# Single-Shuffle Card-based Protocol with Eight Cards per Gate*

Kazunari Tozawa[1], Hiraku Morita[2], and Takaaki Mizuki[3]

[1] The University of Tokyo, Tokyo, Japan
[2] University of St.Gallen, St.Gallen, Switzerland
[3] Tohoku University, Miyagi, Japan

**Abstract.** Card-based cryptography allows us to securely compute arbitrary functions using a deck of physical cards. Its performance is mainly measured by the number of used cards and shuffles, and there is a line of work that aims to reduce either of them. One of the seminal work is by Shinagawa and Nuida (Discrete Applied Mathematics 2021) that shows any Boolean function can be constructed by shuffling only once based on the garbling scheme. Their construction requires $2n + 24g$ cards for an $n$-input Boolean function that is represented by $g$ logical gates. In this paper, we reduce the number of cards to $2n + 8g$ for arbitrary functions while keeping it working with only one shuffle.

**Keywords:** Card-based cryptography · Garbled circuit · XOR shuffle.

## 1 Introduction

Card-based cryptography is unconventional computing which performs cryptographic tasks such as secure computations, which exploits a deck of physical cards [8, 14, 15, 20]. A card-based cryptographic protocol typically uses a two-color deck consisting of $\boxed{\heartsuit}$ and $\boxed{\clubsuit}$ whose backs are all identical $\boxed{?}$, and the following encoding rule is usually used to represent Boolean values:

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

The complexity of a card-based protocol is measured in terms of the number of cards and shuffles, which correspond to the space and time complexities of the protocol, respectively.

As for minimizing the number of required cards, the currently known best result is that $2n+6$ cards are sufficient to construct any $n$-variable Boolean function [19]. However, it needs an exponential number of shuffles. As for minimizing

* This paper appears in Proceedings of UCNC 2023. This version of the contribution has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: `https://doi.org/10.1007/978-3-031-34034-5_12`. 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`.

the number of shuffles, Shinagawa and Nuida [24] proved that only one shuffle is enough to design a protocol securely computing any Boolean function (although it needs a relatively large number of cards as mentioned below). This paper mainly focuses on the latter, i.e., improving the Shinagawa–Nuida single-shuffle construction.

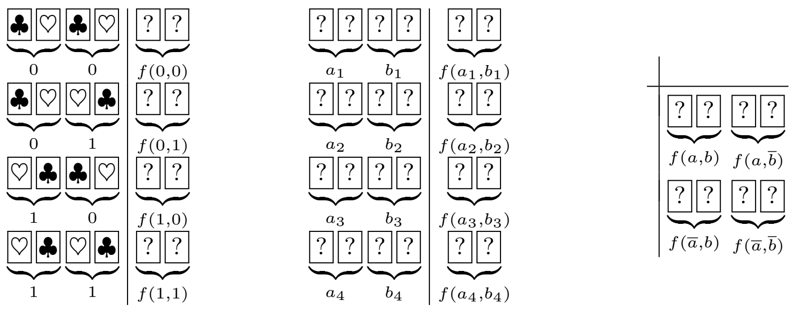
More specifically, Shinagawa and Nuida [24] have investigated the relation between card-based cryptography and garbled circuit techniques. They have proposed a card-based variant of a garbled circuit, called *Card-based Garbled Circuit*. The protocol enables us to compute any Boolean function with the optimal number of shuffles, namely exactly one shuffle. Regarding the number of cards, the protocol for an $n$-input Boolean function requires $2n + 24g$ cards, where $g$ is the number of gates (when describing the function as a circuit).

**Technical Overview of Our Scheme.** Our goal is to minimize the number of cards required to represent a card-based garbled circuit. To address this, we propose a new method to represent garbled gates with a small number of cards, that is, 8 cards per gate. Our method helps to reduce the primitive cost of card-based garbled circuits which can be seen as reducing the size of the garbled tables, allowing the number of cards to be reduced proportionally to the number of gates.

The basic idea of garbled circuit techniques is to consider each gate as an encryption of the corresponding truth table. A truth table of a logic gate consists of 12 cells, so the most straightforward way to represent it in card-based cryptography is to use 24 cards with standard encoding, as shown in Figure 1a. The Shinagawa-Nuida single-shuffle construction shows that this representation allows privacy-preserving computations of any binary gate. The garbling stage of their scheme aims to encrypt the truth table of each gate by turning down the cards and randomly permuting the rows of the table. The resulting truth table looks as shown in Figure 1b. Since the positions of the result values are uniformly random, the evaluation stage of the gate can make all values in the operand cells public.

On the other hand, our scheme uses only 8 cards to represent a truth table as shown in Figure 1c. This reduction comes from the observation that the operand cell values are only positional information for each result cell. In other words, we can omit the cards required for the operand cells, as long as the positional information is recoverable. A similar optimization has already been proposed in the context of garbled circuits, called the *point-and-permute* technique [2]. We propose a variant of the point-and-permute technique in card-based cryptography.

**Our Contributions.** We propose a protocol for any Boolean circuit with a single shuffle. Compared to the existing protocol [24], we reduce the number of cards from $2n + 24g$ to $2n + 8g$, where $g$ is the number of gates.



(a) Plain-text (b) Shinagawa-Nuida (c) Our scheme

Fig. 1: Card-based representations of the truth table for gate $f$

**Related Work.** As mentioned above, the goal of this paper is to provide a generic construction for securely computing any Boolean functions using only one shuffle. The Shinagawa-Nuida's single-shuffle construction [24] has been the only one that achieves the goal. There are several single-shuffle protocols for specific elementary functions such as the AND [4, 16, 18], XOR [18], 3-input equality [6, 23], 3-input majority [25], $n$-input AND [13], and $n$-input XOR [13] functions. Another aspect of previous work is to reduce the number of cards; especially, designing card-minimal protocols has attracted attention [5, 9–12, 21, 25].

Bellare *et al.* [3] proposed garbling schemes, an abstraction of garbled circuit methods, and formalized security properties of garbled circuits: privacy and obliviousness.

## 2 Preliminaries

### 2.1 Notation

We use a bold symbol to represent a vector (or an ordered set). Let $\boldsymbol{v}_i$ denote the $i$-th (starting from 1) element of $\boldsymbol{v}$. We write $\boldsymbol{v} \parallel \boldsymbol{w}$ for the concatenation of $\boldsymbol{v}$ and $\boldsymbol{w}$.

Let $S_n$ denote the symmetric group on $n$ elements. For a vector $\boldsymbol{v}$ of degree $n$, we define a place-permutation action of $S_n$ as $(\boldsymbol{v} \cdot \pi)_i = \boldsymbol{v}_{\pi(i)}$, where $\pi \in S_n$. For two permutations $\pi, \rho$, we use the notation $\pi\rho$ to represent the product of permutations such that $\boldsymbol{v} \cdot (\pi\rho) = (\boldsymbol{v} \cdot \pi) \cdot \rho$ for any vector $\boldsymbol{v}$. We use the notation $(a, b) \in S_n$ to denote the transposition (*i.e.,* a cycle with two elements) that swaps $a$ and $b$.

For $b, c \in \mathbb{Z}_2$, we will write $\overline{b}$, $b \oplus c$, $b \vee c$ and $b \wedge c$ for the negation of $b$, the XOR, OR and AND of $b$ and $c$, respectively.

### 2.2 Card-based Cryptographic Protocol

In this paper, we follow the computation model proposed in [17]. There are two kinds of cards: The face side of a card is either $\boxed{\clubsuit}$ or $\boxed{\heartsuit}$, and the back side is indistinguishable from the other cards, denoted by $\boxed{?}$. Each card during the protocol execution can be in one of the two states: face-top or face-down. A deck is a finite vector on $\{\heartsuit, \clubsuit\} \times \{\text{face-top}, \text{face-down}\}$.

As mentioned before, a bit $x \in \mathbb{Z}_2$ is represented by a single pair of cards as follows:

$$0 = \boxed{\clubsuit}\boxed{\heartsuit} \qquad\qquad 1 = \boxed{\heartsuit}\boxed{\clubsuit}$$

A *commitment* of $x \in \mathbb{Z}_2$, denoted by $\mathsf{Com}(x)$, is a pair of face-down cards representing $x$.

For a current deck $D$ of $n$ cards, a protocol can perform the following operations:

– $(\mathsf{Perm}, \pi)$, where $\pi \in S_n$. The operation converts $D$ into $D \cdot \pi$.
– $(\mathsf{Shuffle}, G)$, where $G$ is a permutation group on $n$ points. The operation converts $D$ into $D \cdot \pi$, where $\pi$ is a random permutation chosen uniformly on $G$.
– $(\mathsf{Open}, S)$, where $S \subseteq \{1, \ldots, n\}$. The operation makes the state of the $i$-th card in $D$ face-top for all $i \in S$.

In particular, when $G$ is isomorphic to some symmetric group $S_k$ for $k \leq n$, $(\mathsf{Shuffle}, G)$ is called a pile-scramble shuffle [7]. Let $\boldsymbol{I}^{(1)}, \ldots, \boldsymbol{I}^{(k)}$ be the disjoint ordered subsets of $\{1, \ldots, n\}$ of the same size $s$. We define $G$ as the permutation group generated by the products of parallel transpositions $(\boldsymbol{I}_1^{(i)}, \boldsymbol{I}_1^{(j)})(\boldsymbol{I}_2^{(i)}, \boldsymbol{I}_2^{(j)}) \cdots (\boldsymbol{I}_s^{(i)}, \boldsymbol{I}_s^{(j)})$ for all $i < j$. In this case, we use the syntactic sugar $(\mathsf{PileShuffle}, \boldsymbol{I}^{(1)}, \ldots, \boldsymbol{I}^{(k)})$ for $(\mathsf{Shuffle}, G)$.

The protocol is in a committed format if the output is also encoded in the same manner as input; $0 = \boxed{\clubsuit}\boxed{\heartsuit}, 1 = \boxed{\heartsuit}\boxed{\clubsuit}$. Our construction in this paper focuses on a committed format so that it will be easy to use the committed output of our protocol as an input of some other protocols.

## 2.3 Garbled Circuit

Let $f = (n, m, g, \mathsf{Wires}, A, B, G)$ be a Boolean circuit. Here, $n$, $m$ and $g$ denote the numbers of inputs, outputs and gates, respectively. All input wires and gates in $f$ are assigned unique numbers belonging to $\mathsf{Inputs} = \{1, \ldots, n\}$ and $\mathsf{Gates} = \{n+1, \ldots, n+g\}$, respectively. The wire coming out of gate $i$ is also assigned $i$, so the wires correspond to $\mathsf{Wires} = \{1, \ldots, n+g\}$. The functions $A, B : \mathsf{Gates} \to \mathsf{Wires} \setminus \mathsf{Outputs}$ respectively specify the first and second input wires of a gate, and $G : \mathsf{Gates} \times \{0,1\} \times \{0,1\} \to \{0,1\}$ specifies the functionality of each gate. We simply write $A_i$, $B_i$ and $G_i$ for $A(i)$, $B(i)$ and $G(i, \cdot, \cdot)$, and write $A_i^{-1}$ and $B_i^{-1}$ for $A^{-1}(i)$ and $B^{-1}(i)$, respectively. We assume that $A_i < B_i < i$ holds for all $i$. Then all output wires in $f$ belong to $\mathsf{Outputs} = \{n+g-m+1, \ldots, n+g\}$.

Garbling scheme [3] consists of the following algorithms:

– $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$: Given a security parameter $k \in \mathbb{N}$ and a function $f : \{0,1\}^n \to \{0,1\}^m$, it outputs a garbled circuit $F$, encoding information $e$, and decoding information $d$

– $X \leftarrow \mathsf{Enc}(e, x)$: Given encoding information $e$ and an input $x \in \{0,1\}^n$, it outputs a garbled input $X$
– $Y \leftarrow \mathsf{Eval}(F, X)$: Given a garbled function $F$ and a garbled input $X$, it outputs a garbled output $Y$
– $y \leftarrow \mathsf{Dec}(d, Y)$: Given decoding information $d$ and a garbled output $Y$, it outputs a plain output $y$

The security properties we focus on are described as follows:

– Privacy: The tuple $(F, X, d)$ should not reveal any information on the input $x$ except the output $f(x)$. Here, there must exist a simulator $\mathcal{S}$ that takes input $(1^k, f, f(x))$ and outputs $(F', X', d')$ that is indistinguishable from $(F, X, d)$ that would be generated by the protocol.
– Obliviousness: The tuple $(F, X)$ should not reveal any information on $x$. Here, there must exist a simulator $\mathcal{S}$ that takes input $(1^k, f)$ and outputs $(F', X')$ that is indistinguishable from $(F, X)$ that would be generated by the protocol.

Note that in this paper we do not use the security parameter $k$ and only consider the security properties in the information-theoretical sense.

### 2.4 Card-based Garbling Scheme

Our card-based garbling protocol (Protocol 2) realizes the standard garbling and encoding algorithms at once, as in Shinagawa and Nuida [24]. In addition, when considering a protocol in a committed form, the decoding phase is omitted. In summary, the functionalities of a card-based garbling scheme are described as follows:

– $I \leftarrow \mathsf{Init}(x, f)$: Given an input $x$ and a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$, it outputs a committed initial state $I$
– $(F, X) \leftarrow \mathsf{CardGb}(I)$: Given a security parameter $k \in \mathbb{N}$ and an initial state regarding an input $x$ and a function $f$, it outputs a garbled circuit $F$, decoding information $d$, and a garbled input $X$
– $Y \leftarrow \mathsf{CardEval}(F, X)$: Given a garbled function $F$ and a garbled input $X$, it outputs a garbled output $Y$

## 3 Main Protocol

In this section, we propose an efficient protocol for card-based garbled circuits. The main idea of our scheme is to use an elaborate shuffling technique for randomization. We show that the XOR shuffle technique for the secret-sharing-based scheme of [1] is also applicable to card-based cryptography. As a result, the XOR shuffle provides an analog of the point-and-permute technique in the field of garbled circuits, reducing the number of cards required to represent the circuit.

### 3.1 Example: for a circuit with one gate

To grasp the intuition of our idea, let us consider the simplest case with a single binary logic gate. In this case, the Boolean circuit is described as $(2, 1, 1, \{1, 2, 3\}, \{3 \mapsto 1\}, \{3 \mapsto 2\}, f)$, where $f$ is the functionality of the logic gate. The procedure has three phases: initialization, garbling, and evaluation.

**Initialization.** Let $x_1$ and $x_2$ be distinct inputs, and $f$ be the binary Boolean function we want to evaluate at the gate. Given the commitments of $x_1$ and $x_2$, the initial state is set as follows:

$$\underbrace{\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}}_{x_1}\underbrace{\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}}_{x_2}\underbrace{\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}}_{f(0,0)}\underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{f(0,1)}\underbrace{\overset{9}{\boxed{?}}\overset{10}{\boxed{?}}}_{f(1,0)}\underbrace{\overset{11}{\boxed{?}}\overset{12}{\boxed{?}}}_{f(1,1)}$$

Here, the first four cards are the commitments of $x_1$ and $x_2$, and the latter eight cards are the commitments of the values of $f(0, 0)$, $f(0, 1)$, $f(1, 0)$, and $f(1, 1)$, *i.e.,* an encryption of the truth table for $f$. See Sect. 3.2 for the detail.

**Garbling.** In order to obliviously select the desired output from the encrypted truth table while keeping the input values secret, we apply an XOR shuffle to the initial state[4]. An XOR Shuffle consists of two consecutive pile-scramble shuffles as follows:

1. $(\mathsf{PileShuffle}, \{1, 5, 6, 7, 8\}, \{2, 9, 10, 11, 12\})$

$$\begin{array}{cccccccccccc} 1&2&3&4&5&6&7&8&9&10&11&12\\ \boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}\\ \circ&\bullet&&&\circ&\circ&\circ&\circ&\bullet&\bullet&\bullet&\bullet \end{array}$$

2. $(\mathsf{PileShuffle}, \{3, 5, 6, 9, 10\}, \{4, 7, 8, 11, 12\})$

$$\begin{array}{cccccccccccc} 1&2&3&4&5&6&7&8&9&10&11&12\\ \boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}\\ &&\circ&\bullet&\circ&\circ&\bullet&\bullet&\circ&\circ&\bullet&\bullet \end{array}$$

A pile scramble shuffle allows us to swap the positions of the white- and black-marked cards with equal probability while maintaining the order of the cards marked each color. After the XOR shuffle, the final state can be one of the following four cases:

$$\underbrace{\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}}_{x_1}\underbrace{\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}}_{x_2}\underbrace{\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}}_{f(0,0)}\underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{f(0,1)}\underbrace{\overset{9}{\boxed{?}}\overset{10}{\boxed{?}}}_{f(1,0)}\underbrace{\overset{11}{\boxed{?}}\overset{12}{\boxed{?}}}_{f(1,1)} \qquad \underbrace{\overset{2}{\boxed{?}}\overset{1}{\boxed{?}}}_{\overline{x_1}}\underbrace{\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}}_{x_2}\underbrace{\overset{9}{\boxed{?}}\overset{10}{\boxed{?}}}_{f(1,0)}\underbrace{\overset{11}{\boxed{?}}\overset{12}{\boxed{?}}}_{f(1,1)}\underbrace{\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}}_{f(0,0)}\underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{f(0,1)}$$

$$\underbrace{\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}}_{x_1}\underbrace{\overset{4}{\boxed{?}}\overset{3}{\boxed{?}}}_{\overline{x_2}}\underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{f(0,1)}\underbrace{\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}}_{f(0,0)}\underbrace{\overset{11}{\boxed{?}}\overset{12}{\boxed{?}}}_{f(1,1)}\underbrace{\overset{9}{\boxed{?}}\overset{10}{\boxed{?}}}_{f(1,0)} \qquad \underbrace{\overset{2}{\boxed{?}}\overset{1}{\boxed{?}}}_{\overline{x_1}}\underbrace{\overset{4}{\boxed{?}}\overset{3}{\boxed{?}}}_{\overline{x_2}}\underbrace{\overset{11}{\boxed{?}}\overset{12}{\boxed{?}}}_{f(1,1)}\underbrace{\overset{9}{\boxed{?}}\overset{10}{\boxed{?}}}_{f(1,0)}\underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{f(0,1)}\underbrace{\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}}_{f(0,0)}$$

[4] Note that a similar procedure was used for changing an integer encoding into two commitments [22].

where $\overline{x}$ denotes the negation of $x$. The key property of the XOR shuffle is that each case has an equal probability thanks to the randomness given by pile-scramble shuffles. Accordingly, the distribution of the two values committed on the first four cards is uniform on $\{0,1\}^2$ regardless of the input values $x_1$ and $x_2$. See Sect. 3.3 for the details.

**Evaluation.** The evaluation phase of our protocol proceeds as follows. The players first reveal the first four cards to obtain the two randomized values, denoted by $b_1$ and $b_2$. Then, depending on the values of $b_1$ and $b_2$, the players choose two of the latter eight cards according to the following rule:

- If $b_1 = 0$ and $b_2 = 0$, take 5 and 6
- If $b_1 = 0$ and $b_2 = 1$, take 7 and 8
- If $b_1 = 1$ and $b_2 = 0$, take 9 and 10
- If $b_1 = 1$ and $b_2 = 1$, take 11 and 12

The correctness of the protocol comes from the definition of XOR shuffle. When $b_1 = x_1 \oplus r_1$ and $b_2 = x_2 \oplus r_2$ for some $r_1, r_2 \in \mathbb{Z}_2$, the latter part are the commitments of the values of $f(r_1, r_2)$, $f(r_1, \overline{r_2})$, $f(\overline{r_1}, r_2)$ and $f(\overline{r_1}, \overline{r_2})$. Therefore, the two chosen cards are the commitment of the desired output value. See Sect. 3.4 for the details.

## 3.2 Initialization Phase

We first define the initial state of the protocol. Let $f = (n, m, g, \mathsf{Wires}, A, B, G)$ be a Boolean circuit. For $i \in \mathsf{Gates}$, we define an eight-card representation of the truth table of gate $i$ as follows:

$$\mathsf{Com}(f,i) := \mathsf{Com}(G_i(0,0)) \,\|\, \mathsf{Com}(G_i(0,1)) \,\|\, \mathsf{Com}(G_i(1,0)) \,\|\, \mathsf{Com}(G_i(1,1)).$$

We now describe a protocol for $\mathsf{Init}$ below. The protocol requires $2n + 8g$ cards.

**Protocol 1.** $\mathsf{Init}$

**Input:** $(\boldsymbol{x}, f)$, where $\boldsymbol{x}$ is a vector of input values, and $f$ is a Boolean circuit.
**Output:** $I$, where $I$ is an initial state consisting of $2n + 8g$ face-down cards.

1. Set $I = (\mathsf{Com}(\boldsymbol{x}_1) \,\|\, \cdots \,\|\, \mathsf{Com}(\boldsymbol{x}_n) \,\|\, \mathsf{Com}(f, n+1) \,\|\, \cdots \,\|\, \mathsf{Com}(f, n+g))$.
2. Output $I$.

For simplicity, we define the offset $\mathsf{a} : \mathsf{Wires} \to \{1, \ldots, 2n+8g\}$, which assigns the first position in the deck to the wire number of the circuit, as follows:

$$\mathsf{a}(i) = \begin{cases} 2i - 1 & i \in \mathsf{Inputs} \\ 8i - 6n - 7 & i \in \mathsf{Gates} \end{cases}$$

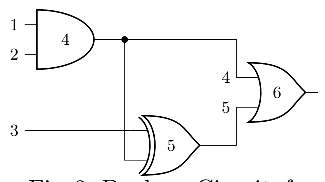


Fig. 2: Boolean Circuit $f$

**Example.** As an example, we consider the Boolean circuit in Fig. 2. Note that this example is the same as the Appendix example in [24]. Formally, the circuit is defined as $f = (3, 1, 3, A, B, G)$ where $A(4) = 1, A(5) = 3, A(6) = 4$, $B(4) = 2, B(5) = 4, B(6) = 5$, $G_4, G_5$ and $G_6$ are AND, XOR, and OR gates, respectively. In this case, the initial state of the protocol is set as 30 face-down cards arranged as follows:

$$\underbrace{\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}}_{x_1}\underbrace{\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}}_{x_2}\underbrace{\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}}_{x_3}\underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{0}\underbrace{\overset{9}{\boxed{?}}\overset{10}{\boxed{?}}}_{0}\underbrace{\overset{11}{\boxed{?}}\overset{12}{\boxed{?}}}_{0}\underbrace{\overset{13}{\boxed{?}}\overset{14}{\boxed{?}}}_{1}\underbrace{\overset{15}{\boxed{?}}\overset{16}{\boxed{?}}}_{0}\underbrace{\overset{17}{\boxed{?}}\overset{18}{\boxed{?}}}_{1}\underbrace{\overset{19}{\boxed{?}}\overset{20}{\boxed{?}}}_{1}\underbrace{\overset{21}{\boxed{?}}\overset{22}{\boxed{?}}}_{0}\underbrace{\overset{23}{\boxed{?}}\overset{24}{\boxed{?}}}_{0}\underbrace{\overset{25}{\boxed{?}}\overset{26}{\boxed{?}}}_{1}\underbrace{\overset{27}{\boxed{?}}\overset{28}{\boxed{?}}}_{1}\underbrace{\overset{29}{\boxed{?}}\overset{30}{\boxed{?}}}_{1}$$

## 3.3 Garbling Phase

Next, the protocol proceeds to the garbling phase. The players perform a series of pile-scramble shuffles according to the circuit. This ensures that the input values are uniformly random after the garbling phase without changing the semantics of the circuit.

Our scheme requires a total of $n + g - m$ pile-scramble shuffles. Each pile-scramble shuffle is defined for each $i \in \mathsf{Wires} \setminus \mathsf{Outputs}$, and runs consecutively in this index order. Unlike the single-gate circuit, a pile-scramble shuffle requires many different positions to be shuffled simultaneously according to the circuit topology. To specify the positions to be exchanged in each pile-scramble shuffle, we define the ordered subsets $\boldsymbol{P}^{(i,1)}, \boldsymbol{P}^{(i,2)} \subseteq \{1, \ldots, 2n + 8g\}$ for $i \in \mathsf{Wires} \setminus \mathsf{Outputs}$ as follows:

$$\boldsymbol{P}^{(i,1)} = \boldsymbol{I}^{(i,1)} \parallel (\|_{j \in A_i^{-1}} \boldsymbol{L}^{(j,1)}) \parallel (\|_{j \in B_i^{-1}} \boldsymbol{R}^{(j,1)})$$
$$\boldsymbol{P}^{(i,2)} = \boldsymbol{I}^{(i,2)} \parallel (\|_{j \in A_i^{-1}} \boldsymbol{L}^{(j,2)}) \parallel (\|_{j \in B_i^{-1}} \boldsymbol{R}^{(j,2)})$$

where $\boldsymbol{I}^{(i,1)}$, $\boldsymbol{I}^{(i,2)}$, $\boldsymbol{L}^{(j,1)}, \boldsymbol{L}^{(j,2)}, \boldsymbol{R}^{(j,1)}, \boldsymbol{R}^{(j,2)}$ are given as:

$$\boldsymbol{I}^{(i,1)} = \begin{cases} \{\mathsf{a}(i) + 0\} & i \in \mathsf{Inputs} \\ \{\mathsf{a}(i) + 0, \mathsf{a}(i) + 2, \mathsf{a}(i) + 4, \mathsf{a}(i) + 6\} & i \in \mathsf{Gates} \end{cases}$$
$$\boldsymbol{I}^{(i,2)} = \begin{cases} \{\mathsf{a}(i) + 1\} & i \in \mathsf{Inputs} \\ \{\mathsf{a}(i) + 1, \mathsf{a}(i) + 3, \mathsf{a}(i) + 5, \mathsf{a}(i) + 7\} & i \in \mathsf{Gates} \end{cases}$$
$$\boldsymbol{L}^{(j,1)} = \{\mathsf{a}(j) + 0, \mathsf{a}(j) + 1, \mathsf{a}(j) + 2, \mathsf{a}(j) + 3\}$$

$$\boldsymbol{L}^{(j,2)} = \{\mathsf{a}(j)+4, \mathsf{a}(j)+5, \mathsf{a}(j)+6, \mathsf{a}(j)+7\}$$
$$\boldsymbol{R}^{(j,1)} = \{\mathsf{a}(j)+0, \mathsf{a}(j)+1, \mathsf{a}(j)+4, \mathsf{a}(j)+5\}$$
$$\boldsymbol{R}^{(j,2)} = \{\mathsf{a}(j)+2, \mathsf{a}(j)+3, \mathsf{a}(j)+6, \mathsf{a}(j)+7\}$$

The pile-scramble shuffle determined by $\boldsymbol{P}^{(i,1)}$ and $\boldsymbol{P}^{(i,2)}$ results in an XOR shuffling, as discussed in Section 3.1. Here we point out the difference between this definition and the single gate case. In a general circuit, there can be non-input wires and branching wires. When $j \notin \mathsf{Inputs}$, the players must randomize all 4 possible commitments for input to $j$, defined as $\boldsymbol{I}^{(i,1)}$ and $\boldsymbol{I}^{(i,2)}$. When $j$ is a branching wire, *i.e.,* when $A_j^{-1} \cup B_j^{-1}$ contains two or more wires, the players must randomize all the truth tables on the output side of wire $j$, defined as $\boldsymbol{L}^{(j,1)}$, $\boldsymbol{L}^{(j,2)}$, $\boldsymbol{R}^{(j,1)}$ and $\boldsymbol{R}^{(j,2)}$.

**Protocol 2.** CardGb

**Input:** $I$, where $I$ is an initial state.
**Output:** $(F, X)$, where $F$ and $X$ are decks with $8g$ and $2n$ face-down cards, respectively.

```
1. For $i \in \mathsf{Wires} \setminus \mathsf{Outputs}$ do:
   (a) Compute ($\mathsf{PileShuffle}$, $\boldsymbol{P}^{(i,1)}$, $\boldsymbol{P}^{(i,2)}$).
2. Parse the resulting deck as $X \parallel F$ and output it.
```

The garbling phase randomizes the circuit semantics as follows. In $X$ and $F$, the Boolean value assigned to each non-output wire is randomized by a single pile-scramble shuffle. Note that each pile-scramble shuffle can be viewed as a group action by a random element in $S_2 \cong \mathbb{Z}_2$. Let $r_i$ denote a random element in $\mathbb{Z}_2$ resulting from the $i$-th pile-scramble shuffle, except that we set $r_i = 0$ if $i \in \mathsf{Outputs}$. For each $i \in \mathsf{Inputs}$, the input value $x_i$ is randomized to $x_i \oplus r_i$ and stored in $X$. For each $j \in \mathsf{Gates}$, the four outcome values are randomized and rearranged to be $G_j(r_{A_j}, r_{B_j}) \oplus r_j$, $G_j(r_{A_j}, \overline{r_{B_j}}) \oplus r_j$, $G_j(\overline{r_{A_j}}, r_{B_j}) \oplus r_j$, and $G_j(\overline{r_{A_j}}, \overline{r_{B_j}}) \oplus r_j$ in order.

Furthermore, we can make a more critical observation that the resulting circuit semantics is determined independently of the order of the $n+g-m$ pile-scramble shuffles. Due to the successive pile-scramble shuffles, the truth table of each gate $j$ is randomized by three kinds of group actions, defined as $\boldsymbol{I}$, $\boldsymbol{L}$, and $\boldsymbol{R}$, by the distinct random elements $r_j$, $r_{A_j}$, and $r_{B_j}$. The key fact here is that, by definition, these group actions are all commutative. This is because they form the group $S_2 \times S_2 \times S_2$ and its natural action on 8 points. Accordingly, the players can apply the pile-scramble shuffles in any order.

From the above observation, we can define a variant of the CardGb protocol that only requires a single shuffle. Let $G_i$ be the permutation group determined by $\boldsymbol{P}^{(i,1)}$ and $\boldsymbol{P}^{(i,2)}$, and let $G_f := \{g_1 g_2 \cdots g_{n+g-m} \in S_{2n+8g} \mid g_j \in G_j\}$. Note that $G_f$ is also a permutation group, since any two of $G_j$ are commutative. Thus, we can combine the $n+g-m$ pile-scramble shuffles into one shuffle as follows.

**Protocol 3.** CardGb (with a single shuffle)

**Input:** $I$, where $I$ is an initial state.
**Output:** $(F,\ X)$, where $F$ and $X$ are decks with $8g$ and $2n$ face-down cards, respectively.

1. Compute $(\mathsf{Shuffle},\ G_f)$.
2. Parse the resulting deck as $X \parallel F$ and output it.

**Example.** Consider the garbling phase of the example given by Fig. 2. In this case, $n + g - m = 5$, so the garbling phase contains 5 consecutive pile-scramble shuffles. Since wire 1 is the first input of gate 4, the first pile-scramble shuffle is described as $(\mathsf{PileShuffle}, \{1, 7, 8, 9, 10\}, \{2, 11, 12, 13, 14\})$:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
| ○ | ● | | | | | ○ | ○ | ○ | ○ | ● | ● | ● | ● | | | | | | | | | | | | | | | | |

Similarly, wires 2 and 3 respectively determine the second pile-scramble shuffle $(\mathsf{PileShuffle}, \{3, 7, 8, 11, 12\}, \{4, 9, 10, 13, 14\})$ and the third pile-scramble shuffle $(\mathsf{PileShuffle}, \{5, 15, 16, 17, 18\}, \{6, 19, 20, 21, 22\})$:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
| | | ○ | ● | | | ○ | ○ | ● | ● | ○ | ○ | ● | ● | | | | | | | | | | | | | | | | |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
| | | | | ○ | ● | | | | | | | | | ○ | ○ | ○ | ○ | ● | ● | ● | ● | | | | | | | | |

Next, consider the pile-scramble shuffle given by wire 4. This wire comes out of gate 4 and goes into the second input of gate 5 and the first input of gate 6. Accordingly, all the cards corresponding to gates 4, 5, and 6 are shuffled in a way that preserves the circuit semantics. Such a shuffle is given as the pile-scramble shuffle $(\mathsf{PileShuffle}, \boldsymbol{P}^{(4,1)}, \boldsymbol{P}^{(4,2)})$, where $\boldsymbol{P}^{(4,1)} = \boldsymbol{I}^{(4,1)} \parallel \boldsymbol{R}^{(5,1)} \parallel \boldsymbol{L}^{(6,1)}$ and $\boldsymbol{P}^{(4,2)} = \boldsymbol{I}^{(4,2)} \parallel \boldsymbol{R}^{(5,2)} \parallel \boldsymbol{L}^{(6,2)}$:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
| | | | | | | ○ | ● | ○ | ● | ○ | ● | ○ | ● | ○ | ○ | ● | ● | ○ | ○ | ● | ● | ○ | ○ | ○ | ○ | ● | ● | ● | ● |

Likewise, the fifth pile-scramble shuffle is defined as $(\mathsf{PileShuffle}, \boldsymbol{P}^{(5,1)}, \boldsymbol{P}^{(5,2)})$, where $\boldsymbol{P}^{(5,1)} = \boldsymbol{I}^{(5,1)} \parallel \boldsymbol{R}^{(6,1)}$ and $\boldsymbol{P}^{(5,2)} = \boldsymbol{I}^{(5,2)} \parallel \boldsymbol{R}^{(6,2)}$.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
| | | | | | | | | | | | | | | ○ | ● | ○ | ● | ○ | ● | ○ | ● | ○ | ○ | ● | ● | ○ | ○ | ● | ● |

Finally, let us review the circuit semantics after successive pile-scramble shuffles. Let $r_i$ denote the bit corresponding to the $i$-th pile-scramble shuffle. Then the following holds:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

$$\underbrace{\boxed{?}\boxed{?}}_{x_1\oplus r_1}\underbrace{\boxed{?}\boxed{?}}_{x_2\oplus r_2}\underbrace{\boxed{?}\boxed{?}}_{x_3\oplus r_3}\underbrace{\boxed{?}\boxed{?}}_{g_0^4\oplus r_4}\underbrace{\boxed{?}\boxed{?}}_{g_1^4\oplus r_4}\underbrace{\boxed{?}\boxed{?}}_{g_2^4\oplus r_4}\underbrace{\boxed{?}\boxed{?}}_{g_3^4\oplus r_4}\underbrace{\boxed{?}\boxed{?}}_{g_0^5\oplus r_5}\underbrace{\boxed{?}\boxed{?}}_{g_1^5\oplus r_5}\underbrace{\boxed{?}\boxed{?}}_{g_2^5\oplus r_5}\underbrace{\boxed{?}\boxed{?}}_{g_3^5\oplus r_5}\underbrace{\boxed{?}\boxed{?}}_{r_4\vee r_5}\underbrace{\boxed{?}\boxed{?}}_{r_4\vee\overline{r_5}}\underbrace{\boxed{?}\boxed{?}}_{\overline{r_4}\vee r_5}\underbrace{\boxed{?}\boxed{?}}_{\overline{r_4}\vee\overline{r_5}}$$

where

$$\begin{array}{llll} g_0^4 = r_1 \wedge r_2, & g_1^4 = r_1 \wedge \overline{r_2}, & g_2^4 = \overline{r_1} \wedge r_2, & g_3^4 = \overline{r_1} \wedge \overline{r_2}, \\ g_0^5 = r_3 \oplus r_4, & g_1^5 = r_3 \oplus \overline{r_4}, & g_2^5 = \overline{r_3} \oplus r_4, & g_3^5 = \overline{r_3} \oplus \overline{r_4}. \end{array}$$

## 3.4 Evaluation Phase

The evaluation phase is an iterative process to obtain a commitment of the desired Boolean value as output. At each step, the players open two designated cards. The first step starts with opening the cards corresponding to the input wires. Then, if a gate takes the values at the two input wires, the players refer to the values to determine the following card positions and open only two out of the eight cards corresponding to the gate.

Let us consider the correctness of our scheme. To prove this, we show that the circuit randomized by the garbling phase provides the same output as the plain-text circuit evaluation when computed through the evaluation phase of our scheme. Let $v_j$ be the output value of gate $j$ in the circuit $f$ when $x$ is input. Then, it suffices to show that the evaluation phase of our scheme gives the output value of gate $j$ as $v_j \oplus r_j$, where $r_j$ is the same as introduced in Section 3.3.

The proof is shown by induction on the circuit structure. The base case follows from the fact that each $i \in \mathsf{Inputs}$ is assigned $x_i \oplus r_i$ in $X$. Assume that the statement holds up to $j-1$. Then $\mathsf{Out}(A_j) = v_{A_j} \oplus r_{A_j}$ and $\mathsf{Out}(B_j) = v_{B_j} \oplus r_{B_j}$. On the other hand, by the definition of $\mathsf{CardGb}$, the four secret values represented by the eight cards starting from $\mathsf{a}(j)$ in $F$ is $G_j(r_{A_j}, r_{B_j}) \oplus r_j$, $G_j(r_{A_j}, \overline{r_{B_j}}) \oplus r_j$, $G_j(\overline{r_{A_j}}, r_{B_j}) \oplus r_j$, and $G_j(\overline{r_{A_j}}, \overline{r_{B_j}}) \oplus r_j$. Hence, according to the definition of the evaluation phase, the players specify the two cards representing $G_j(v_{A_i}, v_{B_j}) \oplus r_j$, which is the desired conclusion.

Remark that the evaluation phase leaks no information about the initial state $I$. During the evaluation phase, $2(n+g-m)$ cards become open in total, so the players know $n+g-m$ Boolean values. However, due to $n+g-m$ pile-scramble shuffles in the garbling phase, there is enough randomness in $F$ and $X$ to hide the secret values. This property can be viewed as the counterpart of the obliviousness property in garbled schemes.

**Example.** Here we demonstrate the evaluation phase for the example given by the circuit in Fig. 2. First, the players open all cards in $X$. Suppose the result is:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

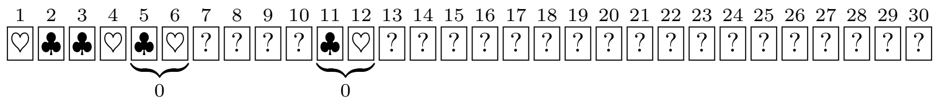
$$\underbrace{\boxed{\heartsuit}\boxed{\clubsuit}}_{1}\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{0}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}$$

**Protocol 4.** CardEval

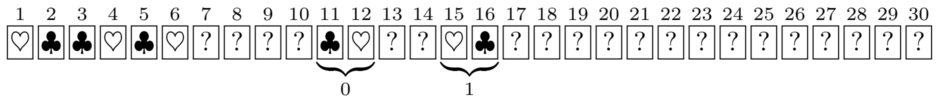
```
Input: $(F, X)$, where $F$ and $X$ are decks consisting of $8g$ and $2n$ face-down cards, respectively.
Output: $Y$, where $Y$ is a deck consisting of $2m$ face-down cards.

1. Set $D = X \parallel F$.
2. Compute $(\mathsf{Open}, \{1, \dots, 2n\})$.
3. Set $\mathsf{Out}(i)$ to the Boolean value represented by $(2i-1)$-th and $2i$-th cards for all $i \in \mathsf{Inputs}$.
4. For $i \in \mathsf{Gates} \setminus \mathsf{Outputs}$ do:
    – If $\mathsf{Out}(A_i) = 0$ and $\mathsf{Out}(B_i) = 0$, compute $(\mathsf{Open}, \{\mathsf{a}(i), \mathsf{a}(i)+1\})$ and set $\mathsf{Out}(i)$ to the value.
    – If $\mathsf{Out}(A_i) = 0$ and $\mathsf{Out}(B_i) = 1$, compute $(\mathsf{Open}, \{\mathsf{a}(i)+2, \mathsf{a}(i)+3\})$ and set $\mathsf{Out}(i)$ to the value.
    – If $\mathsf{Out}(A_i) = 1$ and $\mathsf{Out}(B_i) = 0$, compute $(\mathsf{Open}, \{\mathsf{a}(i)+4, \mathsf{a}(i)+5\})$ and set $\mathsf{Out}(i)$ to the value.
    – If $\mathsf{Out}(A_i) = 1$ and $\mathsf{Out}(B_i) = 1$, compute $(\mathsf{Open}, \{\mathsf{a}(i)+6, \mathsf{a}(i)+7\})$ and set $\mathsf{Out}(i)$ to the value.
5. Set $Y$ to an empty deck.
6. For $i \in \mathsf{Outputs}$ do:
    – If $\mathsf{Out}(A_i) = 0$ and $\mathsf{Out}(B_i) = 0$, append the face-down cards at $\{\mathsf{a}(i), \mathsf{a}(i)+1\}$ to $Y$.
    – If $\mathsf{Out}(A_i) = 0$ and $\mathsf{Out}(B_i) = 1$, append the face-down cards at $\{\mathsf{a}(i)+2, \mathsf{a}(i)+3\}$ to $Y$.
    – If $\mathsf{Out}(A_i) = 1$ and $\mathsf{Out}(B_i) = 0$, append the face-down cards at $\{\mathsf{a}(i)+4, \mathsf{a}(i)+5\}$ to $Y$.
    – If $\mathsf{Out}(A_i) = 1$ and $\mathsf{Out}(B_i) = 1$, append the face-down cards at $\{\mathsf{a}(i)+6, \mathsf{a}(i)+7\}$ to $Y$.
7. Output $Y$.
```

The next step is evaluating gate 4 with inputs 1 and 0. The evaluation protocol picks the third committed value from $\mathsf{a}(4)$ and opens it.



Similarly, gate 5 is evaluated with inputs 0 and 0, opening the first committed value from $\mathsf{a}(5)$.



In the final step of evaluating gate 6 with inputs 0 and 1, the players choose the second committed value from $\mathsf{a}(6)$ and output it as $Y := \boxed{?}^{25}\,\boxed{?}^{26}$.

Let us check the correctness of this case. As discussed in Section 3.3, the garbling phase randomizes the circuit semantics using five random Boolean values $r_i$. The values revealed during the evaluation phase satisfy the following:

$$x_1 \oplus r_1 = 1, \quad x_2 \oplus r_2 = 0, \quad x_3 \oplus r_3 = 0, \quad (\overline{r_1} \wedge r_2) \oplus r_4 = 0, \quad (r_3 \oplus r_4) \oplus r_5 = 1.$$

In addition, $Y$ is a commitment of $r_4 \vee \overline{r_5}$. Accordingly, we have the following:

$$r_4 \vee \overline{r_5} = r_4 \vee (r_3 \oplus r_4) = (\overline{r_1} \wedge r_2) \vee (r_3 \oplus (\overline{r_1} \wedge r_2)) = (x_1 \wedge x_2) \vee (x_3 \oplus (x_1 \wedge x_2)),$$

which is the desired output.

## 3.5 Card-based Protocols for Standard Garbling Scheme

This section introduces card-based protocols for computing the functionalities in Section 2.3 to investigate the relation to the standard garbling schemes. By making some modifications to the protocols in the previous sections, we can define a card-based garbled circuit scheme suitable for the original garbling scheme definition. In this case, our scheme requires $4n + 8g + 2m$ cards in total and one shuffle each in $\mathsf{Gb}$, $\mathsf{Enc}$ and $\mathsf{Dec}$, respectively. On the other hand, this scheme allows us to split the computational procedure into two parts that depend only on $f$ or $x$, respectively. Hereafter, let $\mathsf{XOR}$ denote the functionality for computing the element-wise XOR of two decks in a committed form. Using the protocol proposed in [18], $\mathsf{XOR}$ can be computed with only one shuffle.

First, we define a protocol for the garbling phase $\mathsf{Gb}$. The initial state consists of $4n+8g+2m$ face-down cards and is defined as the output of $\mathsf{Init}(\mathbf{0}, f)$ followed by $m$ commitments of 0, where $\mathbf{0}$ denotes the $2n$ cards consisting of $n$ commitments of 0. The protocol computes $\mathsf{CardGb}$ on the first $2n + 8g$ cards. Then, executes additional $m$ pile-scramble shuffles for each $i \in \mathsf{Outputs}$ as follows.

$$(\mathsf{PileShuffle}, \boldsymbol{I}^{(i,1)} \parallel \{6g + 2m + 2i - 1\}, \boldsymbol{I}^{(i,2)} \parallel \{6g + 2m + 2i\})$$

Finally, the protocol outputs the first $2n$ cards as $e$, the next $8g$ cards as $F$, and the last $2m$ cards as $f$. Note that the additional $m$ pile-scramble shuffles are commutative with each other and also with the other $n + g - m$ shuffles, so all the required pile-scramble shuffles can be combined into one shuffle.

The rest part is straightforward. We can define a protocol for the encoding phase $\mathsf{Enc}(e, x)$ as simply computing $\mathsf{XOR}$ of $e$ and the commitment of $x$. Similarly, our protocol for the decoding phase $\mathsf{Dec}(d, Y)$ simply computes $\mathsf{XOR}$ of $d$ and $Y$, opens the results and outputs the Boolean values. The evaluation phase $\mathsf{Eval}$ is the same as $\mathsf{CardEval}$.

It is easy to verify the correctness and obliviousness of this scheme. The proof is similar to the case of the $\mathsf{CardGb}$ and $\mathsf{CardEval}$ protocols. In addition, this protocol provides the counterpart of the privacy property in garbling schemes. This follows from the fact that the obliviousness proof in Section 3.4 leaks no information on $x$ in an information-theoretic sense.

## 4 Conclusion

Shinagawa and Nuida [24] showed a surprising result that any Boolean function can be securely computed using only one shuffle by combining card-based cryptography with Yao's garbled circuit technique. Their protocol requires $2n + 24g$ cards, where $g$ is the number of gates and $n$ is the number of function inputs. This paper improved upon this existing approach by introducing an XOR shuffle technique that helps to reduce the number of required cards. Consequently, we showed that, instead of having $2n + 24g$ cards, only $2n + 8g$ cards are sufficient for constructing a single-shuffle protocol.

**Acknowledgements** This work was supported by JSPS KAKENHI Grant Numbers JP21H05052, JP21K11881, and JST, CREST Grant Number JPMJCR22M1, Japan.

## References

1. Attrapadung, N., Hanaoka, G., Matsuda, T., Morita, H., Ohara, K., Schuldt, J.C.N., Teruya, T., Tozawa, K.: Oblivious linear group actions and applications. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 630–650. ACM Press (2021). `https://doi.org/10.1145/3460120.3484584`
2. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC. pp. 503–513. ACM Press (1990). `https://doi.org/10.1145/100216.100287`
3. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 784–796. ACM Press (2012). `https://doi.org/10.1145/2382196.2382279`
4. Den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT '89. LNCS, vol. 434, pp. 208–217. Springer, Berlin, Heidelberg (1990), `https://doi.org/10.1007/3-540-46885-4_23`
5. Haga, R., Hayashi, Y., Miyahara, D., Mizuki, T.: Card-minimal protocols for three-input functions with standard playing cards. In: AFRICACRYPT 2022. LNCS, vol. 13503, pp. 448–468. Springer, Cham (2022)
6. Heather, J., Schneider, S., Teague, V.: Cryptographic protocols with everyday objects. Formal Aspects of Computing **26**(1), 37–62 (2014), `https://doi.org/10.1007/s00165-013-0274-7`
7. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Calude, C.S., Dinneen, M.J. (eds.) Unconventional Computation and Natural Computation. LNCS, vol. 9252, pp. 215–226. Springer, Cham (2015). `https://doi.org/10.1007/978-3-319-21819-9\_16`
8. Isuzugawa, R., Miyahara, D., Mizuki, T.: Zero-knowledge proof protocol for Cryptarithmetic using dihedral cards. In: Kostitsyna, I., Orponen, P. (eds.) Unconventional Computation and Natural Computation. LNCS, vol. 12984, pp. 51–67. Springer, Cham (2021), `https://doi.org/10.1007/978-3-030-87993-8_4`

9. 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.) ASIACRYPT 2017. LNCS, vol. 10626, pp. 126–155. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70700-6_5
10. 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
11. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer, Berlin, Heidelberg (2015), https://doi.org/10.1007/978-3-662-48797-6_32
12. Koyama, H., Miyahara, D., Mizuki, T., Sone, H.: A secure three-input AND protocol with a standard deck of minimal cards. In: Santhanam, R., Musatov, D. (eds.) 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
13. Kuzuma, T., Toyoda, K., Miyahara, D., Mizuki, T.: Card-based single-shuffle protocols for secure multiple-input AND and XOR computations. In: ASIA Public-Key Cryptography. pp. 51–58. ACM, NY (2022), https://doi.org/10.1145/3494105.3526236
14. Miyahara, D., Ueda, I., Hayashi, Y.i., Mizuki, T., Sone, H.: Analyzing execution time of card-based protocols. In: Stepney, S., Verlan, S. (eds.) Unconventional Computation and Natural Computation. LNCS, vol. 10867, pp. 145–158. Springer, Cham (2018), https://doi.org/10.1007/978-3-319-92435-9_11
15. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Mauri, G., Dennunzio, A., Manzoni, L., Porreca, A.E. (eds.) Unconventional Computation and Natural Computation. LNCS, vol. 7956, pp. 162–173. Springer, Berlin, Heidelberg (2013), https://doi.org/10.1007/978-3-642-39074-6_16
16. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 598–606. Springer, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-34961-4_36
17. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security **13**(1), 15–23 (2013). https://doi.org/10.1007/s10207-013-0219-4
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. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any Boolean function. In: Jain, R., Jain, S., Stephan, F. (eds.) Theory and Applications of Models of Computation. LNCS, vol. 9076, pp. 110–121. Springer, Cham (2015), https://doi.org/10.1007/978-3-319-17142-5_11
20. Ruangwises, S., Itoh, T.: Physical ZKP for connected spanning subgraph: Applications to Bridges puzzle and other problems. In: Kostitsyna, I., Orponen, P. (eds.) Unconventional Computation and Natural Computation. pp. 149–163. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87993-8_10
21. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. Theor. Comput. Sci. **887**, 99–110 (2021), https://doi.org/10.1016/j.tcs.2021.07.007

22. Shikata, H., Toyoda, K., Miyahara, D., Mizuki, T.: Card-minimal protocols for symmetric boolean functions of more than seven inputs. In: Theoretical Aspects of Computing – ICTAC 2022. LNCS, vol. 13572, pp. 388–406. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17715-6_25
23. Shinagawa, K., Mizuki, T.: The six-card trick: Secure computation of three-input equality. In: Lee, K. (ed.) Information Security and Cryptology. LNCS, vol. 11396, pp. 123–131. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-12146-4_8
24. Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any boolean circuit. Discrete Applied Mathematics **289**, 248–261 (2021). https://doi.org/10.1016/j.dam.2020.10.013
25. Toyoda, K., Miyahara, D., Mizuki, T.: Another use of the five-card trick: Card-minimal secure three-input majority function evaluation. In: Adhikari, A., Küsters, R., Preneel, B. (eds.) INDOCRYPT 2021. LNCS, vol. 13143, pp. 536–555. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-92518-5_24