Card-minimal Protocols for Symmetric Boolean Functions of More Than Seven Inputs*

Hayato Shikata[1], Kodai Toyoda[1], Daiki Miyahara[2,3], and Takaaki Mizuki[1,3]

[1] Tohoku University, Sendai, Japan
[2] The University of Electro-Communications, Tokyo, Japan
[3] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract. Secure computations enable us to obtain the output value of a predetermined function while keeping its input values secret. Card-based cryptography realizes secure computations using a deck of physical cards. Because each input bit is typically encoded with two cards, an obvious lower bound on the number of required cards is $2n$ when securely computing an n-input Boolean function. Although card-based protocols often require helping cards (aside from $2n$ cards needed for input), there exist several protocols that require no helping card, namely, helping-card-free protocols. For example, there are helping-card-free protocols for several fundamental functions, such as the AND, XOR, and three-input majority functions. However, in general, it remains an open problem whether all Boolean functions have their helping-card-free protocols. In this study, we focus our attention on symmetric functions: Whereas the best known result is that any n-input symmetric function can be securely computed using two helping cards, we present a helping-card-free protocol for an arbitrary n-input symmetric function such that $n > 7$. Because much attention has been drawn to constructing card-based protocols using the minimum number of cards, our protocol, which is card-minimal, would be of interest to the research area of card-based cryptography.

Keywords: Card-based Cryptography · Secure computation · Real-life hands-on cryptography · Symmetric function

1 Introduction

A *secure computation*, whose concept was first brought by Yao's seminal paper [38], enables us to obtain the output value of a predetermined function while

* This paper appears in Proceedings of ICTAC 2022. This version of the contribution has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/978-3-031-17715-6_25. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use: https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms.

keeping the input values secret. Various techniques for secure computations have been proposed so far (cf. [5]). While "computer-based (digital)" secure computations have been mainly studied and developed, "physical-tool-based" secure computations, such as using seals [27], balls [18], PEZ dispensers [1,3,28], flash lights [13], coins [12], and a deck of cards [20,21], have also been studied. Physical methods have a couple of advantages over computer-based methods; lay-people do not need to trust black boxes contained in computers and/or software, and the correctness and security of physical-tool based protocols tend to be easily understood without specialized knowledge. In this study, we focus on *card-based cryptography*, which uses a deck of physical cards to perform secure computations; refer to [9,10,25,34] for surveys.

1.1 What is Card-based Cryptography?

Since Den Boer [4] invented the *five-card trick* in 1989, many *card-based cryptographic protocols* have been proposed. In these protocols, a one-bit value is usually encoded by a pair of cards $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, as follows:

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

According to Eq. (1), when two face-down cards $\boxed{?}\boxed{?}$ (whose face side is either $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$) encode a bit $x \in \{0,1\}$, we call them a *commitment* to x, which is expressed as:

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

A card-based cryptographic protocol, or simply a *protocol*, takes commitments as input to perform a secure computation. For example, the aforementioned five-card trick [4], which is a five-card protocol as the name suggests, takes commitments to $a, b \in \{0,1\}$ and one helping card $\boxed{\heartsuit}$ as input:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{b}\boxed{\heartsuit}.$$

By applying some actions on the sequence of these five cards, such as rearranging, shuffling, and turning over cards, this protocol reveals only the value $a \wedge b$ of the AND function.

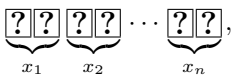
Another example is the *secure NOT computation*, which must be the simplest among all the existing protocols: Given a commitment to $x \in \{0,1\}$, switching the left and right cards of the commitment brings a commitment to its negation $\overline{x}$:

$$\underbrace{\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}}_{x} \rightarrow \underbrace{\overset{2}{\boxed{?}}\overset{1}{\boxed{?}}}_{\overline{x}},$$

where we attach the numbers above to the cards for the sake of convenience, so as to display how the cards are rearranged.

1.2 Helping-card-free Protocols

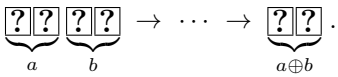
One of the most attractive topics in card-based cryptography is to design *card-minimal* protocols that use the minimum number of cards. As most of the existing protocols follow the encoding rule (1) above (which is a "two-card-per-bit" encoding), this paper also focuses only on protocols whose inputs are given according to Eq. (1). Therefore, since a one-bit value is encoded by two cards, any protocol for an n-input Boolean function $f : \{0,1\}^n \to \{0,1\}$ uses at least $2n$ cards. That is, such a protocol takes n commitments to $x_1, x_2, \ldots, x_n \in \{0,1\}$,



(2)

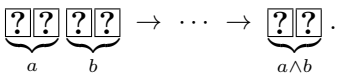
as input. If an n-input protocol does not use any (helping) card aside from the $2n$ cards for the input commitments as in Eq. (2), we call it a *helping-card-free* protocol. For example, the five-card trick [4] mentioned in Sect. 1.1 is not a helping-card-free protocol because it requires one helping card $\boxed{\heartsuit}$ to securely compute the AND function. Thus, a helping-card-free protocol for an n-input Boolean function f takes only $2n$ commitments (as in Eq. (2)) as input, and outputs only the value of $f(x_1, x_2, \ldots, x_n)$ after applying a series of actions such as shuffling and turning over cards. Note that any helping-card-free protocol is automatically a card-minimal protocol. This paper mainly deals with helping-card-free protocols (within the standard[4] computation model of card-based cryptography [24, 25]).

To the best of our knowledge, the first helping-card-free protocol in history (other than the obvious NOT computation seen above) is the XOR protocol [26], invented in 2009, which securely computes the XOR function using only two commitments to $a, b \in \{0,1\}$:



Since the output of this protocol is a commitment to $a \oplus b$, the protocol can be repeated $n-1$ times to obtain a commitment to $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ from n commitments to $x_1, x_2, \ldots, x_n$ (where we set $(a,b) = (x_1, x_2)$, $(a,b) = (x_1 \oplus x_2, x_3)$, and so on). Therefore, we immediately have a helping-card-free n-input XOR protocol.

Next came the AND protocol [23] proposed in 2012. This protocol does not produce a commitment to $a \wedge b$, but later in 2015, Koch et al. [11] constructed a helping-card-free AND protocol that outputs a commitment:



Since the output is a commitment, a helping-card-free n-input AND protocol can be constructed in a similar manner. Independently of this protocol, a helping-card-free n-input AND protocol was developed in 2016 [19]. Recently, a simpler helping-card-free 3-input AND protocol has also been devised [7].

[4] There is another computation model where private actions are allowed, e.g. [2, 14–17, 29, 32].

As for functions other than AND and XOR, a helping-card-free protocol for the 3-input majority function has been constructed very recently [37]. Also, there are helping-card-free protocols for the 3-input equality Boolean function (that outputs 1 when $x_1 = x_2 = x_3$ and 0 otherwise) [6, 35].

Ruangwises and Itoh [33] designed a helping-card-free protocol for any function in the class of the so-called "doubly symmetric" Boolean functions. Note that an n-input Boolean function $f : \{0,1\}^n \to R$ with some set R as its range is said to be *symmetric* if for any i, j, $1 \leq i, j \leq n$, the following holds:

$$f(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) = f(x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n),$$

and that an n-input Boolean function $f : \{0,1\}^n \to R$ is said to be *doubly symmetric* if f is symmetric and the following holds:

$$f(x_1, x_2, \ldots, x_n) = f(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n).$$

For example, the equality Boolean functions are doubly symmetric.

We have reviewed the existing helping-card-free protocols.

1.3 Contribution

As seen in Sect. 1.2, in the literature, we already have helping-card-free protocols for the limited classes of functions, such as the n-input AND and XOR functions and the doubly symmetric Boolean functions. Because the class of symmetric Boolean functions contains all these functions as well as many other important Boolean functions (such as threshold functions), a natural question is: Can one construct a helping-card-free protocol for any symmetric Boolean function?

As an upper bound on the number of required helping cards, in 2015, Nishida et al. [30] proved that two helping cards are sufficient for any n-input symmetric Boolean function $f : \{0,1\}^n \to \{0,1\}$ to be securely computed. Ruangwises and Itoh [33] extended the result to any range R, i.e., they constructed a two-helping-card protocol for any n-input symmetric Boolean function $f : \{0,1\}^n \to R$, where R is any set. Anyway, it is still open to determine whether one can obtain a protocol for any n-input symmetric Boolean function using fewer than two helping cards.

In this paper, we tackle this open problem. Namely, we aim to design a helping-card-free protocol for symmetric Boolean functions. Specifically, we will provide a generic construction of a helping-card-free protocol for an arbitrary n-input symmetric Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that $n \geq 8$. Therefore, we give a partial answer to the open problem.

Our generic construction relies on the two novel sub-protocols, which will be presented in Sect. 3. The first sub-protocol transforms two commitments (to $a, b \in \{0,1\}$) into the result of their addition (namely, $a + b$) without any helping card; in addition, it produces one "free" card, which is very useful because such "free" cards can be used as helping cards in another protocol. Because the result of addition is obtained as an integer in a different encoding, the second sub-protocol transforms such an integer into commitments; in other words, it

"binarizes" an integer. Making use of these two sub-protocols along with other existing protocols, we will design a generic protocol in Sect. 4. Before Sects. 3 and 4, we give some preliminaries in Sect. 2, and we conclude this paper in Sect. 5.

2 Preliminaries

In this section, we introduce a property of the symmetric functions and some of the existing protocols. Hereinafter, a symmetric Boolean function $f : \{0,1\}^n \to \{0,1\}$ is simply called a symmetric function.

2.1 Property of Symmetric Functions

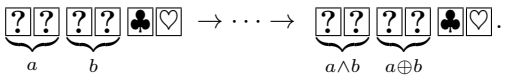
Let $f : \{0,1\}^n \to \{0,1\}$ be a symmetric function, and let $x_i \in \{0,1\}$ for every i, $1 \leq i \leq n$. It is well-known that the value of $f(x_1, \ldots, x_n)$ depends only on the summation of the inputs, i.e., $\sum_{i=1}^{n} x_i$. That is, there exists a function $g : \{0, 1, \ldots, n\} \to \{0,1\}$ such that

$$f(x_1, \ldots, x_n) = g\left(\sum_{i=1}^{n} x_i\right). \tag{3}$$

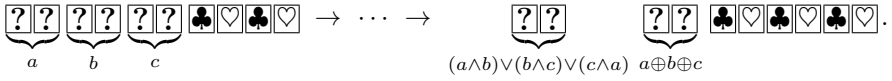
This implies that computing the summation $\sum_{i=1}^{n} x_i$ is one way for computing the symmetric function f.

2.2 Half Adder Protocol and Full Adder Protocol

A half-adder protocol is useful when computing the summation described in Sect. 2.1. The first card-based half-adder protocol was presented in 2013 [22]. After that, Nishida et al. [30] improved it by proposing a half-adder protocol with two helping cards:

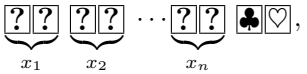


A full-adder protocol using four helping cards was presented in 2013 [22]:



2.3 Protocol for Symmetric Functions with Two Helping Cards

In 2015, Nishida et al. [30] invented a protocol for any symmetric function using two helping cards (as mentioned in Sect. 1.3). Given n commitments and two helping cards,



their protocol produces a sequence of commitments that represents the binary representation of the summation,

$$\underbrace{\boxed{?}\boxed{?}\ \boxed{?}\boxed{?}\ \cdots\ \boxed{?}\boxed{?}}_{(\sum_{i=1}^{n} x_i)_2},$$

using the half-adder protocol introduced in Sect. 2.2; then, their protocol computes $g(\sum_{i=1}^{n} x_i)$ as in Eq. (3) using that sequence. Here, for a nonnegative integer $i \in \{0, 1, \ldots, \ell\}$, we denote the binary representation of i by $(i)_2$, which is represented as $\lceil \log_2(\ell + 1) \rceil$ commitments written as

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\ \cdots\ \boxed{?}\boxed{?}}_{(i)_2}.$$

The range of the protocol above is $\{0, 1\}$, whereas Ruangwises and Itoh [33] constructed a protocol for a symmetric function $f : \{0, 1\}^n \to R$ with an arbitrary range R using two helping cards (as also mentioned in Sect. 1.3). Their protocol used the following ♣*-position* and ♡*-position encodings* (the ♣-pos. encoding and the ♡-pos. encoding for short, respectively). In the ♣-pos. encoding, for $k \geq 2$, one $\boxed{\clubsuit}$ and $k - 1$ $\boxed{\heartsuit}$s are used to represent an integer i $(0 \leq i \leq k - 1)$ by placing $\boxed{\clubsuit}$ at the $(i + 1)$-st position as follows:

$$\overset{1}{\boxed{\heartsuit}}\overset{2}{\boxed{\heartsuit}} \ldots \overset{i+1}{\boxed{\clubsuit}} \ldots \overset{k}{\boxed{\heartsuit}}.$$

In the following, we denote such a sequence of face-down cards by $E_k^{\clubsuit}(i)$, and write it as follows:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\cdots\boxed{?}}_{E_k^{\clubsuit}(i)}.$$

The ♡-pos. encoding and $E_k^{\heartsuit}(i)$ are defined in the same way with the colors (i.e., suits) reversed.

2.4 Addition of Position Encodings

Ruangwises and Itoh [33] proposed the following method for adding two integers represented in the pos. encodings.[5]

1. Assume that we have $E_k^{\clubsuit}(a)$ and $E_k^{\heartsuit}(b)$ for two integers a, b. For convenience, we name each card as follows:

$$E_k^{\clubsuit}(a) : \underset{x_0}{\boxed{?}}\underset{x_1}{\boxed{?}} \cdots \underset{x_{k-1}}{\boxed{?}}, \quad E_k^{\heartsuit}(b) : \underset{y_0}{\boxed{?}}\underset{y_1}{\boxed{?}} \cdots \underset{y_{k-1}}{\boxed{?}}.$$

[5] This method is originated from the previous protocol [36] proposed by Shinagawa et al.

2. Rearrange the sequences as follows:

$$\underset{x_0}{\boxed{?}}\overset{y_{k-1}}{\boxed{?}}\ \underset{x_1}{\boxed{?}}\overset{y_{k-2}}{\boxed{?}}\ \cdots\ \underset{x_{k-1}}{\boxed{?}}\overset{y_0}{\boxed{?}}.$$

3. Apply a *random 2-section cut* (also known as a pile-shifting shuffle [31]) to this sequence. Here, a random 2-section cut means to make each pair of cards into a single bundle and shuffle all the bundles cyclically. Thus, for a random number r, the sequence changes as follows:

$$\left[\underset{x_0}{\boxed{?}}\overset{y_{k-1}}{\boxed{?}}\,\middle|\,\underset{x_1}{\boxed{?}}\overset{y_{k-2}}{\boxed{?}}\,\middle|\,\cdots\,\middle|\,\underset{x_{k-1}}{\boxed{?}}\overset{y_0}{\boxed{?}}\right] \rightarrow \left[\underset{x_{0+r}}{\boxed{?}}\overset{y_{k-1-r}}{\boxed{?}}\,\middle|\,\underset{x_{1+r}}{\boxed{?}}\overset{y_{k-2-r}}{\boxed{?}}\,\middle|\,\cdots\,\middle|\,\underset{x_{k-1+r}}{\boxed{?}}\overset{y_{0-r}}{\boxed{?}}\right].$$

4. Rearrange them back to the first place as follows:

$$E_k^{\clubsuit}(a-r):\ \underset{x_{0+r}}{\boxed{?}}\ \underset{x_{1+r}}{\boxed{?}}\ \cdots\ \underset{x_{k-1+r}}{\boxed{?}}\ ,\quad E_k^{\heartsuit}(b+r):\ \underset{y_{0-r}}{\boxed{?}}\ \underset{y_{1-r}}{\boxed{?}}\ \cdots\ \underset{y_{k-1-r}}{\boxed{?}}\ ,$$

   where a is subtracted by the random number r and b is added by r.
5. Turn over $E_k^{\heartsuit}(b+r)$ and shift $E_k^{\clubsuit}(a-r)$ cyclically to the right by the revealed number, i.e., add $b+r$ to $a-r$; when revealing $E_k^{\heartsuit}(b+r)$, the value of b does not leak because a random value r was added to b:

$$E_k^{\clubsuit}(a-r):\ \underset{x_{0+r}}{\boxed{?}}\ \underset{x_{1+r}}{\boxed{?}}\ \cdots\ \underset{x_{k-1+r}}{\boxed{?}}\ \rightarrow\ E_k^{\clubsuit}(a+b):\ \underset{x_{0-b}}{\boxed{?}}\ \underset{x_{1-b}}{\boxed{?}}\ \cdots\ \underset{x_{k-1-b}}{\boxed{?}}\ .$$

This enables a secure computation of $(a-r)+(b+r)=a+b$ without leaking the values of a and b. That is, $E_k^{\clubsuit}(a+b)$ is obtained. Here, we did an addition of the ♣-pos. encoding and the ♡-pos. encoding, but other combinations are also feasible.

3 Building Blocks

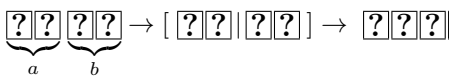
Before describing our proposed protocol in the next section, we present two novel sub-protocols, which will be used in the proposed protocol as building blocks.

3.1 Addition of Two Commitments

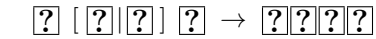
To construct our proposed protocol, we first compute the summation of inputs as implied in Sect. 2.1. For this, we propose the following addition protocol to compute $a+b$ from commitments to $a, b \in \{0,1\}$ without the need of any helping card[6]. That is, somewhat surprisingly, this novel sub-protocol is helping-card-free.

[6] This protocol is inspired by the Mizuki–Kumamoto–Sone AND protocol [23]; the procedure is the same up to the middle.

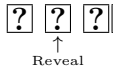
1. Apply a *random bisection cut* [26] (denoted by $[\cdots|\cdots]$) to the sequence of the commitments to a and b as follows:



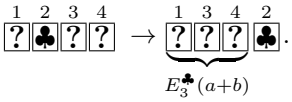
   Here, a random bisection cut is to halve a sequence and shuffle the two halves randomly.
2. Shuffle the two cards in the middle as follows:



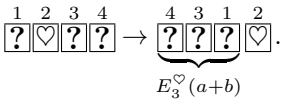
3. Reveal the second card from the left; then, either ♣ or ♡ appears with a probability of 1/2:



   (a) If ♣ appears, rearrange the sequence to obtain $a + b$ in the ♣-pos. encoding, i.e., $E_3^{\clubsuit}(a + b)$, as follows:



   (b) If ♡ appears, rearrange the sequence to obtain $a + b$ in the ♡-pos. encoding, i.e., $E_3^{\heartsuit}(a + b)$, as follows:



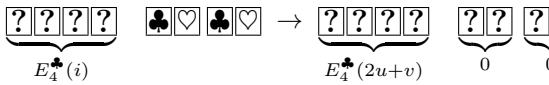
We call this protocol the *helping-card-free two-commitment addition*. The correctness and security of this addition protocol can be proved by drawing the so-called KWH-tree [8, 11]. We depict its KWH-tree in Fig. 1.

As seen above, we obtain either $E_3^{\clubsuit}(a + b)$ or $E_3^{\heartsuit}(a + b)$ (with a probability of 1/2) as well as one free card from the commitments to a and b.

3.2 Binarization of Position Encoding

Our second novel sub-protocol is to "binarize" an integer in the position encoding. Let $0 \le i \le 3$; given $E_4^{\clubsuit}(i)$ and four helping cards, this sub-protocol produces commitments to $(i)_2$. (A protocol for $E_4^{\heartsuit}(i)$ can be constructed in a similar way.)

1. Turn the four helping cards face down (resulting in two commitments to 0 because of the encoding rule (1)):



   Here, we introduce $u, v \in \{0, 1\}$ such that $i = 2u + v$, i.e., u and v are the most and least significant bits of $(i)_2$, respectively.

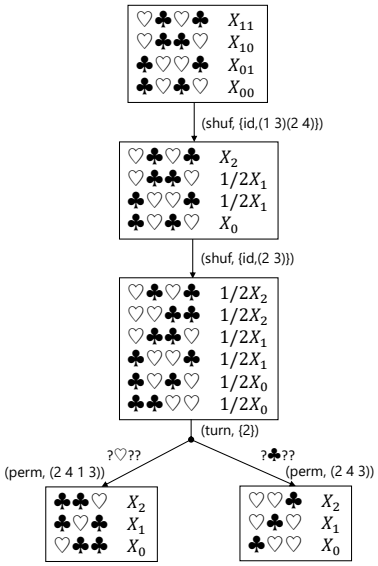


Fig. 1. KWH-tree of the helping-card-free two-commitment addition. Here, $X_0 = X_{00}$, $X_1 = X_{01} + X_{10}$, and $X_2 = X_{11}$.

2. Shuffle $E_4^{\clubsuit}(2u + v)$ and the middle commitment to 0 "synchronously" as follows.
   (a) Rearrange the sequence as follows:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{E_4^{\clubsuit}(2u+v)}\ \underbrace{\boxed{?}\boxed{?}}_{0}\ \underbrace{\boxed{?}\boxed{?}}_{0} \to \boxed{?}\boxed{?}\boxed{?}\ \boxed{?}\boxed{?}\boxed{?}\ \underbrace{\boxed{?}\boxed{?}}_{0}.$$

   (b) Apply a random bisection cut to the first six cards:

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

   (c) Rearrange the sequence as follows:

$$\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\ \ \boxed{?}\boxed{?} \to \underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{E_4^{\clubsuit}(2(u\oplus r_1)+v)}\ \underbrace{\boxed{?}\boxed{?}}_{r_1}\ \underbrace{\boxed{?}\boxed{?}}_{0}.$$

   Here, a random bit $r_1 \in \{0, 1\}$ is added to u and the middle commitment to 0 because of the random bisection cut in Step 2b.
3. Shuffle $E_4^{\clubsuit}((2u \oplus r_1) + v)$ and the right commitment to 0 "synchronously" as follows.
   (a) Rearrange the sequence as follows:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{E_4^{\clubsuit}(2(u\oplus r_1)+v)}\ \underbrace{\boxed{?}\boxed{?}}_{r_1}\ \underbrace{\boxed{?}\boxed{?}}_{0} \to \boxed{?}\boxed{?}\boxed{?}\ \boxed{?}\boxed{?}\boxed{?}\ \underbrace{\boxed{?}\boxed{?}}_{r_1}.$$

Table 1. How to swap commitments in Step 5 of the binarization protocol

Revealed Seq.	Binary	Swapping
$\underbrace{\clubsuit\heartsuit\heartsuit\heartsuit}_{E_4^{\clubsuit}(0)}$	$(0,0)$	$\boxed{?}\boxed{?}$ $\boxed{?}\boxed{?}$
$\underbrace{\heartsuit\clubsuit\heartsuit\heartsuit}_{E_4^{\clubsuit}(1)}$	$(0,1)$	$\boxed{?}\boxed{?}$ $\boxed{?}\boxed{?}$ Swap
$\underbrace{\heartsuit\heartsuit\clubsuit\heartsuit}_{E_4^{\clubsuit}(2)}$	$(1,0)$	$\boxed{?}\boxed{?}$ Swap $\boxed{?}\boxed{?}$
$\underbrace{\heartsuit\heartsuit\heartsuit\clubsuit}_{E_4^{\clubsuit}(3)}$	$(1,1)$	$\boxed{?}\boxed{?}$ Swap $\boxed{?}\boxed{?}$ Swap

(b) Apply a random bisection cut to the first six cards:

$$[\,\boxed{?}\boxed{?}\boxed{?}\,|\,\boxed{?}\boxed{?}\boxed{?}\,]\quad \boxed{?}\boxed{?} \;\to\; \boxed{?}\boxed{?}\boxed{?}\quad \boxed{?}\boxed{?}\boxed{?}\quad \boxed{?}\boxed{?}.$$

(c) Rearrange the sequence as follows:

$$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\quad \overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\quad \underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{r_1} \;\to\; \underbrace{\overset{1}{\boxed{?}}\overset{4}{\boxed{?}}\overset{2}{\boxed{?}}\overset{5}{\boxed{?}}}_{E_4^{\clubsuit}(2(u\oplus r_1)+(v\oplus r_2))}\quad \underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{r_1}\ \underbrace{\overset{3}{\boxed{?}}\overset{6}{\boxed{?}}}_{r_2}.$$

Here, a random bit $r_2 \in \{0,1\}$ is added to v and the right commitment to 0 because of applying the random bisection cut in Step 3b.

4. Reveal $E_4^{\clubsuit}(2(u \oplus r_1) + (v \oplus r_2))$:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{\text{Reveal}}\quad \boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

5. From the revealed integer value in the previous step, obtain commitments to $(i)_2$ by swapping (or not swapping) the commitments to r_1 and r_2 as shown in Table 1: Consider commitments to the binary representation of the revealed integer value (i.e., the second column of the table); if we have 1 among the two-bit sequence, we swap the corresponding commitment; if we have 0, we do not swap the commitment. By rearranging the sequence in this way, we obtain

$$\underbrace{\boxed{?}\boxed{?}}_{u}\ \underbrace{\boxed{?}\boxed{?}}_{v},\ \text{i.e.,}\ \underbrace{\boxed{?}\boxed{?}\ \boxed{?}\boxed{?}}_{(i)_2}.$$

For example, if $E_4^{\clubsuit}(2)$ appears by revealing the sequence in Step 4 as

$$\underbrace{\heartsuit\heartsuit\clubsuit\heartsuit}_{E_4^{\clubsuit}(2)}\quad \underbrace{\boxed{?}\boxed{?}}_{r_1}\ \underbrace{\boxed{?}\boxed{?}}_{r_2},$$

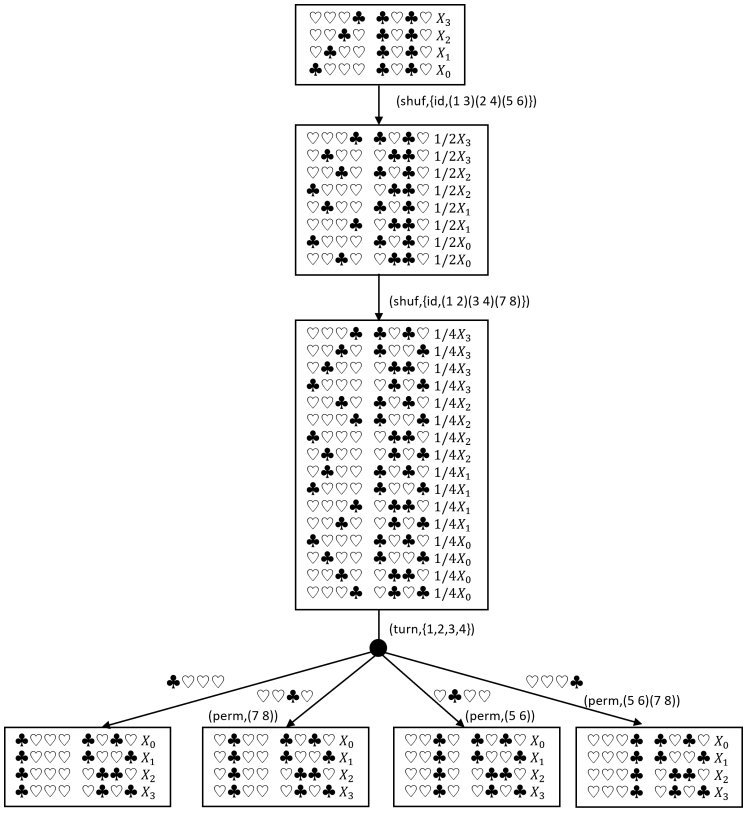


Fig. 2. KWH-tree of our binarization protocol. Note that the probability of the input sequence being $E_4^{\clubsuit}(i)$ is X_i.

we swap the middle commitment to r_1, but do not swap the right commitment as follows:

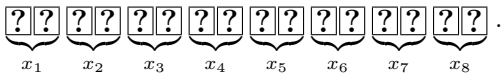
$$\underbrace{\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}}_{E_4^{\clubsuit}(2)} \quad \boxed{?}\boxed{?}\ \boxed{?}\boxed{?} \xrightarrow[\text{Swap}]{} \boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit} \quad \underbrace{\boxed{?}\boxed{?}\ \boxed{?}\boxed{?}}_{(i)_2}.$$

The correctness and security of this protocol can be confirmed by drawing its KWH-tree depicted in Fig. 2. We believe that this sub-protocol is of independent interest.

4 Our Proposed Protocol

We are ready to describe our proposed protocol for securely computing any symmetric function $f : \{0,1\}^n \to \{0,1\}$ such that $n \geq 8$. Let us assume that $n = 8$ for simplicity (the protocol can be easily extended to the case of $n \geq 9$).

Thus, the input to the protocol is a sequence of 16 cards:

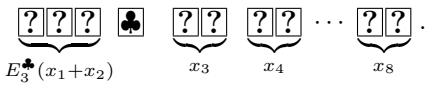


An overview of our protocol is as follows.

1. Add the inputs to obtain sequences of $x_1+x_2+x_7$, x_3+x_4, and $x_5+x_6+x_8$ using our helping-card-free two-commitment addition proposed in Sect. 3.1 and the existing addition protocol introduced in Sect. 2.4 (Sect. 4.1).
2. Binarize the sequences obtained above using our binarization protocol proposed in Sect. 3.2 (Sect. 4.2).
3. Add the binarized sequences using the existing full-adder protocol introduced in Sect. 2.2 (Sect. 4.3).

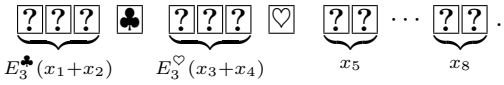
4.1 Adding the Inputs

Computing x_1+x_2. First, we obtain a sequence of x_1+x_2 from the commitments to x_1 and x_2 using our helping-card-free two-commitment addition described in Sect. 3.1. For the sake of explanation, let us assume that the addition result is obtained in the ♣-pos. encoding, i.e., we obtain $E_3^{\clubsuit}(x_1+x_2)$ without loss of generality. In this case, one free card ♣ is also obtained. In summary, the resulting sequence is as follows:

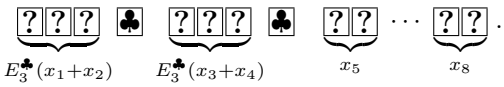


Computing x_3+x_4. We apply the helping-card-free two-commitment addition to obtain a sequence of x_3+x_4. Here, we have two possible cases with a probability of 1/2 as follows.

1. If the revealed card is ♡, the resulting sequence is as follows:



2. If it is ♣, the resulting sequence is as follows:



In the former case, we have ♣♡ as free cards. In the latter case, we have ♣♣ as free cards.

Computing $x_5 + x_6$. For each of the two cases described in the previous paragraph, an addition is done as follows.

Case 1: We have free cards $\boxed{\clubsuit}\boxed{\heartsuit}$.
We use the helping-card-free two-commitment addition for computing $x_5 + x_6$. Without loss of generality, let us assume that $\boxed{\clubsuit}$ appears in this computation. Thus, the resulting sequence is as follows:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\clubsuit}(x_1+x_2)}\ \underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\heartsuit}(x_3+x_4)}\ \underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\clubsuit}(x_5+x_6)}\ \boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\ \underbrace{\boxed{?}\boxed{?}}_{x_7}\ \underbrace{\boxed{?}\boxed{?}}_{x_8}. \tag{4}$$

Case 2: We have free cards $\boxed{\clubsuit}\boxed{\clubsuit}$.
We want to acquire a free card $\boxed{\heartsuit}$ (which is a different color from the current free cards) via the computation of $x_5 + x_6$. To achieve this, we use the existing addition protocol introduced in Sect. 2.4 to compute $x_5 + x_6$ while generating a $\boxed{\heartsuit}$. Remember that the current sequence is:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\clubsuit}(x_1+x_2)}\ \boxed{\clubsuit}\ \underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\clubsuit}(x_3+x_4)}\ \boxed{\clubsuit}\ \underbrace{\boxed{?}\boxed{?}}_{x_5}\ \cdots\ \underbrace{\boxed{?}\boxed{?}}_{x_8}.$$

Recall that the commitments to x_5 and x_6 are encoded by:

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

This can be viewed as representing an integer value at the position of $\boxed{\clubsuit}$, i.e., $E_2^{\clubsuit}(x_5)$ or $E_2^{\clubsuit}(x_6)$. If we swap the two cards comprising each commitment, they are commitments to $\overline{x}_5$ and $\overline{x}_6$ and can be viewed as $E_2^{\heartsuit}(x_5)$ and $E_2^{\heartsuit}(x_6)$, respectively, because they represent values at the position of $\boxed{\heartsuit}$. In this case, adding a $\boxed{\clubsuit}$ to the rightmost does not change the value represented, resulting in $E_3^{\heartsuit}(x_5)$ and $E_3^{\heartsuit}(x_6)$. Based on this, we compute $x_5 + x_6$ as follows[7].

1. Place each of the two $\boxed{\clubsuit}$s on the right of the commitments to $\overline{x}_5$ and $\overline{x}_6$ as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{\overline{x}_5}\boxed{\clubsuit}\ \underbrace{\boxed{?}\boxed{?}}_{\overline{x}_6}\boxed{\clubsuit}.$$

   Then, turn over the face-up cards to have $E_3^{\heartsuit}(x_5)$ and $E_3^{\heartsuit}(x_6)$:

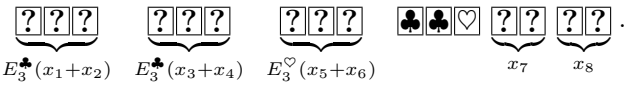
$$\underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\heartsuit}(x_5)}\ \underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\heartsuit}(x_6)}.$$

2. Apply the existing addition protocol introduced in Sect. 2.4 to the sequence to obtain $E_3^{\heartsuit}(x_5 + x_6)$:

$$\underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\heartsuit}(x_5)}\ \underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\heartsuit}(x_6)} \rightarrow \underbrace{\boxed{?}\boxed{?}\boxed{?}}_{E_3^{\heartsuit}(x_5+x_6)}\ \boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}.$$

[7] The idea of adding two pos. encodings of the same color was suggested by Kazumasa Shinagawa.

(Note that ♣♣♡ appear as free cards because the three cards in the ♡-pos. encoding are revealed.) In summary, the whole sequence is as follows:

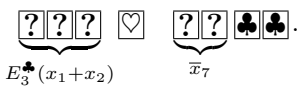


(5)

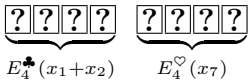
Thus, we have two possibilities of the current sequence (4) or (5). Hereinafter, we assume the sequence (5); the case for the sequence (4) will be easily handled just by exchanging "$x_3 + x_4$" and "$x_5 + x_6$" in the sequel.

Addition of $x_1 + x_2$ and x_7. As shown above, we have now three free cards ♣♣♡[8]. Next, we add $E_3^{\clubsuit}(x_1+x_2)$ to the commitment to x_7 to obtain $E_4^{\clubsuit}(x_1+x_2+x_7)$ as follows.

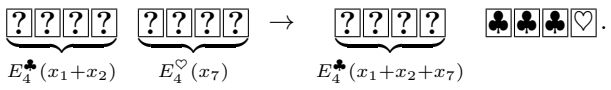
1. Place the free card ♡ on the right of $E_3^{\clubsuit}(x_1 + x_2)$ and place the two ♣s on the right of the commitment to $\overline{x}_7$ as follows:



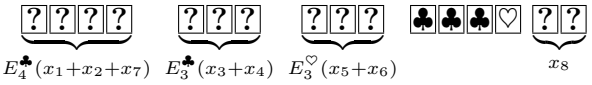
Recall that the value of $x_1 + x_2$ is represented at the position of ♣ in the sequence, and $\overline{x}_7$ is represented at the position of ♡ in the commitment. Therefore, after turning over the face-up cards, each value is represented by the position encoding as follows:



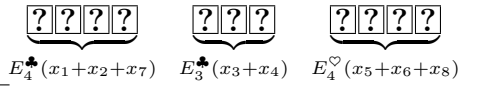
2. Apply the existing addition protocol introduced in Sect. 2.4:



In summary, the whole sequence is as follows:



Addition of x_5+x_6 and x_8 Now we have four free cards of three ♣s and one ♡. Using them, $E_3^{\heartsuit}(x_5+x_6)$, and the commitment to x_8, we obtain $E_4^{\heartsuit}(x_5+x_6+x_8)$ in a similar way as in the above paragraph:

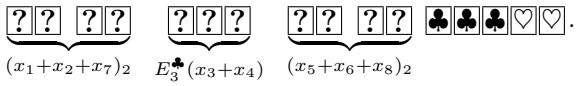


[8] Generally, there are two cards of the same color and one card of the other color.

4.2 Binarization

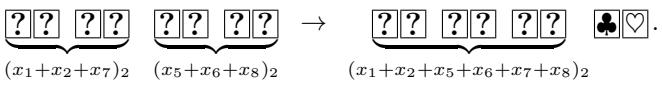
Up to now, we have at least four free cards of two ♣s and two ♡s. Using these, we binarize $E_4^{\clubsuit}(x_1+x_2+x_7)$ and $E_4^{\heartsuit}(x_5+x_6+x_8)$ by our binarization protocol proposed in Sect. 3.2.

After that, the resulting sequence is as follows:

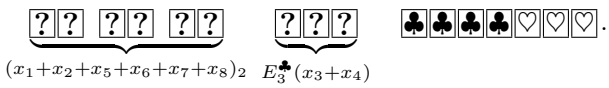


4.3 Full Adder and Binarization

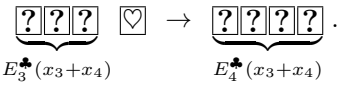
Full adder of $(x_1+x_2+x_7)_2$ *and* $(x_5+x_6+x_8)_2$. Using the existing full adder protocol introduced in Sect. 2.2, we add $(x_1+x_2+x_7)_2$ to $(x_5+x_6+x_8)_2$ with the free cards of two ♣s and two ♡s as follows:



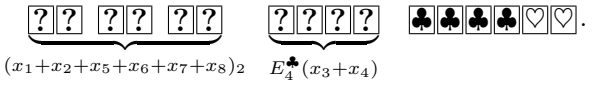
After that, the whole sequence is as follows:



Binarization of $E_3^{\clubsuit}(x_3+x_4)$ *and Overall Addition.* To obtain commitments to $(x_3+x_4)_2$, we first place a ♡ on the right of $E_3^{\clubsuit}(x_3+x_4)$ to obtain $E_4^{\clubsuit}(x_3+x_4)$ as follows:

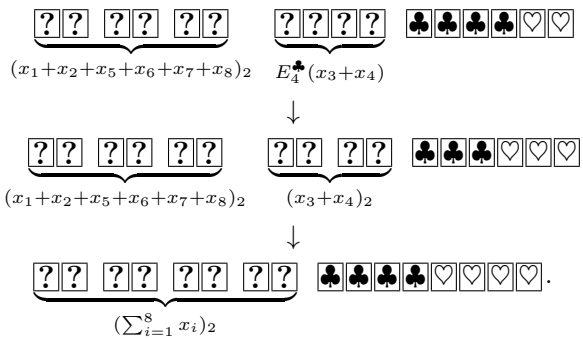


To summarize the situation, we have



Then, we binarize $E_4^{\clubsuit}(x_3+x_4)$ using our binarization protocol proposed in Sect. 3.2. Finally, we add $(x_3+x_4)_2$ to $(x_1+x_2+x_5+x_6+x_7+x_8)_2$ with the free cards using the existing full-adder protocol introduced in Sect. 2.2 as

follows:



Now we have commitments to the summation of the inputs $\sum_{i=1}^{8} x_i$. From these commitments, we compute $g(\sum_{i=1}^{8} x_i)$ as in Eq. (3).

When $n \geq 9$, it suffices to add the remaining commitments to $x_9, x_{10}, \ldots, x_n$ to the summation $\sum_{i=1}^{8} x_i$ using the existing half-adder protocol because we have enough free cards.

5 Conclusion and Future Work

In this study, we proved that any n-input symmetric function such that $n \geq 8$ can be securely computed without the need of any helping cards. That is, we provided a helping-card-free protocol for any n-input symmetric function such that $n \geq 8$. Therefore, our protocol uses the minimum number of cards, i.e., it is card-minimal.

For the case of $n = 2$, the existing protocols [11, 23, 26] immediately imply that any 2-input symmetric function can be securely computed without any helping card. Therefore, the remaining open problem is to determine whether there exists a helping-card-free protocol for any n-input symmetric function such that $3 \leq n \leq 7$.

Acknowledgements

We thank the anonymous referees, whose comments have helped us improve the presentation of the paper. We thank Kazumasa Shinagawa for the idea of computing $x_5 + x_6$ for Case 2 in Sect. 4.1. This work was supported in part by JSPS KAKENHI Grant Number JP21K11881.

References

1. Abe, Y., Iwamoto, M., Ohta, K.: Efficient private PEZ protocols for symmetric functions. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. LNCS, vol. 11891, pp. 372–392. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-36030-6_15

2. Abe, Y., Nakai, T., Kuroki, Y., Suzuki, S., Koga, Y., Watanabe, Y., Iwamoto, M., Ohta, K.: Efficient card-based majority voting protocols. New Gener. Comput. **40**, 173–198 (2022), https://doi.org/10.1007/s00354-022-00161-7
3. Balogh, J., Csirik, J.A., Ishai, Y., Kushilevitz, E.: Private computation using a PEZ dispenser. Theor. Comput. Sci. **306**(1), 69–84 (2003), https://doi.org/10.1016/S0304-3975(03)00210-X
4. 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. LNCS, vol. 434, pp. 208–217. Springer, Berlin, Heidelberg (1990), https://doi.org/10.1007/3-540-46885-4_23
5. Evans, D., Kolesnikov, V., Rosulek, M.: A pragmatic introduction to secure multi-party computation. Foundations and Trends® in Privacy and Security **2**(2-3), 70–246 (2018), https://doi.org/10.1561/3300000019
6. Heather, J., Schneider, S., Teague, V.: Cryptographic protocols with everyday objects. Formal Aspects of Computing **26**(1), 37–62 (2014), https://doi.org/10.1007/s00165-013-0274-7
7. Isuzugawa, R., Toyoda, K., Sasaki, Y., Miyahara, D., Mizuki, T.: A card-minimal three-input AND protocol using two shuffles. In: Chen, C.Y., Hon, W.K., Hung, L.J., Lee, C.W. (eds.) Computing and Combinatorics. LNCS, vol. 13025, pp. 668–679. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-89543-3_55
8. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology—ASIACRYPT 2017. LNCS, vol. 10626, pp. 126–155. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70700-6_5
9. Koch, A.: Cryptographic Protocols from Physical Assumptions. Ph.D. thesis, Karlsruhe Institute of Technology (2019), https://doi.org/10.5445/IR/1000097756
10. Koch, A.: The landscape of security from physical assumptions. In: IEEE Information Theory Workshop. pp. 1–6. IEEE, NY (2021), https://doi.org/10.1109/ITW48936.2021.9611501
11. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology—ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer, Berlin, Heidelberg (2015), https://doi.org/10.1007/978-3-662-48797-6_32
12. Komano, Y., Mizuki, T.: Coin-based secure computations. Int. J. Inf. Secur. pp. 1–14 (2022), https://doi.org/10.1007/s10207-022-00585-8, in press
13. Lafourcade, P., Mizuki, T., Nagao, A., Shinagawa, K.: Light cryptography. In: Drevin, L., Theocharidou, M. (eds.) Information Security Education. Education in Proactive Information Security. IFIPAICT, vol. 557, pp. 89–101. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-23451-5_7
14. Manabe, Y., Ono, H.: Card-based cryptographic protocols for three-input functions using private operations. In: Combinatorial Algorithms. LNCS, vol. 12757, pp. 469–484. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-79987-8_33
15. Manabe, Y., Ono, H.: Card-based cryptographic protocols with a standard deck of cards using private operations. In: Cerone, A., Ölveczky, P.C. (eds.) Theoretical Aspects of Computing – ICTAC 2021. LNCS, vol. 12819, pp. 256–274. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-85315-0_15
16. Manabe, Y., Ono, H.: Secure card-based cryptographic protocols using private operations against malicious players. In: Maimut, D., Oprina, A.G., Sauveron, D. (eds.) Innovative Security Solutions for Information Technology and Communications. LNCS, vol. 12596, pp. 55–70. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-69255-1_5

17. Manabe, Y., Ono, H.: Card-based cryptographic protocols with malicious players using private operations. New Gener. Comput. **40**, 67–93 (2022), https://doi.org/10.1007/s00354-021-00148-w
18. Miyahara, D., Komano, Y., Mizuki, T., Sone, H.: Cooking cryptographers: Secure multiparty computation based on balls and bags. In: Computer Security Foundations Symposium. pp. 1–16. IEEE, NY (2021), https://doi.org/10.1109/CSF51468.2021.00034
19. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theor. Comput. Sci. **622**(C), 34–44 (2016), https://doi.org/10.1016/j.tcs.2016.01.039
20. Mizuki, T.: Preface: Special issue on card-based cryptography. New Gener. Comput. **39**, 1–2 (2021), https://doi.org/10.1007/s00354-021-00127-1
21. Mizuki, T.: Preface: Special issue on card-based cryptography 2. New Gener. Comput. **40**, 47–48 (2022), https://doi.org/10.1007/s00354-022-00170-6
22. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Mauri, G., Dennunzio, A., Manzoni, L., Porreca, A.E. (eds.) Unconventional Computation and Natural Computation. LNCS, vol. 7956, pp. 162–173. Springer, Berlin, Heidelberg (2013), https://doi.org/10.1007/978-3-642-39074-6_16
23. 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. LNCS, vol. 7658, pp. 598–606. Springer, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-34961-4_36
24. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. Int. J. Inf. Secur. **13**(1), 15–23 (2014), https://doi.org/10.1007/s10207-013-0219-4
25. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Trans. Fundam. **E100.A**(1), 3–11 (2017), https://doi.org/10.1587/transfun.E100.A.3
26. 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. LNCS, vol. 5598, pp. 358–369. Springer, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-642-02270-8_36
27. Moran, T., Naor, M.: Basing cryptographic protocols on tamper-evident seals. Theor. Comput. Sci. **411**(10), 1283 – 1310 (2010), https://doi.org/10.1016/j.tcs.2009.10.023
28. Murata, S., Miyahara, D., Mizuki, T., Sone, H.: Public-PEZ cryptography. In: Susilo, W., Deng, R.H., Guo, F., Li, Y., Intan, R. (eds.) Information Security. LNCS, vol. 12472, pp. 59–74. Springer, Cham (2020), https://doi.org/10.1007/978-3-030-62974-8_4
29. Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: Secure computation for threshold functions with physical cards: Power of private permutations. New Gener. Comput. pp. 1–19 (2022), https://doi.org/10.1007/s00354-022-00153-7, in press
30. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any Boolean function. In: Jain, R., Jain, S., Stephan, F. (eds.) Theory and Applications of Models of Computation. LNCS, vol. 9076, pp. 110–121. Springer, Cham (2015), https://doi.org/10.1007/978-3-319-17142-5_11
31. Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. IEICE Trans. Fundam. **101**(9), 1494–1502 (2018), https://doi.org/10.1587/transfun.E101.A.1494

32. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Gener. Comput. **39**(1), 19–40 (2021), https://doi.org/10.1007/s00354-020-00113-z
33. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. Theor. Comput. Sci. **887**, 99–110 (2021), https://doi.org/10.1016/j.tcs.2021.07.007
34. Shinagawa, K.: On the Construction of Easy to Perform Card-Based Protocols. Ph.D. thesis, Tokyo Institute of Technology (2020)
35. Shinagawa, K., Mizuki, T.: The six-card trick: Secure computation of three-input equality. In: Lee, K. (ed.) Information Security and Cryptology. LNCS, vol. 11396, pp. 123–131. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-12146-4_8
36. Shinagawa, K., Mizuki, T., Schuldt, J., Nuida, K., Kanayama, N., Nishide, T., Hanaoka, G., Okamoto, E.: Card-based protocols using regular polygon cards. IEICE Trans. Fundam. **E100.A**(9), 1900–1909 (2017), https://doi.org/10.1587/transfun.E100.A.1900
37. Toyoda, K., Miyahara, D., Mizuki, T.: Another use of the five-card trick: Card-minimal secure three-input majority function evaluation. In: Adhikari, A., Küsters, R., Preneel, B. (eds.) Progress in Cryptology – INDOCRYPT 2021. LNCS, vol. 13143, pp. 536–555. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-92518-5_24
38. Yao, A.C.: Protocols for secure computations. In: Foundations of Computer Science. pp. 160–164. IEEE Computer Society, Washington, DC, USA (1982), https://doi.org/10.1109/SFCS.1982.88