Card-based Single-shuffle Protocols for Secure Multiple-input AND and XOR Computations

Tomoki Kuzuma
Graduate School of Information Sciences, Tohoku University
Sendai, Japan
tomoki.kuzuma.s1[at]dc.tohoku.ac.jp

Raimu Isuzugawa
Graduate School of Information Sciences, Tohoku University
Sendai, Japan

Kodai Toyoda
Graduate School of Information Sciences, Tohoku University
Sendai, Japan

Daiki Miyahara
The University of Electro-Communications
National Institute of Advanced Industrial Science and Technology
Tokyo, Japan

Takaaki Mizuki
Cyberscience Center, Tohoku University
National Institute of Advanced Industrial Science and Technology
Sendai, Japan
mizuki+acm[at]tohoku.ac.jp

ABSTRACT

In card-based cryptography, the numbers of cards and shuffles are the complexity measures of protocols for secure computations, and the smaller these values are, the better. As the state-of-the-art study to minimize the latter measure, Shinagawa and Nuida showed a surprising result that any n-input logical function can be securely computed with only one shuffle, based on the idea of Yao's garbled circuit. When executing their protocol, the number of required cards is $2n+24q$, where the n-input logical function to be computed is represented by q gates. For example, when applied to the n-input AND and XOR functions, the number of gates is $n-1$, and hence, $26n-24$ cards are required. In this paper, we show that the number of required cards can be reduced by focusing on these two specific functions. Specifically, we construct a single-shuffle protocol for the n-input AND function using $4n-2$ cards, and construct a single-shuffle protocol for the n-input XOR function using $2n$ cards.

CCS CONCEPTS

• **Security and privacy → Cryptography**.

KEYWORDS

Card-based cryptography, Secure computation, Real-life hands-on cryptography, AND protocols, XOR protocols

ACM Reference Format:
Tomoki Kuzuma, Raimu Isuzugawa, Kodai Toyoda, Daiki Miyahara, and Takaaki Mizuki. 2022. Card-based Single-shuffle Protocols for Secure Multiple-input AND and XOR Computations. In *Proceedings of the 9th ACM ASIA Public-Key Cryptography Workshop (APKC '22), May 30, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3494105.3526236

1 INTRODUCTION

A method of performing secure computations using a deck of physical cards is called *card-based cryptography* (refer to [7, 20] for surveys). Card-based cryptography uses two types of indistinguishable cards with $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ on the front and $\boxed{?}$ on the back. These cards are used to represent Boolean values as

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

According to this encoding rule, when a bit $x \in \{0,1\}$ is represented by two face-down cards, these two cards are called a *commitment* to x and is denoted as follows:

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

A sequence of steps to perform secure computations with commitments as input is called a card-based cryptographic *protocol*. For example, a protocol starts with commitments to $x_1, x_2, \ldots, x_n \in \{0,1\}$ for a natural number $n\,(\geq 2)$ along with some additional cards, and outputs a commitment to the value of some predetermined function $f(x_1, x_2, \ldots, x_n)$ via a series of operations:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\cdots$$
$$\rightarrow \cdots \rightarrow \underbrace{\boxed{?}\boxed{?}}_{f(x_1,x_2,\ldots,x_n)}.$$

In this research field, the "number of cards" and the "number of shuffles" are measures of the complexity of a protocol; thus, many studies have aimed at reducing these factors.

1.1 Known results

Historically, reducing the number of cards required for the three fundamental functions, i.e., the AND function, the XOR function, and copying commitments, was focused on from the beginning (refer to [20]). Then, the number of shuffles has also gradually

Table 1: The numbers of required cards for single-shuffle secure computations of the n-input AND and XOR functions

	n-AND	n-XOR
Applying Shinagawa–Nuida [33]	$26n - 24$	$26n - 24$
Ours	$4n - 2$	$2n$

attracted attention, and the development of protocols with fewer shuffles has continued (e.g. [1, 2, 4, 11, 16, 34]).

As generic constructions for arbitrary n-input logical functions $f(x_1, x_2, \ldots, x_n)$, the state-of-the-art protocols in terms of the aforementioned two measures are as follows.

On the scale of reducing the number of cards, Nishida et al. [24] showed that any n-input logical function can be computed with $2n + 6$ cards. This means that additional six cards are sufficient because input commitments consist of $2n$ cards. When executing this protocol, the number of required shuffles is $O(2^n)$ because the target function is first expressed as an AND-XOR expansion and then the AND and XOR protocols [21] are applied according to the expansion.

On the other hand, with regard to reducing the number of shuffles, Shinagawa and Nuida [33] showed a surprising result that any n-input logical function can be computed with a single shuffle, based on the idea of Yao's garbled circuit. Because it is impossible to securely compute a non-trivial function without any shuffle, this protocol is optimal in the sense that the number of shuffles is minimal. The number of cards required for their protocol is $2n + 24q$ when the n-input logical function to be computed can be represented by q gates (i.e., $24q$ additional cards are required). In their protocol, for each gate, a truth table is created using 24 cards (as will be seen in Section 2.4). Reducing the number of $2n + 24q$ cards while keeping the number of shuffles at one remains open.

These results strongly suggest that there is a trade-off between the number of cards and the number of shuffles.

1.2 Contributions

In this study, we revisit the Shinagawa–Nuida single-shuffle generic protocol [33] and examine whether it is possible to reduce the number of cards from $2n+24q$ while keeping the number of shuffles at one. Specifically, we consider the n-input AND function and the n-input XOR function as specific functions, and customize the existing generic protocol to have specialized protocols with fewer cards.

Note that when applying the existing generic protocol to the n-input AND and XOR functions, $26n - 24$ cards are required because the number of gates for both the functions is $n - 1$. We consider saving the sequence of cards corresponding to a truth table so that we have a single-shuffle n-input AND protocol with $4n-2$ cards and a single-shuffle n-input XOR protocol with $2n$ cards, as summarized in Table 1. Thus, if we restrict ourselves to specific functions, we have succeeded in reducing the number of cards.

2 PRELIMINARIES

In this section, we first describe the operations used in card-based cryptography. Next, we describe the existing protocol that uses Yao's garbled circuit [33].

2.1 Operations

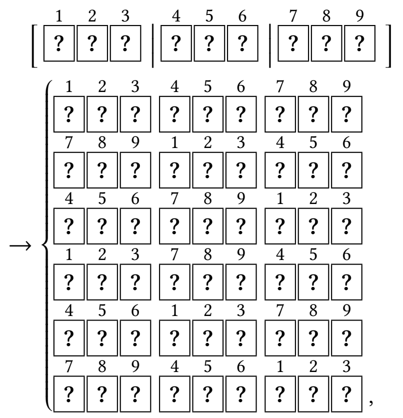
In card-based cryptography, three main operations, i.e., *rearrange*, *turn*, and *shuffle*, are used[i], and their computational model has been established by abstract machine [5, 10, 19].

Rearranging is an operation that rearranges the order of a sequence of cards, and is formulated using a permutation. Turning (over a card) is an operation to change the face of a card. Shuffling is an operation to randomly change the order of a sequence of cards, and is formulated using a set of permutations and a probability distribution on it. There are various shuffling operations, one type of which is introduced in Section 2.2.

2.2 Pile-scramble Shuffle

A shuffle that divides a sequence of cards into multiple *piles* with the same number of cards and changes only the order of these piles randomly is called a *pile-scramble shuffle* [3]; this is one of the most commonly used shuffles in the design of card-based cryptographic protocols. One of the famous shuffles, the random bisection cut [21], can be regarded as a special case of a pile-scramble shuffle.

As an example, if we divide the nine cards into three piles and apply a pile-scramble shuffle, it will transition to one of the six sequences with equal probability as follows:



where we denote a pile-scramble shuffle by $[\,\cdot\,|\,\cdots\,|\,\cdot\,]$.

Recently, the pile-scramble shuffle has been often used when constructing card-based zero-knowledge proof protocols, e.g. [12, 15, 27–31].

2.3 Counting and Combining Shuffles

As mentioned in Section 2.1, a shuffle is represented by a set of permutations Π and a probability distribution $\mathcal{F}$ on it. We allow any such pair $(\Pi, \mathcal{F})$, which includes shuffles that are difficult for humans to implement. However, since this paper is a theoretical

[i]There is another computational model that allows an operation called a *private permutation*, e.g. [13, 14, 22, 23, 26, 35].

Table 2: Truth table for AND gate

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

Table 3: Truth table for AND gate and corresponding commitments

$\underbrace{\boxed{?}\boxed{?}}_{a}$	$\underbrace{\boxed{?}\boxed{?}}_{b}$	
$\underbrace{\boxed{?}\boxed{?}}_{0}$	$\underbrace{\boxed{?}\boxed{?}}_{0}$	$\underbrace{\boxed{?}\boxed{?}}_{0}$
$\underbrace{\boxed{?}\boxed{?}}_{0}$	$\underbrace{\boxed{?}\boxed{?}}_{1}$	$\underbrace{\boxed{?}\boxed{?}}_{0}$
$\underbrace{\boxed{?}\boxed{?}}_{1}$	$\underbrace{\boxed{?}\boxed{?}}_{0}$	$\underbrace{\boxed{?}\boxed{?}}_{0}$
$\underbrace{\boxed{?}\boxed{?}}_{1}$	$\underbrace{\boxed{?}\boxed{?}}_{1}$	$\underbrace{\boxed{?}\boxed{?}}_{1}$

study that aims to help clarify the computational limitations of card-based cryptography, any pair $(\Pi, \mathcal{F})$ is counted as one shuffle[ii].

It should be noted that, in this sense, any two consecutive shuffles $(\Pi_1, \mathcal{F}_1)$ and $(\Pi_2, \mathcal{F}_2)$ can be combined into one shuffle [19].

2.4 The Existing Protocol

The single-shuffle generic protocol proposed by Shinagawa and Nuida [33], which uses Yao's garbled circuit, represents a target function (to be computed) as a logic circuit, and performs the secure computation starting by arranging commitments based on the truth table of each gate.

Table 2 depicts the truth table for the AND gate. In each of these 0-1 values, the protocol places a commitment based on the truth table and the commitments to the inputs of the function, as shown in Table 3, for instance. Now, suppose that we turn over the commitments on the left and in the middle (which are 10 commitments in total). Check the revealed values of a and b against the left four columns of the truth table, and determine the matching row; then, the corresponding commitment on the right side will be a commitment to $a \wedge b$. By performing pile-scramble shuffles in key places as below, we can obtain a commitment to the gate output without leaking the value while maintaining the above function.

[ii]Practical implementations of shuffles were discussed in [9, 17, 25, 32], for example.

Regard Table 3 as a matrix with five rows and six columns. For this card matrix, apply a pile-scramble shuffle to the bottom four rows. Then, apply a pile-scramble shuffle to the first and second columns from the left, and the third and fourth columns, respectively. After that, turn over the cards in the first through fourth columns; then, the cards in the fifth and sixth columns of the row that match the cards in the original input (first row) will be an output commitment. In this case, because the commitments to a and b have an equal probability of being the same or their negation, no information about the values of a and b is leaked. In addition, the rows and values of the commitments corresponding to the values in the truth table are also shuffled, so that no information about the values of the output is leaked. Because the commitments to a and b and the commitments to the values in the truth table are shuffled simultaneously, the output commitment always represents $a \wedge b$. Therefore, the output value can be obtained without leaking the value of the input by applying pile-scramble shuffles. Shinagawa and Nuida's construction [33] is based on such a mechanism.

Furthermore, because the multiple pile-scramble shuffles described above are applied consecutively, they can be combined into a single shuffle, as mentioned in Section 2.3.

The example above is a 2-input AND function, and the number of gates in this function is one; therefore, in addition to the four cards for gate input commitments, 24 cards are used to construct the truth table. Thus, more generally, when an n-input logic function is represented by a logic circuit with q gates, we need a total of $2n + 24q$ cards, where $2n$ cards are for the input commitments and $24q$ cards are for the commitments to the q truth tables. As mentioned above, because multiple pile-scramble shuffles applied (in the whole process) can be combined into one, we can construct a protocol with a single shuffle using $2n + 24q$ cards for an arbitrary logical function[iii].

If we apply this generic protocol to the n-input AND function and the n-input XOR function, because the number of gates for these functions is $n - 1$, the number of required cards is $2n + 24(n - 1) = 26n - 24$. As has been done in conventional Yao's garbled circuit technique (e.g. [36]), it is expected that truth tables can be saved by specializing in the AND or XOR function; we will propose such card-based protocols in the following sections.

3 OUR PROPOSED n-INPUT AND PROTOCOL

In this section, we propose an n-input AND protocol that uses only one shuffle. The number of required cards is $4n - 2$. In Sections 3.1 and 3.2, we state the ideas behind our protocol. In Section 3.3, we describe the single-shuffle protocol and prove its correctness and security.

3.1 Basic Idea 1: How to Compute AND

Arrange the commitments to $x_1, x_2, \ldots, x_n \in \{0, 1\}$ and $n - 1$ commitments to 0 as follows:

[iii]Shinagawa and Nuida [33] also devised a method called *batching*, which can be used to perform the same computation with two pile-scramble shuffles, although more additional cards are required for batching.

$$\begin{array}{rl} 1: & \underbrace{\boxed{?}\boxed{?}}_{x_1} \\ 2: & \underbrace{\boxed{?}\boxed{?}}_{0}\ \underbrace{\boxed{?}\boxed{?}}_{x_2} \\ 3: & \underbrace{\boxed{?}\boxed{?}}_{0}\ \underbrace{\boxed{?}\boxed{?}}_{x_3} \\ & \vdots \\ n-1: & \underbrace{\boxed{?}\boxed{?}}_{0}\ \underbrace{\boxed{?}\boxed{?}}_{x_{n-1}} \\ n: & \underbrace{\boxed{?}\boxed{?}}_{0}\ \underbrace{\boxed{?}\boxed{?}}_{x_n} \end{array} \tag{1}$$

Now, let us turn over the commitment to x_1 on the first line. If its value is $x_1 = 0$, then the left commitment of the second line (which is the commitment to 0) is a commitment to $x_1 \wedge x_2$, and if its value is $x_1 = 1$, then the right commitment (which is the commitment to x_2) is a commitment to $x_1 \wedge x_2$, because the following relationship holds:

$$x_1 \wedge x_2 = \begin{cases} 0 & \text{if } x_1 = 0, \\ x_2 & \text{if } x_1 = 1. \end{cases}$$

Next, let us turn over the commitment to $x_1 \wedge x_2$ obtained now. If the value is 0, then the commitment on the left of the third line is a commitment to $x_1 \wedge x_2 \wedge x_3$, and if it is 1, then the commitment on the right is a commitment to $x_1 \wedge x_2 \wedge x_3$.

Bear this in mind, we can obtain a commitment to $x_1 \wedge x_2 \wedge \cdots \wedge x_n$

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

by the following *Procedure A*, starting from the arrangement in Eq. (1). Note that this arrangement has $2n - 1$ commitments in total, i.e., it consists of $4n - 2$ cards.

[Procedure A]

(1) Suppose that we have $2n - 1$ commitments arranged as in Eq. (1).
(2) Turn over the commitment on the first line and let its value be denoted by v. Let $i := 2$.
(3) If $v = 0$, then turn over the commitment on the left of the line i; if $v = 1$, then turn over the commitment on the right of the line i. Set v to the revealed value (overwriting v).
(4) Let $i := i + 1$. If $i \leq n - 1$, then return to (3).
(5) If $v = 0$, then output the commitment on the left of the line n; if $v = 1$, then output the commitment on the right.

Again, if we perform procedure A with the arrangement in Eq. (1) as input, we will obtain a commitment to $x_1 \wedge x_2 \wedge x_3 \cdots \wedge x_n$. However, because the values of $x_1, x_1 \wedge x_2, x_1 \wedge x_2 \wedge x_3, \ldots, x_1 \wedge x_2 \wedge \cdots \wedge x_{n-1}$ are leaked each time a commitment is turned over, it is of course not a secure computation.

3.2 Basic Idea 2: Randomization

Let us focus on the first and second lines in the arrangement in Eq. (1). If we swap the positions of the two cards that make up the commitment to x_1 on the first line and at the same time swap the positions of the commitments on the second line, the first line becomes a commitment to $\overline{x}_1$, and the second line becomes commitments to x_2 and 0 (in this order):

$$\begin{array}{rl} 1: & \underbrace{\boxed{?}\boxed{?}}_{\overline{x}_1} \\ 2: & \underbrace{\boxed{?}\boxed{?}}_{x_2}\ \underbrace{\boxed{?}\boxed{?}}_{0}. \end{array}$$

Since swapping of the first and second lines is synchronized, performing procedure A on the whole from this state still yields a commitment to $x_1 \wedge x_2 \wedge \cdots \wedge x_n$. Therefore, applying the following shuffle to the first and second lines in the arrangement in Eq. (1) does not affect the output of procedure A:

$$\begin{array}{c} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}} \\ \overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\ \overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \end{array}$$

$$\rightarrow \quad \begin{array}{c} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}} \\ \overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\ \overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \end{array} \quad \text{or} \quad \begin{array}{c} \overset{2}{\boxed{?}}\overset{1}{\boxed{?}} \\ \overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\ \overset{3}{\boxed{?}}\overset{4}{\boxed{?}} \end{array} \tag{2}$$

We call this the shuffle $\mathcal{S}_1$ (which was used in the Mizuki–Sone AND protocol [21]).

In a similar way, consider applying the following shuffle $\mathcal{S}_i$ to the lines i and $i + 1$ for every i, $2 \leq i \leq n - 1$:

$$\begin{array}{rl} i: & \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\ \overset{3}{\boxed{?}}\overset{4}{\boxed{?}} \\ i+1: & \overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\ \overset{7}{\boxed{?}}\overset{8}{\boxed{?}} \end{array}$$

$$\rightarrow \quad \begin{array}{c} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\ \overset{3}{\boxed{?}}\overset{4}{\boxed{?}} \\ \overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\ \overset{7}{\boxed{?}}\overset{8}{\boxed{?}} \end{array} \quad \text{or} \quad \begin{array}{c} \overset{2}{\boxed{?}}\overset{1}{\boxed{?}}\ \overset{4}{\boxed{?}}\overset{3}{\boxed{?}} \\ \overset{7}{\boxed{?}}\overset{8}{\boxed{?}}\ \overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \end{array} \tag{3}$$

Note that the index starts at '1' for simplicity. Applying these shuffles to the arrangement in Eq. (1) also does not affect the output of procedure A. These shuffles can be achieved by applying pile-scramble shuffles and rearranging. In our proposed AND protocol, we use the shuffles $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n-1}$ to randomize the commitments.

3.3 Description

We are now ready to describe our proposed AND protocol. Given n commitments along with $2n-2$ additional cards as input

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\cdots\boxed{\clubsuit}\boxed{\heartsuit},$$

the protocol proceeds as follows.

(1) Turn over all the $2n-2$ additional cards to make $n-1$ commitments to 0 and place them along with the commitments to $x_1, \ldots, x_n$, as shown in the arrangement in Eq. (1).
(2) Combine the $n-1$ shuffles $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n-1}$ defined in Section 3.2 into a single shuffle (as we mentioned in Section 2.3 that this kind of combining is possible), and apply that shuffle to the arrangement.
(3) For the obtained arrangement, apply the procedure A defined in Section 3.1. Its output is a commitment to $x_1 \wedge x_2 \wedge \cdots \wedge x_n$

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

As explained in Sections 3.1 and 3.2, the application of any shuffle $\mathcal{S}_i$ to the arrangement in Eq. (1) does not affect the output of procedure A. Therefore, it is clear that a commitment to $x_1 \wedge x_2 \cdots \wedge x_n$ is correctly obtained in step 3, which guarantees the correctness of our protocol.

Furthermore, in this protocol, the cards are turned over only during the execution of procedure A in step 2, and one commitment value is revealed for each of the lines from the line 1 to $n-1$. For example, on the first line, the value of $x_1 \oplus r_1$ is revealed, where r_1 is a uniformly distributed random bit generated by the shuffle $\mathcal{S}_1$. Therefore, even if the value of this commitment is revealed, the value of the input x_1 is not leaked. In the same way, the value revealed on the line i, $2 \le i \le n-1$, is $(x_1 \wedge \cdots \wedge x_i) \oplus r_i$, and because r_i is a random bit, the value of the input is not leaked. Hence, our proposed protocol is information-theoretically secure, and the security of the protocol is confirmed.

4 OUR PROPOSED n-INPUT XOR PROTOCOL

In this section, we propose an n-input XOR protocol that uses only one shuffle. The number of required cards is $2n$. In Sections 4.1 and 4.2, we state the ideas behind our protocol. In Section 4.3, we describe the protocol and show its correctness and security.

4.1 Basic Idea 1: How to Compute XOR

Arrange the commitments to $x_1, x_2, \cdots, x_n \in \{0,1\}$ as follows:

$$\begin{array}{rc} 1: & \underbrace{\boxed{?}\boxed{?}}_{x_1} \\ 2: & \underbrace{\boxed{?}\boxed{?}}_{x_2} \\ 3: & \underbrace{\boxed{?}\boxed{?}}_{x_3} \\ & \vdots \\ n-1: & \underbrace{\boxed{?}\boxed{?}}_{x_{n-1}} \\ n: & \underbrace{\boxed{?}\boxed{?}}_{x_n} \end{array} \tag{4}$$

Now, let us turn over the commitment to x_1 on the first line. If its value is $x_1 = 0$, then the commitment on the second line (which is a commitment to x_2) is a commitment to $x_1 \oplus x_2$, and if its value is $x_1 = 1$, then the negation of the commitment on the second line (which is a commitment to $\overline{x}_2$) is a commitment to $x_1 \oplus x_2$, because the following relationship holds:

$$x_1 \oplus x_2 = \begin{cases} x_2 & \text{if } x_1 = 0, \\ \overline{x}_2 & \text{if } x_1 = 1. \end{cases}$$

Next, let us turn over the commitment to $x_1 \oplus x_2$ obtained now. If the value is 0, the commitment on the third line is a commitment to $x_1 \oplus x_2 \oplus x_3$, and if it is 1, the negation of the commitment on the third line is a commitment to $x_1 \oplus x_2 \oplus x_3$.

Bear this in mind, we can obtain a commitment to $x_1 \oplus x_2 \oplus \cdots \oplus x_n$

$$\underbrace{\boxed{?}\boxed{?}}_{x_1\oplus x_2\oplus\cdots\oplus x_n}$$

by the following *Procedure B*. Note that the arrangement in Eq. (4) has n commitments in total, i.e., it consists of $2n$ cards.

[Procedure B]

(1) Suppose that we have n commitments arranged as in the arrangement in Eq. (4).
(2) Turn over the commitment on the first line and let its value be denoted by v. Let $i := 2$.
(3) If $v = 0$, then turn over the commitment on the line i; if $v = 1$, then swap the cards on the left and right of the commitment on the line i, turn over the commitment, and set v to the revealed value.
(4) Let $i := i + 1$. If $i \le n-1$, then return to (3).
(5) If $v = 0$, then output the commitment on the line n; if $v = 1$, then after swapping the cards on the left and right of the commitment on the line n, output the commitment.

Again, if we perform procedure B with the arrangement in Eq. (4) as input, we will obtain a commitment to $x_1 \oplus x_2 \oplus \cdots \oplus x_n$. However, the values of $x_1, x_1 \oplus x_2, x_1 \oplus x_2 \oplus x_3, \ldots, x_1 \oplus x_2 \oplus \cdots \oplus x_{n-1}$ are leaked every time when a commitment is revealed.

4.2 Basic Idea 2: Randomization

Let us focus on the lines i and $i+1$ in the arrangement in Eq. (4) for $1 \le i \le n-1$. If we swap the positions of the two cards that make up the commitment to x_i on the line i and swap the positions of the two cards that make up the commitment to x_{i+1} on the line $i+1$, the former one becomes a commitment to $\overline{x}_i$ and the latter one becomes a commitment to $\overline{x}_{i+1}$:

$$\begin{array}{rc} i: & \underbrace{\boxed{?}\boxed{?}}_{\overline{x}_i} \\ i+1: & \underbrace{\boxed{?}\boxed{?}}_{\overline{x}_{i+1}} \end{array} \tag{5}$$

Because swapping in the lines i and $i+1$ is synchronized, performing procedure B on the whole from this state still yields a commitment to $x_1 \oplus x_2 \oplus \cdots \oplus x_n$. Therefore, applying the following shuffle to the lines i and $i+1$ in the arrangement in Eq. (4) does not affect the output of procedure B:

$$\begin{array}{rc} i: & \overset{1}{\boxed{?}}\overset{2}{\boxed{?}} \\ i+1: & \overset{3}{\boxed{?}}\overset{4}{\boxed{?}} \end{array}$$

$$\rightarrow \quad \begin{array}{c} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}} \\ \overset{3}{\boxed{?}}\overset{4}{\boxed{?}} \end{array} \quad \text{or} \quad \begin{array}{c} \overset{2}{\boxed{?}}\overset{1}{\boxed{?}} \\ \overset{4}{\boxed{?}}\overset{3}{\boxed{?}} \end{array} \tag{6}$$

We call this the shuffle $\mathcal{T}_i$ (which was used in the Mizuki–Sone XOR protocol [21]). In our proposed XOR protocol, we use the shuffles $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{n-1}$ to randomize the commitments.

4.3 Description

We are now ready to describe our proposed XOR protocol. Given n commitments as input

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n},$$

the protocol proceeds as follows.

(1) Place the commitments to $x_1, \ldots, x_n$ as shown in the arrangement in Eq. (4).

(2) Combine the $n-1$ shuffles $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{n-1}$ defined in Section 4.2 into a single shuffle, and apply that shuffle to the commitments.

(3) For the obtained arrangement, apply the procedure B defined in Section 4.1. Its output is a commitment to $x_1 \oplus x_2 \oplus \cdots \oplus x_n$

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

As explained in Sections 4.1 and 4.2, the application of any shuffle $\mathcal{T}_i$ to the arrangement in Eq. (4) does not affect the output of procedure B. Therefore, it is clear that a commitment to $x_1 \oplus x_2 \cdots \oplus x_n$ is correctly obtained in step 3, which guarantees the correctness of our protocol.

Furthermore, in this protocol, the cards are turned over only during the execution of procedure B in step 2, and one commitment value is revealed for each of the lines from the lines 1 to $n-1$. For example, on the first line, the value of $x_1 \oplus r_1$ is revealed, where r_1 is a uniformly distributed random bit generated by the shuffle $\mathcal{T}_1$. Therefore, even if the value of this commitment is revealed, the value of the input x_1 is not leaked. In the same way, the value revealed on the line i, $2 \le i \le n-1$, is $(x_1 \oplus \cdots \oplus x_i) \oplus r_i$, and because r_i is a random bit, the value of the input is not leaked. Hence, our proposed protocol is information-theoretically secure, and the security of the protocol is confirmed.

5 DISCUSSION

In this section, we discuss some issues related to our two single-shuffle protocols presented in Sections 3 and 4.

5.1 Optimality

Because we need $2n$ cards for describing an n-bit input (as long as we follow the two-card per-bit encoding), our n-input XOR protocol, which uses exactly $2n$ cards, is optimal in the sense that the number of required cards is minimum.

By contrast, our n-input AND protocol uses $4n-2$ cards (meaning that it requires $2n-2$ helping cards). Therefore, we have a natural question whether there exists a single-shuffle AND protocol using fewer than $4n-2$ cards.

Let us consider the case of $n = 2$. Then, our two-input AND protocol (which is the same as the Mizuki–Sone AND protocol [21]) uses six cards. On the other hand, Koch, Walzer, and Härtel [10] proved that any finite-runtime two-input AND protocol requires five cards. Therefore, the gap between the upper and lower bounds currently known is one: It is open to determine whether a five-card single-shuffle AND protocol is possible or not. For $n \ge 3$, finding/proving optimal single-shuffle n-input AND protocols will be an important open problem, as well; a promising technique to obtain lower bounds on the number of required cards would be the "backwards calculus" developed by Koch [6]. (In addition, 'Table 4' shown in [8] may be useful to narrow down permutations to consider.)

5.2 Non-committed Output

Our proposed protocols as well as the Shinagawa–Nuida generic constructions [33] produce as output a commitment to the value of a predetermined function (such as the AND and XOR functions). If we allow the output to be any kind of encoding, we have a $(4n-3)$-card single-shuffle n-input AND protocol; specifically, in the arrangement (1), if we replace the n-th line with

$$\underbrace{\boxed{?}}_{0}\ \underbrace{\boxed{?}}_{x_n},$$

where we adopt the one-card encoding: $\boxed{\clubsuit} = 0$, $\boxed{\heartsuit} = 1$, then we can obtain the AND value based on the one-card encoding scheme above. Thus, by admitting a non-committed output, we can reduce the number of required cards for AND by one.

It should be noted that, from the non-committed four-card AND protocol given in [18], we can have a four-card non-committed single-shuffle AND protocol by combining the two shuffles used there; therefore, for the case of $n = 2$, this single-shuffle AND protocol is optimal in terms of required cards.

5.3 Two Pile-scramble Shuffles with Batching

As seen in Sections 3 and 4, we have constructed our single-shuffle protocols by combining a bunch of pile-scramble shuffles, so that the resulting single shuffle is very complicated and is no longer practical. If we want to stick to the pile-scramble shuffles, there is a nice solution: the *batching* technique developed also by Shinagawa and Nuida [33].

Briefly, the batching technique enables us to implement a number of "pair-wise disjoint" pile-scramble shuffles by applying one pile-scramble shuffle with the help of additional cards. This technique can be applied to our two protocols, as follows. Remember the shuffles $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n-1}$ appearing our AND protocol; because the shuffles $\mathcal{S}_1, \mathcal{S}_3, \mathcal{S}_5, \ldots$ are pair-wise disjoint, the batching technique enables us to implement these shuffles by a single pile-scramble shuffle. The same is true for $\mathcal{S}_2, \mathcal{S}_4, \mathcal{S}_6, \ldots$. Therefore, we can obtain an AND protocol using only two pile-scramble shuffles; in this case, we need $(4n + 2\lfloor \frac{n}{2} \rfloor \lceil \log_2 \lfloor \frac{n}{2} \rfloor \rceil)$ cards in total. Similarly, we can obtain an XOR protocol which uses only two pile-scramble shuffles if we accept $(2n + 2\lfloor \frac{n}{2} \rfloor \lceil \log_2 \lfloor \frac{n}{2} \rfloor \rceil)$ cards in total.

6 CONCLUSION

Shinagawa and Nuida [33] showed that, surprisingly, an arbitrary function can be computed with only one shuffle, based on the idea of Yao's garbled circuit. In this study, by focusing on specific functions, we tried to reduce the number of required cards while keeping the number of shuffles at one, and showed that the n-input AND function and the n-input XOR function can be computed with $4n - 2$ cards and $2n$ cards, respectively. Because the existing generic protocol requires $26n - 24$ cards to perform the same computations, in a sense, we succeeded in reducing the number of cards.

It is natural in a sense that the number of cards can be reduced by losing the advantage of being generic, but the problem of how far the number of cards can be reduced under the condition that applying a shuffle is allowed only once is considered to be one of the most important themes in clarifying the computational limits of card-based cryptography.

In the future, it is expected that our technique will be extended to functions other than AND and XOR. Because there is a gap between the AND and XOR computations in regard to the number of cards required by our protocols as discussed in Section 5, it is also worthwhile to examine whether this gap is inherent or not.

ACKNOWLEDGMENTS

We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. This work was supported in part by JSPS KAKENHI Grant Numbers JP19J21153 and JP21K11881.

REFERENCES

[1] Yuta Abe, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2021. Five-Card AND Computations in Committed Format Using Only Uniform Cyclic Shuffles. *New Gener. Comput.* 39 (2021), 97–114. Issue 1. https://doi.org/10.1007/s00354-020-00110-2

[2] Yuta Abe, Takaaki Mizuki, and Hideaki Sone. 2021. Committed-format AND protocol using only random cuts. *Nat. Comput.* 20, 4 (2021), 639–645. https://doi.org/10.1007/s11047-021-09862-2

[3] Rie Ishikawa, Eikoh Chida, and Takaaki Mizuki. 2015. Efficient Card-Based Protocols for Generating a Hidden Random Permutation Without Fixed Points. In *Unconventional Computation and Natural Computation (Lecture Notes in Computer Science, Vol. 9252)*, Cristian S. Calude and Michael J. Dinneen (Eds.). Springer, Cham, 215–226. https://doi.org/10.1007/978-3-319-21819-9_16

[4] Raimu Isuzugawa, Kodai Toyoda, Yu Sasaki, Daiki Miyahara, and Takaaki Mizuki. 2021. A Card-Minimal Three-Input AND Protocol Using Two Shuffles. In *Computing and Combinatorics (Lecture Notes in Computer Science, Vol. 13025)*, Chi-Yeh Chen, Wing-Kai Hon, Ling-Ju Hung, and Chia-Wei Lee (Eds.). Springer, Cham, 668–679. https://doi.org/10.1007/978-3-030-89543-3_55

[5] Julia Kastner, Alexander Koch, Stefan Walzer, Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2017. The Minimum Number of Cards in Practical Card-Based Protocols. In *Advances in Cryptology – ASIACRYPT 2017 (Lecture Notes in Computer Science, Vol. 10626)*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer, Cham, 126–155. https://doi.org/10.1007/978-3-319-70700-6_5

[6] Alexander Koch. 2018. The Landscape of Optimal Card-based Protocols. Cryptology ePrint Archive, Report 2018/951. https://eprint.iacr.org/2018/951.

[7] Alexander Koch. 2021. The Landscape of Security from Physical Assumptions. In *2021 IEEE Information Theory Workshop (ITW)*. IEEE, Los Alamitos, CA, USA, 1–6. https://doi.org/10.1109/ITW48936.2021.9611501

[8] Alexander Koch, Michael Schrempp, and Michael Kirsten. 2021. Card-based cryptography meets formal verification. *New Gener. Comput.* 39, 1 (2021), 115–158. https://doi.org/10.1007/s00354-020-00120-0

[9] Alexander Koch and Stefan Walzer. 2020. Foundations for Actively Secure Card-Based Cryptography. In *Fun with Algorithms (LIPIcs, Vol. 157)*, Martin Farach-Colton, Giuseppe Prencipe, and Ryuhei Uehara (Eds.). Schloss Dagstuhl, Dagstuhl, Germany, 17:1–17:23. https://doi.org/10.4230/LIPIcs.FUN.2021.17

[10] Alexander Koch, Stefan Walzer, and Kevin Härtel. 2015. Card-Based Cryptographic Protocols Using a Minimal Number of Cards. In *Advances in Cryptology – ASIACRYPT 2015 (Lecture Notes in Computer Science, Vol. 9452)*, Tetsu Iwata and Jung Hee Cheon (Eds.). Springer, Berlin, Heidelberg, 783–807. https://doi.org/10.1007/978-3-662-48797-6_32

[11] Hiroto Koyama, Kodai Toyoda, Daiki Miyahara, and Takaaki Mizuki. 2021. New Card-Based Copy Protocols Using Only Random Cuts. In *ASIA Public-Key Cryptography Workshop* (Hong Kong) *(APKC '21)*. ACM, New York, NY, USA, 13–22. https://doi.org/10.1145/3457338.3458297

[12] Pascal Lafourcade, Daiki Miyahara, Takaaki Mizuki, Léo Robert, Tatsuya Sasaki, and Hideaki Sone. 2021. How to Construct Physical Zero-Knowledge Proofs for Puzzles with a "Single Loop" Condition. *Theor. Comput. Sci.* 888 (2021), 41–55. https://doi.org/10.1016/j.tcs.2021.07.019

[13] Yoshifumi Manabe and Hibiki Ono. 2021. Secure Card-Based Cryptographic Protocols Using Private Operations Against Malicious Players. In *Innovative Security Solutions for Information Technology and Communications*, Diana Maimut, Andrei-George Oprina, and Damien Sauveron (Eds.). Springer, Cham, 55–70. https://doi.org/10.1007/978-3-030-69255-1_5

[14] Yoshifumi Manabe and Hibiki Ono. 2022. Card-Based Cryptographic Protocols with Malicious Players Using Private Operations. *New Gener. Comput.* (2022), 27 pages. https://doi.org/10.1007/s00354-021-00148-w in press.

[15] Daiki Miyahara, Hiromichi Haneda, and Takaaki Mizuki. 2021. Card-Based Zero-Knowledge Proof Protocols for Graph Problems and Their Computational Model. In *Provable and Practical Security (Lecture Notes in Computer Science, Vol. 13059)*, Qiong Huang and Yu Yu (Eds.). Springer, Cham, 136–152. https://doi.org/10.1007/978-3-030-90402-9_8

[16] Daiki Miyahara, Itaru Ueda, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2021. Evaluating card-based protocols in terms of execution time. *Int. J. Inf. Secur.* 20, 5 (2021), 729–740. https://doi.org/10.1007/s10207-020-00525-4

[17] Kengo Miyamoto and Kazumasa Shinagawa. 2021. Graph Automorphism Shuffles from Pile-Scramble Shuffles. *CoRR* abs/2109.00397 (2021), 16 pages. https://arxiv.org/abs/2109.00397

[18] Takaaki Mizuki, Michihito Kumamoto, and Hideaki Sone. 2012. The Five-Card Trick Can Be Done with Four Cards. In *Advances in Cryptology – ASIACRYPT 2012 (Lecture Notes in Computer Science, Vol. 7658)*, Xiaoyun Wang and Kazue Sako (Eds.). Springer, Berlin, Heidelberg, 598–606. https://doi.org/10.1007/978-3-642-34961-4_36

[19] Takaaki Mizuki and Hiroki Shizuya. 2014. A formalization of card-based cryptographic protocols via abstract machine. *Int. J. Inf. Secur.* 13, 1 (2014), 15–23. https://doi.org/10.1007/s10207-013-0219-4

[20] Takaaki Mizuki and Hiroki Shizuya. 2017. Computational Model of Card-Based Cryptographic Protocols and Its Applications. *IEICE Trans. Fundamentals* E100.A, 1 (2017), 3–11. https://doi.org/10.1587/transfun.E100.A.3

[21] Takaaki Mizuki and Hideaki Sone. 2009. Six-Card Secure AND and Four-Card Secure XOR. In *Frontiers in Algorithmics (Lecture Notes in Computer Science,*

Vol. 5598), Xiaotie Deng, John E. Hopcroft, and Jinyun Xue (Eds.). Springer, Berlin, Heidelberg, 358–369. https://doi.org/10.1007/978-3-642-02270-8_36

[22] Takeshi Nakai, Yuto Misawa, Yuuki Tokushige, Mitsugu Iwamoto, and Kazuo Ohta. 2021. How to Solve Millionaires' Problem with Two Kinds of Cards. *New Gener. Comput.* 39, 1 (2021), 73–96. https://doi.org/10.1007/s00354-020-00118-8

[23] Takeshi Nakai, Satoshi Shirouchi, Yuuki Tokushige, Mitsugu Iwamoto, and Kazuo Ohta. 2022. Secure Computation for Threshold Functions with Physical Cards: Power of Private Permutations. *New Gener. Comput.* (2022), 19 pages. https://doi.org/10.1007/s00354-022-00153-7 in press.

[24] Takuya Nishida, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2015. Card-Based Protocols for Any Boolean Function. In *Theory and Applications of Models of Computation (Lecture Notes in Computer Science, Vol. 9076)*, Rahul Jain, Sanjay Jain, and Frank Stephan (Eds.). Springer, Cham, 110–121. https://doi.org/10.1007/978-3-319-17142-5_11

[25] Akihiro Nishimura, Takuya Nishida, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2018. Card-based protocols using unequal division shuffles. *Soft Comput.* 22 (2018), 361–371. https://doi.org/10.1007/s00500-017-2858-2

[26] Hibiki Ono and Yoshifumi Manabe. 2021. Card-Based Cryptographic Logical Computations Using Private Operations. *New Gener. Comput.* 39 (2021), 19–40. Issue 1. https://doi.org/10.1007/s00354-020-00113-z

[27] Léo Robert, Daiki Miyahara, Pascal Lafourcade, Luc Libralesso, and Takaaki Mizuki. 2021. Physical zero-knowledge proof and NP-completeness proof of Suguru puzzle. *Inf. Comput.* (2021), 14 pages. https://doi.org/10.1016/j.ic.2021.104858 in press.

[28] Léo Robert, Daiki Miyahara, Pascal Lafourcade, and Takaaki Mizuki. 2022. Card-Based ZKP for Connectivity: Applications to Nurikabe, Hitori, and Heyawake. *New Gener. Comput.* (2022), 23 pages. https://doi.org/10.1007/s00354-022-00155-5 in press.

[29] Suthee Ruangwises. 2022. Two Standard Decks of Playing Cards are Sufficient for a ZKP for Sudoku. *New Gener. Comput.* (2022), 17 pages. https://doi.org/10.1007/s00354-021-00146-y in press.

[30] Suthee Ruangwises and Toshiya Itoh. 2020. Physical Zero-Knowledge Proof for Numberlink Puzzle and k Vertex-Disjoint Paths Problem. *New Gener. Comput.* 39 (2020), 3–17. https://doi.org/10.1007/s00354-020-00114-y

[31] Suthee Ruangwises and Toshiya Itoh. 2021. Physical zero-knowledge proof for Ripple Effect. *Theor. Comput. Sci.* 895 (2021), 115–123. https://doi.org/10.1016/j.tcs.2021.09.034

[32] Takahiro Saito, Daiki Miyahara, Yuta Abe, Takaaki Mizuki, and Hiroki Shizuya. 2020. How to Implement a Non-uniform or Non-closed Shuffle. In *Theory and Practice of Natural Computing (Lecture Notes in Computer Science, Vol. 12494)*, Carlos Martín-Vide, Miguel A. Vega-Rodríguez, and Miin-Shen Yang (Eds.). Springer, Cham, 107–118. https://doi.org/10.1007/978-3-030-63000-3_9

[33] Kazumasa Shinagawa and Koji Nuida. 2021. A single shuffle is enough for secure card-based computation of any Boolean circuit. *Discrete Appl. Math.* 289 (2021), 248–261. https://doi.org/10.1016/j.dam.2020.10.013

[34] Kodai Toyoda, Daiki Miyahara, and Takaaki Mizuki. 2021. Another Use of the Five-Card Trick: Card-Minimal Secure Three-Input Majority Function Evaluation. In *Progress in Cryptology – INDOCRYPT 2021 (Lecture Notes in Computer Science, Vol. 13143)*, Avishek Adhikari, Ralf Küsters, and Bart Preneel (Eds.). Springer, Cham, 536–555. https://doi.org/10.1007/978-3-030-92518-5_24

[35] Kenji Yasunaga. 2020. Practical Card-Based Protocol for Three-Input Majority. *IEICE Trans. Fundamentals* E103.A, 11 (2020), 1296–1298. https://doi.org/10.1587/transfun.2020EAL2025

[36] Samee Zahur, Mike Rosulek, and David Evans. 2015. Two Halves Make a Whole. In *Advances in Cryptology - EUROCRYPT 2015 (Lecture Notes in Computer Science, Vol. 9057)*, Elisabeth Oswald and Marc Fischlin (Eds.). Springer, Berlin, Heidelberg, 220–250. https://doi.org/10.1007/978-3-662-46803-6_8