A Card-Minimal Three-Input AND Protocol Using Two Shuffles*

Raimu Isuzugawa[1], Kodai Toyoda[1], Yu Sasaki[1], Daiki Miyahara[1,2], and Takaaki Mizuki[1,2]

[1] Tohoku University, Sendai, Japan
mizuki+lncs[atmark]tohoku.ac.jp
[2] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract. Card-based cryptography typically uses a physical deck comprising black and red cards to perform secure computations, where a one-bit value is encoded using a pair of cards with different colors such that the order of black to red represents 0 and red to black represents 1. One of the most fundamental classes of card-based protocols is the class of "card-minimal" n-input AND protocols, which require $2n$ face-down cards as input to securely evaluate the AND value after applying a number of shuffles; here, the $2n$ cards are minimally required to describe an n-bit input. The best n-input AND protocols currently known use two shuffles for $n = 2$, five shuffles for $n = 3$, and $n + 1$ shuffles for $n > 3$. These upper bounds on the numbers of shuffles have not been improved for several years. In this work, we present a better upper bound for the $n = 3$ case by designing a new card-minimal three-input AND protocol using only two shuffles. Therefore, our proposed protocol reduces the number of required shuffles from five to two; we believe that this is a significant improvement.

1 Introduction

Many card-based protocols have been designed in the history of *card-based cryptography* to perform secure computations using a deck of physical cards. Typically, card-based protocols work on two-colored decks comprising black $\boxed{♣}$ and red $\boxed{♡}$ cards whose backs are denoted by $\boxed{?}$ and indistinguishable. These cards are used to represent Boolean values as follows:

$$\boxed{♣}\boxed{♡} = 0, \quad \boxed{♡}\boxed{♣} = 1. \tag{1}$$

When two face-down cards represent a bit $x \in \{0, 1\}$ according to the above encoding rule (1), we call them a *commitment* to x, denoted as

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

* This paper appears in Proceedings of COCOON 2021. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-89543-3_55.

1.1 Card-Minimal AND Protocols

In 1989, Den Boer [2] designed the first card-based protocol, called the "five-card trick," which takes two commitments to $x_1, x_2 \in \{0,1\}$ and a helping card $\boxed{\heartsuit}$ as input to securely evaluate the AND value $x_1 \wedge x_2$ (without revealing any information about x_1, x_2 more than necessary) through a series of actions, such as shuffling and turning over cards:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\ \underbrace{\boxed{?}\boxed{?}}_{x_2}\ \boxed{\heartsuit} \rightarrow \cdots \rightarrow x_1 \wedge x_2.$$

Thus, this is a two-input AND protocol using one helping card.

Since then, a challenging open problem had been to construct a two-input AND protocol without any helping card. More generally, could we construct an n-input AND protocol using only $2n$ cards?

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\ \underbrace{\boxed{?}\boxed{?}}_{x_2} \cdots \underbrace{\boxed{?}\boxed{?}}_{x_n} \rightarrow \cdots \rightarrow x_1 \wedge x_2 \wedge \cdots \wedge x_n.$$

Because $2n$ cards are necessary for arranging the n input commitments to the values $x_1, x_2, \ldots, x_n \in \{0,1\}$ (as long as we obey the encoding rule (1)), this type of AND protocol (using exactly $2n$ cards) is said to be *card-minimal*. This study addresses the class of card-minimal AND protocols.

1.2 Known Results

The first card-minimal AND protocol was proposed by Kumamoto et al. in 2012 [12]:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\ \underbrace{\boxed{?}\boxed{?}}_{x_2} \xrightarrow{\text{2 RBCs}} \cdots \rightarrow \begin{cases} x_1 \wedge x_2 = 0 & \text{if } \boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{\heartsuit} \\ x_1 \wedge x_2 = 0 & \text{if } \boxed{\clubsuit}\boxed{\heartsuit}\boxed{?}\boxed{?} \\ x_1 \wedge x_2 = 1 & \text{if } \boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{\clubsuit} \\ x_1 \wedge x_2 = 1 & \text{if } \boxed{\heartsuit}\boxed{\heartsuit}\boxed{?}\boxed{?}. \end{cases}$$

That is, using only four cards, a two-input AND protocol was constructed. This protocol uses two shuffles; more precisely, it applies a "random bisection cut (RBC)" twice, which is a kind of shuffling operation (explained later in Sect. 2.3). See the first protocol listed in Table 1.

How about n-input AND protocols for $n \geq 3$? This open problem was solved in 2016 [11]. Specifically, for the case of $n = 3$, a card-minimal three-input AND protocol was proposed by Mizuki [11]:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\ \underbrace{\boxed{?}\boxed{?}}_{x_2}\ \underbrace{\boxed{?}\boxed{?}}_{x_3} \xrightarrow{\text{3 RBCs \& 2 RCs}} \cdots \rightarrow \begin{cases} x_1 \wedge x_2 \wedge x_3 = 1 & \text{if } \boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\cdots \\ x_1 \wedge x_2 \wedge x_3 = 1 & \text{if } \boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}\cdots \\ x_1 \wedge x_2 \wedge x_3 = 0 & \text{otherwise.} \end{cases}$$

Table 1. Existing card-minimal AND protocols and our proposed protocol (using random cuts and/or random bisection cuts)

	#Inputs	#Cards	#Shuffles
Kumamoto et al. [12]	2	4	2
Mizuki [11]	3	6	5
Mizuki [11]	$n\,(\geq 4)$	$2n$	$n{+}1$
This paper	**3**	**6**	**2**

Thus, this six-card protocol uses three random bisection cuts along with two "random cuts (RCs)," for a total of five shuffles; the random cut is another common type of shuffling operation, which will be explained in Sect. 2.2. See the second protocol listed in Table 1.

For the case of $n \geq 4$, Mizuki [11] also proposed a card-minimal n-input AND protocol:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\;\underbrace{\boxed{?}\boxed{?}}_{x_2}\;\underbrace{\boxed{?}\boxed{?}}_{x_3}\;\underbrace{\boxed{?}\boxed{?}}_{x_4}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n}\overset{n+1\text{ RBCs}}{\rightarrow\cdots\rightarrow}\begin{cases} x_1\wedge x_2\wedge\cdots\wedge x_n=0 & \text{if } \boxed{\clubsuit}\cdots \\ x_1\wedge x_2\wedge\cdots\wedge x_n=1 & \text{if } \boxed{\heartsuit}\cdots . \end{cases}$$

This protocol takes n input commitments (such that $n \geq 4$) and uses $n+1$ random bisection cuts to securely evaluate their AND value. See the third protocol shown in Table 1.

1.3 Contribution

In this work, we focus on the number of required shuffles: From Table 1, it is observed that the number of shuffles used in the second protocol, i.e., 5 in the three-input AND protocol [11], is somewhat large. Actually, the three-input protocol [11] is elaborate but rather complicated, and it seems difficult for laypeople to execute practically. Therefore, our goal is to improve this existing three-input AND protocol [11].

Specifically, we will construct a new card-minimal three-input AND protocol using only two shuffles, namely one random bisection cut and one random cut:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\;\underbrace{\boxed{?}\boxed{?}}_{x_2}\;\underbrace{\boxed{?}\boxed{?}}_{x_3}\;\overset{1\text{ RBC \& 1 RC}}{\rightarrow\cdots\rightarrow}\begin{cases} x_1\wedge x_2\wedge x_3=1 & \text{if } \boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\cdots \\ x_1\wedge x_2\wedge x_3=1 & \text{if } \boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}\cdots \\ x_1\wedge x_2\wedge x_3=0 & \text{otherwise.} \end{cases}$$

The performance of this approach is shown in Table 1 in the last row.

Fig. 1 shows the numbers of required shuffles for all the protocols listed in Table 1. As seen from this figure, the previous three-input protocol [11] requires five shuffles while our proposed three-input protocol uses only two shuffles, thereby successfully reducing the number of required shuffles significantly from five to two. As shown later in Sect. 3, our designed protocol is simple enough for laypeople to execute practically. Therefore, we believe that our new protocol is important from both theoretical and practical points of view.

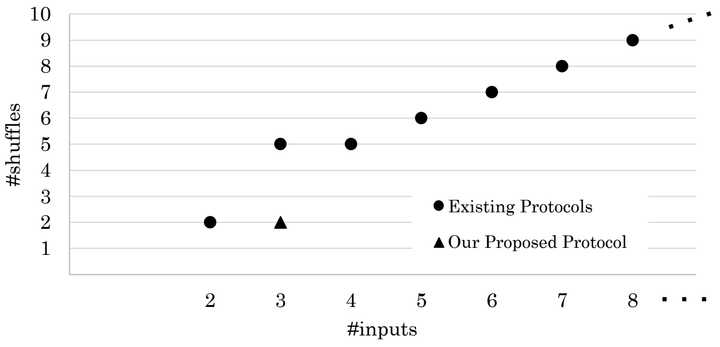

Fig. 1. Numbers of shuffles used in the existing card-minimal n-input AND protocols (for all $n \geq 2$) and our proposed protocol

1.4 Related Work

All the protocols mentioned thus far are not *committed-format* AND protocols because their AND values are not obtained as commitments. Contrarily, there are committed-format protocols, such as committed-format two-input AND protocols, which produce commitments to the AND values:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\ \underbrace{\boxed{?}\boxed{?}}_{x_2} \rightarrow \cdots \rightarrow \underbrace{\boxed{?}\boxed{?}}_{x_1 \wedge x_2}.$$

Since this output is a (hidden) commitment, it can be used as input to another computation; thus, by repeatedly executing a committed-format AND protocol $n-1$ times, we can perform an n-input AND computation. Therefore, card-minimal committed-format two-input AND protocols are considered useful. Unfortunately however, such known AND protocols [4,8,23] require nonuniform or nonclosed shuffles, which are difficult to implement manually (cf. [17,18,25]); furthermore, Kastner et al. [3] proved that there exist no card-minimal two-input AND protocols that use only uniform closed shuffles. It should be noted that both random cuts and random bisection cuts (which all the protocols listed in Table 1 rely on) are uniform closed shuffles, which are easy to implement (as shown in Sects. 2.2 and 2.3).

If we allow the use of helping cards, we have a six-card committed-format AND protocol [15] and a five-card committed-format AND protocol [1] that rely only on random cuts and/or random bisection cuts; however, of course, they are not card-minimal.

Apart from the AND computation, because there is a card-minimal committed-format two-input XOR protocol [15], we can construct a card-minimal n-input XOR protocol for any $n \geq 2$. Recently, Ruangwises and Itoh [24] constructed a general way of designing card-minimal protocols that securely compute any doubly symmetric functions.

In stead of using shuffling operations, there is an alternative approach that relies on *private operations* (e.g., [16, 19–21, 28]); under this somewhat strong assumption, Manabe and Ono [9,22] showed that card-minimal protocols can be constructed for many kinds of Boolean functions, such as the AND, half-adder, full-adder, and symmetric functions.

2 Preliminaries

In this section, we first present a formal treatment of the actions used in card-based protocols (which has been developed in [3, 5, 13, 14]). Then, we formally introduce two shuffling operations, namely a random cut and a random bisection cut.

2.1 Actions in Card-Based Protocols

In card-based protocols, the following three main actions are applied to a sequence of cards; below, we assume a sequence of m cards.

Permute. This is denoted by (perm, π), where π is a permutation applied to the sequence of cards as follows:

$$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{m}{\boxed{?}} \xrightarrow{(\mathsf{perm},\pi)} \overset{\pi^{-1}(1)}{\boxed{?}}\ \overset{\pi^{-1}(2)}{\boxed{?}}\ \cdots\ \overset{\pi^{-1}(m)}{\boxed{?}}\ .$$

Turn. This is denoted by (turn, T), where T is a set of indexes, indicating that for every $t \in T$, the t-th card is turned over as follows:

$$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{t\in T}{\boxed{?}}\cdots\overset{m}{\boxed{?}} \xrightarrow{(\mathsf{turn},T)} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{t\in T}{\boxed{\clubsuit}}\cdots\overset{m}{\boxed{?}}\,.$$

Shuffle. This is denoted by $(\mathsf{shuf}, \Pi, \mathcal{F})$, where Π is a permutation set and $\mathcal{F}$ is a probability distribution on Π, indicating that $\pi \in \Pi$ is drawn according to $\mathcal{F}$ and applied to the sequence of cards as follows:

$$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{m}{\boxed{?}} \xrightarrow{(\mathsf{shuf},\Pi,\mathcal{F})} \overset{\pi^{-1}(1)}{\boxed{?}}\ \overset{\pi^{-1}(2)}{\boxed{?}}\ \cdots\ \overset{\pi^{-1}(m)}{\boxed{?}}\ .$$

Here, the permutation in Π that is applied remains unknown. If the distribution $\mathcal{F}$ is uniform, then its description can be omitted.

2.2 Random Cut

A *random cut* (RC) is the simplest and most easy-to-implement shuffle in card-based cryptography, denoted by $\langle\cdot\rangle$, which shifts a sequence of cards cyclically and randomly. If a random cut is applied to a sequence of m cards, then the resulting sequence becomes one of the following n sequences, each of which occurs with a probability of $1/m$:

$$\left\langle \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\cdots\overset{m-1}{\boxed{?}}\overset{m}{\boxed{?}} \right\rangle \rightarrow \begin{cases} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\cdots\overset{m-1}{\boxed{?}}\overset{m}{\boxed{?}}, \\ \overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\cdots\overset{m}{\boxed{?}}\overset{1}{\boxed{?}}, \\ \qquad\vdots \\ \overset{m-1}{\boxed{?}}\overset{m}{\boxed{?}}\overset{1}{\boxed{?}}\cdots\overset{m-3}{\boxed{?}}\overset{m-2}{\boxed{?}}, \\ \overset{m}{\boxed{?}}\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\cdots\overset{m-2}{\boxed{?}}\overset{m-1}{\boxed{?}}. \end{cases}$$

This random cut is formally described as

$$(\mathsf{shuf}, \{\mathsf{id}, \pi, \pi^2, \ldots, \pi^{m-1}\})$$

for a cyclic permutation $\pi = (1\,2\,3\,\cdots\,m)$, where id denotes the identity permutation.

Hereinafter, we use $\mathsf{RC}_{1,2,\ldots,m}$ to represent $\{\mathsf{id}, \pi, \pi^2, \ldots, \pi^{m-1}\}$. For example, $(\mathsf{shuf}, \mathsf{RC}_{1,2,3,4,5,6})$ is a random cut to a sequence of six face-down cards:

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

A random cut can be easily performed manually; a secure implementation called the Hindu cut is a well-known instance [27].

2.3 Random Bisection Cut

A *random bisection cut* (RBC) is another major shuffle action, which was invented in 2009 [15]. This shuffle, denoted by $[\,\cdot\,|\,\cdot\,]$, bisects a sequence of $2m$ cards and randomly swaps the two halves; the resulting sequence becomes one of the following two sequences, with a probability of $1/2$:

$$\left[\overset{1}{\boxed{?}}\cdots\overset{m}{\boxed{?}} \,\middle|\, \overset{m+1}{\boxed{?}}\cdots\overset{2m}{\boxed{?}} \right] \rightarrow \begin{cases} \overset{1}{\boxed{?}}\cdots\overset{m}{\boxed{?}}\overset{m+1}{\boxed{?}}\cdots\overset{2m}{\boxed{?}}, \\ \overset{m+1}{\boxed{?}}\cdots\overset{2m}{\boxed{?}}\overset{1}{\boxed{?}}\cdots\overset{m}{\boxed{?}}. \end{cases}$$

That is, the resulting sequence either remains unchanged compared with the original or is obtained such that the two halves are swapped, with a probability of $1/2$. The random bisection cut can be expressed as follows:

$$(\mathsf{shuf}, \{\mathsf{id}, (1\ m{+}1)(2\ m{+}2)\cdots(m\ 2m)\}).$$

Secure implementations of a random bisection cut were shown in [26].

3 Our Proposed Protocol

In this section, we present the new card-minimal three-input AND protocol:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\;\underbrace{\boxed{?}\boxed{?}}_{x_2}\;\underbrace{\boxed{?}\boxed{?}}_{x_3} \xrightarrow{\text{1 RBC \& 1 RC}} \cdots \rightarrow \begin{cases} x_1 \wedge x_2 \wedge x_3 = 1 & \text{if } \boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\cdots \\ x_1 \wedge x_2 \wedge x_3 = 1 & \text{if } \boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}\cdots \\ x_1 \wedge x_2 \wedge x_3 = 0 & \text{otherwise.} \end{cases}$$

This protocol uses one random bisection cut and one random cut.

We present the description of our protocol in Sect. 3.1 as well as its pseudocode in Sect. 3.2. We also present an intuitive explanation in Sect. 3.3 as to why the proposed protocol works correctly. Formal proofs of the correctness and security of our proposed protocol (based on the so-called KWH-tree [8]) are omitted owing to length limitations.

3.1 Description

Our card-minimal three-input AND protocol proceeds, as follows.

1. Apply $(\mathsf{shuf}, \{\mathsf{id}, (1\,2)(3\,6)\})$ by performing operations (a)–(c) noted below.
   (a) Swap the second and third cards as well as the fourth and sixth cards:
   $$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \rightarrow \overset{1}{\boxed{?}}\overset{3}{\boxed{?}}\overset{2}{\boxed{?}}\overset{6}{\boxed{?}}\overset{5}{\boxed{?}}\overset{4}{\boxed{?}}.$$
   (b) Apply a random bisection cut to the four cards on the extreme left:
   $$\left[\boxed{?}\boxed{?}\,\middle|\,\boxed{?}\boxed{?}\right]\;\boxed{?}\boxed{?} \rightarrow \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$
   (c) Swap the second and third cards as well as the fourth and sixth cards again:
   $$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \rightarrow \overset{1}{\boxed{?}}\overset{3}{\boxed{?}}\overset{2}{\boxed{?}}\overset{6}{\boxed{?}}\overset{5}{\boxed{?}}\overset{4}{\boxed{?}}.$$
2. Turn over the first card to check its color. If the card is $\boxed{\clubsuit}$, swap the first and second cards as well as the third and sixth cards:
   $$\overset{1}{\boxed{\clubsuit}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \rightarrow \overset{2}{\boxed{?}}\overset{1}{\boxed{\clubsuit}}\overset{6}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{3}{\boxed{?}}.$$
   If the card color is $\boxed{\heartsuit}$, proceed to Step 3 directly.
3. After turning over the revealed card in a face-down manner, apply a random cut to the entire sequence:
   $$\left\langle\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\right\rangle \rightarrow \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

4. Turn over the first, third, and fifth cards. If these are $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$ (apart from cyclic rotation), then $x_1 \wedge x_2 \wedge x_3 = 0$; if the cards are $\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}$ or $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}$, then $x_1 \wedge x_2 \wedge x_3 = 1$:

$$\begin{array}{c}
\boxed{\heartsuit}\boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{\heartsuit}\boxed{?}\ \ \boxed{\clubsuit}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{\clubsuit}\boxed{?}\\
\boxed{\heartsuit}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{\clubsuit}\boxed{?}\ \ \boxed{\clubsuit}\boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{\heartsuit}\boxed{?}\\
\boxed{\clubsuit}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{\heartsuit}\boxed{?}\ \ \boxed{\heartsuit}\boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{\clubsuit}\boxed{?}\\
x_1 \wedge x_2 \wedge x_3 = 0,
\end{array}$$

$$\begin{array}{c}
\boxed{\heartsuit}\boxed{?}\boxed{\heartsuit}\boxed{?}\boxed{\heartsuit}\boxed{?}\ \ \boxed{\clubsuit}\boxed{?}\boxed{\clubsuit}\boxed{?}\boxed{\clubsuit}\boxed{?}\\
x_1 \wedge x_2 \wedge x_3 = 1.
\end{array}$$

3.2 Pseudocode

The pseudocode for our protocol is depicted in Algorithm 1, where $(\mathsf{result}, i, j, k)$ specifies the output positions.

Algorithm 1 Our proposed protocol

input set:

$$\begin{aligned}
\Big\{&\Big(\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit}\Big),\Big(\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit}\Big),\Big(\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit}\Big),\\
&\Big(\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit}\Big),\Big(\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit}\Big),\Big(\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit}\Big),\\
&\Big(\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit}\Big),\Big(\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit},\tfrac{?}{\heartsuit},\tfrac{?}{\clubsuit}\Big)\Big\}
\end{aligned}$$

```
1. (shuf, {id, (1 2)(3 6)})
2. (turn, {1})
3. if visible sequence = (♣, ?, ?, ?, ?, ?) then
4.    (perm, (1 2)(3 6))
5.    (turn, {2})
6. else if visible sequence = (♡, ?, ?, ?, ?, ?) then
7.    (turn, {1})
8. (shuf, $\mathsf{RC}_{1,2,3,4,5,6}$)
9. (result, 1, 3, 5)
```

3.3 Why Our Protocol Works Correctly

Herein, we intuitively explain why our proposed protocol works correctly. Note that the input sequence has eight possibilities depending on the input values $(x_1, x_2, x_3) \in \{0,1\}^3$, as shown in the second column of Table 2. The idea behind

Table 2. Resulting sequences after applying (perm,(1 2)(3 6)) and Step 2

Input	Input Sequence	Apply (perm,(1 2)(3 6))	After Step 2
(0,0,0)	$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}$
(0,0,1)	$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}$
(0,1,0)	$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}$
(0,1,1)	$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}$
(1,0,0)	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$	$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$
(1,0,1)	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}$	$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}$
(1,1,0)	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}$	$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}$
(1,1,1)	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$	$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}$	$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$

our protocol is to assign each possible input to one of two sequence patterns without leaking the input value. One of the two patterns is an alternating pattern of $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, i.e., either $\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$, which is a possible input sequence when $(x_1, x_2, x_3) = (0, 0, 0), (1, 1, 1)$; we call this an *alternating* sequence. The other pattern corresponds to the remaining sequences, which we call *non-alternating* sequences.

Suppose that we apply the permutation $(1\,2)(3\,6)$ to the input sequence; this appears in the shuffle applied in Step 1 and permutation in Step 2. The resulting sequence is the one shown in the third column of Table 2. Note that the second column (i.e., input sequence) and third column of Table 2 are equivalent to the transition possibilities after applying (shuf, {id, $(1\,2)(3\,6)$}) to the input sequence in Step 1. In the third column of Table 2, among the four sequences from the top (which are obtained when $x_1 = 0$), the sequences corresponding to $(0, 0, 1)$, $(0, 1, 0)$, and $(0, 1, 1)$ are still non-alternating, and the sequence corresponding to $(0, 0, 0)$ is converted to a non-alternating sequence. If we combine the four sequences from the top in the third column with the four sequences from the bottom in the second column, we obtain the eight sequences shown in the fourth column of Table 2, where only the sequence corresponding to $(1, 1, 1)$ is alternating. To achieve this, we apply the permutation $(1\,2)(3\,6)$ when the first cards in the sequences in the second and third columns of Table 2 are $\boxed{\clubsuit}$. This is the reason behind performing (shuf, {id, $(1\,2)(3\,6)$}) in Step 1 and (perm,$(1\,2)(3\,6)$) in Step 2 if the first card revealed is $\boxed{\clubsuit}$.

By applying a random cut to the sequence of cards in Step 3, the resulting sequence is one among the following four sequences (up to cyclic rotation):

(a) $\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$, if $(0, 0, 0)$ or $(1, 0, 0)$;
(b) $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$, if $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 1)$, or $(1, 1, 0)$;
(c) $\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}$, if $(0, 1, 1)$;
(d) $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$, if $(1, 1, 1)$.

Here, when the input is either $(0,0,0)$ or $(1,0,0)$, the resulting sequence is equal to that in (a); when the input is $(0,0,1)$, $(0,1,0)$, $(1,0,1)$, or $(1,1,0)$, the sequence is equal to that in (b); when the input is $(0,1,1)$, the sequence is equal to that in (c); when the input is $(1,1,1)$, the sequence is equal to that in (d). Now, if we revealed all the cards in the sequence, then we could obtain the value of $x_1 \wedge x_2 \wedge x_3$; however, information about the input would be leaked because the revealed sequence depends on the input values. Therefore, we need an alternative approach to obtain the output value.

Let us focus on the cards at the odd-numbered positions. If we reveal only these cards, then the revealed cards in every sequence of (a), (b), and (c) will have the same pattern, i.e., either $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$ up to cyclic rotation. By contrast, if (d) occurs, then the revealed cards will be either $\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}$ or $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}$. Hence, the sequences of (a), (b), and (c) become indistinguishable, and we can obtain only the value of $x_1 \wedge x_2 \wedge x_3$.

4 Conclusion

In this work, we proposed a card-minimal three-input AND protocol using only two shuffles. The minimality means that the protocol uses exactly six cards. The shuffles used in our protocol are one random cut and one random bisection cut. Since the existing three-input protocol [11] requires five shuffles, our proposed protocol successfully reduces the number of required shuffles from five to two. We believe that this is a significant improvement and that our protocol is simple enough for easy execution by lay-people.

An interesting open problem is improving the numbers of shuffles for card-minimal n-input AND computations for $n \geq 4$. It is also an intriguing problem to seek lower bounds on the numbers of shuffles using the "formal method approach" recently developed by Koch, Schrempp, and Kirsten [6,7].

This work considers the number of shuffles as the quality metric for evaluating a protocol because the shuffle action is the most time-consuming step (cf. [10]). However, considering the other actions for a more fine-grained analysis would be an interesting line of investigation in the future.

Acknowledgements

We thank the anonymous referees, whose comments have helped us improve the presentation of the paper. We would like to thank Hideaki Sone for his cooperation in preparing a Japanese draft version at an earlier stage of this work. This work was supported in part by JSPS KAKENHI Grant Numbers JP19J21153 and JP21K11881.

References

1. 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

2. Den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology—EUROCRYPT '89. LNCS, vol. 434, pp. 208–217. Springer, Berlin, Heidelberg (1990), https://doi.org/10.1007/3-540-46885-4_23
3. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology—ASIACRYPT 2017. LNCS, vol. 10626, pp. 126–155. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70700-6_5
4. Koch, A.: The landscape of optimal card-based protocols. Cryptology ePrint Archive, Report 2018/951 (2018), https://eprint.iacr.org/2018/951
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., Schrempp, M., Kirsten, M.: Card-based cryptography meets formal verification. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology–ASIACRYPT 2019. LNCS, vol. 11921, pp. 488–517. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-34578-5_18
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. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology—ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer, Berlin, Heidelberg (2015), https://doi.org/10.1007/978-3-662-48797-6_32
9. Manabe, Y., Ono, H.: Card-based cryptographic protocols for three-input functions using private operations. In: Combinatorial Algorithms. LNCS, Springer, Cham (2021), to appear
10. 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 (2021), https://doi.org/10.1007/s10207-020-00525-4
11. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theor. Comput. Sci. **622**(C), 34–44 (2016), https://doi.org/10.1016/j.tcs.2016.01.039
12. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Wang, X., Sako, K. (eds.) Advances in Cryptology—ASIACRYPT 2012. LNCS, vol. 7658, pp. 598–606. Springer, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-34961-4_36
13. 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
14. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Trans. Fundamentals **E100.A**(1), 3–11 (2017), https://doi.org/10.1587/transfun.E100.A.3
15. 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
16. 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

17. Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: An implementation of non-uniform shuffle for secure multi-party computation. In: Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography. pp. 49–55. AsiaPKC '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2898420.2898425, https://doi.acm.org/10.1145/2898420.2898425
18. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Comput. **22**(2), 361–371 (2018), https://doi.org/10.1007/s00500-017-2858-2
19. Ono, H., Manabe, Y.: Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In: Asia Joint Conference on Information Security (AsiaJCIS). pp. 23–28 (2018), https://doi.org/10.1109/AsiaJCIS.2018.00013
20. Ono, H., Manabe, Y.: Card-based cryptographic protocols with the minimum number of cards using private operations. In: Zincir-Heywood, N., Bonfante, G., Debbabi, M., Garcia-Alfaro, J. (eds.) Foundations and Practice of Security. LNCS, vol. 11358, pp. 193–207. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-18419-3_13
21. Ono, H., Manabe, Y.: Card-based cryptographic protocols with the minimum number of rounds using private operations. In: Pérez-Solà, C., Navarro-Arribas, G., Biryukov, A., Garcia-Alfaro, J. (eds.) Data Privacy Management, Cryptocurrencies and Blockchain Technology. LNCS, vol. 11737, pp. 156–173. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-31500-9_10
22. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Gener. Comput. **39**(1), 19–40 (2021), https://doi.org/10.1007/s00354-020-00113-z
23. Ruangwises, S., Itoh, T.: AND protocols using only uniform shuffles. In: van Bevern, R., Kucherov, G. (eds.) Computer Science–Theory and Applications. LNCS, vol. 11532, pp. 349–358. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-19955-5_30
24. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with $2n$ cards. Theor. Comput. Sci. (2021), https://doi.org/10.1016/j.tcs.2021.07.007, in press
25. 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.) Theory and Practice of Natural Computing. LNCS, vol. 12494, pp. 107–118. Springer, Cham (2020), https://doi.org/10.1007/978-3-030-63000-3_9
26. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. Int. J. Inf. Secur. **19**(4), 445–452 (2020), https://doi.org/10.1007/s10207-019-00463-w
27. Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Martín-Vide, C., Mizuki, T., Vega-Rodríguez, M.A. (eds.) Theory and Practice of Natural Computing. LNCS, vol. 10071, pp. 58–69. Springer, Cham (2016), https://doi.org/10.1007/978-3-319-49001-4_5
28. Yasunaga, K.: Practical card-based protocol for three-input majority. IEICE Trans. Fundamentals **E103.A**(11), 1296–1298 (2020), https://doi.org/10.1587/transfun.2020EAL2025