Free-XOR in Card-based Garbled Circuits

No Author Given

No Institute Given

Abstract. This paper shows a free-XOR technique in card-based garbled circuits. Card-based cryptographic protocols were proposed as a secure multiparty computation using physical cards instead of computers. They can be used when users cannot trust software on computers. Shinagawa and Nuida proposed card-based garbled circuits that compute any Boolean functions using a single shuffle. Their protocol uses $24g + 2n$ cards, where g is the number of gates and n is the number of inputs. Tozawa et al. reduced the number of cards to $8g+2n$. This paper introduces the free-XOR technique for standard garbled circuits to card-based garbled circuits. It is unnecessary to prepare a garbled table for XOR gates. The number of cards is reduced to $8g_1+2g_2+2n$, where g_1 is the number of gates other than XOR and g_2 is the number of XOR gates whose output is used as a final output. The card-based garbled circuits proposed by Shinagawa and Nuida have one restriction the final outputs cannot be used for inputs to the other gates. This paper eliminates the restriction with two different techniques.

Keywords: Card-based cryptographic protocols · secure multiparty computation · garbled circuits · exclusive or · free-XOR

1 Introduction

Card-based cryptographic protocols [14, 30, 31] were proposed in which physical cards are used instead of computers to securely compute values. They can be used when computers cannot be used or users cannot trust the software on the computer. Also, the protocols are easy to understand, thus the protocols can be used to teach the basics of cryptography [5, 26]. den Boer [4] first showed a five-card protocol to securely compute the logical AND of two inputs. Since then, many protocols have been proposed to realize primitives to compute any Boolean functions [13, 16, 21, 32, 37, 43, 44, 51, 52] and specific computations such as a specific class of Boolean functions [2, 3, 7, 12, 18–20, 22, 27, 29, 38, 41, 46, 47, 50, 56], universal computation such as Turing machines [6, 15], millionaires' problem [23, 34, 42], voting [1, 28, 35, 36, 39, 55, 59], random permutation [8, 10, 11, 33], grouping [9], ranking [54], lottery [53], and so on.

Shinagawa and Nuida [52] proposed a protocol to compute any Boolean functions using the garbled circuit technique [60]. The number of shuffles used in the protocol is one. Their protocol uses $24g + 2n$ cards, where g is the number of gates and n is the number of inputs. Tozawa et al. [57] reduced the number of cards to $8g + 2n$.

To reduce the size of standard garbled tables, free-XOR technique [17] was shown, in which no garbled table is necessary for XOR gates. This paper introduces the technique to card-based garbled circuits. We show that garbled tables are also unnecessary for XOR gates in card-based garbled circuits. Thus no cards are necessary for internal XOR gates, where, the output of an XOR gate is not a final output. When the output of an XOR gate is a final output, two cards are necessary. The number of cards is thus reduced to $8g_1 + 2g_2 + 2n$, where g_1 is the number of gates other than XOR and g_2 is the number of XOR gates whose output is a final output. The number of shuffles is kept to one.

The card-based garbled circuits proposed by Shinagawa and Nuida [52] have one restriction the final outputs cannot be used for inputs to the other gates. Though each input value in the garbled tables is randomized to hide the value, the output data must not be randomized. That is the reason for the restriction. This paper considers eliminating the restriction with two different techniques. The first technique is preparing a copy of garbled table entries that is used for final outputs. The second technique is remembering the random value and undoing the randomization. Though the former technique needs more cards, the total number of shuffles is kept to one. The latter technique needs one additional shuffle.

Section 2 shows basic notations and definitions of card-based cryptographic protocols. Section 3 shows Shinagawa-Nuida card-based garbled circuit whose size is reduced by [45]. Section 4 shows the new free-XOR technique for card-based garbled circuits. Section 5 discusses eliminating the output restriction in [52]. Section 6 concludes the paper.

2 Preliminaries

This section gives the notations and basic definitions of card-based cryptographic protocols. Most of the results are based on a two-color card model. In the two-color card model, there are two kinds of marks, $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. Cards of the same marks cannot be distinguished. In addition, the back of both types of cards is $\boxed{?}$. It is impossible to determine the mark on the back of a given card of $\boxed{?}$.

One-bit data is represented by two cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$.

One pair of cards that represents one bit $x \in \{0, 1\}$, whose face is down, is called a commitment of x, and denoted as $commit(x)$. It is written as $\underbrace{\boxed{?}\boxed{?}}_{x}$. Note that when these two cards are swapped, $commit(\bar{x})$ can be obtained. Thus, logical negation can be easily computed.

A set of cards placed in a row is called a sequence of cards. A sequence of cards S whose length is n is denoted as $S = s_1, s_2, \ldots, s_n$, where s_i is i-th card of the sequence. $S = \underbrace{\boxed{?}}_{s_1}\underbrace{\boxed{?}}_{s_2}\underbrace{\boxed{?}}_{s_3}\ldots,\underbrace{\boxed{?}}_{s_n}$.

All protocols are executed by two players, Alice and Bob. The players are semi-honest, that is, they obey the rule of the protocol, but they try to obtain secret values.

Next, we discuss the inputs and outputs of the protocols. Most protocols have committed inputs, that is, the inputs are given to the players in a committed manner. The players do not know the input values and they might try to obtain the input values during the protocol execution. The other type of protocol considers the case when each player inputs his/her input value that must be hidden from the other player. They are called non-committed input protocols. Note that committed-input protocols can be used when the players input their own values. Each player makes a commitment to his/her input in advance and they are used as inputs. Thus, committed-input protocols are desirable. On the other hand, non-committed input protocols can be simple and might reduce the number of cards used in the protocol.

Most protocols output the result in a committed manner. They are called committed-output protocols. On the other hand, several protocols terminate by opening some cards and obtaining the result from the sequence of the opened cards. Such protocols are called non-committed output protocols. Committed-output protocols are desirable since the committed output can be used as input for further secure computations.

Next, we show operations on the cards. Opening a card is turning a face-down card into a face-up, thus the players can see the mark on the card. Face-down a card is turning a face-up card to face-down. Rearrangement is a permutation of a sequence of cards, that is, the position of a given sequence of cards is changed.

A shuffle is executed on a sequence of cards S. Its parameter is $(\Pi, \mathcal{F})$, where Π is a set of permutations on S and $\mathcal{F}$ is a probability distribution on Π. For a given sequence S, each permutation $\pi \in \Pi$ is selected by the probability distribution $\mathcal{F}$ and π is applied to S. If π is applied on $S = s_1, s_2, \ldots, s_n$, the result is $s_{\pi^{-1}(1)}, s_{\pi^{-1}(2)}, \ldots, s_{\pi^{-1}(n)}$. Since π is selected from Π, the result is not deterministic. Non-deterministic execution is necessary for card-based protocols. If all operations are deterministic, the relation between the committed input value and the committed output value is known to the players. When the committed output cards are opened to see the final output, the private input data is known to the players using the relation between the input and the output. Thus non-deterministic execution is necessary to hide the private input values.

We show examples of shuffles used in the protocols shown below. A random shuffle is randomly changing the positions of the cards for the given sequence of cards. When $S = s_1, s_2, s_3$, the result of a random shuffle is $S_1 = s_1, s_2, s_3$, $S_2 = s_1, s_3, s_2$, $S_3 = s_2, s_1, s_3$, $S_4 = s_2, s_3, s_1$, $S_5 = s_3, s_1, s_2$, or $S_6 = s_3, s_2, s_1$. The probability of obtaining each result is $1/|S|!$.

A shuffle is uniform if $\mathcal{F}$ is a uniform distribution, that is, $\pi \in \Pi$ is selected uniformly at random. A shuffle is closed if multiple executions of a shuffle are also the same shuffle. Non-uniform shuffles are not desirable since they are difficult to execute by human hands. Using some additional cards or tools, protocols to execute any kinds of shuffles were shown [25, 40, 48, 49, 58].

Closed shuffles are desirable since each one of Alice and Bob can execute one instance of the shuffle to obtain one shuffle result. Even if Alice and Bob are not honest and each player knows the result of his/her shuffle, the final result of the

two shuffles is unknown to the players if there is no collusion between Alice and Bob. The random shuffle shown above is uniform and closed.

Next, we introduce piles of cards. A pile of cards is a sequence of cards whose order cannot be changed using some additional tools such as clips or envelopes. For example, consider a case when cards $s_{i,j}(i = 1, 2, \ldots, n,\ j = 1, 2, \ldots m)$ are given. The players make piles of cards such that $P_i = s_{i,1}, s_{i,2}, \ldots, s_{i,m}(i = 1, 2, \ldots, n)$ using clips or envelopes. The players treat each pile P_i just like a single card during shuffle operations. The order of cards in a pile cannot be changed because of the clip or envelope. Consider the case shuffle π is executed on the above piles $P_i(i = 1, 2, \ldots, n)$. The result is $P_{\pi^{-1}(1)}, P_{\pi^{-1}(2)}, \ldots, P_{\pi^{-1}(n)}$, where $P_{\pi^{-1}(i)} = s_{\pi^{-1}(i),1}, s_{\pi^{-1}(i),2}, \ldots, s_{\pi^{-1}(i),m}$. Random shuffles on piles are called pile-scramble shuffles.

Last, the efficiency of the protocol is evaluated by the number of cards used by the protocol. It corresponds to the space complexity of programs.

The number of shuffles is used to evaluate the time complexity of the protocols since the other operations are simple [24].

3 Card-based garbled circuits

Garbled circuits [60] are a fundamental technique to securely compute any function by two semi-honest players. The original garbled circuits consider the case when Alice has input value x and Bob has input y. They want to compute $f(x, y)$ together without revealing each player's input value to the other player.

Shinagawa and Nuida [52] proposed a card-based cryptographic protocol to compute any Boolean functions using a single shuffle by using the garbled circuit technique. The problem definition differs from the above one. Alice and Bob have functions $f_i(x_1, x_2, \ldots, x_n)(i = 1, 2, \ldots, m)$ to compute from input $x_1, x_2, \ldots, x_n$. The inputs are given by cards in a committed manner. The outputs must be given in a committed manner.

Tozawa et al. [57] reduced the number of cards. Their protocol uses eight cards for each gate.

First, we show the outline of the computation with no security and the secure protocol shown in [57].

For each two-input logic gate, Alice and Bob prepare a table that represents the relation between the inputs and the output as in Fig. 1, which shows the case of $g_1 = x_1 \oplus x_2$. The first(second) row has the values when $x_1 = 0(1)$, respectively. The first(second) column has the values when $x_2 = 0(1)$, respectively.

All the cards are then set to face-down to hide the values of the table. Consider the simple case when x_1, x_2 are private inputs given to the players and $g_1 = x_1 \oplus x_2$ is a final output. The input value x_1 and x_2 are given by face-down cards. The players open the cards and search for the entry that corresponds to the input values. For example, if $x_1 = 1$ ($\boxed{\heartsuit}\boxed{\clubsuit}$) and $x_2 = 0$ ($\boxed{\clubsuit}\boxed{\heartsuit}$), the entry at the second row and the first column has the result. If the cards are opened, the value is $\boxed{\heartsuit}\boxed{\clubsuit}$, which is the correct result of $x_1 \oplus x_2$. The result is obtained

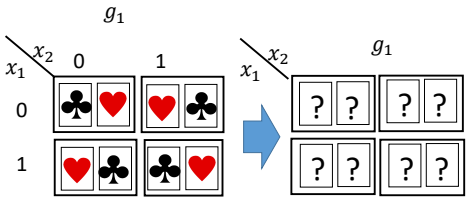

Fig. 1. Table to compute $g_1 = x_1 \oplus x_2$

in a committed manner. Further computation of the other gates can be similarly executed. The final output can be obtained in a committed manner.

Since the players open the input values, the security of inputs is not realized. To solve the problem, Alice and Bob randomize the inputs of the tables and input values together. For each garbled table, make two piles P_1 and P_2. $P_1(P_2)$ consists of the first(second) row of the table and the left(right) card of input x_1, respectively. P_1 and P_2 consist of five cards. Alice and Bob execute a pile-scramble shuffle on P_1 and P_2 as in Fig. 2. With probability $1/2$, P_1 and P_2 are swapped. With probability $1/2$, they are unchanged. After the shuffle, the cards are set back to each position. The result can be represented by a random value $r_1 \in \{0, 1\}$ as follows: the cards that have the input x_1 is changed to $x_1 \oplus r_1$ and the first row of the garbled table has the values when the input is r_1.

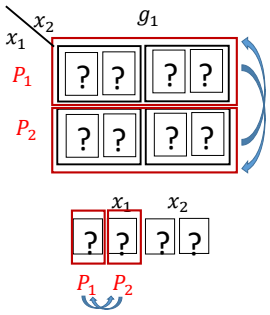

Fig. 2. Randomization for input x_1

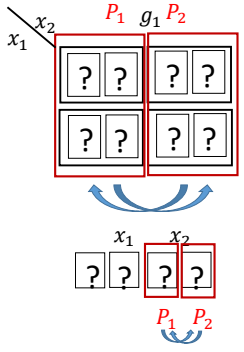

Fig. 3. Randomization for input x_2

Another pile-scramble shuffle is similarly executed for the input x_2 and the two columns of the table, as shown in Fig. 3. The result can be similarly represented by another random value $r_2 \in \{0, 1\}$. The cards that have the input x_2 is changed to $x_2 \oplus r_2$ and the first column of the garbled table has the values when the input is r_2.

When we execute the computation after the pile-scramble shuffles, the players can obtain the correct result of the computation of the gate. For example, consider the case when $x_1 = 1$, $x_2 = 0$, $r_1 = 1$, and $r_2 = 0$. The players see $0 = x_1 \oplus r_1$ and $0 = x_2 \oplus r_2$ when the input cards are opened. Thus the players select the element in the first row and the first column in the table. The result is correct since the entry was initially at the second row and the first column before the shuffles.

Since the players open $x_1 \oplus r_1$ and $x_2 \oplus r_2$, the security of the input values is achieved because r_1 and r_2 are random values unknown to Alice and Bob.

Note that input x_1 might also be an input of another gate $g_2, g_3, \ldots, g_k$. In this case, when the players make piles P_1 and P_2, the entries of the table for $g_2, g_3, \ldots, g_k$ must also be added.

When the output of g_1 is the final output, the computation is finished and the players obtain the committed result. When the output of g_1 is an input of another gate, the further computation is necessary. Let $g'_1, g'_2, \ldots, g'_i$ be the gates that input g_1's output. In this case, since the output cards of g_1 must be opened to select entries of g_j's garbled table, g_1's output value must also be randomized in advance to hide the output value. The randomization of the output must be executed together with the tables of $g'_1, g'_2, \ldots, g'_i$.

For example, Fig. 4 shows the case when the output of g_1 is used as the row input x_3 of gate g_2. Similar to the above case, the players make two piles P_1 and P_2. P_1 (P_2) consist of the left (right) card of each entry of table g_1 and the first (second) row of the table g_2, respectively. Execute a pile-scramble shuffle on P_1 and P_2 and the cards are set back to each position. Using a random value $r \in \{0, 1\}$, the output of g_1 is changed as $g_1 \oplus r$. When the players open the output card of g_1, the players obtain no information about the output since the value is randomized by r. In addition, the computation of g_2 is still correct since the entries of the tables are randomized using the same random value r.

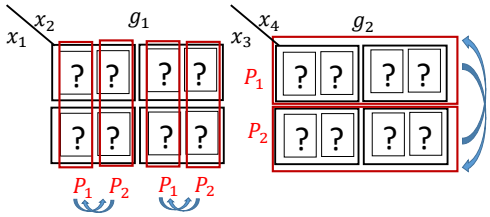

Fig. 4. Randomization when g_2's input x_3 is the output of g_1.

Note that all shuffles of the inputs and table entries are executed in advance to compute.

In summary, the protocol is executed as follows.

1. Prepare one table for each gate that is used to compute $f_i (i = 1, 2, \ldots, m)$.

2. When a value x (an input value or the output of a garbled table) is used for the row input of gate $g_1, g_2, \ldots, g_k$ and the column input of gate $g'_1, g'_2, \ldots, g'_{k'}$, make two piles P_1 (P_2) with the left (right) card(s) of x, the first (second) row of the garbled table of gate $g_1, g_2, \ldots, g_k$, and the first (second) column of the garbled table of gate $g'_1, g'_2, \ldots, g'_{k'}$, respectively. Execute a pile-scramble shuffle to P_1 and P_2. Set back the cards to the initial positions. Execute the above procedure for every value x that will be opened during computation.
3. For each gate, open (randomized) input cards and select the row and column entry that matches the opened value and obtain the committed output of the gate.
4. The final output cards are not opened and they are used as the result.

Though in this example the cards are set as a 2×2 table, they can also be set as one sequence of cards, for example, the output cards for input (0,0), (0,1) (1,0), and (1,1) can be placed in this order as in [57]. Since the players know each position, they can make two piles to be shuffled using the positions.

Note that any kind and any number of shuffles can be combined into one shuffle [52], thus the total number of shuffles is one. Since the final output must not be randomized, the output must not be used as an input of another gate.

4 Free-XOR in card-based garbled circuits

Free-XOR [17] is a technique for garbled circuits. It is unnecessary to prepare a garbled table for each XOR gate. This section shows that a garbled table is also unnecessary for XOR gates in the above card-based garbled circuits. Two cards are necessary when the output of an XOR gate is a final output. No cards are necessary when the output of an XOR gate is input to the other gates.

Before showing the protocol, we need to simplify the discussion. We need to eliminate the case when the output of an XOR gate is an input of another XOR gate. The output of an XOR gate $g_1 = x_1 \oplus x_2$ might be used as an input of another XOR gate such as $g_2 = g_1 \oplus x_3$. g_2 can be written as $g_2 = x_1 \oplus x_2 \oplus x_3$ to eliminate the case when the output of an XOR gate is an input of another XOR gate. A similar transformation can be executed when g_2 is also an input of another XOR gate. Thus, the cases to be considered are: the output of an XOR gate is (1) a final output value or (2) an input of a non-XOR gate, where the number of inputs of the XOR gate is arbitrary (> 1) and the inputs of the XOR gate are not an output of another XOR gate, that is, they are initially-given inputs or outputs of a non-XOR gate. In the above example of g_1 and g_2, when g_2 is the final output and g_1 is an input of a non-XOR gate g_3, we need to compute (1) the output of $g_1 = x_1 \oplus x_2 \oplus x_3$ is a final output and (2) the output of $g_2 = x_1 \oplus x_2$ is an input of non-XOR gate g_3.

Note that the negation of an XOR might be needed. For example, consider the case when the players compute $g_4 = \overline{x_1 \oplus x_2}$. We note that we do not need to consider negation of the XOR gates[1]. As shown above, g_4 is a final output or

[1] By a similar argument, we can see that negation of gates is unnecessary for any gates

an input of another non-XOR gate, for example, $g_5 = g_4 \wedge x_3$. In the former case, swap the pair of the cards that have the output $x_1 \oplus x_2$ and we can compute g_4. In the latter case, we can prepare a garbled table for g_5 in which the input of the first element is negated, just as $\bar{x} \wedge x_3$. Thus, we do not need to consider the negation of XOR gates.

First, consider the case when the output of XOR gate g is a final output value. Let $x_1, x_2, \ldots, x_k$ be inputs to compute $g = \bigoplus_{i=1}^{k} x_i$. $x_i (1 \leq i \leq k)$ are initially-given inputs or outputs of non-XOR garbled tables.

The protocol for XOR gate g is the following steps.

- For the computation of g, prepare one pair of cards, denoted as G. Initially, G is $\boxed{\clubsuit}\boxed{\heartsuit}$ and it is turned into a committed value.
- In Step 2 of the above protocol, when the left card and the right card for value x_i are included in a pile P_1 and P_2, the left card of G is also set into P_1 and the right card of G is also set into P_2 then a pile-scramble shuffle is executed. For each input $x_i (1 \leq i \leq k)$, the above procedure is executed. Fig. 5 shows the randomization of $g = x_1 \oplus x_2$, where x_1 and x_2 are input values. Make P_1 (P_2) be the left (right) cards of G and x_1, respectively. Execute a Pile-scramble shuffle to P_1 and P_2. Then, make P'_1 (P'_2) be the left (right) cards of G and x_2, respectively. Execute a Pile-scramble shuffle to P'_1 and P'_2.
  Note that when x_i is an output of a garbled table, P_1 (P_2) consists of the left (right) cards of the garbled table.

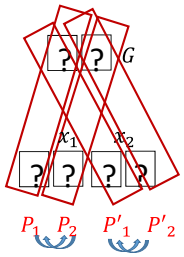

Fig. 5. Randomization of $g = x_1 \oplus x_2$'s input x_1 and x_2.

- When the players compute the gate $g = \bigoplus_{i=1}^{k} x_i$, the appropriate cards that have x_i are opened. Note that the value opened, x'_i, might not be x_i because of the randomization. Swap the two cards of G if the opened values x'_i satisfy $\bigoplus_{i=1}^{k} x'_i = 1$. The final committed pair G is used as the result of $g = \bigoplus_{i=1}^{k} x_i$.

Theorem 1. *The above protocol correctly computes* $g = \bigoplus_{i=1}^{k} x_i$.

Proof. Initially, G has value 0. By the pile-scramble shuffle of input x_i, the value is randomized as $x_i \oplus r_i$ for some $r_i \in \{0,1\}$. At the same time, G is also randomized using r_i thus the value is changed from 0 to $0 \oplus (\bigoplus_{i=1}^{k} r_i) = \bigoplus_{i=1}^{k} r_i$. When the gate g is computed, cards of the inputs are opened. The opened values are $x_i \oplus r_i$. The two cards of G are swapped if $\bigoplus_{i=1}^{k}(x_i \oplus r_i) = 1$. Thus the value of G is changed from $\bigoplus_{i=1}^{k} r_i$ to $\bigoplus_{i=1}^{k} r_i \oplus (\bigoplus_{i=1}^{k}(x_i \oplus r_i)) = \bigoplus_{i=1}^{k} x_i$. Thus the result is correct. □

Next, consider the case when the output of $g = \bigoplus_{i=1}^{k} x_i$ is used as an input of another gate g'. The following protocol shows the case when g is the row input of g'. The case when g is the column input of g' can be similarly shown.

The protocol for XOR gate g is the following steps.

- For the computation of $g = \bigoplus_{i=1}^{k} x_i$, no cards are prepared. Instead, the cards for the input of g' are used. Suppose that g is the row input of g'.
- In Step 2 of the above protocol, when the left (right) card(s) of value x_i are included in a pile P_1 (P_2), respectively, the first (second) row of the table of g' is also set into P_1 (P_2), respectively. Then a pile-scramble shuffle is executed. For each input $x_i (1 \leq i \leq k)$, the above procedure is executed. Fig. 6 shows the randomization of $g = x_1 \oplus x_2$, where x_1 and x_2 are input values and g is the row input of g'. Make pile P_1 (P_2) by the left (right) card of x_1 and first (second) row of g', respectively. Execute pile-scramble shuffle to P_1 and P_2. Next. make pile P_1' (P_2') by the left (right) card of x_2 and first (second) row of g', respectively. Execute pile-scramble shuffle to P_1' and P_2'. Note that when x_i is an output of a garbled table, P_1 (P_2) consists of the left (right) cards of the garbled table.

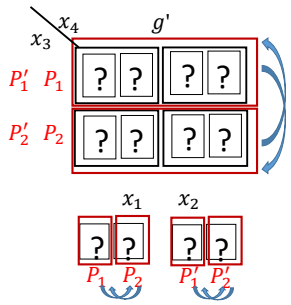

Fig. 6. Randomization of $g = x_1 \oplus x_2$'s input x_1 and x_2 when g is the row input of g'

- When the players compute the gate g', The appropriate cards that have x_i are opened. Note that the value opened, x_i', might not be x_i because of the randomization. The first row is used to compute g' if $\bigoplus_{i=1}^{k} x_i' = 0$. Otherwise, the second row is used.

The output g might be inputs of multiple gates $g_1, g_2, \ldots, g_m$. In this case, all appropriate rows or columns of the tables for gate $g_1, g_2, \ldots, g_m$ are included to pile P_1 and P_2 to shuffle each input x_i.

Theorem 2. *The above protocol correctly computes the input value* $g = \bigoplus_{i=1}^{k} x_i$ *of gate* g'.

Proof. This proof assumes that g is the row input of gate g'. The case when g is the column input can be similarly proved. Initially, the first (second) row of g''s garbled table has the values when the input is 0 (1), respectively.

By the pile-scramble shuffle of input x_i, the value is randomized as $x_i \oplus r_i$ for some $r_i \in \{0, 1\}$. At the same time, the rows of g' are also randomized using r_i thus the first row has the values when the input $0 \oplus (\bigoplus_{i=1}^{k} r_i) = \bigoplus_{i=1}^{k} r_i$ is 0. When the gate g' is computed, cards of g's inputs are opened. The opened values are $x_i \oplus r_i$. The players use the first row if $\bigoplus_{i=1}^{k}(x_i \oplus r_i) = 0$, otherwise, they use the second row to compute g'. $\bigoplus_{i=1}^{k}(x_i \oplus r_i) = 0$ implies that $\bigoplus_{i=1}^{k} x_i = \bigoplus_{i=1}^{k} r_i$. Thus, when the players select the first row, the first row has the values when the input $\bigoplus_{i=1}^{k} r_i$ is 0, that is, $\bigoplus_{i=1}^{k} x_i$ is 0. Therefore, the selection is correct. □

Note that the combined single shuffle becomes complicated since a pair of cards is included in both of x_1 and x_2's shuffles. However, anyway, the shuffles can be executed by a single shuffle since any combination of shuffles can be executed by a single shuffle. The combined single shuffle is uniform and closed since each shuffle is swapping two elements by the probability of 1/2.

The number of cards used by the protocol is $8g_1 + 2g_2 + 2n$, where g_1 is the number of non-XOR gates and g_2 is the number of XOR gates whose output is a final output.

5 Eliminating restriction for outputs

As shown above, the Shinagawa-Nuida protocol has a restriction that the output values cannot be used for inputs to the other circuits. This section discusses eliminating the restriction.

This section discusses the functions in the following form:

$$f_i(x_1, x_2, \ldots, x_n, f_1, f_2, \ldots, f_{i-1})(i = 1, 2, \ldots, m)$$

The definition considers the outputs $f_1, f_2, \ldots, f_{i-1}$ can be used as inputs of f_i. It is unnecessary to use the outputs of some functions as inputs of another function, but it might reduce the number of logic gates. For example, consider the case when we need to compute $f_1 = x_1 \vee x_2$ and $f_2 = (x_1 \vee x_2) \wedge x_3$. We can compute f_2 by $f_2 = f_1 \wedge x_3$.

Note that $f_j(j > i)$ cannot be used in f_i to avoid circular definition such as $f_2 = f_1 \wedge x_1$ and $f_1 = f_2 \wedge x_2$.

In the garbled circuits, each input of a gate must be randomized because the input cards are opened and the value is known to the players. On the other

hand, the output value must not be randomized. Thus, Shinagawa and Nuida added the restriction that output cannot be used as an input of another gate. There are two ways to eliminate this restriction. The first technique is preparing cards for a non-randomized value and the second one is undoing randomization.

Before showing the technique, let us consider the case when the output of an XOR gate g is the final output. As shown in the previous section, no additional cards are necessary to input the output of g to a non-XOR gate or another XOR gate. Thus, we discuss the case when the output of a non-XOR gate is a final output.

The first technique is simple. If the output of a gate is a final output and input of another gate, prepare two pairs of each output value in the garbled table as in Fig. 7, where $g_{i,O}$ are cards for the output and $g_{i,I}$ are cards for the input of gate $g'_1, g'_2, \ldots, g'_k$. When $g'_1, g'_2, \ldots, g'_k$'s inputs are simultaneously randomized, $g_{i,I}$ are included in the randomization, but $g_{i,O}$ are not included. Note that $g_{i,O}$ are included in the shuffles of the rows or columns of the table of g_i. The values in $g_{i,O}$ are used as the final output, and the values in $g_{i,I}$ are used for the garbled table lookup. Since the value $g_{i,O}$ are not randomized, $g_{i,O}$ can be used as the output. Since the values in $g_{i,I}$ are randomized, $g_{i,I}$ can be opened for the garbled table lookup.

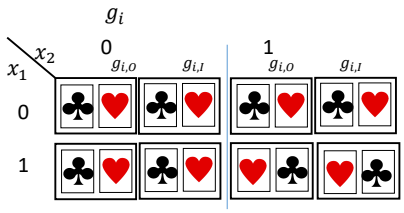

Fig. 7. Output of gate g_i

The second technique is undoing randomization. If an output of a gate g_i is a final output and an input of another gate, prepare one pair of cards O_i whose initial value is $\boxed{\clubsuit}\boxed{\heartsuit}$. The cards are set face-down. The change of the protocol is as follows.

- In Step 2 of the above protocol, when the left (right) card of the output of gate g_i are included in a pile P_1 (P_2), respectively, the left (right) card of O_i is also set into P_1 (P_2), respectively. Then a pile-scramble shuffle is executed. For example, O_i for gate g_i and their randomization is shown in Fig. 8.
- During the computation of gate g_i, one pair of cards, G_i, is selected as the output and opened because the value is used as an input of another gate. After the computation is finished, the cards for G_i are turned face-down.
- Make pile $P_{i,1}$ ($P_{i,2}$) that consists of the left (right) cards of O_i and G_i, respectively. Execute a pile-scramble shuffle on $P_{i,1}$ and $P_{i,2}$, as in Fig. 9,

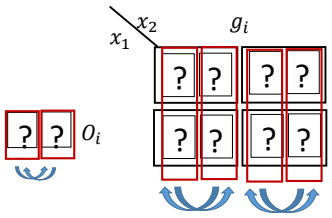

Fig. 8. O_i for output of gate g_i and randomization.

which shows the case when the output is the second row and the second column.

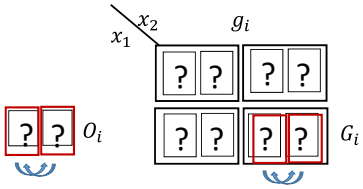

Fig. 9. randomization of G_i and O_i.

Open O_i and swap two cards of G_i if O_i has value 1, as shown in Fig. 10. G_i is used as a final output.

Theorem 3. *The above protocol is secure and correctly outputs g_i.*

Proof. During the randomization of the output value of g_i, O_i is also randomized. The value that the output card G_i has is $g_i \oplus r_i$ for some unknown random value r_i. At the same time, the cards O_i have r_i since $0 \oplus r_i = r_i$. After G_i is turned face-down again, G_i and O_i are randomized using a random value $r'_i \in \{0, 1\}$. G_i has $g_i \oplus r_i \oplus r'_i$ and O_i has $r_i \oplus r'_i$. Then the players open O_i and swap the two cards of G_i if $O_i = 1$. The output is correct since G_i has $g_i \oplus r_i \oplus r'_i \oplus (r_i \oplus r'_i) = g_i$.

The protocol is secure since the players see $g_i \oplus r_i$ and $r_i \oplus r'_i$. The value g_i cannot be known from these values. □

Since the randomization of output G_i must be executed after the garbled table lookup, two shuffles are necessary for the total. Note that the shuffles for each O_i are combined into one shuffle.

The first technique needs eight cards for each output. The number of shuffles is one. The second technique needs two cards for each output, though the number of shuffles becomes two.

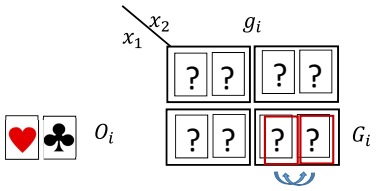

Fig. 10. computation of the output using G_i and O_i.

6 Conclusion

This paper showed the free-XOR technique in card-based garbled circuits. The number of cards is reduced though the shuffle becomes complicated. This paper then showed techniques to eliminate the restriction that an output value cannot be used as an input of another gate. These methods improve the efficiency of card-based garbled circuits.

References

1. Abe, Y., Nakai, T., Kuroki, Y., Suzuki, S., Koga, Y., Watanabe, Y., Iwamoto, M., Ohta, K.: Efficient card-based majority voting protocols. New Generation Computing **40**(1), 173–198 (2022)
2. Abe, Y., Hayashi, Y.i., Mizuki, T., Sone, H.: Five-card and computations in committed format using only uniform cyclic shuffles. New Generation Computing **39**(1), 97–114 (2021)
3. Abe, Y., Mizuki, T., Sone, H.: Committed-format and protocol using only random cuts. Natural Computing pp. 1–7 (2021)
4. den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Proc. of EUROCRYPT '89, LNCS Vol. 434. pp. 208–217 (1990)
5. Cheung, E., Hawthorne, C., Lee, P.: Cs 758 project: Secure computation with playing cards (2013), http://cdchawthorne.com/writings/secure_playing_cards.pdf
6. Dvořák, P., Koucký, M.: Barrington plays cards: The complexity of card-based protocols. arXiv preprint arXiv:2010.08445 (2020)
7. Francis, D., Aljunid, S.R., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Necessary and sufficient numbers of cards for securely computing two-bit output functions. In: Proc. of Second International Conference on Cryptology and Malicious Security(Mycrypt 2016), LNCS Vol. 10311. pp. 193–211 (2017)
8. Hashimoto, Y., Nuida, K., Shinagawa, K., Inamura, M., Hanaoka, G.: Toward finite-runtime card-based protocol for generating hidden random permutation without fixed points. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101-A**(9), 1503–1511 (2018)
9. Hashimoto, Y., Shinagawa, K., Nuida, K., Inamura, M., Hanaoka, G.: Secure grouping protocol using a deck of cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101**(9), 1512–1524 (2018)

10. Ibaraki, T., Manabe, Y.: A more efficient card-based protocol for generating a random permutation without fixed points. In: Proc. of 3rd Int. Conf. on Mathematics and Computers in Sciences and in Industry (MCSI 2016). pp. 252–257 (2016)
11. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Proc. of 14th International Conference on Unconventional Computation and Natural Computation(UCNC 2015), LNCS Vol. 9252. pp. 215–226 (2015)
12. Isuzugawa, R., Toyoda, K., Sasaki, Y., Miyahara, D., Mizuki, T.: A card-minimal three-input and protocol using two shuffles. In: Proc. of 27th International Computing and Combinatorics Conference (COCOON 2021), LNCS Vol. 13025. pp. 668–679. Springer (2021)
13. 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: Proc. of Asiacrypt 2017, Part III, LNCS Vol. 10626. pp. 126–155 (2017)
14. Koch, A.: The landscape of optimal card-based protocols. Mathematical Cryptology **1**(2), 115–131 (2021)
15. Koch, A., Walzer, S.: Private function evaluation with cards. New Generation Computing **40**(1), 115–147 (2022)
16. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Proc. of Asiacrypt 2015, LNCS Vol. 9452. pp. 783–807 (2015)
17. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free xor gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) Proc. of 35th International Colloquium on Automata, Languages, and Programming (ICALP 2008) Part II, LNCS Vol.5126. pp. 486–498. Springer (2008)
18. Koyama, H., Toyoda, K., Miyahara, D., Mizuki, T.: New card-based copy protocols using only random cuts. In: Proceedings of the 8th ACM on ASIA Public-Key Cryptography Workshop. pp. 13–22. APKC '21, Association for Computing Machinery, New York, NY, USA (2021)
19. Kuzuma, T., Isuzugawa, R., Toyoda, K., Miyahara, D., Mizuki, T.: Card-based single-shuffle protocols for secure multiple-input and and xor computations. In: Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop. pp. 51–58 (2022)
20. Manabe, Y., Ono, H.: Card-based cryptographic protocols for three-input functions using private operations. In: Proc. of 32nd International Workshop on Combinatorial Algorithms (IWOCA 2021), LNCS Vol. 12757. pp. 469–484. Springer (2021)
21. Manabe, Y., Ono, H.: Card-based cryptographic protocols with a standard deck of cards using private operations. In: Proc. of 18th International Colloquium on Theoretical Aspects of Computing (ICTAC 2021), LNCS Vol.12819. Springer (2021)
22. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. IACR Cryptology ePrint Archive, Report 2015/1031 (2015)
23. Miyahara, D., Hayashi, Y.i., Mizuki, T., Sone, H.: Practical card-based implementations of yao's millionaire protocol. Theoretical Computer Science **803**, 207–221 (2020)
24. Miyahara, D., Ueda, I., Hayashi, Y.i., Mizuki, T., Sone, H.: Evaluating card-based protocols in terms of execution time. International Journal of Information Security **20**(5), 729–740 (2021)
25. Miyamoto, K., Shinagawa, K.: Graph automorphism shuffles from pile-scramble shuffles. New Generation Computing **40**(1), 199–223 (2022)

26. Mizuki, T.: Applications of card-based cryptography to education. In: IEICE Technical Report ISEC2016-53. pp. 13–17 (2016), (In Japanese)
27. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theoretical Computer Science **622**, 34–44 (2016)
28. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Proc. of 12th International Conference on Unconventional Computing and Natural Computation (UCNC 2013), LNCS Vol. 7956. pp. 162–173 (2013)
29. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Proc. of Asiacrypt 2012, LNCS Vol.7658. pp. 598–606 (2012)
30. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security **13**(1), 15–23 (2014)
31. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **100**(1), 3–11 (2017)
32. Mizuki, T., Sone, H.: Six-card secure and and four-card secure xor. In: Proc. of 3rd International Workshop on Frontiers in Algorithms(FAW 2009), LNCS Vol. 5598. pp. 358–369 (2009)
33. Murata, S., Miyahara, D., Mizuki, T., Sone, H.: Efficient generation of a card-based uniformly distributed random derangement. In: Proc. of 15th International Workshop on Algorithms and Computation (WALCOM 2021), LNCS Vol. 12635. pp. 78–89. Springer International Publishing, Cham (2021)
34. Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: How to solve millionaires' problem with two kinds of cards. New Generation Computing **39**(1), 73–96 (2021)
35. Nakai, T., Shirouchi, S., Iwamoto, M., Ohta, K.: Four cards are sufficient for a card-based three-input voting protocol utilizing private permutations. In: Proc. of 10th International Conference on Information Theoretic Security (ICITS 2017), LNCS Vol. 10681. pp. 153–165 (2017)
36. Nakai, T., Shirouchi, S., Tokushige, Y., Iwamoto, M., Ohta, K.: Secure computation for threshold functions with physical cards: Power of private permutations. New Generation Computing **40**(1), 95–113 (2022)
37. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any boolean function. In: Proc. of 15th International Conference on Theory and Applications of Models of Computation(TAMC 2015), LNCS Vol. 9076. pp. 110–121 (2015)
38. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Securely computing three-input functions with eight cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **98**(6), 1145–1152 (2015)
39. Nishida, T., Mizuki, T., Sone, H.: Securely computing the three-input majority function with eight cards. In: Proc. of 2nd International Conference on Theory and Practice of Natural Computing(TPNC 2013), LNCS Vol. 8273. pp. 193–204 (2013)
40. Nishimura, A., Hayashi, Y.i., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101**(9), 1494–1502 (2018)
41. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**(2), 361–371 (2018)
42. Ono, H., Manabe, Y.: Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In: Proc. of 13th Asia Joint Conference on Information Security(AsiaJCIS 2018). pp. 23–28 (2018)

43. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Generation Computing **39**(1), 19–40 (2021)
44. Ono, H., Manabe, Y.: Minimum round card-based cryptographic protocols using private operations. Cryptography **5**(3) (2021)
45. Ono, T., Nakai, T., Watanabe, Y., Iwamoto, M.: An efficient card-based protocol of any boolean circuit using private operations. In: Proc. of Computer Security Symposium. pp. 72–77 (2022), (In Japanese)
46. Ruangwises, S., Itoh, T.: And protocols using only uniform shuffles. In: Proc. of 14th International Computer Science Symposium in Russia(CSR 2019), LNCS Vol. 11532. pp. 349–358 (2019)
47. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. Theoretical Computer Science **887**, 99–110 (2021)
48. Saito, T., Miyahara, D., Abe, Y., Mizuki, T., Shizuya, H.: How to implement a non-uniform or non-closed shuffle. In: Proc. of 9th International Conference on the Theory and Practice of Natural Computing(TPNC 2020), LNCS Vol. 12494. pp. 107–118. Springer (2020)
49. Shinagawa, K., Miyamoto, K.: Automorphism shuffles for graphs and hypergraphs and its applications. arXiv preprint arXiv:2205.04774 (2022)
50. Shinagawa, K., Mizuki, T.: The six-card trick:secure computation of three-input equality. In: Proc. of 21st International Conference on Information Security and Cryptology (ICISC 2018), LNCS Vol. 11396. pp. 123–131 (2018)
51. Shinagawa, K., Mizuki, T.: Secure computation of any boolean function based on any deck of cards. In: Proc. of 13th International Workshop on Frontiers in Algorithmics (FAW 2019), LNCS Vol. 11458. pp. 63–75. Springer (2019)
52. 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)
53. Shinoda, Y., Miyahara, D., Shinagawa, K., Mizuki, T., Sone, H.: Card-based covert lottery. In: Proc. of 13th International Conference on Information Technology and Communications Security(SecITC 2020), LNCS Vol. 12596. pp. 257–270. Springer (2020)
54. Takashima, K., Abe, Y., Sasaki, T., Miyahara, D., Shinagawa, K., Mizuki, T., Sone, H.: Card-based protocols for secure ranking computations. Theoretical Computer Science **845**, 122–135 (2020)
55. Toyoda, K., Miyahara, D., Mizuki, T.: Another use of the five-card trick: Card-minimal secure three-input majority function evaluation. In: Proc. of 22nd International Conference on Cryptology in India (INDOCRYPT 2021), LNCS Vol. 13143. pp. 536–555. Springer (2021)
56. Toyoda, K., Miyahara, D., Mizuki, T., Sone, H.: Six-card finite-runtime xor protocol with only random cut. In: Proc. of the 7th ACM Workshop on ASIA Public-Key Cryptography. pp. 2–8 (2020)
57. Tozawa, K., Morita, H., Mizuki, T.: Single-shuffle card-based protocol with eight cards per gate. In: Proc. of 20th International Conference on Unconventional Computation and Natural Computation (UCNC 2023), LNCS vol. 14003. pp. 171–185. Springer (2023)
58. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y.i., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. International Journal of Information Security **19**(4), 445–452 (2020)
59. Watanabe, Y., Kuroki, Y., Suzuki, S., Koga, Y., Iwamoto, M., Ohta, K.: Card-based majority voting protocols with three inputs using three cards. In: Proc. of 2018 International Symposium on Information Theory and Its Applications (ISITA). pp. 218–222. IEEE (2018)

60. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (SFCS 1986). pp. 162–167. IEEE (1986)