

Single-Shuffle Card-Based Protocols with Six Cards per Gate

Tomoki Ono¹, Kazumasa Shinagawa^{2,3}, Takeshi Nakai⁴, Yohei Watanabe^{1,3},
and Mitsugu Iwamoto¹

¹ The University of Electro-Communications, Tokyo, Japan
{onotom, watanabe, mitsugu}@uec.ac.jp

² Ibaraki University, Ibaraki, Japan

kazumasa.shinagawa.np92@vc.ibaraki.ac.jp

³ National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

⁴ Toyohashi University of Technology, Aichi, Japan
nakai@cs.tut.ac.jp

Abstract. Card-based cryptography refers to a secure computation with physical cards, and the number of cards and shuffles measures the efficiency of card-based protocols. This paper proposes new card-based protocols for any Boolean circuits with only a single shuffle. Although our protocols rely on Yao’s garbled circuit as in previous single-shuffle card-based protocols, our core construction idea is to encode truth tables of each Boolean gate with fewer cards than previous works while being compatible with Yao’s garbled circuit. As a result, we show single-shuffle card-based protocols with six cards per gate, which are more efficient than previous single-shuffle card-based protocols.

Keywords: Card-based cryptography · Secure computation · Garbled circuit.

1 Introduction

1.1 Background and Motivation

Secure computation protocols allow parties to collaboratively compute a function while keeping each party’s input hidden from the other party. Although secure computation protocols are usually implemented on computers, *card-based cryptography* [3, 4], which is an area focusing on secure computation using physical cards (without computers), has also been eagerly investigated. Let us give an example of a secure card-based AND protocol called the *five-card trick* [3]. Suppose that each of Alice and Bob has two cards, \clubsuit and \heartsuit , and a \heartsuit is placed face-down on a table. Alice (resp., Bob) puts their cards face-down on the left side (resp., the right side) of \heartsuit , following the encoding rule: the order of the cards is $\clubsuit\heartsuit$ if the input is zero; it is $\heartsuit\clubsuit$ if the input is one. After shuffling the five face-down cards without changing the order of the sequence, they face up the cards. The output of the AND protocol is one if the consecutive three heart cards appear; it is zero otherwise.

The major efficiency measures of card-based cryptography are the number of cards and shuffles. The fewer cards and shuffles card-based protocols are realized, the easier it is to execute them. In this work, we focus on the implementability of shuffles; it is unclear how to implement shuffles that yield desired probability distributions, though various attempts have been made thus far [7, 11, 17, 18, 22, 23, 28, 29]. For this reason, we devote effort to constructing card-based protocols with the minimum number of shuffles and as few cards as possible.

A well-known approach to constructing card-based protocols for any function is to realize card-based protocols for a Boolean gate such as AND and XOR since any function can be realized by combining Boolean gates [4, 12, 15]. Hence, improving card-based protocols for Boolean gates is one of the mainstream research topics [1, 2, 4–6, 8, 9, 12–14, 19–21, 25, 26]. However, this approach increases the number of shuffles required for the resulting card-based protocols for any function (or Boolean circuit) since the number of shuffles depends on the number of gates consisting of the Boolean circuit. Therefore, we aim to directly propose card-based protocols for any Boolean circuit consisting of various Boolean gates, not any Boolean gate, with a single shuffle. Note that, as stated in [24], it is impossible to realize card-based protocol for any non-trivial function without shuffles; The lower bound of shuffles required for secure card-based protocols is one.

1.2 Prior Works

Shinagawa and Nuida [24] showed a single-shuffle card-based protocol for any n -variable Boolean circuit $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ with $24q + 2n$ cards, where q is number of gates in the Boolean circuit. Tozawa et al. [27] improved the Shinagawa–Nuida protocol and reduced the number of cards to $8q + 2n$ without additional shuffles. Kuzuma et al. [10] focused on a restricted class of Boolean circuits and showed single-shuffle card-based protocols for an n -variable AND function with $4n - 2$ cards and an n -variable XOR function with $2n$ cards. Note that, allowing multiple shuffles, Nishida et al. [16] showed a card-based protocol with $2n + 6$ cards for any n -variable Boolean circuit $f : \{0, 1\}^n \rightarrow \{0, 1\}$, which is the most efficient protocol in terms of the number of cards.

1.3 Our Contribution

This paper proposes new single-shuffle card-based protocols based on Yao’s garbled circuits [30]. The core construction idea is to encode truth tables of each Boolean gate with fewer cards than previous protocols while being compatible with Yao’s garbled circuit. Unlike previous single-shuffle card-based protocols such as Shinagawa–Nuida [24] and Tozawa et al. [27], each output of the truth table is represented by a single card, and we add two more extra cards to make the truth tables encoded with single cards compatible with Yao’s technique. As a result, we show two single-shuffle card-based protocols for any Boolean circuit $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ with six cards per gate: One is a non-committed-format protocol with $2n + 6q$ cards, and the other is a committed-format protocol with

Table 1. A comparison among protocols with one shuffle for any Boolean circuit. q , n , and m are the number of gates, bit-length of the input, and bits-length of the output, respectively.

	Format	Number of cards	Shuffle type
Shinagawa–Nuida [24]	committed	$24q + 2n$	uniform closed
Tozawa et al. [27]	committed	$8q + 2n$	uniform closed
Our protocol (Section 3.6)	non-committed	$6q + 2n$	uniform
Our protocol (Section 3.7)	committed	$6q + 2(n + m)$	uniform

$2(n + m) + 6q$ cards, where q is the number of gates in f , and a protocol is said to be *committed* if it outputs cards face-down and the output follows the same encoding rule as the input.

Table 1 shows a comparison among the existing protocols and our protocols. Since the number of gates q is greater than or equal to the number of the output gates m , our protocol is more efficient than those of Shinagawa–Nuida [24] and Tozawa et al. [27] in terms of the number of cards. It should be noted that our protocols use a uniform shuffle, which is not closed (see Section 2.2 for the definition), although Shinagawa–Nuida [24] and Tozawa et al. [27] used a uniform closed shuffle.

1.4 Organization

In Section 2, we introduce basic definitions. In Section 3, we construct our single-shuffle protocols both in the non-committed-format setting and the committed-format setting. In Section 4, we conclude our paper.

2 Preliminaries

For an integer $k \geq 1$, we denote the k -th symmetric group by S_k . For two permutations $\pi_1, \pi_2 \in S_k$, the composition of them is denoted by $\pi_2 \circ \pi_1$. Here, permutations are applied from right to left, i.e., $\pi_2 \circ \pi_1 \in S_k$ means that permutation π_1 is applied and then π_2 is applied. For two subsets $A, B \subseteq S_k$, we define $AB := \{\pi_A \circ \pi_B \mid \pi_A \in A, \pi_B \in B\}$.

2.1 Syntax of Boolean Circuits

A Boolean circuit C is defined by a 6-tuple (n, m, q, L, R, G) where $n \geq 1$ is the number of input wires, $m \geq 1$ is the number of output wires, $q \geq 1$ is the number of gates, L and R are functions that specify the left and right wires in each gate, respectively, and G is a function that specifies the truth table of each gate. The detailed specification is given in the following.

- The number of wires in C is $n + q$, where n wires are the input wires and q wires are the output wires of gates. Each input wire corresponds to




- 1, 2, \dots , n , and each output wire of gates corresponds to $n+1, n+2, \dots, n+q$. The last m wires $n+q-m+1, n+q-m+2, \dots, n+q$ correspond to the output wires of C . A gate g is identified with the output wire of g , i.e., each gate also corresponds to $n+1, n+2, \dots, n+q$.
- Each gate g has two input wires: the left input wire of g is $L(g)$ and the right input wire of g is $R(g)$. We assume that $L(g) \leq R(g) < g$, i.e., the input wires $L(g), R(g)$ are smaller than g , and the left wire is smaller than or equal to the right wire. This restriction prevents the loop of the circuit.
 - A wire w , which is not an output wire of C is called the inner wire, i.e., each inner wire corresponds to $1, 2, \dots, n+q-m$. An inner wire is an input wire or an input wire of some gate. An inner wire w can be branched, i.e., there might exist two or more gates having w as its input wire, or some gate can be taken w as the left and right input wires.
 - For an inner wire w , $L^{-1}(w)$ is defined by the set of all gates whose left input wire is w , i.e., $L^{-1}(w) = \{g \in \{n+1, n+2, \dots, n+q\} \mid L(g) = w\}$. We define $R^{-1}(w)$ in the same way.
 - For a gate g , $G(g)$ represent the truth table of g . When g computes a function $f : \{0, 1\}^2 \rightarrow \{0, 1\}$, $G(g)$ represents a 4-bit binary string defined by

$$G(g) = (f(0, 0), f(0, 1), f(1, 0), f(1, 1)).$$

In this paper, for simplicity, we assume that all gates are the NAND gates, i.e., $G(g) = (1, 1, 1, 0)$ for all gates g . This is based on the fact that any Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ can be constructed by only NAND gates. We note that our protocol can also be applied to a circuit with other gates.

Example of Boolean Circuit. A Boolean circuit $C = (3, 1, 3, L, R, G)$ is given in Figure 1, where the number of the input wires is $n = 3$, the number of the output wires is $m = 1$, and the number of the gates is $q = 3$. Each input wire corresponds to 1, 2, 3 and each gate corresponds to 4, 5, 6. The functions L and R are defined by $L(4) = 1, R(4) = 2, L(5) = 3, R(5) = 4, L(6) = 4$ and $R(6) = 5$. Then we have $L^{-1}(1) = \{4\}, R^{-1}(1) = \emptyset, L^{-1}(2) = \emptyset, R^{-1}(2) = \{4\}, L^{-1}(3) = \{5\}, R^{-1}(3) = \emptyset, L^{-1}(4) = \{6\}, R^{-1}(4) = \{5\}, L^{-1}(5) = \emptyset$ and $R^{-1}(5) = \{6\}$. Since all gates are the NAND gates, $G(g) = (1, 1, 1, 0)$ for all $4 \leq g \leq 6$.

2.2 Card-based Protocols

In this paper, we use two-colored cards: the front side of a card is either  or , and the back side is . All cards with the same suit are indistinguishable, and the backs of all cards are also indistinguishable.

In card-based protocols, three operations are used: permutation, shuffle, and turn. Let k be the number of cards. A permutation operation (**perm**, π) for $\pi \in S_k$ is a deterministic operation that rearranges the order of the cards according to π . A shuffle operation (**shuffle**, Π, \mathcal{F}) for a subset $\Pi \subseteq S_k$ and a probability distribution \mathcal{F} over Π is a probabilistic operation that randomly rearranges the order of the cards according to a permutation $\pi \in \Pi$ drawn from \mathcal{F} . It is assumed

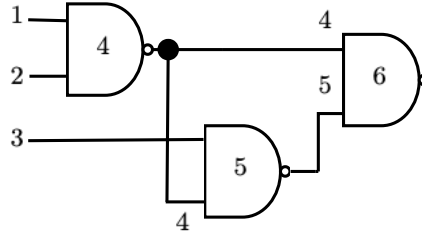


Fig. 1. An example of Boolean circuits

that no player can know which permutation π is actually drawn from \mathcal{F} . A turn operation (turn, T) for $T \subseteq \{1, 2, \dots, k\}$ is a deterministic operation that turns over cards in T from face-down to face-up or from face-up to face-down.

Let S be a shuffle ($\text{shuffle}, \Pi, \mathcal{F}$). If \mathcal{F} is a uniform distribution, S is called a uniform shuffle. If Π is a subgroup of S_k , S is called a closed shuffle. If S is uniform and closed, it is called a uniform closed shuffle.

2.3 Card-based Garbled Circuits

Shinagawa–Nuida [24] developed a card-based garbled circuit, which is a card-based protocol based on garbled circuits. Tozawa et al. [27] improved the card-based garbled circuit in terms of the number of cards. A card-based garbled circuit consists of three phases: initialization phase, garbling phase, and evaluation phase as follows:

Initialization phase: Given a Boolean circuit $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ and a sequence of input commitments to x_1, x_2, \dots, x_n , it outputs a sequence of face-down cards I , which we call an initial state. The objective of this phase is to encode the circuit and its input into a sequence of face-down cards.

Garbling phase: Given an initial state I , it outputs two sequences of face-down cards \tilde{C} and \tilde{X} , which we call a garbled circuit and a garbled input, respectively. The objective of this phase is to randomize the inputs and the intermediate values of the circuit without changing the functionality of the circuit.

Evaluation phase: Given a garbled circuit \tilde{C} and a garbled input \tilde{X} , it outputs the output value or a commitment of the output value. The purpose of this phase is to obtain the output value by evaluating each gate of the garbled circuit \tilde{C} with the garbled input \tilde{X} .

3 Our Single-Shuffle Protocols

3.1 Idea of Our Protocol

In many card-based protocols, 0 and 1 are represented by $\heartsuit\clubsuit$ and $\clubsuit\heartsuit$, respectively, and [27] succeeded in realizing garbled circuits with eight cards that by encoding every 2×2 truth table with eight cards.

Here, we briefly describe our idea for realizing the garbled circuits with *six* cards that consist of three hearts and three clubs. For instance, we represent the truth table of the NAND and AND gates as follows, where 0 and 1 are represented by \clubsuit and \heartsuit , respectively.

NAND	0	1
0	\heartsuit	\heartsuit
1	\heartsuit	\clubsuit

AND	0	1
0	\clubsuit	\clubsuit
1	\clubsuit	\heartsuit

Facing down cards in the above truth table conceals all values in the truth table, but the negation of them is not possible due to the encoding rule with one card that prevents us from converting the NAND gate to the AND gate⁵. To overcome this obstacle, we append two \clubsuit to the NAND truth table as the third column and permute it. Then we have the following and by deleting the third column, we obtain the truth table of AND in a committed format.

\heartsuit	\heartsuit	\clubsuit	permutation →	\clubsuit	\clubsuit	\heartsuit
\heartsuit	\clubsuit	\clubsuit		\clubsuit	\heartsuit	\heartsuit

3.2 Preliminaries for Our Protocol

In our protocol, each input wire is represented by two cards $\clubsuit\heartsuit$ and each gate is represented by six cards $\clubsuit\clubsuit\clubsuit\heartsuit\heartsuit\heartsuit$. Since we have n input wires and q gates, we use $2n + 6q$ cards in total.

To clarify the position of the cards, we define $2n + 6q$ indices: $P_i[a]$ ($1 \leq i \leq n$ and $a \in \{0, 1\}$) and $P_g[b][c]$ ($n + 1 \leq g \leq n + q$, $b \in \{0, 1\}$, and $c \in \{0, 1, 2\}$). Two indices $P_i[0], P_i[1]$ correspond to the input wire i and six indices $P_g[0][0], P_g[1][0], P_g[0][1], P_g[1][1], P_g[0][2], P_g[1][2]$ correspond to the gate g . We assume that all indices are distinct. We give an example of distinct $2n + 6q$ indices in the following:

- $P_i[a] = 2i - 1 + a$ for $1 \leq i \leq n$ and $a \in \{0, 1\}$;
- $P_g[b][c] = 2n + 1 + 6(g - (n + 1)) + 3b + c$ for $n + 1 \leq g \leq n + q$, $b \in \{0, 1\}$, and $c \in \{0, 1, 2\}$.

The above indices are consecutive from 1 to $2n + 6q$.

⁵ The reason utilizing eight cards in [27] comes from this point.

3.3 Initialization Phase

Given a Boolean circuit $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ and a sequence of input commitments to x_1, x_2, \dots, x_n , the initialization phase makes a sequence of face-down cards on the indices of the position $P_i[a]$ and $P_g[b][c]$.

First, the sequence of input commitments is arranged as follows:

$$\underbrace{\begin{matrix} P_1[0] & P_1[1] \\ \boxed{?} & \boxed{?} \end{matrix}}_{x_1} \underbrace{\begin{matrix} P_2[0] & P_2[1] \\ \boxed{?} & \boxed{?} \end{matrix}}_{x_2} \dots \underbrace{\begin{matrix} P_n[0] & P_n[1] \\ \boxed{?} & \boxed{?} \end{matrix}}_{x_n}.$$

For each gate $g \in \{n + 1, n + 2, \dots, n + q\}$, we identify the indices $P_g[a][b]$ ($a \in \{0, 1\}$ and $b \in \{0, 1, 2\}$) with the cells of a 2×3 matrix as follows:

$P_g[0][0]$	$P_g[0][1]$	$P_g[0][2]$
$P_g[1][0]$	$P_g[1][1]$	$P_g[1][2]$

Then, we place six cards $\clubsuit \clubsuit \clubsuit \heartsuit \heartsuit \heartsuit$ as follows:

\heartsuit	\heartsuit	\clubsuit
\heartsuit	\clubsuit	\clubsuit

We note that the above matrix represents the NAND gate: for two inputs $a, b \in \{0, 1\}$, the card on $P_g[a][b]$ is \clubsuit if $a = b = 1$ and \heartsuit otherwise. It can be regarded as the NAND gate by $\clubsuit = 0, \heartsuit = 1$. We also note that two additional \clubsuit s on $P_g[0][2], P_g[1][2]$ are needed to garble the gate as explained in Section 3.1.

Then, we apply a turn operation so that all cards are face-down. Now we have a sequence of $2n + 6q$ face-down cards on the indices of the position $P_i[a]$ and $P_g[b][c]$. This is the output of this phase.

3.4 Garbling Phase

Next, the protocol proceeds to the garbling phase. This phase just applies a uniform shuffle (shuffle, Π, \mathcal{F}) to the sequence of $2n + 6q$ cards outputted by the initialization phase. In the following, we will define $\Pi \subseteq S_{2n+6q}$ by three steps: (1) defining four permutations, (2) defining a shuffle for randomizing a wire, and (3) composing all shuffles.

Defining Four Permutations. For an input wire $i \in \{1, 2, \dots, n\}$, a permutation π_i is defined by

$$\pi_i := (P_i[0], P_i[1]).$$

It represents the bit flip of the i -th input commitment. For a gate $g \in \{n + 1, n + 2, \dots, n + q\}$, a permutation π_g is defined by

$$\pi_g := (P_g[0][0], P_g[0][2]) \circ (P_g[1][0], P_g[1][2]) \circ (P_g[0][1], P_g[1][1]).$$

It represents the bit flip of the truth table of g as follows:

$$\begin{array}{|c|c|c|} \hline \heartsuit & \heartsuit & \clubsuit \\ \hline \heartsuit & \clubsuit & \clubsuit \\ \hline \end{array} \xrightarrow{\pi_g} \begin{array}{|c|c|c|} \hline \clubsuit & \clubsuit & \heartsuit \\ \hline \clubsuit & \heartsuit & \heartsuit \\ \hline \end{array}.$$

For a gate g , a permutation τ_g is defined by

$$\tau_g := (P_g[0][0], P_g[1][0]) \circ (P_g[0][1], P_g[1][1]).$$

It represents a swap of the rows of the truth table of g as follows:

$$\begin{array}{|c|c|c|} \hline 1 & 3 & 5 \\ \hline 2 & 4 & 6 \\ \hline \end{array} \xrightarrow{\tau_g} \begin{array}{|c|c|c|} \hline 2 & 4 & 5 \\ \hline 1 & 3 & 6 \\ \hline \end{array}.$$

A permutation σ_g is defined by

$$\sigma_g := (P_g[0][0], P_g[0][1]) \circ (P_g[1][0], P_g[1][1]).$$

It represents a swap of the columns of the truth table of g as follows:

$$\begin{array}{|c|c|c|} \hline 1 & 3 & 5 \\ \hline 2 & 4 & 6 \\ \hline \end{array} \xrightarrow{\sigma_g} \begin{array}{|c|c|c|} \hline 3 & 1 & 5 \\ \hline 4 & 2 & 6 \\ \hline \end{array}.$$

Defining a Shuffle for Randomizing a Wire. For a wire $w \in \{1, 2, \dots, n + q\}$, a permutation $\hat{\pi}_w$ is defined by

$$\hat{\pi}_w := \pi_w \circ \prod_{g \in L^{-1}(w)} \tau_g \circ \prod_{g' \in R^{-1}(w)} \sigma_{g'}.$$

By applying it, the value of the wire w is flipped and for each gate g , the rows of g are swapped if w is the left input wire of g and the columns of g are swapped if w is the right input wire of g . Define $\Pi_w := \{\text{id}, \hat{\pi}_w\}$. A uniform shuffle (shuffle, Π_w, \mathcal{F}_w) is a shuffle that randomizes the value of the wire w and all gates having w as input.

Composing All Shuffles. The subset $\Pi \subseteq S_{n+q}$ is defined by

$$\Pi := \Pi_1 \Pi_2 \cdots \Pi_{n+q-m} = \{\pi'_1 \circ \pi'_2 \circ \cdots \pi'_{n+q-m} \mid \pi'_i \in \Pi_i\}.$$

The uniform shuffle (shuffle, Π, \mathcal{F}) is now obtained. We note that it is a shuffle by composing $n + q$ uniform shuffles (shuffle, Π_w, \mathcal{F}_w).

3.5 Evaluation Phase

Finally, the protocol proceeds to the evaluation phase. In this phase, the players evaluate the circuit by opening cards as follows. Let v_w ($1 \leq i \leq n + q$) be an indeterminate on $\{0, 1\}$. The protocol proceeds by determining these values and finally outputs $v_{n+q-m+1}, v_{n+q-m+2}, \dots, v_{n+q}$ as the output values.

First, the players open all cards corresponding to the input wires: the value of v_i ($1 \leq i \leq n$) is set to the value of the i -th commitment according to the encoding rule $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$. Next, each gate $g = n + 1, \dots, n + q$ is evaluated (in this order) by opening the card on the position $P_g[v_{L(g)}][v_{R(g)}]$: the value of v_g is set to the value of the card according to the encoding rule $\boxed{\clubsuit} = 0$ and $\boxed{\heartsuit} = 1$. Note that the values of $v_{L(g)}$ and $v_{R(g)}$ are determined before g is executed since $L(g) \leq R(g) < g$. By repeating this process, we finally obtain the output values $v_{n+q-m+1}, v_{n+q-m+2}, \dots, v_{n+q}$.

3.6 Description of Our Protocol in the Non-committed Format

We summarize our protocol in the following.

1. First, we enter the initialization phase. Given a Boolean circuit f and the input commitments to x_1, \dots, x_n , this phase outputs a sequence of $6q + 2n$ face-down cards as an initial state.
2. Next, we enter the garbling phase. Given an initial state, this phase applies a shuffle ($\text{shuffle}, \Pi, \mathcal{F}$) defined in Section 3.4. We regard the resulting sequence of $2n$ cards corresponding to the input commitments as the garbling input and the remaining sequence of $6q$ cards as the garbled circuit.
3. Finally, we enter the evaluation phase. This phase opens the garbled input and some cards of the garbled circuit. We output a m -bit string corresponding to the cards of the output gates.

In the following, we prove the correctness and security of our protocol.

Correctness: Recall that for each wire $1 \leq w \leq n + q - m$, the permutation $\hat{\pi}_w$ is defined as follows:

$$\hat{\pi}_w := \pi_w \circ \prod_{g \in L^{-1}(w)} \tau_g \circ \prod_{g' \in R^{-1}(w)} \sigma_{g'}.$$

Let w be a wire and g be a gate such that $L(g) = w$ (resp., $R(g) = w$). From the definition of $\hat{\pi}_w$, we can observe that the bit flip introduced by $\hat{\pi}_w$ and the swap of columns (resp., rows) introduced by τ_g (resp., $\sigma_{g'}$) is synchronized, which guarantees that the functionality of the circuit remains the same. Therefore, this protocol is correct.

Security: In order to prove the security, it is sufficient to show that the opened values v_i ($1 \leq i \leq n + q - m$) except the output values are independently and uniformly random bits. (We note that once this fact is proven, a simulator can be constructed in the same way as Shinagawa–Nuida [24].) In the following, we prove this fact by reverse induction from $n + q - m$ to 1.

Let A_w be the shuffle for randomizing a wire w , i.e., $A_w := (\text{shuffle}, \Pi_w, \mathcal{F}_w)$. First, v_{n+q-m} is a uniformly random bit due to the effect of uniform shuffles A_{n+q-m} and $\{A_w \mid w \in L^{-1}(n + q - m) \cup R^{-1}(n + q - m)\}$. Next, suppose that $v_{i+1}, v_{i+2}, \dots, v_{n+q-m}$ are independently and uniformly random bits.

The uniform property of v_i is obvious due to the effect of uniform shuffles A_i and $\{A_w \mid w \in L^{-1}(i) \cup R^{-1}(i)\}$. Since A_i does not appear in the wires greater than i , the randomness introduced by the shuffle A_i is independent from $v_{i+1}, v_{i+2}, \dots, v_{n+q-m}$. Thus $v_i, v_{i+1}, \dots, v_{n+q-m}$ are also independently and uniformly random bits.

Therefore, v_i ($1 \leq i \leq n + q - m$) are independently and uniformly random bits. This proves the security.

3.7 Our Protocol in the Committed Format

Although our protocol in Section 3.6 is a non-committed-format protocol, we can convert it to a committed-format protocol by appending $2m$ additional cards, where m is the number of the output wires. The committed-format protocol is the same as our -committed-format protocol except that for each output gate $g \in \{n + q - m + 1, \dots, n + q\}$, we use the eight-card truth table as in Tozawa et al. [27] instead of our six-card truth table. More concretely, we use a truth table of an output gate g as follows:

♥	♣	♥	♣
♥	♣	♣	♥

The shuffle in the committed-format protocol can be defined in the same way as in Section 3.4. By applying it, we obtain a committed-format protocol. Since each output gate requires two additional cards, the number of cards in this protocol is $6q + 2n + 2m$.

4 Conclusion

This paper proposed new single-shuffle card-based protocols for any Boolean circuit. Our protocols are based on Yao's garbled circuit as in previous single-shuffle protocols [24, 27]. Namely, the truth tables of gates in the Boolean circuit are garbled (or randomized) while keeping the output of the circuit consistent. Our core technique to reduce the number of cards is to propose a new encoding of the truth table: each value of the truth table is represented by one card, whereas the previous works used two cards per value. We also used two additional cards to apply Yao's technique to our protocol. Therefore, our protocols require only six cards per gate. Specifically, we proposed a non-committed single-shuffle card-based protocol with $6q + 2n$ cards and then modified it to make it a committed protocol with $2m$ additional cards. Since our protocols require uniform shuffles, it would be interesting to construct a committed card-based protocol with single uniform closed shuffles and a comparable number of cards to ours.

Acknowledgment This work was supported by JSPS KAKENHI Grant Numbers JP23H00468, JP23H00479, JP23K17455, JP23K16880, JP22H03590, JP21K17702, JP21H03395, JP21H03441, JP18H05289, JST CREST JPMJCR22M1, JPMJCR23M2, and MEXT Leading Initiative for Excellent Young Researchers.

References

1. Abe, Y., Hayashi, Y., Mizuki, T., Sone, H.: Five-card AND protocol in committed format using only practical shuffles. In: 5th ACM on ASIA Public-Key Cryptography Workshop. pp. 3–8. APKC '18, ACM, New York (2018), <https://doi.org/10.1145/3197507.3197510>
2. Abe, Y., Hayashi, Y., Mizuki, T., Sone, H.: Five-card AND computations in committed format using only uniform cyclic shuffles. *New Gener. Comput.* **39**(1), 97–114 (2021), <https://doi.org/10.1007/s00354-020-00110-2>
3. den Boer, B.: More efficient match-making and satisfiability: *The Five Card Trick*. In: *Advances in Cryptology - EUROCRYPT 1989, Workshop on the Theory and Application of Cryptographic Techniques*, Houthalen, Belgium, April 10-13, 1989, Proceedings. pp. 208–217 (1989)
4. Crépeau, C., Kilian, J.: Discreet solitary games. In: *Advances in Cryptology — CRYPTO' 93*. pp. 319–330. Springer Berlin Heidelberg (1994)
5. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Takagi, T., Peyrin, T. (eds.) *Advances in Cryptology—ASIACRYPT 2017*. LNCS, vol. 10626, pp. 126–155. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70700-6_5
6. Koch, A.: The landscape of security from physical assumptions. In: *IEEE Information Theory Workshop*. pp. 1–6. IEEE, NY (2021), <https://doi.org/10.1109/ITW48936.2021.9611501>
7. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: 10th International Conference on Fun with Algorithms (FUN 2021). Ed.: M. Farach-Colton. *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 157, pp. Art.–Nr.: 17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik GmbH (LZI) (2020). <https://doi.org/10.4230/LIPIcs.FUN.2021.17>
8. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: *Advances in Cryptology – ASIACRYPT 2015*. pp. 783–807. Springer Berlin Heidelberg (2015)
9. Koyama, H., Toyoda, K., Miyahara, D., Mizuki, T.: New card-based copy protocols using only random cuts. In: *ASIA Public-Key Cryptography Workshop*. pp. 13–22. ACM, NY (2021), <https://doi.org/10.1145/3457338.3458297>
10. Kuzuma, T., Isuzugawa, R., Toyoda, K., Miyahara, D., Mizuki, T.: Card-based single-shuffle protocols for secure multiple-input AND and XOR computations. In: *APKC '22: Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop, APKC@AsiaCCS 2022, Nagasaki, Japan, 30 May 2022*. pp. 51–58. ACM (2022)
11. Miyamoto, K., Shinagawa, K.: Graph automorphism shuffles from pile-scramble shuffles. *New Gener. Comput.* **40**, 199–223 (2022), <https://doi.org/10.1007/s00354-022-00164-4>
12. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: *Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 20-23, 2009*. Proceedings. pp. 358–369 (2009)
13. Mizuki, T., Uchiike, F., Sone, H.: Securely computing XOR with 10 cards. *The Australasian Journal of Combinatorics* **36**, 279–293 (2006)
14. Niemi, V., Renvall, A.: Secure multiparty computations without computers. *Theor. Comput. Sci.* **191**(1-2), 173–183 (1998), [http://dx.doi.org/10.1016/S0304-3975\(97\)00107-2](http://dx.doi.org/10.1016/S0304-3975(97)00107-2)

15. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any boolean function. In: Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings. pp. 110–121 (2015). https://doi.org/10.1007/978-3-319-17142-5_11
16. Nishida, T., Hayashi, Y.i., 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. pp. 110–121. Springer International Publishing, Cham (2015)
17. Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: An implementation of non-uniform shuffle for secure multi-party computation. In: ACM International Workshop on ASIA Public-Key Cryptography. pp. 49–55. AsiaPKC '16, ACM, New York (2016), <https://doi.acm.org/10.1145/2898420.2898425>
18. Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. IEICE Trans. Fundam. **101**(9), 1494–1502 (2018), <https://doi.org/10.1587/transfun.E101.A.1494>
19. Nishimura, A., Nishida, T., ichi Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**, 361–371 (2018)
20. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Five-card secure computations using unequal division shuffle. In: Dediu, A.H., Magdalena, L., Martín-Vide, C. (eds.) Theory and Practice of Natural Computing. LNCS, vol. 9477, pp. 109–120. Springer, Cham (2015), https://doi.org/10.1007/978-3-319-26841-5_9
21. Ruangwises, S., Itoh, T.: AND protocols using only uniform shuffles. In: van Bevern, R., Kucherov, G. (eds.) Computer Science–Theory and Applications. LNCS, vol. 11532, pp. 349–358. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-19955-5_30
22. Saito, T., Miyahara, D., Abe, Y., Mizuki, T., Shizuya, H.: How to implement a non-uniform or non-closed shuffle. In: Martín-Vide, C., Vega-Rodríguez, M.A., Yang, M.S. (eds.) Theory and Practice of Natural Computing. LNCS, vol. 12494, pp. 107–118. Springer, Cham (2020), https://doi.org/10.1007/978-3-030-63000-3_9
23. Shinagawa, K., Miyamoto, K.: Automorphism shuffles for graphs and hypergraphs and its applications. IEICE Trans. Fundam. **E106.A**(3), 306–314 (2023), <https://doi.org/10.1587/transfun.2022CIP0020>
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)
25. Stiglic, A.: Computations with a deck of cards. Theor. Comput. Sci. **259**(1-2), 671–678 (2001). [https://doi.org/10.1016/S0304-3975\(00\)00409-6](https://doi.org/10.1016/S0304-3975(00)00409-6)
26. Toyoda, K., Miyahara, D., Mizuki, T., Sone, H.: Six-card finite-runtime XOR protocol with only random cut. In: ACM Workshop on ASIA Public-Key Cryptography. pp. 2–8. APKC '20, ACM, New York (2020), <https://doi.org/10.1145/3384940.3388961>
27. Tozawa, K., Morita, H., Mizuki, T.: Single-shuffle card-based protocol with eight cards per gate. In: Unconventional Computation and Natural Computation. pp. 171–185. Springer Nature Switzerland, Cham (2023)
28. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. Int. J. Inf. Sec. **19**(4), 445–452 (2020). <https://doi.org/10.1007/s10207-019-00463-w>
29. Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Theory and Practice of Natural Computing - 5th International Conference, TPNC 2016, Sendai, Japan, December 12-13, 2016, Proceedings. pp. 58–69 (2016). https://doi.org/10.1007/978-3-319-49001-4_5

30. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS. pp. 162–167. IEEE Computer Society (1986), <http://dblp.uni-trier.de/db/conf/focs/focs86.html/Yao86>