Six Standard Playing Cards Are Sufficient for All Three-Input Boolean Functions*

Masanori Kabutomori[1] and Takaaki Mizuki[2]

[1] School of Engineering, Tohoku University, Sendai 980–8578, Japan
[2] Cyberscience Center, Tohoku University, Sendai 980–8578, Japan

Abstract. This paper deals with secure computation using a standard deck of playing cards. Since the first protocol using playing cards was devised by Niemi and Renvall in 1999, several secure computation protocols have been proposed. In 2022, Haga et al. proposed a generic protocol that securely computes three-input Boolean functions using six playing cards. However, this did not cover all three-input Boolean functions, and determining whether there exist six-card protocols for all such functions remained an open problem. In this paper, we construct protocols using six playing cards for each three-input Boolean function for which a protocol has not yet been built, thereby resolving this open problem.

Keywords: Card-based cryptography · Secure computation · Playing cards.

1 Introduction

The method of realizing secure computation and other cryptographic tasks using a physical deck of cards is referred to as *card-based cryptography* [6,17]. In many existing card-based protocols, a special deck of cards, called a "two-color deck," is used, in which their front sides are $\boxed{\heartsuit}$ and $\boxed{\clubsuit}$, and their back sides $\boxed{?}$ are indistinguishable. On the other hand, there are also several existing studies using commercially available playing cards (like $\boxed{2_\clubsuit}\boxed{6_\diamondsuit}\boxed{4_\spadesuit}$), and this paper deals with secure computation protocols using such a standard deck of playing cards.

1.1 Protocols Using Playing Cards

Secure computation protocols using commercially available playing cards were first proposed by Niemi and Renvall [20] in 1999. Since there are 52 unique

* This paper appears in Proceedings of COCOON 2026. 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/10.1007/978-981-92-3309-0_29. 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.

playing cards (excluding jokers), determined by the combination of a rank (from 1 to 13) and a suit ($\clubsuit, \spadesuit, \heartsuit, \diamondsuit$), a deck of playing cards can be mapped to integers from 1 to 52, so that we have 52 numbered cards $\boxed{1}\boxed{2}\boxed{3}\cdots$ whose backs $\boxed{?}$ are all identical. Niemi and Renvall encoded a one-bit value using a pair of cards $\boxed{i}$ and $\boxed{j}$ with $1 \leq i < j \leq 52$ as follows:

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

That is, if the card on the left is smaller, it represents 0; if the card on the left is larger, it represents 1. When a bit $x \in \{0, 1\}$ is represented with $\boxed{i}$ and $\boxed{j}$ placed face down, these two cards are called a *commitment* to x and are written as:

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

where the set $\{i, j\}$ is called the *base* of the commitment, and this notation is sometimes omitted.

Secure computation protocols are usually constructed using commitments defined above as input. For example, a secure computation of the NOT operation can be realized easily: given a commitment to x, swapping the left and right cards of the commitment yields a commitment to $\overline{x}$:

$$\underbrace{\boxed{?}\boxed{?}}_{[x]^{\{i,j\}}} \quad \rightarrow \quad \boxed{?}\boxed{?} \quad \rightarrow \quad \underbrace{\boxed{?}\boxed{?}}_{[\overline{x}]^{\{i,j\}}}.$$

1.2 Previous Research and Open Problem

The aforementioned Niemi and Renvall [20] specifically proposed an AND protocol:

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

That is, given two commitments to $a, b \in \{0, 1\}$, it produces a commitment[3] to $a \wedge b$. This protocol uses 5 cards and 7.5 shuffles (expected value), the outline of which will be presented in Section 2.4. Niemi and Renvall also constructed XOR and copy protocols[4].

Following this research, several protocols have been devised. The progress regarding AND and XOR protocols is shown in Table 1. As can be seen from this table, protocols using four cards exist for both AND and XOR. Since the

[3] Such a protocol that produces a commitment as output (without leaking any information) is called a *committed-format* protocol; this paper only considers committed-format protocols.

[4] This implies that any Boolean function can be securely computed (provided that sufficient numbers of cards and shuffles are available) because we can construct a protocol for it by combining the AND, XOR, and copy protocols based on its circuit.

Table 1. Existing protocols for two-input Boolean functions (where #Cards and #Shuffles mean the numbers of required cards and shuffles, respectively).

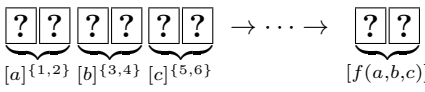
Function	#Cards	#Shuffles	Authors
AND	5	7.5 (exp.)	Niemi & Renvall [20]
	8	4	Mizuki [15]
	4	6 (exp.)	Koch et al. [7]
XOR	4	7 (exp.)	Niemi & Renvall [20]
	4	1	Mizuki [15]
	4	1	Koyama et al. [10]

Table 2. Existing protocols for three-input Boolean functions (#Cards and #Shuffles mean the numbers of required cards and shuffles, respectively). For details on the "140 functions," see Section 3.1.

Function	#Cards	#Shuffles	Authors
3-AND	6	8.5 (exp.)	Koyama et al. [9]
140 functions	6	8.5 (exp.)	Haga et al. [2]

two-bit input is given as two commitments, four cards are necessary, making these protocols optimal in terms of the number of cards. Furthermore, protocols for all other two-input Boolean functions besides AND and XOR can also be constructed with four cards by modifying the AND or XOR protocol (along with the NOT computation). Thus, the problem of finding *card-minimal* protocols for all two-input Boolean functions was already resolved.

Next, we look at three-input Boolean functions $f : \{0,1\}^3 \to \{0,1\}$. How about the card-minimality problem for three inputs? We want to construct a protocol using only three input commitments to $a, b, c \in \{0,1\}$, i.e., only six cards:



In 2021, Koyama et al. [9] constructed a three-input AND protocol (with six cards) with an expected number of 8.5 shuffles. Subsequently, by extending this protocol, Haga et al. [2] constructed six-card protocols for 140 out of 256 three-input Boolean functions. These are listed in Table 2. Thus, the previous studies did not fully cover three-input Boolean functions.

Therefore, whether six-card protocols can be constructed for the remaining 116 three-input Boolean functions is considered an open problem, and we tackle this problem.

1.3 Our Contribution

The main contribution of this paper is to show that six-card protocols can be constructed for all 116 of the three-input Boolean functions previously unaddressed, thereby completely resolving this open problem.

Since dealing with all 256 three-input Boolean functions individually is tedious, we consider *NPN equivalence classes*. In NPN equivalence classes, functions that can be transformed into each other by negation of input variables (Negation), permutation of input variable order (Permutation), and negation of the output result (Negation) are considered to be in the same equivalence class. In card-based protocols, the NOT operation and variable swapping are trivial, so if a secure computation protocol exists for a certain function, a protocol also exists for all functions in the same NPN equivalence class (with the same numbers of cards and shuffles). Therefore, for each equivalence class, we only need to construct a protocol for a single representative function.

It is known that there are a total of 14 NPN equivalence classes for three-input Boolean functions [23]. We list the representative functions[5] for the 10 equivalence classes excluding the trivial ones (i.e., 0, a, $a \wedge b$, and $a \oplus b$):

$$
\begin{aligned}
\mathsf{And3}(a,b,c) &= a \wedge b \wedge c \\
\mathsf{XorAnd}(a,b,c) &= a \wedge (b \oplus c) \\
\mathsf{OrAnd}(a,b,c) &= a \wedge (b \vee c) \\
\mathsf{Onehot}(a,b,c) &= (a \wedge \overline{b} \wedge \overline{c}) \oplus (\overline{a} \wedge b \wedge \overline{c}) \oplus (\overline{a} \wedge \overline{b} \wedge c) \\
\mathsf{Majority}(a,b,c) &= (a \wedge b) \vee (b \wedge c) \vee (a \wedge c) \\
\mathsf{Equality}(a,b,c) &= (a \wedge b \wedge c) \oplus (\overline{a} \wedge \overline{b} \wedge \overline{c}) \\
\mathsf{Dot}(a,b,c) &= a \oplus (c \vee (a \wedge b)) \\
\mathsf{Mux}(a,b,c) &= (a \wedge b) \vee (\overline{a} \wedge c) \\
\mathsf{AndXor}(a,b,c) &= a \oplus (b \wedge c) \\
\mathsf{Xor3}(a,b,c) &= a \oplus b \oplus c\,.
\end{aligned}
$$

Among these 10 representatives, the ones that cannot be handled by the method of Haga et al. [2] are three: Onehot, Dot, and Xor3. Since Xor3 can be realized by repeating existing two-input XOR protocols [15, 20] twice, the truly unresolved ones are Onehot and Dot. For these, we construct six-card protocols in this paper, whose number of shuffles is 10.5 (expected value). Table 3 summarizes the previous research and the results of this paper. Thus, this paper proves that there always exists a six-card protocol for every three-input Boolean function, successfully closing the open problem.

1.4 Related Work

Closely related to the setting of this paper, partial opening actions on playing cards were devised in [14], followed by [4, 5]; efficient protocols have been

[5] The names of the functions are partially based on literature [12].

Table 3. Number of shuffles for six-card three-input Boolean function protocols

	And3	XorAnd	OrAnd	Onehot	Majority	Equality	Dot	Mux	AndXor	Xor3
Koyama et al. [9]	8.5	—	—	—	—	—	—	—	—	—
Haga et al. [2]	8.5	8.5	8.5	—	8.5	8.5	—	8.5	8.5	—
Reducing to 2-input	—	7	—	—	—	—	—	—	7	2
This paper	—	—	—	10.5	—	—	10.5	—	—	—

designed. If private operations are allowed, efficient protocols can be obtained [11, 19].

Aside from secure computations, Ruangwises [22] developed zero-knowledge proof protocols for Sudoku using playing cards. Such zero-knowledge proof protocols for puzzles have been constructed with the help of a two-color deck, e.g., [1, 3, 21].

1.5 Organization of This Paper

The remainder of this paper is organized as follows. In Section 2, we describe the computational model of card-based protocols, and introduce some kinds of shuffles and previous methods. In Section 3, we classify the protocols of Haga et al. [2] into NPN equivalence classes. In Section 4, we present our protocols to resolve the open problem. Finally, Section 5 provides a conclusion.

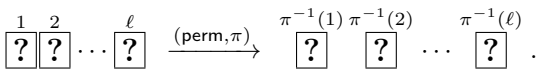
2 Preliminaries

In this section, we briefly introduce the computational model of card-based protocols, two kinds of shuffles, and existing methods.

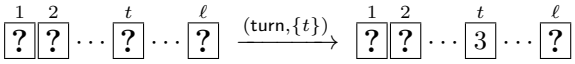
2.1 Computational Model of Card-Based Protocols

Card-based protocols are formally defined via abstract machines [16]. In the computational model, a protocol applies a series of three actions, turn, perm, and shuf, to a sequence of cards. We briefly explain these three actions below.

Permute. Apply a certain permutation $\pi \in S_\ell$ to a sequence of cards. Here, S_ℓ represents the symmetric group of degree ℓ, and this action is written as (perm, π):



Turn. Turn over the t-th card from the left of a sequence. This action is written as $(\mathsf{turn}, \{t\})$:



Shuffle. For a sequence of cards, apply a permutation $\pi \in \Pi$ chosen uniformly at random from a certain set of permutations $\Pi \subseteq S_\ell$. This action is written as (shuf, Π):

$$\overset{1}{?}\,\overset{2}{?}\cdots\overset{\ell}{?} \xrightarrow{(\mathsf{shuf},\Pi)} \overset{\pi^{-1}(1)}{?}\;\overset{\pi^{-1}(2)}{?}\;\cdots\;\overset{\pi^{-1}(\ell)}{?}\ .$$

We here assume that nobody knows which permutation was applied among the permutations included in Π.

2.2 Random Cut

A random cut is a shuffle that cyclically shifts a sequence of cards. When a random cut is applied to a sequence of ℓ cards, it becomes one of the ℓ possible sequences, each with a probability of $1/\ell$, written as follows:

$$\left\langle \overset{1}{?}\,\overset{2}{?}\cdots\overset{\ell-1}{?}\,\overset{\ell}{?} \right\rangle \rightarrow \begin{cases} \overset{1}{?}\,\overset{2}{?}\ \cdots\ \overset{\ell-1}{?}\,\overset{\ell}{?} \\ \overset{2}{?}\,\overset{3}{?}\ \cdots\ \overset{\ell}{?}\,\overset{1}{?} \\ \vdots \\ \overset{\ell-1}{?}\,\overset{\ell}{?}\ \cdots\ \overset{\ell-3}{?}\,\overset{\ell-2}{?} \\ \overset{\ell}{?}\,\overset{1}{?}\ \cdots\ \overset{\ell-2}{?}\,\overset{\ell-1}{?} \end{cases}.$$

Formally, it can be written as $(\mathsf{shuf}, \{(1\,2\,\ldots\,\ell)^i \mid 1 \le i \le \ell\})$, where $(1\,2\,\ldots\,\ell)$ represents a cyclic permutation.

2.3 Finding a Specific Card by Random Cut

Here, we show the procedure for finding a card with a specific number from a sequence of face-down cards using the random cut [20].

Suppose that there are four face-down cards

$$?\,?\,?\,?$$

known to be composed of $1\,2\,3\,4$ (whose order is secret), and we want to identify 1. Then, it suffices to repeatedly apply a random cut and turn over the first card until the revealed card becomes 1:

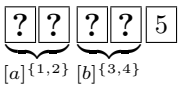
$$?\,?\,?\,? \rightarrow \left\langle ?\,?\,?\,? \right\rangle \rightarrow 2\,?\,?\,? \rightarrow \left\langle ?\,?\,?\,? \right\rangle \rightarrow 1\,?\,?\,?.$$

By virtue of the randomization provided by the random cut, no information other than the fact that the first card is 1 leaks. The expected number of shuffles (i.e., random cuts) required is four.

When applying this method to identify one of k cards from a sequence of ℓ cards, the expected number of shuffles is ℓ/k. As seen in the next subsection, this method plays an important role in the Niemi–Renvall AND protocol [20].

2.4 Niemi–Renvall AND Protocol

As mentioned in Section 1.2, the Niemi–Renvall AND protocol takes



as input. Let us swap the second and third cards; then, the five-card sequence changes as follows:

(a,b)	Sequence				
$(0,0)$	1	2	3	4	5
$(0,1)$	1	2	4	3	5
$(1,0)$	2	1	3	4	5
$(1,1)$	2	1	4	3	5

$\rightarrow$

(a,b)	Sequence				
$(0,0)$	1	3	2	4	5
$(0,1)$	1	4	2	3	5
$(1,0)$	2	3	1	4	5
$(1,1)$	2	4	1	3	5

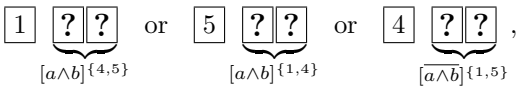
.

Using the method explained in Section 2.3, find $\boxed{2}$ and $\boxed{3}$ by an expected number of $(2.5+4)$ random cuts, and remove them; then, we have a three-card sequence $\boxed{?}\boxed{?}\boxed{?}$ satisfying:

(a,b)	Sequence									
$(0,0)$	1	4	5	or	5	1	4	or	4	5	1
$(0,1)$	1	4	5	or	5	1	4	or	4	5	1
$(1,0)$	1	4	5	or	5	1	4	or	4	5	1
$(1,1)$	1	5	4	or	5	4	1	or	4	1	5

.

After applying a random cut, reveal the first card; then, we obtain:

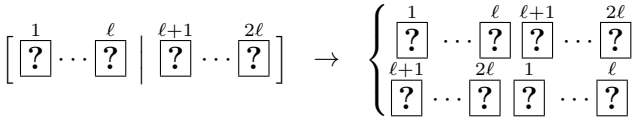


as desired (remember that the NOT operation is trivial).

This is the Niemi–Renvall AND protocol [20], enhanced by Koch, Schrempp, and Kirsten [7].

2.5 Random Bisection Cut

A random bisection cut [18] is a shuffle that transforms a sequence of 2ℓ cards into one of the following two states with a probability of 1/2:



Formally, it can be written as $(\mathsf{shuf}, \{\mathsf{id},\ (1\ \ell+1)(2\ \ell+2)\ ...\ (\ell\ 2\ell)\})$, where id is the identity permutation.

3 Analyzing Haga et al's Protocols by NPN Equivalence Classes

In this section, for a total of 256 three-input Boolean functions, we identify the functions already solved by existing research by Haga et al. [2], based on the NPN equivalence classes.

Table 4. Haga et al.'s results and NPN equivalence class classification

h \ g	0	$a \wedge b$	$a \wedge \overline{b}$	a	$\overline{a} \wedge b$	b	$a \oplus b$	$a \vee b$	$\overline{a \vee b}$	$\overline{a \oplus b}$	$\overline{b}$	$a \vee \overline{b}$	$\overline{a}$	$\overline{a} \vee b$	$\overline{a \wedge b}$	1
0	#1	#2	#2	#3	#2	#3	#4	#5	#2	#4	#3	#5	#3	#5	#5	#6
$a \wedge b$	#2	#3	#4	#5	#4	#5	#10	#8	#9	#7	#10	#11	#10	#11	#12	#5
$a \wedge \overline{b}$	#2	#4	#3	#5	#9	#10	#7	#11	#4	#10	#5	#8	#10	#12	#11	#5
a	#3	#5	#5	#6	#10	#11	#12	#5	#10	#12	#11	#5	#13	#10	#10	#3
$\overline{a} \wedge b$	#2	#4	#9	#10	#3	#5	#7	#11	#4	#10	#10	#12	#5	#8	#11	#5
b	#3	#5	#10	#11	#5	#6	#12	#5	#10	#12	#13	#10	#11	#5	#10	#3
$a \oplus b$	#4	#10	#7	#12	#7	#12	#13	#7	#10	#14	#12	#10	#12	#10	#7	#4
$a \vee b$	#5	#8	#11	#5	#11	#5	#7	#3	#12	#10	#10	#4	#10	#4	#9	#2
$\overline{a \vee b}$	#2	#9	#4	#10	#4	#10	#10	#12	#3	#7	#5	#11	#5	#11	#8	#5
$\overline{a \oplus b}$	#4	#7	#10	#12	#10	#12	#14	#10	#7	#13	#12	#7	#12	#7	#10	#4
$\overline{b}$	#3	#10	#5	#11	#10	#13	#12	#10	#5	#12	#6	#5	#11	#10	#5	#3
$a \vee \overline{b}$	#5	#11	#8	#5	#12	#10	#10	#4	#11	#7	#5	#3	#10	#9	#4	#2
$\overline{a}$	#3	#10	#10	#13	#5	#11	#12	#10	#5	#12	#11	#10	#6	#5	#5	#3
$\overline{a} \vee b$	#5	#11	#12	#10	#8	#5	#10	#4	#11	#7	#10	#9	#5	#3	#4	#2
$\overline{a \wedge b}$	#5	#12	#11	#10	#11	#10	#7	#9	#8	#10	#5	#4	#5	#4	#3	#2
1	#6	#5	#5	#3	#5	#3	#4	#2	#5	#4	#3	#2	#3	#2	#2	#1

3.1 Existing Protocols by Haga et al.

Haga et al. [2] considered three-input Boolean functions as follows:

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

That is, they represented a three-input Boolean function f using two two-input Boolean functions, g and h, and constructed protocols based on this decomposition. In brief, the protocol privately rearranges the four cards of commitments to a and b using a random bisection cut based on the value of commitment to c. Then, similar to the Niemi–Renvall AND protocol explained in Section 2.4, the protocol finds two specific cards from five cards through repeated random cuts, applies a final random cut, and turns over the first card to obtain an output commitment. The expected number of shuffles is 8.5.

This general-purpose protocol can securely compute the functions highlighted in yellow in Table 4. In this table, a three-input Boolean function f is determined by defining two two-input Boolean functions g and h.

Table 5. Representative functions of equivalence classes

Class	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14
Rep.	0	And3	$a \wedge b$	XorAnd	OrAnd	a	Onehot	Majority	Equality	Dot	Mux	AndXor	$a \oplus b$	Xor3

Consequently, among the 256 possible functions, six-card protocols have already been constructed for 140 functions. The 116 functions not highlighted in yellow in Table 4 are those that remained unresolved in the existing research by Haga et al. [2].

3.2 Analysis Based on NPN Equivalence Classes

In Table 4, we indicate which NPN equivalence class each of the 256 functions belongs to using numbers from #1 to #14. These numbers represent the NPN equivalence classes, defined as shown in Table 5.

Recall that Haga et al.'s generic protocol [2] works only for the functions highlighted in yellow in Table 4. Therefore, we find that the NPN equivalence classes not supported by Haga et al.'s generic protocol are #7, #10, and #14 (because these are the numbers not highlighted in yellow across all entries)[6].

Among these three, #14 (namely, $\mathsf{Xor3}(a, b, c) = a \oplus b \oplus c$) can be realized by applying the existing two-input XOR protocol [15] twice. Thus, the truly unresolved classes are #7 (Onehot) and #10 (Dot). In the next section, we construct six-card protocols for these two functions.

Note that for #4 (XorAnd) and #12 (AndXor), protocols can be constructed by combining the existing two-input XOR protocol [15] and two-input AND protocol [7]. The total number of shuffles is 7 (expected value), as shown in Table 3.

Furthermore, #1, #3, #6, and #13 are trivial even without using the protocol by Haga et al. [2]. This is because #1 (0) and #6 (a) are a constant and a single-variable function, respectively, requiring no actions; #3 ($a \wedge b$) and #13 ($a \oplus b$) are essentially two-input functions, to which the existing two-input AND protocol [7] or two-input XOR protocol [15] can be applied.

4 Proposed Protocols

Recall that the truly unresolved classes are #7 (Onehot) and #10 (Dot). Thus, in this section, we construct a six-card protocol for each of the two functions Onehot and Dot.

[6] For example, the class #11 has both 'highlighted' and 'non-highlighted' functions, and any non-highlighted function in the class can be computed with six cards by transforming it into any 'highlighted' function in the class.

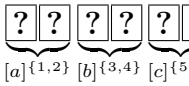
4.1 Our Protocol for Onehot

In this subsection, we construct a protocol for Onehot. To this end, we exploit the following decomposition:

$$\mathsf{Onehot}(a,b,c) = \begin{cases} b \oplus c & \text{if } a = 0 \\ \overline{b \vee c} & \text{if } a = 1 \end{cases} .$$

Given three commitments to $a, b, c \in \{0, 1\}$, our protocol proceeds as follows.

1. Place the three input commitments as:

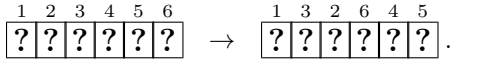


Then, the six cards (depending on (a, b, c)) are as follows:

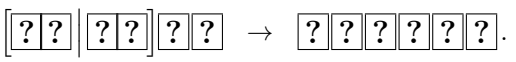
(a,b,c)	Sequence					
$(0,0,0)$	1	2	3	4	5	6
$(0,0,1)$	1	2	3	4	6	5
$(0,1,0)$	1	2	4	3	5	6
$(0,1,1)$	1	2	4	3	6	5

(a,b,c)	Sequence					
$(1,0,0)$	2	1	3	4	5	6
$(1,0,1)$	2	1	3	4	6	5
$(1,1,0)$	2	1	4	3	5	6
$(1,1,1)$	2	1	4	3	6	5

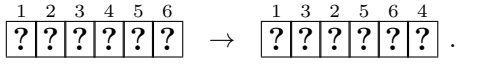
2. Through steps (a)–(d) below, we swap the third and sixth cards only when $a = 0$ without revealing the value of a:
   (a) Permute the sequence by $(\mathsf{perm}, (2\ 3)(4\ 5\ 6))$:



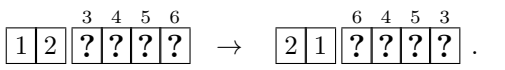
   (b) Apply a random bisection cut to the leftmost four cards:



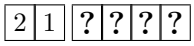
   (c) Permute the sequence by $(\mathsf{perm}, (2\ 3)(4\ 6\ 5))$:



   Note that the first two cards are either $\boxed{1}\boxed{2}$ or $\boxed{2}\boxed{1}$ with equal probability.
   (d) Turn over the first and second cards. If $\boxed{1}\boxed{2}$ appears, permute the sequence by $(\mathsf{perm}, (1\ 2)(3\ 6))$:



   Then, the six cards



satisfy:

(a,b,c)	Sequence						(a,b,c)	Sequence					
$(0,0,0)$	2	1	6	4	5	3	$(1,0,0)$	2	1	3	4	5	6
$(0,0,1)$	2	1	5	4	6	3	$(1,0,1)$	2	1	3	4	6	5
$(0,1,0)$	2	1	6	3	5	4	$(1,1,0)$	2	1	4	3	5	6
$(0,1,1)$	2	1	5	3	6	4	$(1,1,1)$	2	1	4	3	6	5

That is, when $a = 1$, the original arrangement of b and c remains unchanged; in contrast, when $a = 0$, the first card of b and the second card of c are swapped.

3. Using the method explained in Section 2.3, find $\boxed{3}$ or $\boxed{6}$ from the last four cards by an expected number of two random cuts.
   (a) If $\boxed{3}$ is found, move $\boxed{1}$ just before the sixth card:

$$\boxed{2}\boxed{1}\boxed{3}\boxed{?}\boxed{?}\boxed{?} \quad \to \quad \boxed{2}\boxed{3}\boxed{?}\boxed{?}\boxed{1}\boxed{?}.$$

Then, we have:

(a,b,c)	Sequence					(a,b,c)	Sequence					
$(0,0,0)$	2	3	6	4	1	5	$(1,0,0)$	2	3	4	5	1	6
$(0,0,1)$	2	3	5	4	1	6	$(1,0,1)$	2	3	4	6	1	5
$(0,1,0)$	2	3	5	4	1	6	$(1,1,0)$	2	3	5	6	1	4
$(0,1,1)$	2	3	6	4	1	5	$(1,1,1)$	2	3	6	5	1	4

Observing the cyclic order of $\boxed{3}$ and $\boxed{6}$ starting from $\boxed{1}$ reveals that it matches the value of $b \oplus c$ when $a = 0$, and that of $\overline{b \vee c}$ when $a = 1$; so if we remove $\boxed{4}$ and $\boxed{5}$, we will obtain:

$$\left.\begin{array}{ll} \boxed{1}\underbrace{\boxed{?}\boxed{?}}_{[b\oplus c]^{\{3,6\}}} & \text{if } a = 0 \\ \boxed{1}\underbrace{\boxed{?}\boxed{?}}_{[\overline{b\vee c}]^{\{3,6\}}} & \text{if } a = 1 \end{array}\right\} = \boxed{1}\underbrace{\boxed{?}\boxed{?}}_{[\mathsf{Onehot}(a,b,c)]}.$$

   (b) If $\boxed{6}$ is found, move $\boxed{1}$ just before the fifth card:

$$\boxed{2}\boxed{1}\boxed{6}\boxed{?}\boxed{?}\boxed{?} \quad \to \quad \boxed{2}\boxed{6}\boxed{?}\boxed{1}\boxed{?}\boxed{?}.$$

Then, we have:

(a,b,c)	Sequence					(a,b,c)	Sequence					
$(0,0,0)$	2	6	4	1	5	3	$(1,0,0)$	2	6	3	1	4	5
$(0,0,1)$	2	6	3	1	5	4	$(1,0,1)$	2	6	5	1	3	4
$(0,1,0)$	2	6	3	1	5	4	$(1,1,0)$	2	6	4	1	3	5
$(0,1,1)$	2	6	4	1	5	3	$(1,1,1)$	2	6	5	1	4	3

Note that, similar to the case (a) above, removing $\boxed{4}$ and $\boxed{5}$ brings a commitment to $\mathsf{Onehot}(a,b,c)$.

4. Ignoring $\boxed{2}$, turn over the face-up cards:

$$\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

Find and remove $\boxed{4}$ and $\boxed{5}$ by using the method explained in Section 2.3; the number of random cuts is 6.5 (expected value):

$$\boxed{4}\boxed{5}\quad\boxed{?}\boxed{?}\boxed{?}.$$

5. Apply a random cut to the face-down three cards:

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

6. Turn over the first card to obtain a commitment to $\mathsf{Onehot}(a,b,c)$ (or its negation) as follows:

$$\boxed{?}\boxed{?}\boxed{?}\xrightarrow{(\mathsf{turn},\{1\})}\begin{cases}\boxed{1}\ \overbrace{\boxed{?}\boxed{?}}^{[\mathsf{Onehot(a,b,c)}]}\\ \boxed{3}\ \overbrace{\boxed{?}\boxed{?}}^{[\overline{\mathsf{Onehot(a,b,c)}}]}\\ \boxed{6}\ \overbrace{\boxed{?}\boxed{?}}^{[\mathsf{Onehot(a,b,c)}]}\end{cases}.$$

This is our six-card protocol for Onehot. The expected number of shuffles is 10.5 in total.

Up to Step 2, our protocol follows the same procedure as Haga et al. [2]. The key difference and novelty lie in the subsequent steps: while Haga et al.'s protocol only identifies and removes two cards, our proposed protocol strategically repositions $\boxed{1}$ after a single card is identified.

4.2 Our Protocol for Dot

Similar to our protocol for Onehot presented in the previous subsection, a protocol for Dot can be constructed using six cards, based on the following decomposition:

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

Due to page limitations, we omit the details.

5 Conclusion

In this paper, we addressed the problems left unresolved in existing research by Haga et al. [2] and showed that secure computation protocols can be constructed

for all three-input Boolean functions using six playing cards. The required number of shuffles is summarized in Table 3. Reducing these shuffle counts remains a subject for future work.

Our results imply that the card-minimality problem is resolved for all Boolean functions with up to three inputs. Solving the card-minimality problem for n-input Boolean functions with $n \geq 4$ (in the playing-card setting) is also an intriguing direction for future work. It should be noted that, as an upper bound, $2n + 8$ cards have been shown to be sufficient [25].

Recently, the application of card-based cryptography to games such as Old Maid [24], President [13], and Tagiron [8] has attracted growing attention; designing such protocols with only standard playing cards is a promising direction for future study.

Acknowledgements

We thank the anonymous reviewers, whose comments have helped us improve the presentation of the paper. This work was supported in part by JSPS KAKENHI Grant Numbers JP24K02938.

References

1. Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: Du, D.Z., Duan, Z., Tian, C. (eds.) Computing and Combinatorics (COCOON). LNCS, vol. 11653, pp. 166–177. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-26176-4_14
2. Haga, R., Hayashi, Y., Miyahara, D., Mizuki, T.: Card-minimal protocols for three-input functions with standard playing cards. In: Batina, L., Daemen, J. (eds.) AFRICACRYPT 2022. LNCS, vol. 13503, pp. 448–468. Springer, Cham (2022), https://doi.org/10.1007/978-3-031-17433-9_19
3. Hatsugai, K., Asano, K., Abe, Y.: A physical zero-knowledge proof for Sumplete, a puzzle generated by ChatGPT. In: Wu, W., Tong, G. (eds.) Computing and Combinatorics (COCOON). LNCS, vol. 14422, pp. 398–410. Springer, Cham (2024), https://doi.org/10.1007/978-3-031-49190-0_29
4. Honda, Y., Shinagawa, K.: Efficient card-based protocols with a standard deck of playing cards using partial opening. In: Minematsu, K., Mimura, M. (eds.) Advances in Information and Computer Security. LNCS, vol. 14977, pp. 85–100. Springer, Singapore (2024), https://doi.org/10.1007/978-981-97-7737-2_5
5. Honda, Y., Shinagawa, K.: Efficient three-input and four-input AND protocols using playing cards with partial-open actions. In: Kim, Y., Miyaji, A., Tibouchi, M. (eds.) Cryptology and Network Security. LNCS, vol. 16351, pp. 199–212. Springer, Singapore (2025), https://doi.org/10.1007/978-981-95-4434-9_9
6. 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
7. Koch, A., Schrempp, 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. Koizumi, K., Mizuki, T.: An application of secure computation to Tagiron. In: Hartisch, M., Hsueh, C.H., van den Herik, J. (eds.) Advances in Computer Games. LNCS, vol. 16463, pp. 191–204. Springer Nature Switzerland, Cham (2026)
9. 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.) Computer Science – Theory and Applications. LNCS, vol. 12730, pp. 242–256. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-79416-3_14
10. Koyama, H., Toyoda, K., Miyahara, D., Mizuki, T.: New card-based copy protocols using only random cuts. In: ACM ASIA Public-Key Cryptography Workshop. pp. 13–22. ACM, NY (2021), https://doi.org/10.1145/3457338.3458297
11. Manabe, Y., Ono, H.: Card-based cryptographic protocols with a standard deck of cards using private operations. New Gener. Comput. **42**, 305–329 (2024), https://doi.org/10.1007/s00354-024-00257-2
12. Marakkalage, D.S., Testa, E., Riener, H., Mishchenko, A., Soeken, M., De Micheli, G.: Three-input gates for logic synthesis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **40**(10), 2184–2188 (2021). https://doi.org/10.1109/TCAD.2020.3032625
13. Miyahara, D., Lafourcade, P., Mizuki, T., Shinagawa, K.: Playing President with virtual players: How to play multiple cards of a kind. In: Iacono, J. (ed.) Fun with Algorithms. LIPIcs, vol. 366, pp. 34:1–34:15. Schloss Dagstuhl, Dagstuhl, Germany (2026), https://doi.org/10.4230/LIPIcs.FUN.2026.34
14. Miyahara, D., Mizuki, T.: Secure computations through checking suits of playing cards. In: Li, M., Sun, X. (eds.) Frontiers in Algorithmics. LNCS, vol. 13461, pp. 110–128. Springer, Cham (2023), https://doi.org/10.1007/978-3-031-20796-9_9
15. Mizuki, T.: Efficient and secure multiparty computations using a standard deck of playing cards. In: Foresti, S., Persiano, G. (eds.) Cryptology and Network Security. LNCS, vol. 10052, pp. 484–499. Springer, Cham (2016), https://doi.org/10.1007/978-3-319-48965-0_29
16. 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
17. 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
18. 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
19. Nakai, T., Iwanari, K., Ono, T., Abe, Y., Watanabe, Y., Iwamoto, M.: Card-based cryptography with a standard deck of cards, revisited: Efficient protocols in the private model. New Gener. Comput. **42**, 345–358 (2024), https://doi.org/10.1007/s00354-024-00269-y
20. Niemi, V., Renvall, A.: Solitaire zero-knowledge. Fundam. Inf. **38**(1,2), 181–188 (1999), https://doi.org/10.3233/FI-1999-381214
21. Otsuji, T., Fulla, P., Fukunaga, T.: NP-Completeness and physical zero-knowledge proof of Hotaru Beam. In: Chen, Y., Gao, X., Sun, X., Zhang, A. (eds.) Computing and Combinatorics (COCOON). pp. 239–251. Springer, Singapore (2024), https://doi.org/10.1007/978-981-96-1090-7_20

22. Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for Sudoku. In: Chen, C.Y., Hon, W.K., Hung, L.J., Lee, C.W. (eds.) Computing and Combinatorics (COCOON). LNCS, vol. 13025, pp. 631–642. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-89543-3_52
23. Sasao, T.: Switching theory for logic synthesis. Boston: Kluwer Academic Publishers (1999)
24. Shinagawa, K., Miyahara, D., Mizuki, T.: How to play Old Maid with virtual players. Theory of Computing Systems **69**(1), 13 (2025), https://doi.org/10.1007/s00224-024-10203-w
25. Shinagawa, K., Mizuki, T.: Secure computation of any Boolean function based on any deck of cards. In: Chen, Y., Deng, X., Lu, M. (eds.) Frontiers in Algorithmics. Lecture Notes in Computer Science, vol. 11458, pp. 63–75. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-18126-0_6