General Conversion Scheme of Card-based Protocols for Two-colored Cards to Updown Cards

Takumi Sakurai
Nagoya University
Nagoya, Japan
email: sakurai.takumi.j1@s.mail.nagoya-u.ac.jp

Yuichi Kaji
Nagoya University
Nagoya, Japan
email: kaji.yuichi.a0@f.mail.nagoya-u.ac.jp

***Abstract*—Besides the majorly investigated two-colored cards, there are studies of card-based protocols that use updown cards printed with rotationally asymmetric symbols. A card-based protocol for updown cards is advantageous in making the protocol simple and efficient, but not so much effort has been made to develop updown card protocols, and not so much is known about the relation between protocols for two-colored cards and updown cards. In fact, the number of cards for computing an arbitrary function is not known. This study discusses the sufficient condition of two-colored cards protocols under which the protocol can be converted for updown cards, and describes the actual conversion procedure. With the conversion, it is clarified that there are updown card protocols that compute an arbitrary Boolean function with three additional cards, and protocols that compute a symmetric Boolean function with only one additional card.**

Keywords-Card-based cryptography; Secure multi-party computation; Updown cards.

I. INTRODUCTION

Card-based cryptography is a technique for secure multi-party computation using physical cards [5][6][7][8][10][12]. Participants in a computation encode their input bits by using cards with symbols, such as those on playing cards. The cards are placed with their faces down so that the input bits are kept secret. The cards are shuffled, permuted, and flipped according to a specific rule. At the end, the participants learn only the information corresponding to the computation's output from the cards.

Card-based cryptography enables secure multi-party computation without specialized knowledge or equipment, such as a computer. Therefore, card-based cryptography is regarded as an appealing material for the lectures of security and zero-knowledge proof for puzzles [1][13]. On the other hand, the procedure should be as simple as possible because all operations must be performed manually by human operators. Therefore, reducing the number of cards is an important issue, and research has been conducted on the minimum number of cards that are required to compute meaningful functions and to solve other problems.

In card-based cryptography, we often consider using two-colored cards (hereafter called *TC cards*) that are printed with either "♣" or "♡" on their front and "?" on their back. A single bit is encoded by a pair of cards placed so that

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

A *commitment* is a pair of face-down cards that encodes a single bit according to the above encoding rule. In this paper, the commitment to $x \in \{0,1\}$ is denoted as

$$\underset{x}{\boxed{?}\boxed{?}}$$

which represents nobody can see the faces of the cards. If a bit value x is represented as a commitment, then its negation $\bar{x}$ is easily obtained by swapping the places of the two cards of the commitment.

We call a *protocol* the entire procedure of taking an input in the form of a commitment, performing operations, such as permutation, flip, and shuffle, and finally determining the output from a sequence of cards. A protocol for TC cards in this paper follows the Mizuki-Shizuya model [4], which allows only these operations: *permutation* that rearranges the position of the cards, *flip* that faces down or up the cards, and *shuffle* that secretly and probabilistically applies the permutation.

Protocols in which the output is obtained in the form of a commitment are called *committed-format protocols*. Committed-format protocols are important because they allow us to construct complicated protocols from simpler ones. For example, if we have committed-format protocols for basic logical operations, such as AND, OR, and NOT and for copying a Boolean value, then we can construct a committed-format protocol for an arbitrary Boolean function. The result of the function is obtained by opening the commitment of the final result, and no information leaks out about the inputs and the intermediate values that are used in the computation.

In addition to commonly studied TC cards, there is a direction of studies of card-based cryptography that uses cards of a single type. Such cards are called updown cards (hereafter called *UD cards*) and assumed to have "↑" on the front and "(blank)" on the back [5]. Let

$$\boxed{\downarrow} = 0, \boxed{\uparrow} = 1$$

be the encoding of a single bit on these cards. For UD cards, a commitment is defined as a single card representing a single bit but its face down, and protocols are realized by performing operations, such as the permutation, flip, shuffle (that applies the permutations and rotations), and *rotation* that rotates a card by 180 degrees and reverses the upside and the downside of a card. The protocol for the logical NOT is realized by simply rotating the commitment (a single card) of the input.

Generally speaking, protocols for UD cards require fewer cards than protocols for TC cards. This is especially important because cards are operated by a human. Besides the advantage

in efficiency, a smaller number of cards is favorable in discussing the computational capabilities of the card-based protocol. Since two TC cards are expressed by one UD card, the number of possible combinations of cards can be reduced, though the operation on UD cards are more complex than those on TC cards. However, not so many investigations have been made for protocols with UD cards, while TC cards are eagerly studied. For example, there are TC protocols that can safely compute arbitrary n-variable Boolean functions with $2n + 6$ cards ($2n$ cards for the commitments of n bits and six additional cards for "working memory"), and symmetric functions with $2n + 2$ cards [7]. On the other hand, no such general protocol is known for UD cards. To promote the study of UD protocols, it is convenient if we can transform a TC protocol to a corresponding UD protocol. However, it is likely that not all TC protocols can be transformed to UD protocols.

In this study, we illustrate a general method for converting a TC protocol that satisfies certain constraints into an UD protocol, where the latter uses half the number of cards of the former. It is demonstrated that the TC protocols in [7] fulfill the constraint described above. Consequently, our conversion method brings UD protocols that compute arbitrary n-variable Boolean function with $n + 3$ cards and any n-variable symmetric function with $n + 1$ cards. To avoid possible misunderstanding, we remark that this study is a compilation of many known results and protocols, rather than a proposal of a novel protocol that is based on a new idea. The compilation, however, indicates a strong relationship between TC protocols and UD protocols, which has not been recognized explicitly.

This paper is organized as follows. In Section II, we introduce AND and XOR protocols with TC cards. In Section III, we show the basic idea of converting TC protocols to UD protocols. In Section IV, we introduce AND and XOR protocols with UD cards, discussing equivalency to TC cards. In Section V, using converting constraints, we show the example of convertible TC protocols. In Section VI, this paper is concluded.

II. TC CARDS PROTOCOLS

This section reviews committed-format TC protocols that realize computing AND and XOR of input bits.

A. TC Cards AND Protocol with Six Cards

For a pair of bits (x, y) and a bit value i, we define the functions "get" and "shift" as

$$\mathrm{get}^i(x,y) = \begin{cases} x & (i=0), \\ y & (i=1), \end{cases}$$

$$\mathrm{shift}^i(x,y) = \begin{cases} (x,y) & (i=0), \\ (y,x) & (i=1). \end{cases}$$

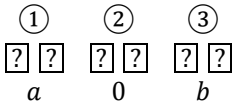
For an input bit $a, b \in \{0,1\}$, the logical conjunction $a \wedge b$ can be written as

$$a \wedge b = \mathrm{get}^a(0,b) = \mathrm{get}^{a\oplus r}\big(\mathrm{shift}^r(0,b)\big)$$

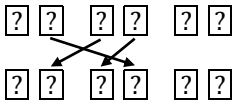
with any bit $r \in \{0,1\}$ [8]. This principle describes committed-format protocols for AND computation.

Mizuki and Sone's AND protocol is realized with six TC cards, including four cards for the commitments of two input bits [6]. The protocol consists of the following five steps. (Hereafter, card positions are given address numbers from left to right for clarity.)

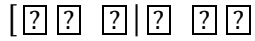
1. The cards are arranged as follows, with input $a, b \in \{0,1\}$ as the commitments placed at positions 1 and 3. The additional two cards encode 0 and are placed face down at position 2.



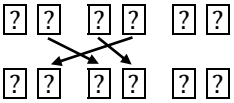
2. Permute the cards as follows.



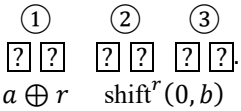
3. Shuffle the cards in such a way that the left and right halves of the cards are each bundled and randomly swapped, which is called a *random bisection cut* and denoted by $[\cdot \mid \cdot]$. In practice, we can use card sleeves or clips and throw them to realize this shuffle.



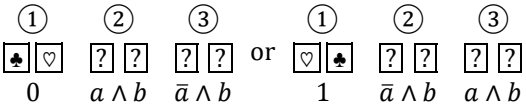
4. Permute the cards as follows.



If the random bisection cut in Step 3 does not change the order of the cards, then this permutation cancels the permutation in Step 2, resulting the commitments to a, 0, and b placed in this order from the left. If the random bisection cut in the previous Step 3 changes the order of the cards, then this permutation brings the commitment to the negation of a, which is written as $a \oplus 1$, in position 1. We can also confirm that the commitment to b moves to position 2 and the commitment to 0 moves to position 3, and hence they swap their positions in the original order. Summarizing the two cases, we have the commitments to $a \oplus r$ and $\mathrm{shift}^r(0,b)$ after the permutation of this step, where $r = 0$ and 1 represent the two different cases of the random bisection cut in Step 3. The results is therefore illustrated as



5. Flip the cards at position 1 and open the commitment to $a \oplus r$ over. If $a \oplus r = 0$, then the cards at positions 2 and 3 are the commitments to $a \wedge b$ and $\bar{a} \wedge b$, respectively. If $a \oplus r = 1$, then the cards at positions 2 and 3 are the commitments to $\bar{a} \wedge b$ and $a \wedge b$, respectively. Therefore, the commitment to $a \wedge b$ is obtained at position 2 if $a \oplus r = 0$, and at 3 if $a \oplus r = 1$. Remark that opening $a \oplus r$ does not reveal the value of a because r is unknown.

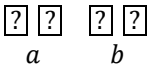


Given the commitments to a, 0, and b, this protocol computes $a \wedge b = \mathrm{get}^a(0, b)$. If the commitments to $x, y, z \in \{0,1\}$ are provided instead of a, 0, and b, then this same protocol computes $\mathrm{get}^x(y, z)$.

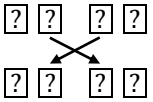
B. TC Cards XOR Protocol with Four Cards

Here, we introduce committed-format protocols that compute exclusive-or $a \oplus b$ for bits $a, b \in \{0,1\}$. Mizuki and Sone's XOR protocol is realized with only four input TC cards, thus using no additional cards [6].

1. The cards are arranged as follows, with input $a, b \in \{0,1\}$ as the commitments.

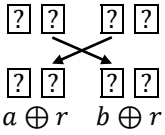


2. Permute the cards as follows.

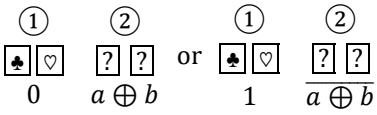


3. Make a random bisection cut.

4. Permute the cards as follows. The commitment after this permutation is represented by a bit r that represents the result of the bisection cut in the previous step.



5. Flip the cards at position 1. This does not reveal the value of a because r has been chosen randomly. If the cards at position 1 were the commitment to 0, then the cards at position 2 are the commitment to $a \oplus b$. If the cards at position 1 were the commitment to 1, then the cards at position 2 are the commitment to the complement to $a \oplus b$. In this latter case, swap the cards at position 2, and we obtain the commitment to $a \oplus b$.



The protocol extracts one card from each commitment, combines them into one, makes a random bisection cut and returns them to their original placement (Steps 2-4). This operation adds a common random bit r to the two input bits. This principle can be easily extended to the case where the input is more than two bits. For example, let $a, b, c \in \{0,1\}$ be the input bits and arrange the cards to represent the commitments of the three input bits. All commitments can be shuffled by dividing them into left and right bundles of cards to give them the random bit as well. Finally, the value of $a \oplus r$ is checked, yielding the commitments to $a \oplus b$ and $a \oplus c$. If the value of b and c are 0, then $a \oplus b = a$ and $a \oplus c = a$. Therefore, two copies of the commitments to a can be obtained while keeping the value of a secret.

III. Conversion to UD Protocols

Given an arbitrary UD protocol, it is clearly possible to construct a TC card protocol that is computationally equivalent to the given UD protocol [11]. Specifically, the commitment represented by one UD card is replaced by a pair of two TC cards representing the same commitment. Though there are some differences between the operations on the UD cards and those on the TC cards, a permutation on the UD cards can be converted to a permutation on the commitment of the TC cards, a rotate operation of a UD card can be converted to a swap operation of the two cards that constitute a commitment in the TC cards, and so on.

If we were able to reverse the conversion from the UD protocol to the TC protocol, we could obtain an UD protocol that can be executed based on the same principles as the TC protocol with exactly half the number of cards. It is always possible to replace one card in the UD protocol with two cards in the TC protocol, while the converse is not always true. For example, in the TC protocol, it is possible for the shuffle or other operations to separate the two cards of a commitment that represent one bit. Obviously, such operations cannot be simulated with UD cards. It is strongly conjectured that the TC protocols can be converted to the UD protocol only if the TC protocols satisfy certain conditions.

We name a TC protocol *commitment-preserving* if the protocol uses only the following four types of operations.

1. Permutation that does not destroy the commitment
2. Shuffle that does not destroy the commitment
3. Flip operation that turns a card face up or face down
4. NOT operation that exchanges the cards of the commitment

Note that destroying the commitment means separating the two cards that are used to constitute a commitment of a single bit. For example, all the permutations in previous section's protocols destroy the commitments, and reversing the order of even cards is one of the operations that do not destroy the commitments, tracing with UD cards. The permutation 1. Prohibits this kind of permutations, and the permutations that fulfill this condition 1. can be converted to a permutation and a rotation in the UD protocol. The shuffle of 2. can be converted to a shuffle that applies a permutation and a rotation in the UD protocol. The flip of 3. can face up one of two cards that constitute a commitment of a TC protocol. Another card of the commitment may not be opened, but the operation of opening a single card completely reveals the value of the commitment. Therefore, the flip of 3. brings the same effect as flipping two cards of a commitment of a TC protocol, which is simulated by a simple flip of a UD card representing the corresponding commitment. The NOT operation in 4. can be converted to a rotation of the cards in the UD protocol.

For TC protocols, we can make use of KWH-tree developed by Koch, Walzer, and Härtel for verifying the security and the correctness [3]. However, the security and correctness of the converted UD protocol depend on the original TC protocol because this conversion only traces the original.

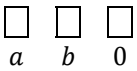
IV. UD CARDS PROTOCOLS

A commitment-preserving protocol with TC cards can be converted to a protocol with half the number of UD cards, but many TC protocols do not satisfy those conditions. We can see that AND and XOR protocols, illustrated in Section II, break the commitments by permutations and shuffles, and they cannot be converted to UD protocols in a naive manner. Fortunately, for the two protocols of AND and XOR, there are UD protocols that realize the same computation as the TC protocols with half the number of cards [5][12]. These UD protocols can lead to commitment-preserving AND and XOR protocols with TC cards. This implies that there are non-commitment-preserving protocols that perform equivalent computations of commitment-preserving protocols. Therefore, we add these non-commitment-preserving protocols to the convertible operations. This section briefly introduces the UD cards AND and XOR protocols in [5][12].

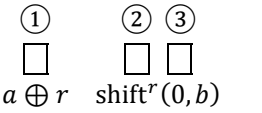
A. UD cards AND Protocols with Three Cards

Mizuki and Shizuya's AND protocol [5] with UD cards is designed based on the same principle as the TC protocol in the previous section. The only difference is that a commitment consists of a single card, and we need just three cards to complete the computation.

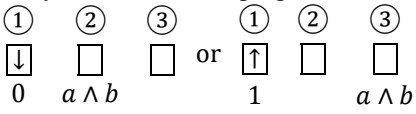
1. The cards are arranged as follows, with input $a, b \in \{0,1\}$ as the commitments. The additional cards are also placed face down.



2. Repeat a shuffle operation for an arbitrary time, where one shuffle operation consists of a rotation of the card at position 1 and a swapping of the cards at position 2 and 3. The commitments after shuffle are represented by a random bit $r \in \{0,1\}$ corresponding to this shuffle as follows.



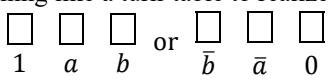
3. Flip the cards at position 1. The output position can be determined by $a \oplus r$ while keeping the value of a secret.



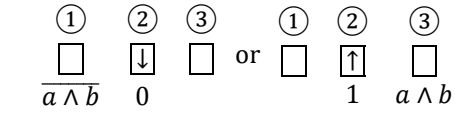
Since the shuffle in Step 2 is difficult to perform, there is another protocol that is easy to implement [12].

1. The cards are arranged as follows, with input $a, b \in \{0,1\}$ as the commitments. The additional cards are also placed face down.

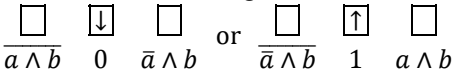
2. Shuffle the entire row of cards in a rotation, which is called a *tornado shuffle*. The value of each commitment is either of the following two results. In practice, we can use a device something like a turn-table to realize this suffle.



3. Flip the card at position 2, which was a commitment to either of a or $\bar{a}$. If the opened value is 0, then take the card at position 1 and rotate it. If the opened value is 1, then take the card at position 3.



The TC cards AND protocol in Section II creates commitments $a \wedge b$ and $\bar{a} \wedge b$ with six cards for the input bits $a, b \in \{0,1\}$. The UD cards AND protocol in this section, which uses the less realistic shuffle, is based on the same principle as the TC cards AND protocol, and it is obviously equivalent to the TC cards AND protocol. For the second protocol that uses a tornado shuffle in this section, we can confirm that the cards are arranged as



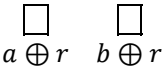
after the protocol. Therefore, the equivalent computation to the TC protocol is realized. It can be said that the commitments to $a \wedge b$ and $\bar{a} \wedge b$ can be obtained with three cards together with the input for input bits $a, b \in \{0,1\}$. It can also be confirmed that $\text{get}^x(y, z)$ can be computed for both UD protocols. Therefore, the AND protocols with TC cards in Section II can be added to the convertible operations of commitment-preserving protocols in Section III.

B. UD cards XOR Protocol with Two Cards

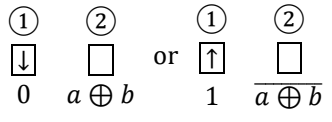
Mizuki and Shizuya's XOR protocol with UD cards can be constructed according to the same principle as the TC cards case [5]. The protocol uses only two input cards, and no additional card is required.

1. The cards are arranged as follows, with input $a, b \in \{0,1\}$ as the commitments.

2. The cards are shuffled by bundling them and rotating them. The commitment after this shuffle is represented by a random bit $r \in \{0,1\}$ as follows.



3. Flip the cards at position 1. If it is 0, then the card at position 2 is the commitment to $a \oplus b$. If the opened card is 1, then the card at position 2 is the commitment to $\overline{a \oplus b}$. In the latter case, rotate the card at position 2 and we obtain the commitment to $a \oplus b$.



This protocol can be extended for three or more input bits and used to obtain multiple copies of an input commitment as in the TC cards case.

The UD cards XOR protocol in this section follows the same principle as the TC protocol. Then, it clearly achieves an equivalent computation with half the number of cards. Therefore, the XOR protocols with TC cards in Section II can be added to the convertible operations of commitment-preserving protocols in Section III.

V. Example of Applications of the Conversion Method

The conversion method described in the previous section can be applied to any commitment-preserving protocol. In this section, we demonstrate that TC card protocols for computing arbitrary Boolean function and symmetric functions are commitment-preserving, namely, they use only the operations that are listed in the previous section. This means that the TC protocols can be converted to protocols for UD cards, which brings efficient protocols for computing arbitrary Boolean functions and symmetric functions by using UD cards.

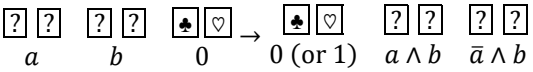
Note that all the permutations of the TC protocols discussed in this section are the operations of commitment-conserving protocols.

A. Preparation

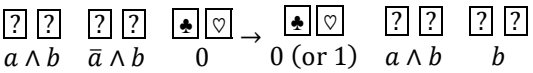
To efficiently compute the arbitrary Boolean function and the symmetric functions, an *improved AND* protocol and a half adder protocol have been developed [7].

The improved AND protocol produces two commitments, one for $a \wedge b$ and one for b, for given three commitments to a, b, and 0. The protocol is composed of the AND and XOR protocols in Section II.

1. For input bits $a, b \in \{0,1\}$, use the AND protocol to the commitments to a, b, and 0, which yields the commitments to $a \wedge b$ and $\bar{a} \wedge b$.



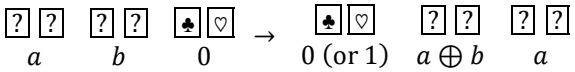
2. Use the XOR protocol to the commitments to $a \wedge b$, $\bar{a} \wedge b$ and 0, which yields the commitments to $(a \wedge b) \oplus (\bar{a} \wedge b) = b$ and $(a \wedge b) \oplus 0 = a \wedge b$.



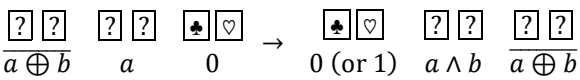
Notice that the two cards that were used as the commitment to 0 (or 1) after Step 1 can be used to encode 0 in this second step, and hence the improved AND protocol is realized with six cards. Notice also that the AND and XOR protocols are used here. Thus, the improved AND protocol can be converted to an UD protocol.

The half-adder protocol is composed of the XOR, NOT, and improved AND protocols.

1. For input bits $a, b \in \{0,1\}$, use the XOR protocol to the commitments to a, b, and 0, which yields the commitments to $a \oplus b$ and a.



2. The NOT protocol yields the commitment to $\overline{a \oplus b}$.
3. The improved AND protocol yields the commitments to $a \wedge (\overline{a \oplus b}) = a \wedge b$ and $\overline{a \oplus b}$, where the two opened cards of Step 1 is reused to realize the improved AND protocol.



4. The NOT protocol yields the commitments to $a \oplus b$. We now have the commitments to $a \wedge b$ and $a \oplus b$.

Notice that the protocol uses six cards, and that the half adder protocol is also the commitment-preserving protocol and can be converted to an UD protocol.

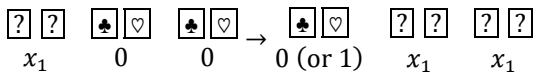
B. Arbitrary n-variable Boolean Function

A protocol for computing arbitrary n-variable Boolean functions with $2n + 6$ two-color cards is proposed by Nishida et al. [7]. The Boolean function, say $f(x_1, x_2, \dots, x_n)$, is expressed as

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= x_1 x_2 \cdots x_n f(1,1,\dots,1) \\ &\oplus \overline{x_1} x_2 \cdots x_n f(0,1,\dots,1) \\ &\vdots \\ &\oplus \overline{x_1}\,\overline{x_2} \cdots \overline{x_n} f(0,0,\dots,0) \end{aligned}$$

from the Shannon expansion [9]. Remark that the values of the function f in the right-hand side are either of 0 or 1. Terms with $f(\cdot) = 0$ dismisses from the expression while terms with $f(\cdot) = 1$ brings the AND value of literals of $x_1, \dots, x_n$. The entire function is therefore obtained as the XOR of the AND values of the literals that make the value of $f(\cdot) = 1$. The protocol for computing $f(x_1, x_2, \dots, x_n)$ is described as follows. Use $2n$ cards to represent $x_1, x_2, \dots, x_n$ as commitments, and additional six cards to represent three commitments to 0. Among the three commitments to 0, two are used as a working memory to compute an AND value of literals of $x_1, x_2, \dots, x_n$. The remaining one commitment to 0 is used to record an intermediate value of the XOR of AND values. In the following description, only relevant commitments are shown in the explanation.

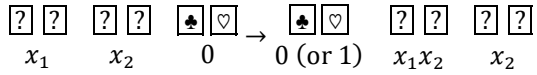
1. Assume that $f(1,1,\dots,1) = 1$. In this case, for input bits $x_1, x_2, \dots, x_n \in \{0,1\}$, the commitment to $x_1 x_2 \cdots x_n$ is created as follows while keeping the input commitments unchanged.
   a. Perform the XOR protocol for the commitments to x_1 and the two commitments of the working memory. The protocol yields the two commitments to x_1 and two free cards.



   Open the two free cards and reformat the cards as a commitment to 0. This brings $2n$ cards for the commitments to $x_1, x_2, \dots, x_n$, a pair of cards for the commitment to a copy of x_1, a pair of cards for the commitment to 0 for the working memory and a pair of

cards (0 at this moment) to record the intermediate value.

b. The improved AND protocol for the commitments to x_1, x_2 and 0 yields the commitments to x_1x_2, x_2 and two free cards.



Similarly to the previous step, we have a commitment to x_1x_2, to 0 for a working memory, and 0 for an intermediate value.

c. Repeatedly use the improved AND protocols like b. operations, which yields the commitment to $x_1x_2 \cdots x_n$. These whole operations use four additional cards. Regard this commitment as an intermediate value, and we still have four free cards for two commitments to 0 for working memory.

2. Similarly, create a commitment to $\overline{x_1}x_2 \cdots x_n$ if $f(0,1,\dots,1) = 1$. This operation uses four cards for the working memory.
3. The XOR protocol yields the commitment to $x_1x_2 \cdots x_n \oplus \overline{x_1}x_2 \cdots x_n$, which is regarded as an intermediate value.
4. In the same way, create the commitment to the term that makes the value of $f(\cdot) = 1$ while the input commitments are maintained, and the XOR protocol is repeated. Finally, the commitment to $f(x_1, x_2, \dots, x_n)$ is obtained at the commitment of the result of the XOR protocols.

From the above, the protocol with $2n + 6$ TC cards for any n-variable Boolean function is commitment-preserving and can thus be converted to a protocol with $n + 3$ UD cards.

C. Arbitrary n-variable Symmetric Boolean Function

A Boolean function is said to be symmetric if its function value is invariant to the permutation of inputs, that is, $f(x_1, \dots, x_n) = f(x_{\pi(1)}, \dots, x_{\pi(n)})$ holds for an arbitrary permutation π. For a symmetric Boolean function, we have a TC protocol that uses only two extra cards rather than six [7][8].

The value of a symmetric Boolean function is irrelevant to the order of inputs. This implies that the function value is determined by the number of 1's contained in the input. Consequently, a symmetric Boolean function f is characterized as

$$f(x_1, \dots, x_n) = g\left(\sum_{i=1}^{n} x_i\right)$$

where g is a mapping from $\{0, 1, \dots, n\}$ to $\{0, 1\}$. Such a mapping g is further characterized by $X \subseteq \{0, 1, \dots, n\}$ where X is defined as $X = \{s | 0 \le s \le n, g(s) = 1\}$. The number of subsets of $\{0, 1, \dots, n\}$ is 2^{n+1}, and it is understood that the number of symmetric Boolean functions with n inputs is 2^{n+1}. However, it is not necessary to consider all symmetric Boolean functions, since the negation of commitments can be easily performed. We will use NPN equivalence classes based on input-output negation and input reordering for the discussion that follows [9].

For $n \le 2$, symmetric functions are obviously computed by a combination of the AND, XOR, and NOT protocols. These protocols use at most two additional cards.

For $n = 3$, symmetric function f for input bits $a, b, c \in \{0,1\}$ is characterized by

$$S_X^3(a, b, c) = \begin{cases} 1 & (a + b + c \in X) \\ 0 & (\text{otherwise}) \end{cases}.$$

where $X \subseteq \{0,1,2,3\}$. From the NPN equivalence class, the symmetric functions are limited to six patterns as follows [8][9]. For example, $S_{\{0,1,2,3\}}^3$ can be computed by using the negation output of $S_{\emptyset}^3$.

- $S_{\emptyset}^3(a, b, c) = 0$
- $S_{\{3\}}^3(a, b, c) = a \wedge b \wedge c$
- $S_{\{1,2\}}^3(a, b, c) = \text{get}^{a \oplus b}(a \oplus c, 1)$
- $S_{\{1,3\}}^3(a, b, c) = a \oplus b \oplus c$
- $S_{\{2,3\}}^3(a, b, c) = \text{get}^{a \oplus b}(a, c)$
- $S_{\{0,2,3\}}^3(a, b, c) = \text{get}^{a \oplus b \oplus c}(1, a \wedge c)$

Each of them is constructed by XOR and AND protocols in TC cards. These patterns use at most two additional cards.

When $n = 4$, the value of $\sum_{i=1}^{4} x_i$ can be encoded in terms of $2(\lfloor \log_2 4 \rfloor + 1) = 6$ cards by using the half adder protocol repeatedly. It uses two additional cards and yields four free cards. The function g is defined as

$$g\left(\sum_{i=1}^{4} x_i\right) = \begin{cases} g(4) & (s_1 = 1) \\ g'(s_2, s_3) & (s_1 = 0) \end{cases}$$

where (s_1, s_2, s_3) is the bits of $\sum_{i=1}^{4} x_i$. The function $g'(s_2, s_3)$ is a Boolean function with the lower two bits of $\sum_{i=1}^{4} x_i$ as inputs. This function can be computed with two additional cards (using free cards above). Then, it can be expressed as $g(\sum_{i=1}^{4} x_i) = \text{get}^{s_1}(g'(s_2, s_3), g(4))$, which is computed by the AND protocol. Thus, the whole protocol uses two additional cards.

When $n \ge 5$, we define a function g similarly. By the half adder protocol, the value of $\sum_{i=1}^{n} x_i$ can be encoded by $2(\lfloor \log_2 n \rfloor + 1)$ cards. Since the total number of cards used in the protocol is $2n + 2$, the number of cards not used to represent the value of $\sum_{i=1}^{n} x_i$ is $(2n + 2) - 2(\lfloor \log_2 n \rfloor + 1) = 2(n - \lfloor \log_2 n \rfloor)$. From $n \ge 5$, $2(n - \lfloor \log_2 n \rfloor) \ge 6$, which is more than 6 cards, can be used freely. Then, by using these six cards as additional cards, g can be computed using the protocol of this section.

From the above, the protocol with $2n + 2$ TC cards for any n-variable symmetric Boolean function is commitment-preserving and can thus be converted to a protocol with $n + 1$ UD cards.

VI. Conclusion and Future Work

In this paper, we proposed a general method to convert a TC protocol to an UD protocol by restricting the operations of the TC protocol. It is also shown that the TC protocols, which allow multi-party computation of arbitrary n-variable Boolean function with $2n + 6$ cards and n-variable symmetric

functions with $2n + 2$ cards, can be converted to an UD protocol with half the number of cards.

We focused on the fact that the AND and XOR protocols on the UD cards can perform calculations equivalent to those on the TC protocols. Since UD protocols can be converted to TC protocols [11], there are the AND and XOR TC protocols not destroying commitments, which ignore the ease of operations. On the other hand, we did not mention anything about the possibility of converting other TC protocols that are not commitment-preserving. In fact, a TC protocol [10] that computes symmetric functions of 8 or more variables with only input cards, i.e., with no additional cards, has been proposed, but it has also been shown that AND computation is not possible on UD cards without additional cards [2]. Thus, the calculation cannot be converted to an UD protocol unless it has the commitment-preserving protocol. This gap between the two classes is being clarified. In addition, the minimum number of cards required for Boolean functions and symmetric functions as a computational capability of the TC protocols is still unresolved. Such studies are expected to contribute to the clear computational capability of TC protocols in the future.

REFERENCES

[1] R. Gradwohl, M. Naor, B. Pinkas, and G. N. Rothblum, "Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles," Internatinal Conference on Fun with Algorithms, pp. 166–182, 2007.

[2] S. Iino, Y. Li, K. Sakiyama, and D. Miyahara, "On the impossibility of n-card and protocols," 42nd Symposium on Cryptography and Information Security, 4D2-3, 2025 (in Japanese).

[3] A. Koch, S. Walzer, and K. Härtel, "Card-based cryptographic protocols using a minimal number of cards," International Conference on the Theory and Application of Cryptology and Information Security, pp. 783–807, 2015.

[4] T. Mizuki and H. Shizuya, "A formalization of card-based cryptographic protocols via abstract machine," International Journal of Information Security, 13, pp. 15–23, 2014.

[5] T. Mizuki and H. Shizuya, "Practical card-based cryptography," International Conference on Fun with Algorithms, pp. 313–324, 2014.

[6] T. Mizuki and H. Sone, "Six-card secure and and four-card secure xor," International Workshop on Frontiers in Algorithmics, pp. 358–369, 2009.

[7] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone, "Card-based protocols for any boolean function," 12th Annual Conference on Theory and Applications of Models of Computation, pp. 110–121, 2015.

[8] T. Nishida, T. Mizuki, and H. Sone, "Securely computing the three-input majority function with eight cards," Second International Conference on Theory and Practice of Natural Computing, pp. 193–204, 2013.

[9] T. Sasao, "Switching theory for logic synthesis," Kluwer Academic Publishers, 1999.

[10] H. Shikata, K. Toyoda, D. Miyahara, and T. Mizuki, "Card-minimal protocols for symmetric boolean functions of more than seven inputs," International Colloquium on Theoretical Aspect of Computing, pp. 388–406, 2022 (in Japanese).

[11] K. Shinagawa, "Card types and encodings of card-based cryptography," presentation slide, Organizing Mathematical Unsolved and New Problems in Card-based Cryptography through Industry-academia Collaboration. [Online]. Available from: https://joint.imi.kyushu-u.ac.jp/wp-content/uploads/2023/06/IMI_shinagawa.pdf (accessed 2025-07-03) (in Japanese)

[12] K. Shinagawa, K. Nuida, T. Nishide, G. Hanaoka, and E. Okamoto, "Committed AND protocol using three cards with more handy shuffle," 2016 International Symposium on Information Theory and Its Applications, pp. 700–702, 2016.

[13] K. Shinagawa, "A report on a lecture for elementary and junior high school using card-based cryptography," 39th Symposium on Cryptography and Information Security, 2F4-4, 2022 (in Japanese).