Secure Computation of Any Boolean Function Based on Any Deck of Cards*

Kazumasa Shinagawa[1,2] and Takaaki Mizuki[3]

[1] Tokyo Institute of Technology, Meguro, Japan
[2] Institute of Advanced Industrial Science and Technology (AIST), Kōtō, Japan
[3] Tohoku University, Sendai, Japan

Abstract. It is established that secure computation can be achieved by using a deck of physical cards. Almost all existing card-based protocols are based on a *specific* deck of cards. In this study, we design card-based protocols that are executable using *any* deck of cards (e.g., playing cards, UNO, and trading cards). Specifically, we construct a card-based protocol for any Boolean function based on any deck of cards. As corollaries of our result, a standard deck of playing cards (having 52 cards) enables secure computation of any 22-variable Boolean function, and UNO (having 112 cards) enables secure computation of any 53-variable Boolean function.

Key words: Secure computation; Card-based protocols; Playing cards

1 Introduction

Secure computation enables parties having secret inputs to compute a joint function of their inputs without revealing information about the inputs that is not trivially revealed by knowing the output. It is established that secure computation can be achieved by using a deck of physical cards; this is known as *card-based cryptography* (e.g., [1,2,5]). Card-based protocols enable participants, including those unfamiliar with mathematics, to be convinced about the correctness and security of their computations. In this study, we design card-based protocols based on *general* decks of cards; almost all the existing protocols are based on a specific deck of cards such as a two-colored deck consisting of two types of cards: $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. We refer to this two-colored deck as a *deck of binary cards*.

1.1 A Deck of Binary Cards

A *deck of binary cards* consists of a finite number of cards whose faces display either $\boxed{\clubsuit}$ or $\boxed{\heartsuit}$ and the backs display an identical symbol $\boxed{?}$. All cards with an identical symbol are indistinguishable. The following encoding rule is used:

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

* This paper appears in Proceedings of FAW 2019. The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-18126-0_6.

Two face-down cards $\boxed{?}\boxed{?}$ representing a bit $x \in \{0,1\}$ is referred to as a *commitment to* x. Given a collection of input commitments to $x_1, x_2, \cdots, x_n \in \{0,1\}$, a card-based protocol for a function $f : \{0,1\}^n \to \{0,1\}$ generates a commitment to the output value $f(x_1, x_2, \cdots, x_n)$ as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\underbrace{\boxed{?}\boxed{?}}_{x_3}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n} \quad \to \quad \underbrace{\boxed{?}\boxed{?}}_{f(x_1,x_2,\cdots,x_n)} \quad .$$

The first card-based protocol is the *five-card trick* proposed by den Boer [2]. It is an AND protocol using a deck of five cards $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}$; it reveals the output value directly[4] rather than generating a commitment to the output value. Mizuki and Sone [5] showed that every Boolean function can be securely computed in a finite runtime by using a deck of binary cards; they constructed a six-card AND, a four-card XOR, and a six-card COPY protocols; these are state-of-the-art finite-runtime protocols under the condition that shuffles used in a protocol are *random cut* and/or *random bisection cut*; these shuffles are known to be physically implementable [8]. Whereas they did not consider the actual number of cards, Nishida et al. [7] showed that any n-variable Boolean function can be securely computed by using a deck of $2n + 6$ binary cards.

1.2 A Deck of Playing Cards

Whereas almost all existing protocols are based on a deck of binary cards, there are certain exceptions [4, 6]. Niemi and Renvall [6] and Mizuki [4] constructed card-based protocols based on a *standard deck of playing cards*, where all the cards are distinguishable while their backs display an identical symbol $\boxed{?}$. By creating a total order on the set of 52 cards, we can assume that a standard deck of playing cards is of the following form: $\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}\cdots\boxed{52}$. The encoding rule is as follows: $\boxed{i}\boxed{j} = 0$, $\boxed{j}\boxed{i} = 1$, where i, j are integers such that $i < j$. A *commitment* to x is defined by a pair of face-down cards with the encoding rule and denoted by $[x]^{\{i,j\}}$; here, $\{i, j\}$ is called the *base* of the commitment. Following the encoding rule, Niemi and Renvall [6] showed that any Boolean function can be securely computed by a Las-Vegas protocol. Mizuki [4] showed that it can be achieved by a *finite-runtime* protocol. Specifically, Mizuki [4] constructed an eight-card AND, a four-card XOR, and a six-card COPY protocols. Note that the eight-card AND protocol requires two more cards than the binary card protocol.

1.3 This Work: Any Deck of Cards

In this study, we design card-based protocols based on various decks of cards. Specifically, our protocols are operable on all decks in a certain class of decks: *majority-free decks*. Majority-free decks satisfy the following properties:

[4] This type of protocols is called *non-committed format*. Meanwhile, the other type of protocols, where the output is not revealed, is called *committed format*. We focus on committed format protocols throughout this paper.

1. It consists of a finite number of cards.
2. The backs of all the cards in the deck have an identical symbol.
3. For each symbol on the faces, the number of cards having the symbol does not exceed half of the total number of cards in the deck.

The third condition is necessary in order to use the two-cards-per-bit encoding. It excludes the following type of decks: $\boxed{1}\boxed{2}\boxed{1}\boxed{1}\boxed{1}\boxed{1}$. In the above-mentioned deck, whereas a commitment whose base is $\{1, 2\}$ can be created, the remaining four cards are ineffective. The third condition guarantees that we can create k commitments for $2k$ cards.

We then classify the class of majority-free decks into the following three types according to the symbols on the faces:

Type-0 A deck in which all the symbols are distinct.
Type-1 A deck in which all the symbols are distinct except for a single symbol, wherein two or more cards display this symbol.
Type-2 A deck that is neither type-0 nor type-1, i.e., a deck in which there exist two or more symbols, each of which are displayed by two or more cards.

A majority-free deck of binary cards, whose number of cards is at least four, is a type-2 deck because $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ are two such symbols. A standard deck of playing cards is a type-0 deck because all the cards are distinct. A deck of playing cards with two jokers whose faces are identical is a canonical example of a type-1 deck because two jokers are the exception. A deck of UNO is a type-2 deck because $\boxed{1}$ with green and $\boxed{2}$ with green are two such symbols. A deck of trading cards (e.g., "Pokémon Trading Card Game") might be any type of deck of cards.

1.4 Our Result

In this study, we construct a card-based protocol for any Boolean function based on any majority-free deck of cards. Our result is summarized by Theorem 1:

Theorem 1 *Let f be any n-variable Boolean function. Then, for any $i \in \{0, 1, 2\}$, we can securely compute f using a type-i majority-free deck of $2n + 8 - i$ cards.*

Table 1 presents a comparison between the previous work by Nishida et al. [7] and our work. It is noteworthy that our result shows that *any* type-2 deck provides a protocol for any function as efficient (in terms of the number of cards) as the deck of binary cards.

Our result also implies the following corollaries:

- The standard deck of playing cards (having 52 cards), which is a type-0 deck, enables secure computation of any 22-variable Boolean function.
- A deck of playing cards with a pair of jokers and a spare card (55 cards in total), which is a type-1 deck, enables secure computation of any 24-variable Boolean function.
- A deck of UNO (having 112 cards), which is a type-2 deck, enables secure computation of any 53-variable Boolean function.

Table 1. A comparison between our work and the existing work

	Type of deck	# of cards
Nishida et al. [7]	binary (type-2)	$2n+6$
Nishida et al. [7] & Mizuki [4]	type-0	$2n+8$
Ours	type-1	$2n+7$
Ours	type-2	$2n+6$

1.5 Related Works

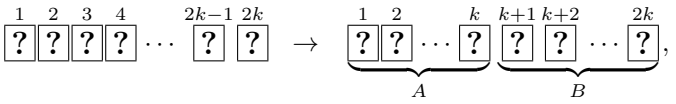
Koch et al. [3] showed the effectiveness of *non-uniform* and/or *non-closed* shuffles by constructing a four-card Las-Vegas AND protocol and a five-card finite-runtime AND protocol using them. In contrast, we focus on constructing protocols using *uniform* and *closed* shuffles, specifically, a random bisection cut. This is because it can be conveniently implemented manually [8].

2 Preliminaries

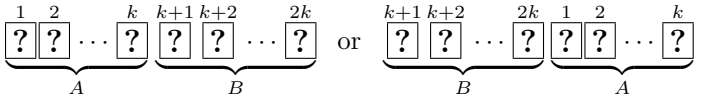
In this section, we review a few existing protocols and define fundamental notations. In Section 2.1, we introduce a random bisection cut, which is a shuffle operation used in existing and our protocols. In Sections 2.2 and 2.3, we introduce an XOR protocol [5] and an input-preserving AND protocol [7]. Although they are assumed to be used with a deck of binary cards, we will use them for various decks of cards in our protocols. In Section 2.4, we define a few notations for various decks of cards.

2.1 Random Bisection Cut

A *random bisection cut* is a shuffle operation, which can be applied to a sequence of $2k$ cards for any integer k. Given $2k$ cards, it first bisects the sequence of $2k$ cards into two sequences of k cards: A and B:



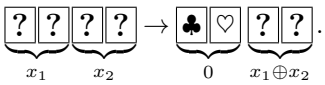
and then switches them randomly. Each case occurs with a probability of 1/2:



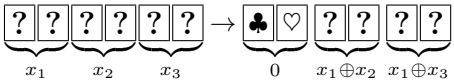
We require that nobody can learn which case will occur. Ueda et al. [8] demonstrated that a random bisection cut is implementable physically.

2.2 Existing XOR protocol

Using a deck of binary cards, Mizuki and Sone [5] constructed a four-card XOR protocol and a six-card AND protocol. The four-card XOR protocol accepts two commitments to $x_1, x_2 \in \{0,1\}$ as inputs and outputs a commitment to the XOR value $x_1 \oplus x_2$ as follows:



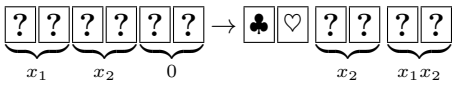
We use a modified version of the protocol: a six-card XOR protocol; it accepts three commitments to $x_1, x_2, x_3 \in \{0,1\}$ as inputs and outputs two commitments to $x_1 \oplus x_2$ and $x_1 \oplus x_3$ as follows:



The four-card XOR protocol is immediately obtained from the six-card XOR protocol by omitting the rightmost two cards. The construction of the XOR protocol is omitted due to the page limitation.

2.3 Existing Input-Preserving AND protocol

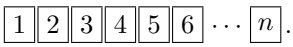
Using a deck of binary cards, Mizuki and Sone [5] constructed a six-card AND protocol; it accepts two commitments to $x_1, x_2 \in \{0,1\}$ as inputs and outputs a commitment to the AND value x_1x_2. Nishida et al. [7] improved it to a six-card input-preserving AND (hereafter, IP-AND) protocol; it outputs a commitment to the AND value x_1x_2 together with the input commitment to x_2 as follows:



The construction of the IP-AND protocol is omitted due to the page limitation.

2.4 Notations for Various Decks of Cards

Without loss of generality, we can assume that each card has a natural number on the face as follows:



As in [4], a commitment to $x \in \{0,1\}$ of base $\{i,j\}$ is denoted by

When we do not consider the base of the commitment, we denote it by

For any type-2 deck of cards, we assume that each of the numbers of $\boxed{1}$ and $\boxed{2}$ is at least 2 and call the base $\{1, 2\}$ a *special base of type-2*. Similarly, for any type-1 deck of cards, we assume that the number of $\boxed{1}$ is at least 2 and call the base containing 1 a *special base of type-1*. Unlike the type-2 case, the other card of a special base of type-1 is arbitrary. For any type-i deck, $i \in \{1, 2\}$, a commitment having a special base is denoted by

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\dagger}}.$$

We note that we can create at least two commitments having a special base because type-2 decks contain two pairs of $\boxed{1}\boxed{2}$, and type-1 decks contain two $\boxed{1}$. Although the dagger $\dagger$ is inconsequential for a type-0 deck, we use the notation $[\cdot]^{\dagger}$ even for a type-0 deck in order to express type-0/1/2 commitments simultaneously.

We denote a face-up card by $\boxed{*}$ when we do not care about the face-up symbol of the card. For example, a sequence of face-up cards $\boxed{1}\boxed{3}\boxed{4}\boxed{5}$ can be denoted by

$$\boxed{1}\boxed{*}\boxed{*}\boxed{*}.$$

Here, the special card $\boxed{1}$ is explicitly written.

3 Our Input-Preserving AND Protocol

In this section, our IP-AND protocol is presented. The key primitive is an *opaque commitment pair* (OC pair) introduced by Mizuki [4]. In Section 3.1, an OC pair is introduced. In Section 3.2, we present a new technique for producing an OC pair. In Section 3.3, we present our construction.

3.1 Opaque Commitment Pair

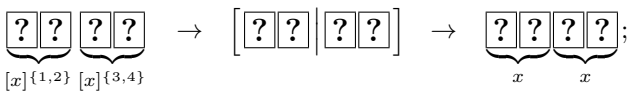
We first explain why a straight-forward application of the existing IP-AND protocol is ineffective for a general deck of cards. Suppose that the following sequence is input to the existing IP-AND protocol:

$$\underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{1,2\}}}\ \underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{3,4\}}}\ \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{5,6\}}}.$$

In the last step of the protocol [7] (Section 2.1: Improved AND Protocol), the commitment to $x_1 x_2$ is turned over, and it reveals whether the base of the commitment is $\{3, 4\}$ or $\{5, 6\}$. The former case implies that $x_1 x_2 = x_2$, and the latter case implies that $x_1 x_2 = 0$; these imply that $x_1 = 1$ and $x_1 = 0$, respectively. Therefore, the secret input x_1 is revealed publicly.

Mizuki [4] solved the above problem by producing an OC pair, which is a pair of commitments of two bases such that it is unknown as to which commitment

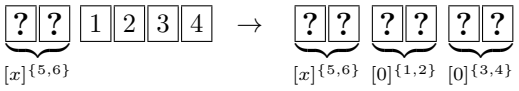
is of which base. For example, we apply a random bisection cut to a pair of two commitments of bases $\{1,2\}$ and $\{3,4\}$ as follows:



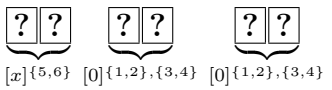
Then, the pair becomes an OC pair because it is unknown as to which commitment is of base $\{1,2\}$. We denote the above commitment by $[x]^{\{1,2\},\{3,4\}}$. Mizuki's technical idea is to use an OC pair $[x_2]^{\{3,4\},\{5,6\}}$ and $[0]^{\{3,4\},\{5,6\}}$ rather than $[x_2]^{\{3,4\}}$ and $[0]^{\{5,6\}}$. Now, the existing IP-AND protocol is effective because the (revealed) base of the commitment to x_1x_2 is independent of the secret input x_1.

We call a protocol for producing an OC pair of $(x,0)$ from a commitment to x an *OC pair generation*. Mizuki's OC pair generation for type-0 decks proceeds as follows:

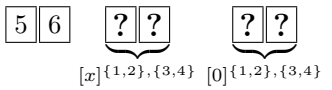
1. Place six cards as follows:



2. Apply a random bisection cut to the rightmost four cards; then, we have an opaque commitment pair to two 0s as follows:

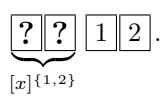


3. Apply the four-card XOR protocol (Section 2.2); then, we have an OC pair of $(x,0)$:

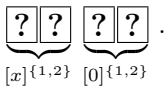


For type-2 decks, an OC pair is not required when the input commitment has the special base $\{1,2\}$ and two cards $\boxed{1}\boxed{2}$ are free. We regard the following trivial protocol as an OC pair generation for type-2 decks:

1. Place four cards as follows:



2. Turning over the face-up cards, we have



Now, we have OC pair generations for type-0 and type-2 decks. In the next section, we present a new technique of OC pair generation for type-1 decks.

3.2 New Technique of Opaque Commitment Pair Generation

In this section, we present a new technique of OC pair generation for type-1 decks. It produces an opaque commitment pair $[x]^{\{1,2\},\{1,3\}}$ and $[0]^{\{1,2\},\{1,3\}}$ from a commitment $[x]^{\{1,*\}}$ with three free cards $\boxed{1}\boxed{2}\boxed{3}$. The key observation is that the left card of the commitment $[0]^{\{1,2\},\{1,3\}}$ is always $\boxed{1}$. It proceeds as follows:

1. Place a commitment to x with three cards as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,*\}}}\ \boxed{1}\boxed{2}\boxed{3}.$$

2. Turn over all the face-up cards; then, apply a random bisection cut to the rightmost two cards:

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,*\}}}\ \boxed{?}\left[\boxed{?}\middle|\boxed{?}\right].$$

3. Apply the four-card XOR protocol (Section 2.2) to the first and second commitments; then, we have

$$\boxed{1}\boxed{*}\ \underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,2\},\{1,3\}}}\ \boxed{?}.$$

4. Rearrange the order of the sequence according to a permutation (1 3)(2 5 4):

$$\boxed{1}\boxed{*}\ \underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,2\},\{1,3\}}}\ \boxed{?}\ \rightarrow\ \underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,2\},\{1,3\}}}\ \boxed{1}\boxed{?}\boxed{*}.$$

5. Turn over $\boxed{1}$; then, we have an opaque commitment pair:

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,2\},\{1,3\}}}\ \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{1,2\},\{1,3\}}}\ \boxed{*}.$$

We denote an OC pair by $[x]_{\mathsf{OC}}$ and $[0]_{\mathsf{OC}}$. For type-0 decks, it comprises $[x]^{\{1,2\},\{3,4\}}$ and $[0]^{\{1,2\},\{3,4\}}$. For type-1 decks, it comprises $[x]^{\{1,2\},\{1,3\}}$ and $[0]^{\{1,2\},\{1,3\}}$. For type-2 decks, it comprises $[x]^{\{1,2\}}$ and $[0]^{\{1,2\}}$.

3.3 Description of Our Input-Preserving AND protocol

In this section, we construct a new IP-AND protocol by our OC pair generation technique. It proceeds as follows:

1. Place two commitments to x_1, x_2 with free cards as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{[x_2]^\dagger}\underbrace{\boxed{*}\cdots\boxed{*}}_{\text{free cards}},$$

   where the free cards are as follows:

$$\text{type-2}: \boxed{1}\boxed{2} \quad \text{type-1}: \boxed{1}\boxed{*}\boxed{*} \quad \text{type-0}: \boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

2. Apply the OC pair generation technique to the commitment to x_2 with the free cards; then, we have

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{[x_2]_{\mathsf{OC}}}\underbrace{\boxed{?}\boxed{?}}_{[0]_{\mathsf{OC}}}\underbrace{\boxed{*}\cdots\boxed{*}}_{\text{free cards}},$$

   where the OC pair is as follows:
   - type-2 : $\{1, 2\}$
   - type-1 : $\{1, 2\}, \{1, 3\}$
   - type-0 : $\{1, 2\}, \{3, 4\}$

   and the free cards are as follows:

$$\text{type-2}: \text{ none} \quad \text{type-1}: \boxed{*} \quad \text{type-0}: \boxed{*}\boxed{*}.$$

3. Apply the existing IP-AND protocol to the leftmost six cards; then, we have

$$\underbrace{\boxed{?}\boxed{?}}_{x_2}\underbrace{\boxed{?}\boxed{?}}_{[x_1x_2]^\dagger}\underbrace{\boxed{*}\cdots\boxed{*}}_{\text{free cards}},$$

   where the free cards are as follows:

$$\text{type-2}: \boxed{1}\boxed{2} \quad \text{type-1}: \boxed{1}\boxed{*}\boxed{*} \quad \text{type-0}: \boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

4 Our Protocol for Any Boolean Function

In this section, we construct a protocol for any n-variable Boolean function using any type-i deck of $2n + 8 - i$ cards. This establishes Theorem 1. In Section 4.1, a swap protocol, which is a subprotocol of our AND–XOR protocol, is constructed. In Section 4.2, the AND–XOR protocol, which is a subprotocol of our main protocol, is constructed. In Section 4.3, the main protocol is constructed.

4.1 Swap Protocol

It appears that a general-purpose protocol could be immediately obtained by plugging our IP-AND protocol into Nishida's AND–XOR protocol. However, this is not true for type-2 and type-1 decks. This is because the number of duplicate cards such as $\boxed{1}$ is limited. Therefore, we have to *reuse* the duplicate cards a number of times. It is feasible to reuse them by constructing a *swap protocol*, which exchanges the bases of two commitments. It proceeds as follows:

1. Place the two commitments $[x_1]^{\{1,2\}}, [x_2]^{\{3,4\}}$ with two free cards $\boxed{5}\boxed{6}$ as follows:
$$\underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{1,2\}}}\ \underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{3,4\}}}\ \boxed{5}\boxed{6} \ \rightarrow\ \underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{1,2\}}}\ \underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{3,4\}}}\ \underbrace{\boxed{?}\boxed{?}}_{[0]^{\{5,6\}}}.$$
2. Apply the four-card XOR protocol (Section 2.2) to the first and third commitments; then, we have
$$\boxed{1}\boxed{2}\ \underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{3,4\}}}\ \underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{5,6\}}}.$$
3. Turn over the face-up cards, and apply the four-card XOR protocol to the second and first commitments; then, we have
$$\underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{1,2\}}}\ \boxed{3}\boxed{4}\ \underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{5,6\}}}.$$
4. Turn over the face-up cards, and apply the four-card XOR protocol to the third and second commitments; then, we have
$$\underbrace{\boxed{?}\boxed{?}}_{[x_2]^{\{1,2\}}}\ \underbrace{\boxed{?}\boxed{?}}_{[x_1]^{\{3,4\}}}\ \boxed{5}\boxed{6}.$$

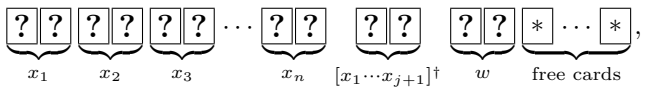
4.2 AND–XOR Protocol

In this subsection, we present our *AND–XOR protocol* based on various decks of cards. Given a collection of $n+1$ commitments to $x_1, x_2, \cdots, x_n, w \in \{0,1\}$, an AND–XOR protocol produces a commitment to $w \oplus x_1 x_2 \cdots x_n$ by preserving a collection of n commitments to $x_1, x_2, \cdots, x_n$.

Our AND–XOR protocol uses $2n+8-i$ cards for type-i decks as follows:

1. Place $2n+8-i$ cards as follows:
$$\underbrace{\boxed{?}\boxed{?}}_{[x_1]^\dagger}\underbrace{\boxed{?}\boxed{?}}_{[x_2]^\dagger}\underbrace{\boxed{?}\boxed{?}}_{x_3}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n}\underbrace{\boxed{?}\boxed{?}}_{w}\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{*}\cdots\boxed{*}}_{\text{free cards}},$$
where the free cards are as follows:
$$\text{type-2}: \text{ no cards.} \quad \text{type-1}: \boxed{*} \quad \text{type-0}: \boxed{*}\boxed{*}$$
2. Apply the six-card XOR protocol (Section 2.2) to the commitments to $x_1, 0$, and 0; then, we have
$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{[x_2]^\dagger}\underbrace{\boxed{?}\boxed{?}}_{x_3}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n}\underbrace{\boxed{?}\boxed{?}}_{w}\underbrace{\boxed{*}\cdots\boxed{*}}_{\text{free cards}}.$$
with the following free cards:
$$\text{type-2}: \boxed{1}\boxed{2} \quad \text{type-1}: \boxed{1}\boxed{*}\boxed{*} \quad \text{type-0}: \boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

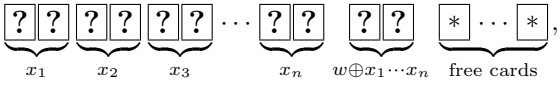
3. For $j = 1$ to $n-1$, adopt the following procedure:
   (a) Apply our IP-AND protocol (Section 3.3) to the commitment to $x_1 \cdots x_j$ and the commitment $[x_{j+1}]^\dagger$ with free cards; then, we have



   with the following free cards:

$$\text{type-2}: \boxed{1}\boxed{2} \quad \text{type-1}: \boxed{1}\boxed{*}\boxed{*} \quad \text{type-0}: \boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

   If $j = n-1$, skip Step 3-(b), and go to Step 4.
   (b) If the deck is type-0, do not take action. Otherwise, apply the swap protocol to the commitment $[x_1 \cdots x_{j+1}]^\dagger$ and the commitment to x_{j+2}.
4. Apply the four-card XOR protocol (Section 2.2) to the commitment $[x_1 \cdots x_n]^\dagger$ and the commitment to w; then, we have



   where the free cards are as follows:

$$\text{type-2}: \boxed{1}\boxed{2}\boxed{1}\boxed{2} \quad \text{type-1}: \boxed{1}\boxed{*}\boxed{1}\boxed{*}\boxed{*} \quad \text{type-0}: \boxed{*}\boxed{*}\boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

4.3 Description of Our Protocol for Any Boolean Function

In this subsection, we prove Theorem 1 by constructing a protocol for any n-variable Boolean function f based on various decks of cards.

Similar to Nishida et al.'s construction, our construction is based on the fact that any n-variable function $f(x_1, x_2, x_3, \cdots, x_n)$ can be expressed as the *Shannon expansion*:

$$\begin{aligned} f(x_1, x_2, x_3, \cdots, x_n) = {} & \bar{x}_1\bar{x}_2\bar{x}_3 \cdots \bar{x}_n f(0,0,0,\cdots,0) \oplus x_1\bar{x}_2\bar{x}_3 \cdots \bar{x}_n f(1,0,0,\cdots,0) \\ & \oplus \bar{x}_1 x_2 \bar{x}_3 \cdots \bar{x}_n f(0,1,0,\cdots,0) \oplus x_1 x_2 \bar{x}_3 \cdots \bar{x}_n f(1,1,0,\cdots,0) \\ & \oplus \cdots \oplus x_1 x_2 x_3 \cdots x_n f(1,1,1,\cdots,1). \end{aligned}$$

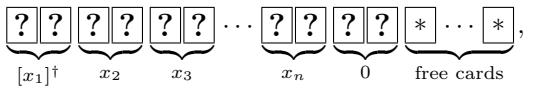
That is, the function f can be expressed as

$$f(x_1, x_2, x_3, \cdots, x_n) = T_1 \oplus T_2 \oplus \cdots \oplus T_\ell,$$

where each T_i is the AND value of n literals such as $\bar{x}_1 x_2 \bar{x}_3 \cdots x_n$.

It proceeds as follows:

1. Place $2n + 8 - i$ cards as follows:



   where the free cards are as follows:

$$\text{type-2}: \boxed{1}\boxed{2}\boxed{*}\boxed{*} \quad \text{type-1}: \boxed{1}\boxed{*}\boxed{*}\boxed{*}\boxed{*} \quad \text{type-0}: \boxed{*}\boxed{*}\boxed{*}\boxed{*}\boxed{*}\boxed{*}.$$

2. Let $T_1 \oplus T_2 \oplus \cdots \oplus T_\ell$ be the Shannon expansion of f. For $i = 1, 2, \cdots, \ell$, add T_i to the rightmost commitment by using our AND–XOR protocol[5].
3. Output the rightmost commitment.

Note that the numbers of cards used in the above construction are identical to that of our AND–XOR protocols. Thus, we obtain Theorem 1.

5 Conclusion

In this study, we showed that any n-variable Boolean function can be securely computed by using a type-i majority-free deck of $2n+8-i$ cards for $i \in \{0, 1, 2\}$. An important open problem is to determine whether $2n+8-i$ cards are necessary or not. Another noteworthy open problem is to show a similar result without majority-freeness.

Acknowledgments

This work was supported in part by JSPS KAKENHI Grant Numbers 17J01169 and 17K00001.

References

1. C. Crépeau and J. Kilian. Discreet solitary games. In *Advances in Cryptology - CRYPTO '93*, pp. 319–330, 1993.
2. B. den Boer. More efficient match-making and satisfiability: *The Five Card Trick*. In *Advances in Cryptology - EUROCRYPT '89*, pp. 208–217, 1989.
3. A. Koch, S. Walzer, and K. Härtel. Card-based cryptographic protocols using a minimal number of cards. In *ASIACRYPT 2015*, pp. 783–807, 2015.
4. T. Mizuki. Efficient and secure multiparty computations using a standard deck of playing cards. In *Cryptology and Network Security - 15th International Conference, CANS 2016*, pp. 484–499, 2016.
5. T. Mizuki and H. Sone. Six-card secure AND and four-card secure XOR. In *Frontiers in Algorithmics, Third International Workshop, FAW 2009*, pp. 358–369, 2009.
6. V. Niemi and A. Renvall. Solitaire zero-knowledge. *Fundam. Inform.*, 38(1-2):181–188, 1999.
7. T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone. Card-based protocols for any boolean function. In *Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015*, pp. 110–121, 2015.
8. I. Ueda, A. Nishimura, Y. Hayashi, T. Mizuki, and H. Sone. How to implement a random bisection cut. In *Theory and Practice of Natural Computing - 5th International Conference, TPNC 2016*, pp. 58–69, 2016.

[5] Prior to executing the AND–XOR protocol, for each complementary literal $\bar{x}_j$ in T_i, we negate the corresponding commitment by swapping the two cards.