kilian [3]	4	10	6
Niemi-Renvall [10]	2	12	2.5
Stiglic [14]	2	8	2
Mizuki-Sone [7]	2	6	1
○ コミット型 XOR 演算			
Crépeau-Kilian [3]	4	14	6
Mizuki-Uchiike-Sone [8]	2	10	2
Mizuki-Sone [7]	2	4	1
○ コミット型半加算器			
Mizuki-Asiedu-Sone [5]	2	8	1
○ コミット型全加算器			
Mizuki-Asiedu-Sone [5]	2	10	1
○ コミット型 3 変数対称関数演算			
Nishida-Mizuki-Sone [11]	2	8	1

1.1 用語と記号

カードベース暗号プロトコルの説明のために、いくつかの用語と記号を与える。

2 枚のカードでブール値を扱うために、次のような符号化ルールを定義する。

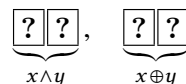
$$\clubsuit \heartsuit = 0, \quad \heartsuit \clubsuit = 1. \quad (1)$$

また、符号化ルール (1) の下でビット $x \in \{0,1\}$ の値と等しく、裏にして $\boxed{?}\boxed{?}$ と置かれて

いる 2 枚のカードを x のコミットメントと呼び、



と表す。表 1 にある「コミット型」とは、出力がコミットメントとして得られるプロトコルのことで、例えばビット x と y のコミットメントを入力とする AND 演算や XOR 演算では、



のようにそれぞれ出力を得ることができる。

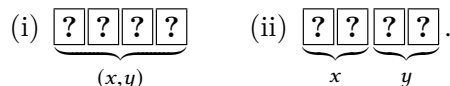
二つのビットからなるペア (x,y) に対して、操作 get と shift を次のように定義する。

$$\begin{aligned} \text{get}^0(x,y) &= x; & \text{get}^1(x,y) &= y; \\ \text{shift}^0(x,y) &= (x,y); & \text{shift}^1(x,y) &= (y,x). \end{aligned}$$

これらの記号を使うと、AND 演算は任意のビット $r \in \{0,1\}$ を用いて

$$x \wedge y = \text{get}^{x \oplus r}(\text{shift}^r(0,y)) \quad (2)$$

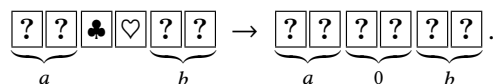
と書ける [11]。以降、二つのビット x と y に関して、(i) の表現は (ii) を意味する。



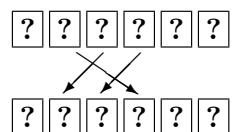
1.2 AND プロトコル

既存の AND プロトコル [7] は、 $a \in \{0,1\}$ と $b \in \{0,1\}$ のコミットメントが与えられたとき、2 枚のカードを加えることでコミット型の AND 演算を実現する。

1. $a, 0, b$ のコミットメントを置く。



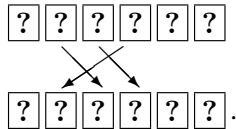
2. 次のように並び替える。



3. カード列を真中で分け、左右の列同士をランダムに入れ替える（これをランダム二等分割カット [7] といい, $[\cdot|\cdot]$ で表す).

$$[\boxed{???}|\boxed{???}] \rightarrow \boxed{???|???}.$$

4. 次のように並び替える.

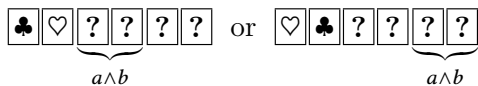


このとき, r を一様ランダムなビットとして

$$\underbrace{\boxed{??}}_{a \oplus r} \quad \underbrace{\boxed{??|??|??}}_{\text{shift}^r(0,b)}$$

のように並んでいる.

5. 左端の2枚のカードをめくると $a \oplus r$ の値が分かり, 式 (2) より次の位置



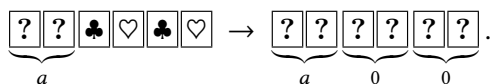
に $a \wedge b$ のコミットメントが得られる.

$a \oplus r$ のコミットメントをめくっても, r がランダムのためビット a に関する情報は全く分からないままなので, 安全な計算が実現されている. また, めくった2枚のカードは別の計算に再利用できる (フリーになる) というカードベースプロトコルの特徴にも注意してほしい.

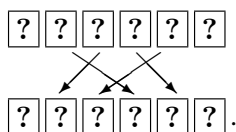
1.3 コピープロトコル

a のコミットメントが与えられたとき, 4枚のカードを加えることで, a のコミットメントを二つに複製することができる [7].

1. $a, 0, 0$ のコミットメントを置く.



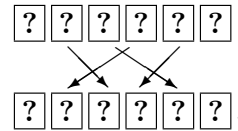
2. 次のように並び替える.



3. ランダム二等分割カットを施す.

$$[\boxed{???}|\boxed{???}].$$

4. 次のように並び替える.

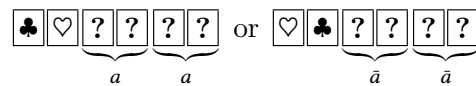


このとき, r を一様ランダムなビットとして

$$\underbrace{\boxed{??}}_{a \oplus r} \quad \underbrace{\boxed{??}}_r \quad \underbrace{\boxed{??}}_r$$

と並んでいる.

5. 左端の2枚をめくると, $r = a$ と $r = \bar{a}$ のどちらであるかが分かり,



として a のコミットメントが二つ得られる.

なお, 符号化ルール (1) の下では, コミットメントの左右のカードを入れ替えるだけで NOT 演算ができる. 従って NOT 演算は自明であり, 本稿では x のコミットメントから \bar{x} のコミットメントへ変換する記述は省略する.

このコピープロトコルではステップ 4 を終えたとき, 右側二つのコミットメントは $r = 0 \oplus r$ になっている. よって, ステップ 1 において, 例えば $a, b, 0$ のコミットメントを置いてスタートすると, ステップ 4 を終えたときの状態は

$$\underbrace{\boxed{??}}_{a \oplus r} \quad \underbrace{\boxed{??}}_{b \oplus r} \quad \underbrace{\boxed{??}}_r$$

となり, $a \oplus b$ と a のコミットメントを出力として得られる [11].

また, 既存の XOR プロトコル [7] では, a と b のコミットメントから新たにカードを加えることなく, $a \oplus b$ のコミットメントだけを得られる (表 1 を参照).

1.4 本稿の結果

これまでに紹介した AND/XOR/NOT 演算から、次の補題は明らかである。

補題 1.1 コミットメント $\underbrace{[?][?]}_{x_1} \underbrace{[?][?]}_{x_2}$ と 2 枚の

追加カード $\clubsuit \heartsuit$ が与えられたとき、任意の 2 変数論理関数 $f(x_1, x_2)$ の値のコミットメントを安全に出力できる。

表 1 に示した通り、次の結果も既知である。

補題 1.2 ([11]) x_1, x_2, x_3 のコミットメントと 2 枚の追加カード $\clubsuit \heartsuit$ が与えられたとき、任意の 3 変数対称関数 $f(x_1, x_2, x_3)$ の値のコミットメントを安全に出力できる。

さらにコピープロトコルを組み合わせることで、任意の (多変数) 論理関数を安全に計算することも明らかである。

しかし、計算にあたり何枚のカードがあれば十分かについては明らかにされておらず、本稿ではこの問題に取り組み、必要なカード枚数の少ない効率的な手法を提案する。

本稿の構成を示す。まず 2 節で既存の AND プロトコルと半加算器プロトコルを改良する。改良されたプロトコルを利用して、3 節では n 個のコミットメントが与えられたとき、任意の n 変数論理関数を追加カード 6 枚で安全に計算できる手法を与える。続く 4 節において、対称関数の場合は追加カードが 2 枚で十分であることを示す (これは補題 1.2 の一般化になっていることに注意)。最後に 5 節で結論を述べる。

2 AND と半加算器プロトコルの改良

本節では、任意の論理関数を安全に計算する際に役立つ、新しいプロトコルを提案する。これは 1.2 節で紹介した AND プロトコルを改良し、後述のようにこの改良により、任意の論理関数を計算するために必要な追加カードの枚数を (8 枚から 6 枚に) 減らすことができる。

1.2 節の AND プロトコルをもう一度見てほしい。プロトコルの出力として $a \wedge b$ のコミットメントを得たときに、別の 2 枚の裏になっているカードは式 (2) より、 $\bar{a} \wedge b$ のコミットメントになっていることが分かる。

$$\underbrace{\clubsuit \heartsuit [?][?]}_{a \wedge b} \text{ or } \underbrace{\heartsuit \clubsuit [?][?]}_{\bar{a} \wedge b} \underbrace{[?][?]}_{a \wedge b}$$

ここでこれら 6 枚のカードを、表になっている 2 枚のカード (フリーカード) を再利用して次のように並べ直し、

$$\underbrace{[?][?]}_{a \wedge b} \underbrace{[?][?]}_{\bar{a} \wedge b} \underbrace{[?][?]}_0$$

1.3 節で紹介したコピープロトコルをステップ 2 から適用すると、 $ab \oplus \bar{a}b$ と ab のコミットメントを得ることができる (AND 演算を意味する \wedge 記号は以降適宜省略する)。即ち $ab \oplus \bar{a}b = (a \oplus \bar{a})b = b$ より、

$$\underbrace{[?][?]}_b \underbrace{[?][?]}_{a \wedge b}$$

となるカード列が得られる。

よってこの改良を施すことで、AND 演算を行いつつ片方の入力コミットメントを維持することが可能になる。

さらに、1.3 節で $a, b, 0$ のコミットメントから

$$\underbrace{[?][?]}_a \underbrace{\clubsuit \heartsuit [?][?]}_{a \oplus b}$$

となるカード列を得られることを紹介したが、ここで上の改良版 AND プロトコルを適用することで、 $a \oplus b$ と $a(\overline{a \oplus b})$ のコミットメントが得られる。つまり $a(\overline{a \oplus b}) = a\bar{a} \oplus ab = ab$ より、

$$\underbrace{[?][?]}_{a \oplus b} \underbrace{[?][?]}_{a \wedge b}$$

となるカード列が得られる。これは a と b を入力とする半加算器の出力になっており、既存のプロトコル [5] では 4 枚の追加カードで実現していた (表 1 を参照) のを、2 枚の追加カードだ

けで実現できている。従ってこれは改良された半加算器プロトコルである。

本節をまとめる。まず、既存の AND プロトコル [7] を改良し、AND 演算を片方の入力を維持したまま出力できるようにした。その手順は以下の通りである。

1. $a, 0, b$ のコミットメントを置く。

$$\underbrace{??\clubsuit}_{a} \underbrace{? \heartsuit}_{b} \rightarrow \underbrace{??}_{a} \underbrace{0}_{0} \underbrace{??}_{b}.$$

2. 1.2 節の AND プロトコルを適用し、次のようなカード列を得る。

$$\underbrace{\clubsuit \heartsuit}_{a \wedge b} \underbrace{??}_{\bar{a} \wedge b} \text{ or } \underbrace{\heartsuit \clubsuit}_{\bar{a} \wedge b} \underbrace{??}_{a \wedge b}.$$

3. 次のように並べ直す。

$$\underbrace{??}_{a \wedge b} \underbrace{??}_{\bar{a} \wedge b} \underbrace{??}_{0}.$$

4. 1.3 節のコピープロトコルをステップ 2 から適用し、

$$\underbrace{\clubsuit \heartsuit}_{b} \underbrace{??}_{a \wedge b} \text{ or } \underbrace{\heartsuit \clubsuit}_{\bar{b}} \underbrace{??}_{a \wedge b}.$$

として b と $a \wedge b$ のコミットメントを得る。

この結果を次の補題にまとめる。

補題 2.1 コミットメント $\underbrace{??}_{x_1} \underbrace{??}_{x_2}$ と 2 枚の

追加カード $\underbrace{\clubsuit \heartsuit}$ が与えられたとき、 $x_1 x_2$ と x_2 のコミットメントを安全に出力できる。

加えて、半加算器プロトコル [5] の改良も行った。既存のプロトコルでは計算を実行するために追加カードを 4 枚要していたが、2 枚で済むようになった。その手順は以下の通りである。

1. $a, b, 0$ のコミットメントを置く。

$$\underbrace{??}_{a} \underbrace{??}_{b} \underbrace{\clubsuit \heartsuit}_{0} \rightarrow \underbrace{??}_{a} \underbrace{??}_{b} \underbrace{??}_{0}.$$

2. 1.3 節のコピープロトコルを適用し、次のようなカード列を得る。

$$\underbrace{\clubsuit \heartsuit}_{a \oplus b} \underbrace{??}_{a} \text{ or } \underbrace{\heartsuit \clubsuit}_{a \oplus b} \underbrace{??}_{\bar{a}}.$$

3. 次のように並べ直す。

$$\underbrace{??}_{a} \underbrace{\clubsuit \heartsuit}_{\bar{a} \oplus b}.$$

4. 本節の AND プロトコルを適用し、

$$\underbrace{\clubsuit \heartsuit}_{\bar{a} \oplus b} \underbrace{??}_{a \wedge b} \text{ or } \underbrace{\heartsuit \clubsuit}_{a \oplus b} \underbrace{??}_{\bar{a} \wedge b}.$$

のように $a \oplus b$ と $a \wedge b$ のコミットメントを得る。

この結果から次の定理が得られる。

定理 2.2 コミットメント $\underbrace{??}_{x_1} \underbrace{??}_{x_2}$ と 2 枚の

追加カード $\underbrace{\clubsuit \heartsuit}$ が与えられたとき、 $x_1 \oplus x_2$ と $x_1 x_2$ のコミットメントを安全に出力できる。

3 任意の多変数論理関数の演算

ここまで構築したプロトコルを応用して、任意の論理関数を安全に計算できるカードベースプロトコルを提案する。結論から言うと、任意の n 変数論理関数は、 n 個のコミットメントと 6 枚の追加カードによって安全に計算できる。

3.1 プロトコルの概要

任意の n 変数論理関数 $f(x_1, x_2, \dots, x_n)$ は、シャノン展開により一意の AND-XOR 二段論理式で表現できる [12]。

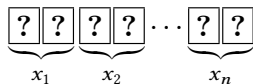
$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= \bar{x}_1 \bar{x}_2 \cdots \bar{x}_n f(0, 0, \dots, 0) \\ &\oplus x_1 \bar{x}_2 \cdots \bar{x}_n f(1, 0, \dots, 0) \\ &\oplus \bar{x}_1 x_2 \cdots \bar{x}_n f(0, 1, \dots, 0) \\ &\oplus x_1 x_2 \cdots \bar{x}_n f(1, 1, \dots, 0) \\ &\vdots \\ &\oplus x_1 x_2 \cdots x_n f(1, 1, \dots, 1). \end{aligned}$$

即ち $f(x_1, x_2, \dots, x_n)$ は 2^n 個の積項の排他的論理和で一意的に表される。ただし対応する f の値が 0 ならばその積項は消去できる。以下ではプロトコルを記述する便宜上、 2^n 個の積項を仮定する。 x_1, x_2, \dots, x_n のコミットメントを入力としたプロトコルの流れは次のようになる。

1. 最初の積項 $\bar{x}_1 \bar{x}_2 \cdots \bar{x}_n$ のコミットメントを作る。その際に x_1 から x_n の全ての入力コミットメントを維持しておく。
2. 維持しておいた全ての入力から、2 番目の積項 $x_1 \bar{x}_2 \cdots \bar{x}_n$ のコミットメントを作る。その際にも全ての入力を維持しておく。
3. $\bar{x}_1 \bar{x}_2 \cdots \bar{x}_n$ と $x_1 \bar{x}_2 \cdots \bar{x}_n$ のコミットメントの XOR 演算を行う。
4. $\bar{x}_1 x_2 \cdots \bar{x}_n$ を同様にして作る。
5. 先ほど生成した $\bar{x}_1 \bar{x}_2 \cdots \bar{x}_n \oplus x_1 \bar{x}_2 \cdots \bar{x}_n$ と、今作られた積項 $\bar{x}_1 x_2 \cdots \bar{x}_n$ の XOR 演算を行う。
6. 同様の手順を最後の積項 $x_1 x_2 \cdots x_n$ まで繰り返し、 $f(x_1, x_2, \dots, x_n)$ の値を生成する。

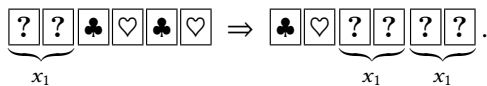
3.2 プロトコルの詳細

プロトコルの詳細を説明する。プロトコルの入力は n 個のコミットメント



と、6 枚の追加カード $\clubsuit \clubsuit \clubsuit \heartsuit \heartsuit \heartsuit$ である。これらのカードを用いて、 $f(x_1, x_2, \dots, x_n)$ の値のコミットメントを安全に出力する。

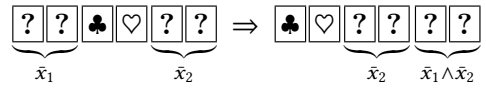
1. 1.3 節のコピープロトコルを適用し、 x_1 を二つに複製する。



複製した x_1 の片方は、維持する入力としてわきによけておく。

2. 2 節の AND プロトコルを適用することで、

補題 2.1 により



を得る。 x_2 は維持するためにわきによける。

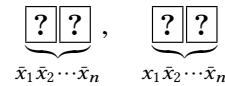
3. 補題 2.1 により、 $\bar{x}_1 \bar{x}_2$ と x_3 から $\bar{x}_1 \bar{x}_2 \bar{x}_3$ と x_3 を得て、 x_3 をわきによける。これを x_n まで繰り返す。

一度状況を整理すると、今手元にあるものは x_1, x_2, \dots, x_n のコミットメントと、積項



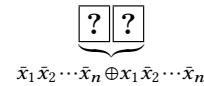
と、4 枚のフリーカード $\clubsuit \clubsuit \heartsuit \heartsuit$ である。

4. 同様にして 2 番目の積項 $x_1 \bar{x}_2 \cdots \bar{x}_n$ を作る。今手元にあるものは、二つの積項



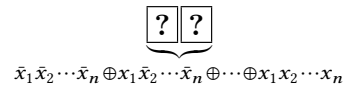
と x_1, x_2, \dots, x_n のコミットメントと、2 枚のフリーカード $\clubsuit \heartsuit$ である。

5. $\bar{x}_1 \bar{x}_2 \cdots \bar{x}_n \oplus x_1 \bar{x}_2 \cdots \bar{x}_n$ を作り、あるものは



と x_1, x_2, \dots, x_n のコミットメントと、4 枚のフリーカード $\clubsuit \clubsuit \heartsuit \heartsuit$ になる。

6. ステップ 4, 5 を $x_1 x_2 \cdots x_n$ まで繰り返し、 $f(x_1, x_2, \dots, x_n)$ の値のコミットメント



を得る。

残った x_1, x_2, \dots, x_n のコミットメントと 4 枚のフリーカード $\clubsuit \clubsuit \heartsuit \heartsuit$ は別の計算に使える。以上から次の定理が導かれる。

定理 3.1 f を任意の n 変数論理関数とする。 x_1, x_2, \dots, x_n のコミットメントと 6 枚の追加カード $\clubsuit \clubsuit \clubsuit \heartsuit \heartsuit \heartsuit$ が与えられたとき、

$f(x_1, x_2, \dots, x_n)$ の値と x_1, x_2, \dots, x_n のコミットメントを安全に出力できる。

ステップ 1-3 の中で、フリーカードは 4 枚しか使用していないことに注目してほしい。従って、ステップ 2, 3 において改良版 AND プロトコルを利用せず、既存の AND/コピープロトコルで同様の処理を行った場合、ステップ 1-3 に 6 枚のフリーカードを使うため、全体としてさらに 2 枚の追加カードが必要になってしまう。

4 対称関数の演算

前節で、任意の論理関数を安全に計算するカードベースプロトコルは、 n 個の入力コミットメントと 6 枚の追加カードで実現できることを示した。本節では対称関数の演算の場合、追加カードは 2 枚だけでよいことを示す。

n 変数対称関数 $f(x_1, x_2, \dots, x_n)$ の出力は、入力に含まれる 1 の数によって定まる。従って、集合 $A \subseteq \{0, 1, \dots, n\}$ を用いて $f = S_A^n$ と書ける。例えば 3 変数対称関数である 3 入力多数決は $f(x_1, x_2, x_3) = S_{\{2,3\}}^3$ と書け、 x_1, x_2, x_3 を加算した値 $\sum_{i=1}^3 x_i$ を表す 2 ビットの列から、MSB を出力する関数 $g(\sum_{i=1}^3 x_i) = \text{MSB}$ により表現できる。つまり、 n 変数対称関数 $f: \{0, 1\}^n \rightarrow \{0, 1\}$ は、 $f = g(\sum x_i)$ となる関数 $g: \{0, 1, \dots, n\} \rightarrow \{0, 1\}$ で表すことができる。

2 節の半加算器を適用することで、定理 2.2 により、 $\sum x_i$ を表すコミットメント列の生成は、追加カード 2 枚で実現できる。

補題 4.1 x_1, x_2, \dots, x_n のコミットメントと 2 枚の追加カード $\clubsuit \heartsuit$ が与えられたとき、 $\sum_{i=1}^n x_i$ を表す $\lceil \log_2 n \rceil + 1$ ビットのコミットメント列を安全に出力できる。

x_1, x_2, \dots, x_n のコミットメント（と 2 枚の追加カード）から $\sum x_i$ を生成する過程で、新たにフリーカードが生まれることに注意してほしい。なぜなら $\sum x_i$ のビット数 $\lceil \log_2 n \rceil + 1$ は n 以下だ

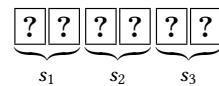
からである。具体的には、全部で $2(n - \lceil \log_2 n \rceil)$ 枚のフリーカードが生じる。

以上で、本節の主要な結果を述べる準備が整った。即ち、任意の対称関数は追加カード 2 枚で計算できる。

定理 4.2 $n \geq 4$ とし、 f を n 変数対称関数とする。 x_1, x_2, \dots, x_n のコミットメントと 2 枚の追加カード $\clubsuit \heartsuit$ が与えられたとき、 $f(x_1, x_2, \dots, x_n)$ の値のコミットメントを安全に出力できる。

(証明) $g: \{0, 1, \dots, n\} \rightarrow \{0, 1\}$ を $g(\sum x_i) = f$ なる関数とする。補題 4.1 により、 $\sum x_i$ を表す $\lceil \log_2 n \rceil + 1$ ビットのコミットメント列と $2(n - \lceil \log_2 n \rceil)$ 枚のフリーカードを得る。もし $n \geq 5$ ならば、 $2(n - \lceil \log_2 n \rceil) \geq 6$ により 6 枚以上のフリーカードがあるので、定理 3.1 により (g の定義域を $\{0, 1\}^{\lceil \log_2 n \rceil + 1}$ と見なして) g の値のコミットメントを安全に出力できる。従って $n = 4$ としてよい。

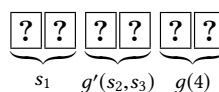
$\sum_{i=1}^4 x_i$ を表す 3 ビットのコミットメント列は



と書ける (s_1 を MSB とする)。 $0 \leq \sum_{i=1}^4 x_i \leq 4$ より、 $s_1 = 1$ ならば、必ず $\sum_{i=1}^4 x_i = 4$ である。従って、 g の定義域を下位 2 ビットにしぼった論理関数 $g': \{0, 1\}^2 \rightarrow \{0, 1\}$ を用いて、

$$g\left(\sum_{i=1}^4 x_i\right) = \begin{cases} g(4) & \text{if } s_1 = 1; \\ g'(s_2, s_3) & \text{if } s_1 = 0 \end{cases}$$

と書ける。 g' は 2 変数なので、補題 1.1 により、 s_2 と s_3 のコミットメントと 2 枚のフリーカードで $g'(s_2, s_3)$ の値のコミットメントを安全に出力できる。よって



というカード列を得られることが分かる。ここから $g(\sum_{i=1}^4 x_i) = \text{get}^{s_1}(g'(s_2, s_3), g(4))$ であるこ

とを利用して, 1.2 節の AND プロトコルをステップ 2 から適用すれば,

$$\begin{array}{ccc} \spadesuit & \heartsuit & \boxed{?} \boxed{?} \boxed{?} \boxed{?} \quad \text{or} \quad \heartsuit \spadesuit \boxed{?} \boxed{?} \boxed{?} \boxed{?} \\ g(\sum_{i=1}^4 x_i) & & g(\sum_{i=1}^4 x_i) \end{array}$$

となり, $f(x_1, x_2, x_3, x_4) = g(\sum_{i=1}^4 x_i)$ の値のコミットメントを得る. \square

5 おわりに

既存の研究では, 特定の関数に対して専用のプロトコルを設計し, それに基づいて安全な計算に必要なカードの枚数を求めてきた.

本稿では, AND プロトコルを改良し, それを AND-XOR 二段論理式に適用することで, 任意の論理関数を安全に計算するのに十分なカードの枚数を求めた. 結果, 任意の n 変数関数の出力コミットメントは n 個のコミットメントと, 6 枚の追加カード $\spadesuit \spadesuit \spadesuit \heartsuit \heartsuit \heartsuit$ から安全に生成可能であることを示した.

さらに, 計算したい関数が対称関数である場合, 改良した半加算器プロトコルによって追加カードは 2 枚 $\spadesuit \heartsuit$ まで減らせることも示した.

参考文献

[1] Balogh, J., Csirik, J.A., Ishai, Y., Kushilevitz, E.: Private computation using a PEZ dispenser. *Theoretical Computer Science* 306(1–3), 69–84 (2003)

[2] den Boer, B.: More efficient match-making and satisfiability: the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) *Advances in Cryptology — EUROCRYPT '89*, Lecture Notes in Computer Science, vol. 434, pp. 208–217. Springer Berlin Heidelberg (1990)

[3] Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) *Advances in Cryptology — CRYPTO '93*, Lecture Notes in Computer Science, vol. 773, pp. 319–330. Springer Berlin Heidelberg (1994)

[4] Goldreich, O.: *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA (2004)

[5] Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In:

Mauri, G., Dennyunzio, A., Manzoni, L., Porreca, A.E. (eds.) *Unconventional Computation and Natural Computation*, Lecture Notes in Computer Science, vol. 7956, pp. 162–173. Springer Berlin Heidelberg (2013)

[6] Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Wang, X., Sako, K. (eds.) *Advances in Cryptology — ASIACRYPT 2012*, Lecture Notes in Computer Science, vol. 7658, pp. 598–606. Springer Berlin Heidelberg (2012)

[7] Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) *Frontiers in Algorithmics*, Lecture Notes in Computer Science, vol. 5598, pp. 358–369. Springer Berlin Heidelberg (2009)

[8] Mizuki, T., Uchiike, F., Sone, H.: Securely computing XOR with 10 cards. *The Australasian Journal of Combinatorics* 36, 279–293 (2006)

[9] Moran, T., Naor, M.: Polling with physical envelopes: a rigorous analysis of a human-centric protocol. In: Vaudenay, S. (ed.) *Advances in Cryptology — EUROCRYPT 2006*, Lecture Notes in Computer Science, vol. 4004, pp. 88–108. Springer Berlin Heidelberg (2006)

[10] Niemi, V., Renvall, A.: Secure multiparty computations without computers. *Theoretical Computer Science* 191(1–2), 173–183 (1998)

[11] Nishida, T., Mizuki, T., Sone, H.: Securely computing the three-input majority function with eight cards. In: Dediu, A.H., Martín-Vide, C., Truthe, B., Vega-Rodríguez, M.A. (eds.) *Theory and Practice of Natural Computing*, Lecture Notes in Computer Science, vol. 8273, pp. 193–204. Springer Berlin Heidelberg (2013)

[12] Sasao, T.: *Switching Theory for Logic Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edn. (1999)

[13] Schneider, T.: *Engineering Secure Two-Party Computation Protocols: Design, Optimization, and Applications of Efficient Secure Function Evaluation*. Springer Berlin Heidelberg (2012)

[14] Stiglic, A.: Computations with a deck of cards. *Theoretical Computer Science* 259(1–2), 671–678 (2001)

[15] Yao, A.C.C.: Protocols for secure computations. In: *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, FOCS 1982. pp. 160–164. IEEE Computer Society, Washington, DC, USA (1982)