



# Card-Minimal Protocols for Three-Input Functions with Standard Playing Cards

Rikuo Haga<sup>1</sup>, Yuichi Hayashi<sup>1,4</sup>, Daiki Miyahara<sup>2,4</sup>,  
and Takaaki Mizuki<sup>3,4</sup>

<sup>1</sup> Nara Institute of Science and Technology, 8916-5 Takayama,  
Ikoma, Nara 630-0192, Japan  
haga.rikuo.hm5@is.naist.jp

<sup>2</sup> The University of Electro-Communications, 1-5-1 Chofugaoka,  
Chofu, Tokyo 182-8585, Japan




<sup>3</sup> Tohoku University, Aramak-Aza-Aoba, Aoba, Sendai 980-8576, Japan

<sup>4</sup> National Institute of Advanced Industrial Science and Technology (AIST),  
2-3-26 Aomi, Koto-ku, Tokyo 135-0064, Japan



**Abstract.** A protocol realizing a secure computation using a deck of physical cards is called a card-based cryptographic protocol. Since Niemi and Renvall first proposed a few protocols using a commercially available deck of playing cards in 1999, several protocols for the two-input AND and XOR functions have been proposed. By combining these existing protocols, one can construct a protocol for any Boolean function using a standard deck of playing cards. However, the minimal numbers of cards needed for Boolean functions having more than two inputs have not been revealed so much. Recently, Koyama et al. developed a card-minimal three-input AND protocol. In this study, by extending Koyama’s AND protocol, we construct a card-minimal protocol for the three-input majority function. Furthermore, carrying the idea behind these protocols further, we provide a generic card-minimal three-input protocol, which covers many important three-input Boolean functions.

**Keywords:** Card-based cryptography · Secure computation · Standard deck of playing cards

## 1 Introduction

*Card-based cryptographic protocols* realize a secure computation using a deck of physical cards (refer to [5,25,37] for surveys). Many researches on card-based cryptography typically use a two-colored deck of cards whose fronts are red  or black  and whose backs are indistinguishable . The Boolean values are encoded as follows:

$$\img alt="black club symbol" data-bbox="381 804 396 819"/>  = 0, \quad \img alt="red heart symbol" data-bbox="496 804 511 819"/>  = 1. \tag{1}$$

When two face-down cards   represent a bit  $x \in \{0, 1\}$  according to Eq. (1), we call them a *commitment* to  $x$  and denote it as follows:

$$\underbrace{\img alt="question mark symbol" data-bbox="458 873 473 888"/> \img alt="question mark symbol" data-bbox="478 873 493 888"}_x$$

Given commitments as input, a *committed-format* protocol produces a commitment to the output value of some predetermined function. For example, the (two-input) AND protocol designed in [26] produces a commitment to  $a \wedge b$  via a series of actions, given two commitments to  $a, b \in \{0, 1\}$  and two helping cards as input:



### 1.1 Card-Based Protocols with a Standard Deck of Cards

The protocols using a two-colored deck of cards cannot be implemented with a single *standard deck* of commercially available playing cards. The reason is that such playing cards contain numbers (such as A, 2, 3, 4, . . . , J, Q, K) in addition to suits ( $\clubsuit, \heartsuit, \spadesuit, \diamond$ ), i.e., all the cards are distinct. Therefore, we need to prepare either multiple decks of playing cards or a tailor-made deck of cards to implement the protocols.

Fortunately, Niemi and Renvall [30] solved this problem by constructing a few protocols using a single standard deck of commercially available playing cards. They regarded a deck of playing cards as a total order on natural numbers from 1 to 52 because there are 52 combinations of numbers and suits in playing cards (excluding the joker); we denote these cards by  $\boxed{1}\boxed{2}\boxed{3} \dots \boxed{52}$ . In their protocols, a bit  $x \in \{0, 1\}$  is encoded using  $\boxed{i}$  and  $\boxed{j}$  satisfying  $1 \leq i < j \leq 52$ , as follows:

$$\boxed{i}\boxed{j} = 0, \quad \boxed{j}\boxed{i} = 1. \tag{2}$$

That is, if the number on the left card is smaller, it represents 0, and if the number on the left card is larger, it represents 1. Thus, similar to the two-colored-deck case (as defined in Eq. (1)), using two cards  $\boxed{i}$  and  $\boxed{j}$  (of different numbers), we can create a *commitment* to  $x \in \{0, 1\}$ , denoted by

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{i,j\}}},$$

where the set  $\{i, j\}$  is called the *base* of the commitment. (We sometimes omit the description of the base.) For example,

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{1,4\}}}$$

is a commitment to  $x$  of base  $\{1, 4\}$ ; if  $x = 0$ , the order of the sequence is  $\boxed{1}\boxed{4}$ , and if  $x = 1$ , it is  $\boxed{4}\boxed{1}$ .

### 1.2 Existing Protocols

Including the Niemi–Renvall protocols mentioned above, there are several existing protocols (working on a standard deck) in the literature, as shown in Table 1. In this subsection, we briefly review these protocols one by one.

**Table 1.** Existing protocols for Boolean functions using standard playing cards

Protocol	# of cards	# of shuffles	Finite?	Authors
2-AND	5	7.5 (exp.)		Niemi & Renvall [30]
	8	4	✓	Mizuki [22]
	4	6 (exp.)		Koch et al. [6]
2-XOR	4	7 (exp.)		Niemi & Renvall [30]
	4	1	✓	Mizuki [22]
3-AND	6	8.5 (exp.)		Koyama et al. [12]

Throughout the paper, ‘2-AND,’ ‘2-XOR,’ and ‘3-AND’ mean the two-input AND, two-input XOR, and three-input AND functions, respectively; we also use similar notations for other functions. In addition, when simply writing ‘AND protocol’ or ‘XOR protocol,’ it means a two-input protocol, i.e., a 2-AND protocol or 2-XOR protocol.

**Two-Input AND and XOR.** As mentioned in Sect. 1.1, Niemi and Renvall [30] proposed the first protocols working on a standard deck. Specifically, they constructed a protocol for the two-input AND function (namely, 2-AND) using five cards:

$$\underbrace{[?][?]_{[a]^{\{1,2\}}}} \quad \underbrace{[?][?]_{[b]^{\{3,4\}}}} \quad [5] \quad \rightarrow \quad \dots \quad \rightarrow \quad \underbrace{[?][?]_{[a \wedge b]^{\{1,4\}}}} .$$

Therefore, aside from the two input commitments to  $a, b \in \{0, 1\}$ , this AND protocol uses one helping card, namely [5]. The protocol (with the slight modification by Koch et al. [6]) uses 7.5 shuffles in expectation; thus, it is a *Las Vegas* protocol (and it is not a finite-runtime protocol). See the first protocol listed in Table 1. We call this the *Niemi–Renvall AND protocol*, whose detailed explanation will be shown in Sect. 2.4.

Niemi and Renvall [30] also constructed a 2-XOR protocol with four cards:

$$\underbrace{[?][?]_{[a]^{\{1,2\}}}} \quad \underbrace{[?][?]_{[b]^{\{3,4\}}}} \quad \rightarrow \quad \dots \quad \rightarrow \quad \underbrace{[?][?]_{[a \oplus b]^{\{1,2\}}}} .$$

Because the two input commitments need four cards as long as we follow the encoding rule in Eq. (2), this XOR protocol, which does not use any helping card, is *card-minimal*<sup>1</sup>. As shown in Table 1, the protocol uses seven shuffles in expectation.

<sup>1</sup> This paper (and the literature) assume the encoding (2), i.e., a two-card-per-bit encoding, when discussing the card-minimality of protocols; thus, an  $n$ -input (Boolean function) protocol always needs  $2n$  cards for input commitments, and such a protocol using only  $2n$  cards is card-minimal.

In 2016, Mizuki [22] proposed AND and XOR protocols with eight and four cards, respectively:

$$\underbrace{??}_{[a]^{\{1,2\}}} \underbrace{??}_{[b]^{\{3,4\}}} \underbrace{5678} \rightarrow \dots \rightarrow \underbrace{??}_{[a \wedge b]^{\{5,6\}}} \text{ or } \underbrace{??}_{[a \wedge b]^{\{7,8\}}}$$

and

$$\underbrace{??}_{[a]^{\{1,2\}}} \underbrace{??}_{[b]^{\{3,4\}}} \rightarrow \dots \rightarrow \underbrace{??}_{[a \oplus b]^{\{3,4\}}}.$$

The AND and XOR protocols use four and one shuffles, respectively, and both the protocols are finite-runtime; see Table 1. While the XOR protocol is card-minimal, the AND protocol needs four helping cards.

As seen thus far, there had been card-minimal XOR protocols, whereas no card-minimal AND protocol had been found until 2019: Koch et al. [6] constructed a card-minimal AND protocol in 2019:

$$\underbrace{??}_{[a]^{\{1,2\}}} \underbrace{??}_{[b]^{\{3,4\}}} \rightarrow \dots \rightarrow \underbrace{??}_{a \wedge b}.$$

As seen in Table 1, this is a Las Vegas protocol, which uses six shuffles in expectation.

**Three-Input AND.** If we execute the above-mentioned card-minimal 2-AND protocol designed by Koch et al. [6] twice, we can securely compute 3-AND without any helping card, although it needs 12 shuffles in expectation.

In 2021, Koyama et al. [12] improved upon this by nicely making use of the Niemi–Renvall AND protocol. That is, they proposed a card-minimal 3-AND protocol with 8.5 shuffles (in expectation):

$$\underbrace{??}_{[a]^{\{1,2\}}} \underbrace{??}_{[b]^{\{3,4\}}} \underbrace{??}_{[c]^{\{5,6\}}} \rightarrow \dots \rightarrow \underbrace{??}_{[a \wedge b \wedge c]^{\{1,4\}}}.$$

Hereinafter, we call this protocol *Koyama’s AND protocol*.

Thus, there have already been card-minimal protocols for 3-AND. In addition, one can easily construct a card-minimal 3-XOR protocol by executing one of the existing 2-XOR protocols twice. However, aside from 2-AND and 2-XOR, there are many other three-input Boolean functions, and it is open to determine whether all the three-input Boolean functions can be securely computed without any helping card.

For example, the three-input majority function  $\text{maj} : \{0, 1\}^3 \rightarrow \{0, 1\}$  defined as

$$\text{maj}(a, b, c) = \begin{cases} 0 & \text{if } a + b + c \leq 1, \\ 1 & \text{if } a + b + c \geq 2 \end{cases}$$

can be securely computed by combining the existing protocols (including the “copy” protocols, which will be mentioned in Sect. 1.4), because it suffices to apply AND, OR, and copy protocols by following a circuit such as

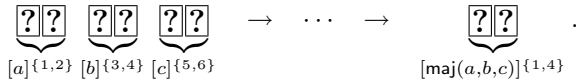
$$\text{maj}(a, b, c) = (a \wedge b) \vee (b \wedge c) \vee (c \wedge a).$$

Note that a 2-OR protocol is obtained immediately by a 2-AND protocol with De Morgan’s laws, or we will directly display a 2-OR protocol in Sect. 3.1. When following the circuit for 3-majority above, we need to duplicate some input commitments, and hence, we need some helping cards, implying that such a construction is not card-minimal. Thus, designing card-minimal protocols for three-input Boolean functions (including 3-majority) is considered to be non-trivial.

### 1.3 Contribution

In this study, we focus on designing card-minimal protocols for three-input Boolean functions by extending the idea behind Koyama’s AND protocol [12] further. Specifically, the contribution of this paper is twofold:

- For the three-input majority function, we construct a protocol using six cards, i.e., we design a card-minimal 3-majority protocol:



As will be explained, our protocol is based on Koyama’s AND protocol [12], and uses the same number of shuffles, namely 8.5 shuffles (in expectation). Note that the 3-majority is one of the most important three-input Boolean functions in terms of practical use.

- We generalize the idea behind Koyama’s AND protocol so that we obtain a generic card-minimal three-input protocol, which accommodates many three-input Boolean functions (namely, 140 functions), including important functions such as 3-OR, 3-XOR, 3-NAND, 3-NOR, 3-XNOR, and the 3-minority.

### 1.4 Related Work

Aside from the existing AND and XOR protocols introduced in Sect. 1.2, there are “copy” protocols working on a standard deck [13, 22, 30]. A copy protocol duplicates a commitment without revealing any information about the value of the commitment. Using such a copy protocol as well as 2-AND, 2-XOR, and NOT protocols<sup>2</sup>, we can construct a protocol for any Boolean function. However, determining whether there exist card-minimal protocols for multi-input functions remains an open problem (except the  $n$ -AND and  $n$ -XOR functions).

---

<sup>2</sup> A NOT protocol can be simply constructed: swapping two cards comprising a commitment produces a commitment to the negation.

There are also attractive applications using a standard deck of cards: zero-knowledge proof protocols for Sudoku [34] and millionaire protocols [18]. Moreover, under another computation model which accepts “private operations” such as revealing a card behind player’s back, card-minimal AND, XOR, and copy protocols were constructed [16].

As mentioned at the beginning of this section, many card-based protocols work on a two-color deck of cards; under several kinds of settings of decks (including the standard-deck and two-color-deck settings), the research area on card-based cryptography has grown rapidly recently from both theoretical and practical aspects. Examples are: constructing zero-knowledge proof protocols [15,32,33,35], investigating computation models [3,9,23,39] and shuffles [8,21,36], designing private-operation-model protocols [1,17,28,29,31], seeking practical and/or efficient protocols [2,7,14,20,40], and making use of other physical objects [4,11,19,27,38].

### 1.5 Outline

In Sect. 2, we introduce operations used in card-based cryptography and describe the existing protocol [30] and technique [12]. In Sect. 3, we show how to construct a three-input majority protocol by extending the ideas behind the Niemi–Renvall AND protocol and Koyama’s AND protocol. In Sect. 4, we construct a generic protocol which covers many three-input Boolean functions by generalizing the ideas further. Section 5 summarizes our study.

## 2 Preliminaries

In this section, we introduce the description of operations formalized in the computational model of card-based cryptography [24]. We also introduce the two practical shuffles called the “random cut” and “random bisection cut.” Finally, we describe the Niemi–Renvall AND protocol [30] and the useful technique [12] called the “swap operation by commitment value.”

### 2.1 Operations

We here introduce three operations, namely *rearrangement*, *turn*, and *shuffle*. We assume that we have a sequence of  $n$  face-down cards for some natural number  $n (\geq 2)$ .

*Rearrangement.* This applies some permutation  $\pi \in S_n$  to the sequence, where  $S_n$  denotes the symmetric group of degree  $n$ . This is written as  $(\text{perm}, \pi)$ , and the sequence changes as follows:

$$\begin{matrix} 1 & 2 & & n \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix} \xrightarrow{(\text{perm}, \pi)} \begin{matrix} \pi^{-1}(1) & \pi^{-1}(2) & & \pi^{-1}(n) \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix} .$$

*Turn.* This reveals the  $t$ -th card from the left in the sequence to check its number. This is written as  $(\text{turn}, \{t\})$ , and the sequence changes as follows (for example):

$$\begin{matrix} 1 & 2 & & t & & n \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} & \cdots & \boxed{?} \end{matrix} \xrightarrow{(\text{turn}, \{t\})} \begin{matrix} 1 & 2 & & t & & n \\ \boxed{?} & \boxed{?} & \cdots & \boxed{7} & \cdots & \boxed{?} \end{matrix}.$$

*Shuffle.* This applies a permutation  $\pi$  drawn from a permutation set  $\Pi \subseteq S_n$  according to a probability distribution  $\mathcal{F}$  on  $\Pi$ . This is written as  $(\text{shuf}, \Pi, \mathcal{F})$ , and the sequence changes as follows:

$$\begin{matrix} 1 & 2 & & n \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix} \xrightarrow{(\text{shuf}, \Pi, \mathcal{F})} \begin{matrix} \pi^{-1}(1) & \pi^{-1}(2) & & \pi^{-1}(n) \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{matrix} \text{ for } \pi \leftarrow \mathcal{F}.$$

Note that no one learns which permutation in  $\Pi$  was applied. If  $\mathcal{F}$  is uniform, then we simply write it as  $(\text{shuf}, \Pi)$ .

### 2.2 Random Cut

A *random cut*, denoted by  $\langle \cdot \rangle$ , is an operation that shuffles a sequence by cyclically shifting it. Applying a random cut to a sequence of  $n$  cards results in one of  $n$  possibilities, each occurring with a probability of  $1/n$ :

$$\left\langle \begin{matrix} 1 & 2 & & n-1 & n \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} & \boxed{?} \end{matrix} \right\rangle \rightarrow \begin{cases} \begin{matrix} 1 & 2 & & n-1 & n \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} & \boxed{?} \end{matrix} & (1/n), \\ \begin{matrix} 2 & 3 & & n & 1 \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} & \boxed{?} \end{matrix} & (1/n), \\ \vdots & \\ \begin{matrix} n-1 & n & & n-3 & n-2 \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} & \boxed{?} \end{matrix} & (1/n), \\ \begin{matrix} n & 1 & & n-2 & n-1 \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} & \boxed{?} \end{matrix} & (1/n). \end{cases}$$

Thus, this operation can be written as  $(\text{shuf}, \langle(1\ 2 \cdots n)\rangle)$ , where  $\langle(i_1\ i_2 \cdots i_\ell)\rangle$  denotes the cyclic group generated by a (cyclic) permutation  $(i_1\ i_2 \cdots i_\ell)$ .

### 2.3 Random Bisection Cut (RBC)

A random bisection cut (RBC) [26], denoted by  $[\cdot | \cdot]$ , is a shuffling operation, which bisects a sequence of cards and then randomly swaps the two halves. Thus, when an RBC is applied to a sequence of  $2n$  cards, the sequence becomes either the original one, or the one in which the first  $n$  cards are swapped with the last  $n$  cards, as follows:

$$\left[ \begin{matrix} 1 & & n \\ \boxed{?} & \cdots & \boxed{?} \end{matrix} \mid \begin{matrix} n+1 & & 2n \\ \boxed{?} & \cdots & \boxed{?} \end{matrix} \right] \rightarrow \begin{cases} \begin{matrix} 1 & & n & & n+1 & & 2n \\ \boxed{?} & \cdots & \boxed{?} & \mid & \boxed{?} & \cdots & \boxed{?} \end{matrix} & (1/2), \\ \begin{matrix} n+1 & & 2n & & 1 & & n \\ \boxed{?} & \cdots & \boxed{?} & \mid & \boxed{?} & \cdots & \boxed{?} \end{matrix} & (1/2). \end{cases}$$

This operation can be written as  $(\text{shuf}, \{\text{id}, (1\ n+1)(2\ n+2) \cdots (n\ 2n)\})$ , where  $\text{id}$  denotes the identity permutation.

### 2.4 The Niemi–Renvall AND Protocol

The Niemi–Renvall AND protocol [30] takes as input two commitments to  $a, b \in \{0, 1\}$  as well as an additional card and outputs a commitment to  $a \wedge b$ . This protocol proceeds as follows.

- Place the two input commitments and the additional card  $\boxed{5}$  as follows, and turn over the face-up card:

$$\boxed{5} \quad \underbrace{\boxed{??} \quad \boxed{??}}_{[a]^{\{1,2\}} \quad [b]^{\{3,4\}}} \quad \rightarrow \quad \underbrace{\boxed{?} \quad \boxed{??}}_{[a]^{\{1,2\}}} \quad \underbrace{\boxed{?} \quad \boxed{?}}_{[b]^{\{3,4\}}}.$$

- Swap the third and fourth cards:

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{matrix} \quad \rightarrow \quad \begin{matrix} 1 & 2 & 4 & 3 & 5 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{matrix}.$$

The initial and swapped sequences for each input are described in the third and fourth columns of Table 2. Observe that the order of  $\boxed{1}$ ,  $\boxed{4}$ , and  $\boxed{5}$  in the swapped sequence is  $\boxed{5} \rightarrow \boxed{4} \rightarrow \boxed{1}$  if and only if  $a \wedge b = 1$ . Therefore, we try to remove the two cards  $\boxed{2}$  and  $\boxed{3}$  in the next steps.

- Apply a random cut to the sequence:

$$\langle \boxed{?} \boxed{?} \boxed{?} \boxed{?} \boxed{?} \rangle \quad \rightarrow \quad \boxed{?} \boxed{?} \boxed{?} \boxed{?} \boxed{?}.$$

- Turn over the first card. Remove the revealed card if it is either  $\boxed{2}$  or  $\boxed{3}$ ; otherwise, turn the card face down. Return to Step 3 unless both  $\boxed{2}$  and  $\boxed{3}$  are already removed.
- Now, the sequence is one of the three possibilities as described in the fifth column of Table 2. Apply a random cut to the sequence again and then turn over the first card. We can obtain a commitment to  $a \wedge b$  (as output), as follows:<sup>3</sup>

$$\langle \boxed{?} \boxed{?} \boxed{?} \rangle \quad \rightarrow \quad \boxed{?} \boxed{?} \boxed{?} \xrightarrow{(\text{turn}, \{1\})} \left\{ \begin{array}{ll} \boxed{1} \quad \underbrace{\boxed{??}}_{[a \wedge b]^{\{4,5\}}} & (1/3), \\ \boxed{4} \quad \underbrace{\boxed{??}}_{[\overline{a \wedge b}]^{\{1,5\}}} & (1/3), \\ \boxed{5} \quad \underbrace{\boxed{??}}_{[a \wedge b]^{\{1,4\}}} & (1/3). \end{array} \right.$$

If the first card is  $\boxed{4}$ , then we obtain a commitment to the negation of  $a \wedge b$ ; we can obtain a commitment to  $a \wedge b$  by swapping the two cards comprising the commitment.

The correctness of this protocol is clear from Table 2. In addition, no information about the input and output is leaked when a card is turned over because we always apply a random cut before turning over a card.

<sup>3</sup> This step was proposed by Koch et al. [6], reducing the number of shuffles.



**Table 2.** The sequence of five cards for each input during the Niemi–Renvall protocol

Input $(a, b)$	$a \wedge b$	Initial	After swap	$\boxed{2}$ and $\boxed{3}$ removed
$(0, 0)$	0	$\boxed{5} \boxed{1} \boxed{2} \boxed{3} \boxed{4}$	$\boxed{5} \boxed{1} \boxed{3} \boxed{2} \boxed{4}$	$\boxed{1} \boxed{4} \boxed{5}$ or $\boxed{4} \boxed{5} \boxed{1}$ or $\boxed{5} \boxed{1} \boxed{4}$
$(0, 1)$	0	$\boxed{5} \boxed{1} \boxed{2} \boxed{4} \boxed{3}$	$\boxed{5} \boxed{1} \boxed{4} \boxed{2} \boxed{3}$	$\boxed{1} \boxed{4} \boxed{5}$ or $\boxed{4} \boxed{5} \boxed{1}$ or $\boxed{5} \boxed{1} \boxed{4}$
$(1, 0)$	0	$\boxed{5} \boxed{2} \boxed{1} \boxed{3} \boxed{4}$	$\boxed{5} \boxed{2} \boxed{3} \boxed{1} \boxed{4}$	$\boxed{1} \boxed{4} \boxed{5}$ or $\boxed{4} \boxed{5} \boxed{1}$ or $\boxed{5} \boxed{1} \boxed{4}$
$(1, 1)$	1	$\boxed{5} \boxed{2} \boxed{1} \boxed{4} \boxed{3}$	$\boxed{5} \boxed{2} \boxed{4} \boxed{1} \boxed{3}$	$\boxed{1} \boxed{5} \boxed{4}$ or $\boxed{4} \boxed{1} \boxed{5}$ or $\boxed{5} \boxed{4} \boxed{1}$

### 2.5 Swapping by Commitment Value

Koyama et al. [12] proposed a sub-protocol called the *swapping by commitment value* based on the idea of behind the two-input XOR protocol [22] proposed by Mizuki. This led to the construction of the 3-AND protocol [12]. Given two target cards  $\boxed{?} \boxed{?}$  and a commitment to  $c \in \{0, 1\}$  of base  $\{i, j\}$ , the swapping by commitment value is to swap the two cards  $\boxed{?} \boxed{?}$  if and only if  $c = 1$ , without leaking any information about the value of  $c$  as follows:

$$\begin{array}{c}
 \begin{array}{cc}
 \overset{1}{?} \overset{2}{?} \\
 \boxed{?} \boxed{?}
 \end{array}
 \quad
 \underbrace{\boxed{?} \boxed{?}}_{[c]^{i,j}}
 \quad
 \rightarrow
 \quad
 \begin{cases}
 \begin{array}{cc}
 \overset{1}{?} \overset{2}{?} \\
 \boxed{?} \boxed{?}
 \end{array}
 & \text{if } c = 0, \\
 \begin{array}{cc}
 \overset{2}{?} \overset{1}{?} \\
 \boxed{?} \boxed{?}
 \end{array}
 & \text{if } c = 1.
 \end{cases}
 \end{array}$$

The procedure is shown below.

1. Place the two target cards and the commitment to  $c$  as follows:

$$\begin{array}{c}
 \boxed{?} \boxed{?} \quad \boxed{?} \boxed{?} \\
 \underbrace{\hspace{1.5cm}}_{[c]^{i,j}}
 \end{array}
 .$$

2. Swap the second and third cards, i.e., apply  $(\text{perm}, (23))$ .
3. Apply  $(\text{shuf}, \{\text{id}, (13)(24)\})$ , i.e., apply an RBC as follows:

$$\boxed{\boxed{?} \boxed{?} \mid \boxed{?} \boxed{?}} \rightarrow \boxed{?} \boxed{?} \boxed{?} \boxed{?} .$$

4. Apply  $(\text{perm}, (23))$  again. Then, the sequence becomes one of the following two possibilities depending on the value of  $c$ :

$$\begin{aligned}
 c = 0 &\rightarrow \begin{cases} \begin{array}{cc} \overset{1}{?} \overset{2}{?} \\ \boxed{?} \boxed{?} \end{array} \boxed{i} \boxed{j} & (1/2), \\ \begin{array}{cc} \overset{2}{?} \overset{1}{?} \\ \boxed{?} \boxed{?} \end{array} \boxed{j} \boxed{i} & (1/2). \end{cases} \\
 c = 1 &\rightarrow \begin{cases} \begin{array}{cc} \overset{1}{?} \overset{2}{?} \\ \boxed{?} \boxed{?} \end{array} \boxed{j} \boxed{i} & (1/2), \\ \begin{array}{cc} \overset{2}{?} \overset{1}{?} \\ \boxed{?} \boxed{?} \end{array} \boxed{i} \boxed{j} & (1/2). \end{cases}
 \end{aligned}$$

Observe that the order of the first and second cards are desirable if the order of  $\boxed{i}$  and  $\boxed{j}$  is  $\boxed{i} \boxed{j}$ .

5. Turn over the third and fourth cards to reveal the order of  $\boxed{i}$  and  $\boxed{j}$ .
  - (a) If  $\boxed{i} \boxed{j}$  appears, then output the first and second cards.
  - (b) If  $\boxed{j} \boxed{i}$  appears, then swap the first and second cards and output them.

Thus, the above sub-protocol achieves the desired functionality without leaking any information about  $c$ .

### 3 Three-Input Majority Protocol

In this section, we construct a card-minimal protocol for the three-input majority function  $\text{maj}(a, b, c)$  working on a standard deck. The idea behind our proposed protocol is based on the Niemi–Renvall AND protocol [30] and Koyama’s AND protocol [12].

To construct a 3-majority protocol, we utilize the following equation:

$$\text{maj}(a, b, c) = \begin{cases} a \wedge b & \text{if } c = 0, \\ a \vee b & \text{if } c = 1. \end{cases} \tag{3}$$

To compute  $\text{maj}(a, b, c)$ , observe that, if  $c = 0$ , it suffices to compute  $a \wedge b$  using the Niemi–Renvall AND protocol introduced in Sect. 2.4; otherwise, we want to compute  $a \vee b$ . Therefore, we first construct an OR protocol by modifying the Niemi–Renvall protocol and then construct a 3-majority protocol.

#### 3.1 Two-Input or Protocol

We construct a two-input OR protocol by changing the rearrangement positions in the Niemi–Renvall AND protocol. The protocol takes as input two commitments to  $a, b$  as well as an additional card and outputs a commitment to  $a \vee b$ , as follows.

1. Place the two input commitments and the additional card  $\boxed{5}$  and turn it over as follows:

$$\begin{array}{ccc} \boxed{5} & \underbrace{\boxed{??} \boxed{??}}_{[a]^{\{1,2\}} \quad [b]^{\{3,4\}}} & \rightarrow \quad \underbrace{\boxed{??} \boxed{??}}_a \quad \underbrace{\boxed{??} \boxed{??}}_b \end{array}$$

2. Rearrange the sequence as follows, i.e., apply  $(\text{perm}, (2\ 3\ 5\ 4))$ :

$$\begin{array}{ccc} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \boxed{??} & \boxed{??} & \boxed{??} & \boxed{??} & \boxed{??} \end{array} & \rightarrow & \begin{array}{ccccc} 1 & 4 & 2 & 5 & 3 \\ \boxed{??} & \boxed{??} & \boxed{??} & \boxed{??} & \boxed{??} \end{array} \end{array}$$

The input and rearranged sequences for each input are described in the third and fourth columns of Table 3. Observe that the order of  $\boxed{1}$ ,  $\boxed{4}$ , and  $\boxed{5}$  in the rearranged sequence is  $\boxed{5} \rightarrow \boxed{4} \rightarrow \boxed{1}$  if and only if  $a \vee b = 1$ .

3. Apply Steps 3, 4, and 5 of the Niemi–Renvall AND protocol shown in Sect. 2.4 to obtain a commitment to  $a \vee b$ .

**Table 3.** The sequence of five cards for each input during the 2-OR protocol

Input( $a, b$ )	$a \vee b$	Initial	Rearranged (Step 2)	Removing <span style="border: 1px solid black; padding: 0 2px;">2</span> and <span style="border: 1px solid black; padding: 0 2px;">3</span>
(0, 0)	0	<span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">2</span> <span style="border: 1px solid black; padding: 0 2px;">3</span> <span style="border: 1px solid black; padding: 0 2px;">4</span>	<span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">3</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">2</span>	<span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">5</span> or <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> or <span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">4</span>
(0, 1)	1	<span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">2</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">3</span>	<span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">3</span> <span style="border: 1px solid black; padding: 0 2px;">2</span>	<span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> or <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">5</span> or <span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">1</span>
(1, 0)	1	<span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">2</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">3</span> <span style="border: 1px solid black; padding: 0 2px;">4</span>	<span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">3</span> <span style="border: 1px solid black; padding: 0 2px;">2</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">1</span>	<span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> or <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">5</span> or <span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">1</span>
(1, 1)	1	<span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">2</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">3</span>	<span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">2</span> <span style="border: 1px solid black; padding: 0 2px;">3</span> <span style="border: 1px solid black; padding: 0 2px;">1</span>	<span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> or <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">1</span> <span style="border: 1px solid black; padding: 0 2px;">5</span> or <span style="border: 1px solid black; padding: 0 2px;">5</span> <span style="border: 1px solid black; padding: 0 2px;">4</span> <span style="border: 1px solid black; padding: 0 2px;">1</span>

### 3.2 Idea

Remember that in Step 2 of the Niemi–Renvall AND protocol and our OR protocol, we rearrange the sequence of cards, i.e., the AND protocol uses  $(\text{perm}, (3\ 4))$  and the OR protocol uses  $(\text{perm}, (2\ 3\ 5\ 4))$ .

Observe that if we apply  $(\text{perm}, (3\ 4))$ , namely

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \end{array} \rightarrow \begin{array}{cccccc} 1 & 2 & 4 & 3 & 5 & \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \end{array},$$

and apply  $(\text{perm}, (2\ 3)(4\ 5))$ , namely

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \end{array} \rightarrow \begin{array}{cccccc} 1 & 3 & 2 & 5 & 4 & \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \end{array},$$

the resulting sequence becomes the same as the one after executing Step 2 of our OR protocol. In other words,  $((2\ 3)(4\ 5))(3\ 4) = (2\ 3\ 5\ 4)$ .

Therefore, after applying  $(\text{perm}, (3\ 4))$ , if we do nothing, it results in the AND protocol. If we apply  $(\text{perm}, (2\ 3)(4\ 5))$  after applying  $(\text{perm}, (3\ 4))$ , it results in the OR protocol. Therefore, it suffices to perform the swap operation by commitment value [12] introduced in Sect. 2.5 to apply  $(\text{perm}, (2\ 3)(4\ 5))$  if and only if  $c = 1$  (see Eq. (3) again).

### 3.3 Description of Protocol

We are ready to describe the procedure for our 3-majority protocol. The protocol takes three commitments to  $a, b, c$  as input and outputs a commitment to  $\text{maj}(a, b, c)$ .

1. Place three input commitments as follows:

$$\underbrace{\boxed{?} \boxed{?}}_{[a]^{\{1,2\}}} \quad \underbrace{\boxed{?} \boxed{?}}_{[b]^{\{3,4\}}} \quad \underbrace{\boxed{?} \boxed{?}}_{[c]^{\{5,6\}}}.$$

2. Swap the second and the third cards:

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array} \rightarrow \begin{array}{cccccc} 1 & 3 & 2 & 4 & 5 & 6 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array}.$$

**Table 4.** The sequence of six cards for each input in our 3-majority protocol, where the sequences in the fourth column are in a case when the order of the revealed fifth and sixth cards are  $\boxed{5}\boxed{6}$  in Step 3d.

Input( $a, b, c$ )	$\text{maj}(a, b, c)$	Initial	After Swap (Step 3d)	Removing $\boxed{2}\boxed{3}\boxed{6}$
(0, 0, 0)	0	$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}$	$\boxed{1}\boxed{3}\boxed{2}\boxed{4}\boxed{5}\boxed{6}$	$\boxed{1}\boxed{4}\boxed{5}$
(0, 0, 1)	0	$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{6}\boxed{5}$	$\boxed{3}\boxed{1}\boxed{4}\boxed{2}\boxed{5}\boxed{6}$	$\boxed{1}\boxed{4}\boxed{5}$
(0, 1, 0)	0	$\boxed{1}\boxed{2}\boxed{4}\boxed{3}\boxed{5}\boxed{6}$	$\boxed{1}\boxed{4}\boxed{2}\boxed{3}\boxed{5}\boxed{6}$	$\boxed{1}\boxed{4}\boxed{5}$
(0, 1, 1)	1	$\boxed{1}\boxed{2}\boxed{4}\boxed{3}\boxed{6}\boxed{5}$	$\boxed{4}\boxed{1}\boxed{3}\boxed{2}\boxed{5}\boxed{6}$	$\boxed{4}\boxed{1}\boxed{5}$
(1, 0, 0)	0	$\boxed{2}\boxed{1}\boxed{3}\boxed{4}\boxed{5}\boxed{6}$	$\boxed{2}\boxed{3}\boxed{1}\boxed{4}\boxed{5}\boxed{6}$	$\boxed{1}\boxed{4}\boxed{5}$
(1, 0, 1)	1	$\boxed{2}\boxed{1}\boxed{3}\boxed{4}\boxed{6}\boxed{5}$	$\boxed{3}\boxed{2}\boxed{4}\boxed{1}\boxed{5}\boxed{6}$	$\boxed{4}\boxed{1}\boxed{5}$
(1, 1, 0)	1	$\boxed{2}\boxed{1}\boxed{4}\boxed{3}\boxed{5}\boxed{6}$	$\boxed{2}\boxed{4}\boxed{1}\boxed{3}\boxed{5}\boxed{6}$	$\boxed{4}\boxed{1}\boxed{5}$
(1, 1, 1)	1	$\boxed{2}\boxed{1}\boxed{4}\boxed{3}\boxed{6}\boxed{5}$	$\boxed{4}\boxed{2}\boxed{3}\boxed{1}\boxed{5}\boxed{6}$	$\boxed{4}\boxed{1}\boxed{5}$

3. Apply the swap operation by the commitment to  $c$  [12] to apply (perm, (1 2)(3 4)) if and only if  $c = 1$  as follows:

(a) Rearrange the sequence as follows:

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{matrix} \rightarrow \begin{matrix} 1 & 3 & 5 & 2 & 4 & 6 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{matrix}.$$

(b) Apply (shuf, {id, (1 4)(2 5)(3 6)}), i.e., apply an RBC as follows:

$$\left[ \boxed{?}\boxed{?}\boxed{?} \mid \boxed{?}\boxed{?}\boxed{?} \right] \rightarrow \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

(c) Rearrange the sequence as follows:

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{matrix} \rightarrow \begin{matrix} 1 & 4 & 2 & 5 & 3 & 6 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{matrix}.$$

(d) Turn over the fifth and sixth cards. If their order is  $\boxed{5}\boxed{6}$ , do nothing; if it is  $\boxed{6}\boxed{5}$ , swap the first and second cards as well as the third and fourth cards. The sequence for each input is described in Table 4 where the order of the revealed two cards  $\boxed{5}\boxed{6}$  does not matter.

4. Execute Steps 3, 4, and 5 of the Niemi–Renvall AND protocol to obtain a commitment to  $\text{maj}(a, b, c)$ , where we use the first through fourth cards as input, and the  $\boxed{5}$  turned over in Step 3d as an additional card (i.e., place the  $\boxed{5}$  in the first from the left).

### 3.4 Correctness and Security

The correctness of this protocol is clear from Table 4 because when the input  $(a, b, c)$  satisfies  $\text{maj}(a, b, c) = 0$ , the resulting sequence after Step 3 is  $\boxed{1}\boxed{4}\boxed{5}$

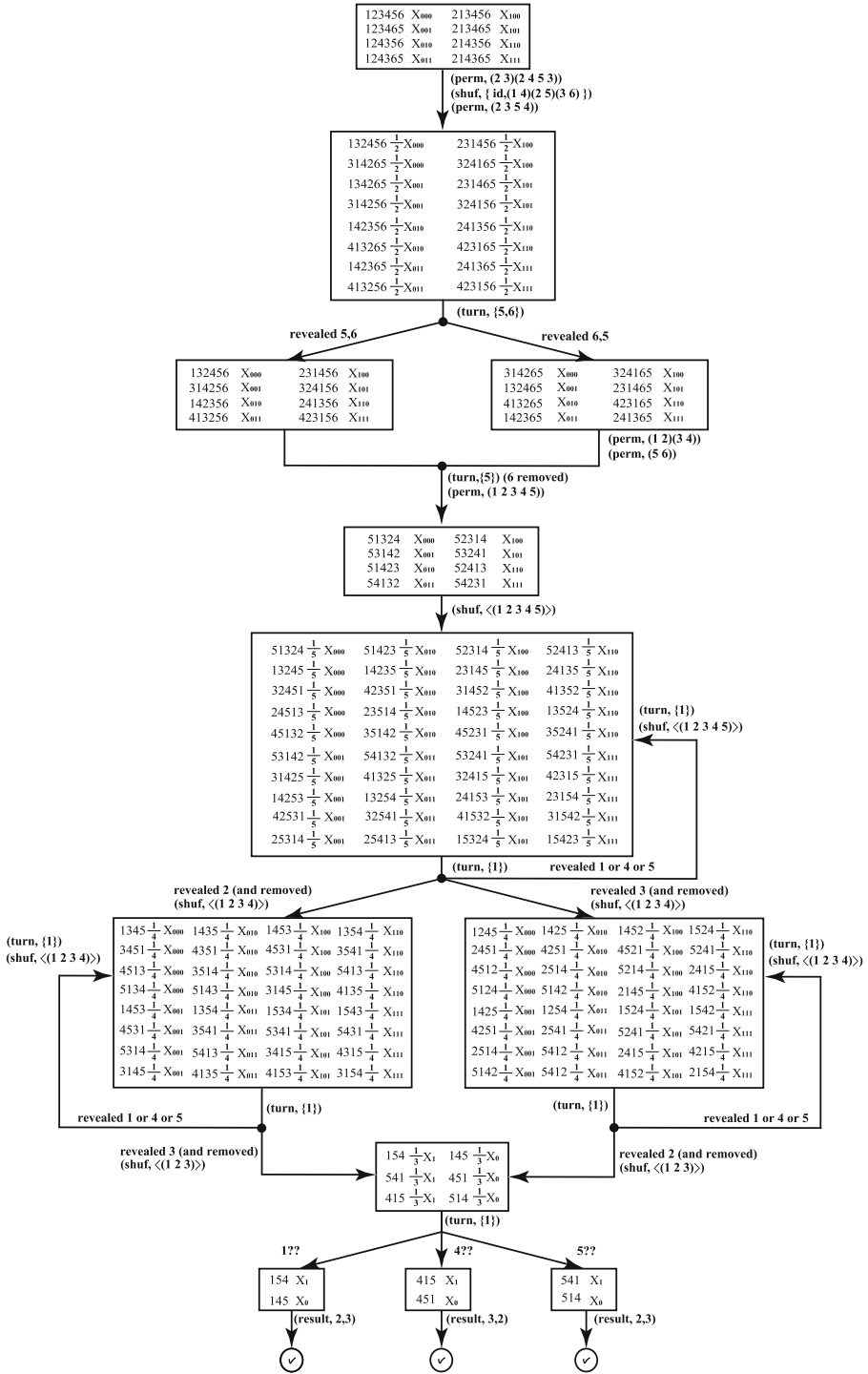


Fig. 1. The KWH-tree of three-input majority protocol

(where  $\boxed{2}$ ,  $\boxed{3}$ , and  $\boxed{6}$  are removed); otherwise, the sequence is  $\boxed{4}\boxed{1}\boxed{5}$ . As for the security, we execute the swap operation by commitment value  $\boxed{12}$  in Step 3 and then the part of the Niemi–Renvall AND protocol (and Steps 1 and 2 just place and swap the sequence, respectively), no information about the input and output is leaked.

More formally, we use the *KWH-tree* [10] to prove the security (and correctness) of this protocol; we depict the KWH-tree of our three-input majority protocol in Fig. 1. In the diagram, states of a sequence of cards are expressed as nodes, and operations on the sequence of cards are expressed as edges. Because the sum of the probability distributions of the nodes is equal to the probability distribution of the input, the protocol is guaranteed to be secure.

### 4 Generic Protocol for Three-Input Functions

In this section, we generalize our 3-majority protocol described in Sect. 3 so as to obtain a generic card-minimal protocol for three-input Boolean functions.

After we describe the idea behind the generalization in Sect. 4.1, we generalize the Niemi–Renvall AND protocol and the swap operation by commitment value  $\boxed{12}$  in Sects 4.2 and 4.3, respectively.

Before going into the subsections, we define a notation; hereinafter,  $\pi_{ijkl}$  denotes a permutation in  $S_4$  such that

$$\pi_{ijkl} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ i & j & k & \ell \end{pmatrix}$$

for four distinct integers  $i, j, k, \ell \in \{1, 2, 3, 4\}$ . For example,  $\pi_{1234} = \text{id}$  and  $\pi_{1324} = (2\ 3)$ .

#### 4.1 Idea

Our idea is that, as  $\text{maj}(a, b, c)$  is represented with the two elementary functions of  $a$  and  $b$  depending on the value of  $c$  (as in Eq. (3)), every three-input Boolean function  $f : \{0, 1\}^3 \rightarrow \{0, 1\}$  can be also written as follows:

$$f(a, b, c) = \begin{cases} g(a, b) & \text{if } c = 1, \\ h(a, b) & \text{if } c = 0, \end{cases} \tag{4}$$

where there exist two functions  $g, h : \{0, 1\}^2 \rightarrow \{0, 1\}$ .

Remember that in our 3-majority protocol proposed in Sect. 3, we first apply the permutation  $\pi_{1324}$  (to compute 2-AND) and then, if  $c = 1$ , apply the permutation  $\pi_{2143}$  by the swap operation (to compute 2-OR);  $\pi_{1324}$  leads to 2-AND, and  $\pi_{2143}\pi_{1324}$  leads to 2-OR. If we replace these two permutations with other ones, then we will obtain (possibly) another three-input protocol.

Bearing this in mind, we first investigate what two-input function (as a candidate for  $g$  or  $h$  in Eq. (4)) will be computed for every permutation  $\pi_{ijkl} \in S_4$  (in Sect. 4.2). Then, we enumerate all possible swap operations (in Sect. 4.3).

**Table 5.** Output  $\text{NR}^\pi$  of the generalized Niemi–Renvall protocol with  $\pi$

Permutation $\pi$	$\text{NR}^\pi(a, b)$
$\pi_{1234}, \pi_{1243}, \pi_{2134}, \pi_{2143}$	0
$\pi_{1324}$	$a \wedge b$
$\pi_{1423}$	$a \wedge \bar{b}$
$\pi_{1342}, \pi_{1432}$	$a$
$\pi_{2314}$	$\bar{a} \wedge b$
$\pi_{3124}, \pi_{3214}$	$b$
N/A	$a \oplus b$
$\pi_{3142}$	$a \vee b$
$\pi_{2413}$	$\bar{a} \vee \bar{b}$
N/A	$\bar{a} \oplus \bar{b}$
$\pi_{4123}, \pi_{4213}$	$\bar{b}$
$\pi_{4132}$	$a \vee \bar{b}$
$\pi_{2341}, \pi_{2431}$	$\bar{a}$
$\pi_{3241}$	$\bar{a} \vee b$
$\pi_{4231}$	$\bar{a} \wedge \bar{b}$
$\pi_{3412}, \pi_{3421}, \pi_{4312}, \pi_{4321}$	1

### 4.2 Generalizing the Niemi–Renvall and Protocol

In this subsection, we generalize the Niemi–Renvall AND protocol by considering all permutations for Step 2 of the protocol.

Assume that we apply a permutation  $\pi \in S_4$  (instead of the original permutation) in Step 2 of the Niemi–Renvall AND protocol. Then, at the end of the protocol, we should obtain a commitment to a certain two-input function; we denote this function by  $\text{NR}^\pi : \{0, 1\}^2 \rightarrow \{0, 1\}$ .

We examined all  $4!$  possibilities for  $\pi$  and write  $\text{NR}^\pi(a, b)$  in Table 5. This table tells us that aside from 2-XOR and 2-XNOR, all two-input functions can be obtained.

### 4.3 Generalizing Swap Operation by Commitment Value

In this subsection, consider all possible swapping operations.

Assume that we have four cards along with a commitment to  $c \in \{0, 1\}$  of base  $\{5, 6\}$ :

$$\boxed{?}\boxed{?}\boxed{?}\boxed{?} \underbrace{\boxed{?}\boxed{?}}_{[c]^{\{5,6\}}}.$$

We want to apply a permutation in  $S_4$  to the first four cards if and only if  $c = 1$ . What are the possible permutations? We can consider two kind of swap operations.

*(i j)-swap.* Remember that the swap operation introduced in Sect. 2.5 swaps two cards (or does not) depending on the value of  $c$ . As a natural extension, let us consider a swap operation such that the  $i$ -th and  $j$ -th cards (among the leftmost four cards) for  $1 \leq i < j \leq 4$  are swapped or not; we call this the  $(i j)$ -swap, which can be achieved as follows.

1. Apply the permutation corresponding to  $(i j)$  according to Table 6.
2. Apply  $(\text{shuf}, \{\text{id}, (35)(46)\})$ , i.e., apply an RBC as follows:

$$\boxed{??} \left[ \boxed{??} \mid \boxed{??} \right] \rightarrow \boxed{??} \boxed{??} \boxed{??} \boxed{??}.$$

3. Apply the inverse of the permutation applied in Step 1.
4. Turn over the fifth and sixth cards (namely, apply  $(\text{turn}, \{5, 6\})$ ). If the order of the revealed cards are  $\boxed{6} \boxed{5}$ , swap the  $i$ -th and  $j$ -th cards (namely,  $(\text{perm}, (i j))$ ); otherwise, do nothing.

*(i j)(k ℓ)-swap.* Remember that our 3-majority protocol uses  $(\text{perm}, (2\ 3)(4\ 5))$  in the swap operation, and note that the permutations  $(2\ 3)$  and  $(4\ 5)$  are disjoint. Therefore, we can consider a swap operation such that the  $i$ -th and  $j$ -th cards as well as the  $k$ -th and  $\ell$ -th cards are swapped or not for  $1 \leq i < j \leq 4$  and  $\{k, \ell\} = \{1, 2, 3, 4\} - \{i, j\}$ ; we call this the  $(i j)(k \ell)$ -swap, which can be achieved as follows.

1. Apply the permutation corresponding to  $(i j)(k \ell)$  according to Table 7.
2. Apply  $(\text{shuf}, \{\text{id}, (14)(25)(36)\})$ , i.e., apply an RBC as follows:

$$\left[ \boxed{??} \boxed{??} \mid \boxed{??} \boxed{??} \right] \rightarrow \boxed{??} \boxed{??} \boxed{??} \boxed{??}.$$

3. Apply the inverse of the permutation applied in Step 1.
4. Turn over the fifth and sixth cards (namely, apply  $(\text{turn}, \{5, 6\})$ ). If the order of the revealed cards are  $\boxed{6} \boxed{5}$ , swap the  $i$ -th and  $j$ -th cards as well as  $k$ -th and  $\ell$ -th cards (namely,  $(\text{perm}, (i j)(k \ell))$ ); otherwise, do nothing.

#### 4.4 Description of Protocol

We are now ready to describe our generic protocol for three-input Boolean function.

Our protocol owns two permutations  $\pi, \sigma \in S_4$  as parameter, where either  $\sigma = (i j)$  for  $1 \leq i < j \leq 4$ , or  $\sigma = (i j)(k \ell)$  for  $1 \leq i < j \leq 4$  and  $\{k, \ell\} = \{1, 2, 3, 4\} - \{i, j\}$ ; it proceeds as follows.

1. Place three input commitments as:

$$\underbrace{\boxed{??}}_{[a]^{\{1,2\}}} \quad \underbrace{\boxed{??}}_{[b]^{\{3,4\}}} \quad \underbrace{\boxed{??}}_{[c]^{\{5,6\}}}.$$

2. Apply  $(\text{perm}, \pi)$ .
3. Apply  $\sigma$ -swap.
4. Apply Steps 3, 4, and 5 of the Niemi–Renvall AND protocol.



**Table 6.** Actions for Steps 1 and 3 of the  $(i j)$ -swap

$(i j)$	Action for Step 1	Action for Step 3 (Inverse of Step 1)
(1 2)	(perm, (1 3)(2 5 4))	(perm, (1 3)(2 4 5))
(1 3)	(perm, (1 3 5 4 2))	(perm, (1 2 4 5 3))
(1 4)	(perm, (1 3 2)(4 5))	(perm, (1 2 3)(4 5))
(2 3)	(perm, (2 3 5 4))	(perm, (2 4 5 3))
(2 4)	(perm, (2 3)(4 5))	(perm, (2 3)(4 5))
(3 4)	(perm, (4 5))	(perm, (4 5))

**Table 7.** Actions for Steps 1 and 3 of the  $(i j)(k \ell)$ -swap

$(i j)(k \ell)$	Action for Step 1	Action for Step 3 (Inverse of Step 1)
(1 2)(3 4)	(perm, (2 4 5 3))	(perm, (2 3 5 4))
(1 3)(2 4)	(perm, (3 4 5))	(perm, (3 5 4))
(1 4)(2 3)	(perm, (3 5))	(perm, (3 5))

### 4.5 Covered Functions

In this subsection, we comprehensively reveal what three-input functions our generic protocol computes.

Executing our generic protocol with parameter  $\pi, \sigma \in S_4$  is equivalent to executing a protocol for the three-input Boolean function  $f$  such that

$$f(a, b, c) = \begin{cases} \text{NR}^{\sigma\pi}(a, b) & \text{if } c = 1, \\ \text{NR}^{\pi}(a, b) & \text{if } c = 0. \end{cases}$$

For example, if we take  $\pi, \sigma$  as in the first and second columns of Table 8, we have  $\text{NR}^{\pi}$  and  $\text{NR}^{\sigma\pi}$  as in the fourth and fifth columns, and hence, the corresponding three-input Boolean functions are shown in the sixth column. This table tells us that major three-input Boolean functions are covered by our generic protocol.

From the user’s perspective, given a three-input function  $f$ , we want to find two permutations  $\pi, \sigma \in S_4$  which lead to  $f$ . Table 9 helps us: We first find  $g, h$  such that

$$f(a, b, c) = \begin{cases} g(a, b) & \text{if } c = 1, \\ h(a, b) & \text{if } c = 0; \end{cases}$$

then, using Table 9, find the corresponding parameter  $\pi, \sigma$ .

Although not all three-input Boolean functions have a corresponding parameter  $\pi, \sigma$ , our generic protocol covers 140 three-input Boolean functions among the 256 ones.

**Table 8.** Covered main functions

$\pi$	$\sigma$	$\sigma\pi$	$NR^\pi$	$NR^{\sigma\pi}$	$f(a, b, c)$
$\pi_{1234}$	(2 3)	$\pi_{1324}$	AND	0	3-AND [12]
$\pi_{3412}$	(2 3)	$\pi_{3142}$	1	OR	3-OR
$\pi_{3412}$	(1 4)(2 3)	$\pi_{2143}$	NAND	OR	3-XOR
$\pi_{4321}$	(2 3)	$\pi_{4231}$	NAND	1	3-NAND
$\pi_{2413}$	(2 3)	$\pi_{2143}$	0	NOR	3-NOR
$\pi_{2413}$	(1 3)(2 4)	$\pi_{1324}$	AND	NOR	3-XNOR
$\pi_{1324}$	(1 2)(3 4)	$\pi_{3142}$	OR	AND	3-majority
$\pi_{2413}$	(1 2)(3 4)	$\pi_{4231}$	NOR	NAND	3-minority

**Table 9.** Parameter  $\pi, \sigma$  leading to  $g, h$

$\begin{matrix} g \\ h \end{matrix}$	0	$a \wedge b$	$a \wedge \bar{b}$	$a$	$\bar{a} \wedge b$	$b$	$a \oplus b$	$a \vee b$	$\overline{a \vee b}$	$\overline{a \oplus b}$	$\bar{b}$	$a \vee \bar{b}$	$\bar{a}$	$\bar{a} \vee b$	$\overline{a \wedge b}$	1
0	$\pi_{1234}$ (3 4)	$\pi_{1234}$ (2 3)	$\pi_{1243}$ (2 3)	$\pi_{1234}$ (2 4)	$\pi_{2134}$ (2 3)	$\pi_{1234}$ (1 3)	N/A	$\pi_{2143}$ (1 4)	$\pi_{2143}$ (2 3)	N/A	$\pi_{2143}$ (1 3)	$\pi_{2134}$ (1 4)	$\pi_{2143}$ (2 4)	$\pi_{1243}$ (1 4)	$\pi_{1234}$ (1 4)	$\pi_{2143}$ (1 4)(2 3)
$a \wedge b$	$\pi_{1324}$ (2 3)	N/A	$\pi_{1324}$ (2 4)	$\pi_{1324}$ (3 4)	$\pi_{1324}$ (1 3)	$\pi_{1324}$ (1 2)	N/A	$\pi_{1324}$ (1 2)(3 4)	$\pi_{1324}$ (1 3)(2 4)	N/A	N/A	N/A	N/A	N/A	$\pi_{1324}$ (1 4)(2 3)	$\pi_{1324}$ (1 4)
$a \wedge \bar{b}$	$\pi_{1423}$ (2 3)	$\pi_{1423}$ (2 4)	N/A	$\pi_{1423}$ (3 4)	$\pi_{1423}$ (1 3)(2 4)	N/A	N/A	$\pi_{1423}$ (1 2)	N/A	$\pi_{1423}$ (1 3)	N/A	$\pi_{1423}$ (1 2)	$\pi_{1423}$ (1 2)(3 4)	N/A	$\pi_{1423}$ (1 4)(2 3)	$\pi_{1423}$ (1 4)
$a$	$\pi_{1342}$ (2 4)	$\pi_{1342}$ (3 4)	$\pi_{1432}$ (3 4)	$\pi_{1432}$ (2 3)	N/A	$\pi_{1342}$ (1 2)(3 4)	N/A	$\pi_{1342}$ (1 2)	N/A	N/A	$\pi_{1342}$ (1 3)(2 4)	$\pi_{1432}$ (1 2)	$\pi_{1432}$ (1 4)(2 3)	N/A	N/A	$\pi_{1432}$ (1 3)
$\bar{a} \wedge b$	$\pi_{2314}$ (2 3)	$\pi_{2314}$ (1 3)	$\pi_{2314}$ (1 3)(2 4)	N/A	N/A	$\pi_{2314}$ (1 2)	N/A	N/A	$\pi_{2314}$ (2 4)	N/A	N/A	$\pi_{2314}$ (1 4)(2 3)	$\pi_{2314}$ (3 4)	$\pi_{2314}$ (1 2)(3 4)	N/A	$\pi_{2314}$ (1 4)
$b$	$\pi_{3124}$ (1 3)	$\pi_{3124}$ (1 2)	N/A	$\pi_{3124}$ (1 2)(3 4)	$\pi_{3214}$ (1 2)	$\pi_{3124}$ (2 3)	N/A	$\pi_{3124}$ (3 4)	N/A	N/A	$\pi_{3214}$ (1 4)(2 3)	N/A	$\pi_{3214}$ (1 2)(3 4)	$\pi_{3214}$ (3 4)	N/A	$\pi_{3124}$ (2 4)
$a \oplus b$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$a \vee b$	$\pi_{3142}$ (1 4)	$\pi_{3142}$ (1 2)(3 4)	N/A	$\pi_{3142}$ (1 2)	N/A	$\pi_{3142}$ (3 4)	N/A	N/A	$\pi_{3142}$ (1 4)(2 3)	N/A	N/A	$\pi_{3142}$ (1 3)	N/A	$\pi_{3142}$ (2 4)	$\pi_{3142}$ (1 3)(2 4)	$\pi_{3142}$ (2 3)
$\overline{a \vee b}$	$\pi_{2413}$ (2 3)	$\pi_{2413}$ (1 3)(2 4)	$\pi_{2413}$ (1 3)	N/A	$\pi_{2413}$ (1 4)	N/A	N/A	$\pi_{2413}$ (1 4)(2 3)	N/A	N/A	$\pi_{2413}$ (1 2)	N/A	$\pi_{2413}$ (3 4)	N/A	$\pi_{2413}$ (1 2)(3 4)	$\pi_{2413}$ (1 4)
$\overline{a \oplus b}$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$\bar{b}$	$\pi_{4123}$ (1 3)	N/A	$\pi_{4123}$ (1 2)	$\pi_{4213}$ (1 3)(2 4)	N/A	$\pi_{4123}$ (1 4)	N/A	N/A	$\pi_{4213}$ (1 2)	N/A	$\pi_{4123}$ (2 3)	$\pi_{4123}$ (3 4)	$\pi_{4123}$ (1 3)(2 4)	N/A	$\pi_{4213}$ (3 4)	$\pi_{4123}$ (2 4)
$a \vee \bar{b}$	$\pi_{4132}$ (1 4)	N/A	$\pi_{4132}$ (1 2)(3 4)	$\pi_{4132}$ (1 2)	$\pi_{4132}$ (1 4)(2 3)	N/A	N/A	$\pi_{4132}$ (1 3)	N/A	N/A	$\pi_{4132}$ (3 4)	N/A	N/A	$\pi_{4132}$ (1 3)(2 4)	$\pi_{4132}$ (2 4)	$\pi_{4132}$ (2 3)
$\bar{a}$	$\pi_{2341}$ (2 4)	N/A	N/A	$\pi_{2341}$ (1 4)	$\pi_{2341}$ (3 4)	$\pi_{2341}$ (1 2)(3 4)	N/A	N/A	$\pi_{2431}$ (3 4)	N/A	$\pi_{2341}$ (1 3)(2 4)	N/A	$\pi_{2341}$ (2 3)	$\pi_{2341}$ (1 2)	$\pi_{2431}$ (1 2)	$\pi_{2431}$ (2 3)
$\bar{a} \vee b$	$\pi_{3241}$ (1 4)	N/A	$\pi_{3241}$ (1 4)(2 3)	N/A	$\pi_{3241}$ (1 2)(3 4)	$\pi_{3241}$ (3 4)	N/A	$\pi_{3241}$ (2 4)	N/A	N/A	N/A	$\pi_{3241}$ (1 3)(2 4)	$\pi_{3241}$ (1 2)	N/A	$\pi_{3241}$ (1 3)	$\pi_{3241}$ (2 3)
$\overline{a \wedge b}$	$\pi_{4231}$ (1 4)	$\pi_{4231}$ (1 4)(2 3)	N/A	N/A	N/A	N/A	N/A	$\pi_{4231}$ (1 3)(2 4)	$\pi_{4231}$ (1 2)(3 4)	N/A	$\pi_{4231}$ (3 4)	$\pi_{4231}$ (2 4)	$\pi_{4231}$ (1 2)	$\pi_{4231}$ (1 3)	N/A	$\pi_{4231}$ (2 3)
1	$\pi_{3412}$ (1 4)(2 3)	$\pi_{4321}$ (1 4)	$\pi_{3421}$ (1 4)	$\pi_{3412}$ (1 3)	$\pi_{4312}$ (1 4)	$\pi_{3421}$ (2 4)	N/A	$\pi_{3412}$ (2 3)	$\pi_{3412}$ (1 4)	N/A	$\pi_{4312}$ (2 4)	$\pi_{4312}$ (2 3)	$\pi_{4321}$ (1 3)	$\pi_{3421}$ (2 3)	$\pi_{4321}$ (2 3)	$\pi_{3412}$ (3 4)

## 5 Conclusion

In this study, we showed how to construct a card-minimal 3-majority protocol by extending the Niemi–Renvall AND protocol [30] and Koyama’s AND pro-

tocol [12]. Furthermore, we constructed a generic card-minimal protocol that covers many three-input Boolean functions as shown in Table 9.

Although the proposed protocol accommodates many major functions as seen in Table 8, not all the three-input Boolean functions can be computed by it. It is open to determine whether there exists a six-card protocol for every three-input Boolean function. While 3-XOR and 3-XNOR can be computed without any helping card by using the existing protocols, we conjecture that some functions, say

$$f(a, b, c) = \begin{cases} a \wedge b & \text{if } c = 1, \\ a \oplus b & \text{if } c = 0, \end{cases}$$

would need helping cards.

**Acknowledgements.** We thank the anonymous referees, whose comments have helped us improve the presentation of the paper. We also thank Hiroto Koyama for his cooperation in preparing a Japanese draft version of Sect. 3 at an earlier stage of this work. This work was supported in part by JSPS KAKENHI Grant Numbers JP21K11881 and JP19H01104.

## References

1. Abe, Y., et al.: Efficient card-based majority voting protocols. *New Gener. Comput.* **40**, 173–198 (2022). <https://doi.org/10.1007/s00354-022-00161-7>
2. Abe, Y., Hayashi, Y., Mizuki, T., Sone, H.: Five-card AND computations in committed format using only uniform cyclic shuffles. *New Gener. Comput.* **39**(1), 97–114 (2021). <https://doi.org/10.1007/s00354-020-00110-2>
3. Dvořák, P., Koucký, M.: Barrington plays cards: the complexity of card-based protocols. In: Bläser, M., Monmege, B. (eds.) *Theoretical Aspects of Computer Science. LIPIcs*, vol. 187, pp. 26:1–26:17. Schloss Dagstuhl, Dagstuhl (2021). <https://doi.org/10.4230/LIPIcs.STACS.2021.26>
4. Isuzugawa, R., Miyahara, D., Mizuki, T.: Zero-knowledge proof protocol for cryptarithmic using dihedral cards. In: Kostitsyna, I., Orponen, P. (eds.) *UCNC 2021. LNCS*, vol. 12984, pp. 51–67. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-87993-8\\_4](https://doi.org/10.1007/978-3-030-87993-8_4)
5. Koch, A.: *Cryptographic protocols from physical assumptions*. Ph.D. thesis, Karlsruhe Institute of Technology (2019). <https://doi.org/10.5445/IR/1000097756>
6. Koch, A., Schrempf, M., Kirsten, M.: Card-based cryptography meets formal verification. In: Galbraith, S.D., Moriai, S. (eds.) *ASIACRYPT 2019. LNCS*, vol. 11921, pp. 488–517. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-34578-5\\_18](https://doi.org/10.1007/978-3-030-34578-5_18)
7. Koch, A., Schrempf, M., Kirsten, M.: Card-based cryptography meets formal verification. *New Gener. Comput.* **39**(1), 115–158 (2021). <https://doi.org/10.1007/s00354-020-00120-0>
8. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: Farach-Colton, M., Prencipe, G., Uehara, R. (eds.) *Fun with Algorithms. LIPIcs*, vol. 157, pp. 17:1–17:23. Schloss Dagstuhl, Dagstuhl (2020). <https://doi.org/10.4230/LIPIcs.FUN.2021.17>

9. Koch, A., Walzer, S.: Private function evaluation with cards. *New Gener. Comput.* 1–33 (2022, in press). <https://doi.org/10.1007/s00354-021-00149-9>
10. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) *ASIACRYPT 2015*. LNCS, vol. 9452, pp. 783–807. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48797-6\\_32](https://doi.org/10.1007/978-3-662-48797-6_32)
11. Komano, Y., Mizuki, T.: Coin-based secure computations. *Int. J. Inf. Secur.* 1–14 (2022, in press). <https://doi.org/10.1007/s10207-022-00585-8>
12. Koyama, H., Miyahara, D., Mizuki, T., Sone, H.: A secure three-input AND protocol with a standard deck of minimal cards. In: Santhanam, R., Musatov, D. (eds.) *CSR 2021*. LNCS, vol. 12730, pp. 242–256. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-79416-3\\_14](https://doi.org/10.1007/978-3-030-79416-3_14)
13. Koyama, H., Toyoda, K., Miyahara, D., Mizuki, T.: New card-based copy protocols using only random cuts. In: *ASIA Public-Key Cryptography Workshop*, pp. 13–22. ACM, New York (2021). <https://doi.org/10.1145/3457338.3458297>
14. Kuzuma, T., Toyoda, K., Miyahara, D., Mizuki, T.: Card-based single-shuffle protocols for secure multiple-input AND and XOR computations. In: *ASIA Public-Key Cryptography*, pp. 1–8. ACM, New York (2022, to appear). <https://doi.org/10.1145/3494105.3526236>
15. Lafourcade, P., Miyahara, D., Mizuki, T., Robert, L., Sasaki, T., Sone, H.: How to construct physical zero-knowledge proofs for puzzles with a “single loop” condition. *Theor. Comput. Sci.* **888**, 41–55 (2021). <https://doi.org/10.1016/j.tcs.2021.07.019>
16. Manabe, Y., Ono, H.: Card-based cryptographic protocols with a standard deck of cards using private operations. In: Cerone, A., Ölveczky, P.C. (eds.) *ICTAC 2021*. LNCS, vol. 12819, pp. 256–274. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-85315-0\\_15](https://doi.org/10.1007/978-3-030-85315-0_15)
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., Hayashi, Y., Mizuki, T., Sone, H.: Practical card-based implementations of Yao’s millionaire protocol. *Theor. Comput. Sci.* **803**, 207–221 (2020). <https://doi.org/10.1016/j.tcs.2019.11.005>
19. 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, New York (2021). <https://doi.org/10.1109/CSF51468.2021.00034>
20. Miyahara, D., Ueda, I., Hayashi, Y., Mizuki, T., Sone, H.: Evaluating card-based protocols in terms of execution time. *Int. J. Inf. Secur.* **20**(5), 729–740 (2020). <https://doi.org/10.1007/s10207-020-00525-4>
21. Miyamoto, K., Shinagawa, K.: Graph automorphism shuffles from pile-scramble shuffles. *New Gener. Comput.* **40**, 199–223 (2022). <https://doi.org/10.1007/s00354-022-00164-4>
22. Mizuki, T.: Efficient and secure multiparty computations using a standard deck of playing cards. In: Foresti, S., Persiano, G. (eds.) *CANS 2016*. LNCS, vol. 10052, pp. 484–499. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-48965-0\\_29](https://doi.org/10.1007/978-3-319-48965-0_29)
23. Mizuki, T., Komano, Y.: Information leakage due to operative errors in card-based protocols. *Inf. Comput.* 1–15 (2022, in press). <https://doi.org/10.1016/j.ic.2022.104910>
24. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. *Int. J. Inf. Secur.* **13**(1), 15–23 (2013). <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.) *FAW 2009. LNCS*, vol. 5598, pp. 358–369. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-02270-8\\_36](https://doi.org/10.1007/978-3-642-02270-8_36)
27. Murata, S., Miyahara, D., Mizuki, T., Sone, H.: Efficient generation of a card-based uniformly distributed random derangement. In: Uehara, R., Hong, S.-H., Nandy, S.C. (eds.) *WALCOM 2021. LNCS*, vol. 12635, pp. 78–89. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-68211-8\\_7](https://doi.org/10.1007/978-3-030-68211-8_7)
28. 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.* 1–19 (2022, in press). <https://doi.org/10.1007/s00354-022-00153-7>
29. Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: How to solve millionaires’ problem with two kinds of cards. *New Gener. Comput.* **39**(1), 73–96 (2021). <https://doi.org/10.1007/s00354-020-00118-8>
30. Niemi, V., Renvall, A.: Solitaire zero-knowledge. *Fundam. Inf.* **38**(1,2), 181–188 (1999). <https://doi.org/10.3233/FI-1999-381214>
31. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. *New Gener. Comput.* **39**(1), 19–40 (2020). <https://doi.org/10.1007/s00354-020-00113-z>
32. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Card-based ZKP for connectivity: applications to Nurikabe, Hitori, and Heyawake. *New Gener. Comput.* **40**, 149–171 (2022). <https://doi.org/10.1007/s00354-022-00155-5>
33. Robert, L., Miyahara, D., Lafourcade, P., Libralesso, L., Mizuki, T.: Physical zero-knowledge proof and NP-completeness proof of Suguru puzzle. *Inf. Comput.* 1–14 (2021, in press). <https://doi.org/10.1016/j.ic.2021.104858>
34. Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for Sudoku. *New Gener. Comput.* 1–17 (2022, in press). <https://doi.org/10.1007/s00354-021-00146-y>
35. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for ripple effect. *Theor. Comput. Sci.* **895**, 115–123 (2021). <https://doi.org/10.1016/j.tcs.2021.09.034>
36. Saito, T., Miyahara, D., Abe, Y., Mizuki, T., Shizuya, H.: How to implement a non-uniform or non-closed shuffle. In: Martín-Vide, C., Vega-Rodríguez, M.A., Yang, M.-S. (eds.) *TPNC 2020. LNCS*, vol. 12494, pp. 107–118. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-63000-3\\_9](https://doi.org/10.1007/978-3-030-63000-3_9)
37. Shinagawa, K.: On the construction of easy to perform card-based protocols. Ph.D. thesis, Tokyo Institute of Technology (2020)
38. Shinagawa, K.: Card-based cryptography with dihedral symmetry. *New Gener. Comput.* **39**(1), 41–71 (2021). <https://doi.org/10.1007/s00354-020-00117-9>
39. Takashima, K., Miyahara, D., Mizuki, T., Sone, H.: Actively revealing card attack on card-based protocols. *Nat. Comput.* 1–13 (2021, in press). <https://doi.org/10.1007/s11047-020-09838-8>
40. 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.) *INDOCRYPT 2021. LNCS*, vol. 13143, pp. 536–555. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-92518-5\\_24](https://doi.org/10.1007/978-3-030-92518-5_24)