Card-based Cryptographic Protocols with a Standard Deck of Cards Using Private Operations

Yoshifumi Manabe[1*] and Hibiki Ono[1]

[1*]School of Informatics, Kogakuin University, 1-24-2, Nishishinjuku, Shinjuku, 1638677, Tokyo, Japan.

*Corresponding author(s). E-mail(s): manabe@cc.kogakuin.ac.jp;

Abstract

This paper shows new kinds of card-based cryptographic protocols with a standard deck of cards using private operations. They are multi-party secure computations executed by multiple players without computers. Most card-based cryptographic protocols use a special deck of cards that consists of many cards with two kinds of marks. Though these protocols are simple and efficient, the users need to prepare such special cards. Few protocols were shown that use a standard deck of playing cards. Though the protocols with a standard deck of cards can be easily executed in our daily lives, the numbers of cards used by these protocols are larger than the ones that use the special deck of cards. This paper shows logical AND, logical XOR, and copy protocols for a standard deck of cards that use the minimum number of cards. Any Boolean functions can be computed with a combination of the above protocols. The new protocols use private operations that are executed by a player at a place where the other players cannot see. The results show the effectiveness of private operations in card-based cryptographic protocols. This paper also shows protocols that use private input operations. When each player privately inputs his/her secret value, this type of protocol is used. Last, we show asymmetric card protocols to further reduce the number of cards.

Keywords: Multi-party secure computation, card-based cryptographic protocols, private operations, logical computations, copy, playing cards

1 Introduction

1.1 Overview of card-based cryptographic protocols

Card-based cryptographic protocols [1–3] 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 [4, 5] to accelerate the social implementation of advanced cryptography [6]. den Boer [7] 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 [8–15] and specific computations such as a specific class of Boolean functions [16–34], universal computation such as Turing machines [35, 36], millionaires' problem [37–40], voting [41–48], random permutation [49–52], garbled circuits [15, 53–55], grouping [56], ranking [57], lottery [58], and so on.

This paper considers computations of logical AND and logical XOR functions and copy operations since any Boolean function can be realized with a combination of these computations.

Note that in this paper, all players are assumed to be semi-honest. Few works are done for the case when some players are malicious or make mistakes [59–64].

1.2 Standard deck of cards

Most of the above works are based on a two-color card model. In the two-color card model, there are two kinds of cards, $\boxed{\clubsuit}$ 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 in the back of a given card of $\boxed{?}$. Though the model is simple, such special cards are not available in our daily lives.

To solve the problem, card-based cryptographic protocols using a standard deck of playing cards were shown [14, 26, 38, 64–69]. Playing cards are available at many houses and are easy to buy. The standard deck of playing cards is also used for zero-knowledge proof of puzzle solutions [70, 71]. This paper discusses protocols to calculate logical functions. Niemi and Renvall first showed protocols that use a standard deck of playing cards [65]. They showed logical XOR, logical AND, and copy protocols since any Boolean functions can be realized by a combination of these protocols. Their protocols are 'Las Vegas' type protocols, that is, the execution times of the protocols are not limited. The protocols are expected to terminate within a finite time, but if the sequence of the random numbers is bad, the protocols do not terminate forever. Mizuki showed fixed time logical XOR, logical AND, and copy protocols [66]. Though the number of cards used by the XOR protocol is the minimum, the ones used by the logical AND and copy protocols are not the minimum. Koch et al. showed a four-card 'Las Vegas' type AND protocol and it is impossible to obtain a four-card finite time protocol with the model without private operations [67]. Koyama et al. showed a three-input 'Las Vegas' type AND protocol with the minimum number of cards [68]. Koyama et al. showed an efficient 'Las Vegas' type copy protocol [26]. Shinagawa and Mizuki showed protocols to compute any n-variable function using a standard deck of

playing cards and a deck of UNO[1] cards [14]. Miyahara et al. showed a protocol that solves Yao's millionares' problem using a standard deck of playing cards [38]. Miyahara and Mizuki showed new protocols that use a special primitive that opens the suit of a playing card [69]. This paper discusses protocols that publically or privately open the cards.

1.3 Private operations

Randomization or a private operation is the most important primitive in these card-based protocols. If every primitive executed in a card-based protocol is deterministic and public, the relationship between the private input values and the output values is known to the players. When the output value is disclosed, the private input value can be known to the players from the relationship. Thus, all protocols need some random or private operation.

First, public randomization primitives have been discussed and then recently, private operations are considered. Many protocols use random bisection cuts [8], which randomly execute swapping two decks of cards or not swapping. If the random value used in the randomization is disclosed, the secret input value is known to the players. If some player privately brings a high-speed camera, the random value selected by the randomization might be known by analyzing the image. Though the size of a high-speed camera is very large, the size might become very small shortly. To prepare for the situation, we need to consider using private operations.

Operations that a player executes in a place where the other players cannot see are called private operations. These operations are considered to be executed under the table or in the back. Private operations are shown to be the most powerful primitives in card-based cryptographic protocols. They were first introduced to solve the millionaires' problem [39]. Using three private operations shown later, committed-input and committed-output logical AND, logical XOR, and copy protocols can be achieved with the minimum number of cards on the two-color card model [12].

So the research question is whether we can achieve the minimum number of cards for a standard deck of cards if we use private operations. We show positive results to the question.

1.4 Our results

This paper shows new logical AND and copy protocols with a standard deck of playing cards that achieves the minimum number of cards by using private operations. The comparisons of AND protocols, copy protocols, XOR protocols, and protocols to compute any n-variable Boolean function are shown in Table 1, Table 2, Table 3, and Table 4, respectively. The results show that private operations are also effective for a standard deck of cards.

Another class of private operations is private input operations that are used when each player inputs his/her private value [37, 44, 72, 73]. All of the above works are based on the two-color model. We show primitives to calculate Boolean functions using private operations for a standard deck of cards.

[1] https://www.letsplayuno.com

Table 1 Comparison of AND protocols with a standard deck of cards.

Article	# of cards	Note
Niemi et al. [65]	5	Las Vegas algorithm
Koch et al. [67]	4	Las Vegas algorithm
Mizuki [66]	8	Fixed time algorithm
This paper	4	Fixed time algorithm

Table 2 Comparison of copy protocols with a standard deck of cards

Article	# of cards	Note
Niemi et al. [65]	6	Las Vegas algorithm
Koyama et al. [26]	6	Las Vegas algorithm
Mizuki [66]	6	Fixed time algorithm
This paper	4	Fixed time algorithm

Table 3 Comparison of XOR protocols with a standard deck of cards.

Article	# of cards	Note
Niemi et al. [65]	4	Las Vegas algorithm
Mizuki [66]	4	Fixed time algorithm
This paper	4	Fixed time algorithm

Several protocols that use asymmetric cards [12, 60, 73, 74] are shown. Asymmetric cards are also used to evaluate the closeness of two values in the Likert scale [75–78]. Since one-bit data can be represented using one asymmetric card, the number of cards can be reduced. Since a standard deck of cards has some number of asymmetric cards, we can execute asymmetric card protocols using a standard deck of cards.

In Section 2, basic notations and the private operations introduced in [12] are shown. Section 3 shows logical AND, copy, and logical XOR protocols. Then, protocols to compute any n-variable Boolean function are shown. Section 4 shows protocols that use private input operations. Section 5 shows asymmetric card protocols. Section 6 concludes the paper.

The preliminary version of this paper was presented as "Yoshifumi Manabe and Hibiki Ono: Card-based Cryptographic Protocols with a Standard Deck of Cards Using Private Operations, Proc. of 18th International Colloquium on Theoretical Aspects of Computing (ICTAC 2021), LNCS Vol.12819, pp.256-274 (2021). This paper improved the AND protocol in Section 3 to simplify the algorithm. Section 4 and Section 5 are new sections that show protocols that use private input operations and asymmetric card protocols.

Table 4 Comparison of protocols to compute any n-variable Boolean function with a standard deck of cards.

Article	# of cards	Note
Shinagawa et al. [14]	$2n+8$	Fixed time algorithm
This paper's Protocol 7	$2^{n+1}+2$	Fixed time algorithm
This paper's Protocol 8	$2n+4$	Fixed time algorithm

2 Preliminaries

2.1 Basic notations

This section gives the notations and basic definitions of card-based protocols with a standard deck of cards. A deck of playing cards consists of 52 distinct mark cards, which are named as 1 to 52. The number of each card (for example, 1 is the ace of spade and 52 is the king of club) is common knowledge among the players. The back of all cards is the same $\boxed{?}$. It is impossible to determine the mark in the back of a given card of $\boxed{?}$.

One-bit data is represented by two cards as follows: $\boxed{\mathbf{i}}\boxed{\mathbf{j}} = 0$ and $\boxed{\mathbf{j}}\boxed{\mathbf{i}} = 1$ if $i < j$.

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 $\underbrace{\boxed{?}\boxed{?}}_{x}$. The base of a commitment is the pair of cards used for the commitment. If card i and $j (i < j)$ are used to set $commit(x)$ (That is, set $\boxed{\mathbf{i}}\boxed{\mathbf{j}}$ if $x = 0$ and set $\boxed{\mathbf{j}}\boxed{\mathbf{i}}$ if $x = 1$), the commitment is written as $commit(x)^{\{i,j\}}$ and written as $\underbrace{\boxed{?}\boxed{?}}_{x^{\{i,j\}}}$. When the base information is obvious or unnecessary, it is not written.

Note that when these two cards are swapped, $commit(\bar{x})^{\{i,j\}}$ can be obtained. Thus, logical negation can be computed without private operations.

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. $S = \underbrace{\boxed{?}}_{s_1}\underbrace{\boxed{?}}_{s_2}\underbrace{\boxed{?}}_{s_3}\ldots,\underbrace{\boxed{?}}_{s_n}$. A sequence whose length is even is called an even sequence. $S_1||S_2$ is a concatenation of sequence S_1 and S_2.

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

Section 3 consider the case when the inputs of the protocols are given in a committed format, that is, the players do not know the input values. Section 4 assumes that each player has his/her input bit that must not be known to the other player.

The output of the protocol must be given in a committed format so that the result can be used as an input to further computation. If the players need to obtain the output value, they just open the committed output. Thus committed output is desirable.

A protocol is secure when the following two conditions are satisfied: (1) If the output cards are not opened, each player obtains no information about the private

input values from the view of the protocol for the player (the sequence of the cards opened to the player). (2) When the output cards are opened, each player obtains no additional information about the private input values other than the information by the output of the protocol. For example, if the output cards of an AND protocol for input x and y are opened and the value is 1, the players can know that $x = 1$ and $y = 1$. If the output value is 0, the players must not know whether the input (x, y) is $(0, 0)$, $(0, 1)$, or $(1, 0)$.

The following protocols use random numbers. Random numbers can be generated without computers using coin-flipping or some similar methods. During the protocol executions, cards are sent and received between the players. The communication is executed by handing the cards between the players to avoid information leakage during the communication. If the players are not in the same place during the protocol execution, a trusted third party (for example, the post office) is necessary to send and receive cards between players.

2.2 Private operations

We show three private operations introduced in [12]: private random bisection cuts, private reverse cuts, and private reveals.

Primitive 1. *(Private random bisection cut)*

A private random bisection cut is the following operation on an even sequence $S_0 = s_1, s_2, \ldots, s_{2m}$*. A player selects a random bit* $b \in \{0, 1\}$ *and outputs*

$$S_1 = \begin{cases} S_0 & \text{if } b = 0 \\ s_{m+1}, s_{m+2}, \ldots, s_{2m}, s_1, s_2, \ldots, s_m & \text{if } b = 1 \end{cases}$$

The player executes this operation in a place where the other players cannot see. The player must not disclose the bit b*.*

Note that if the private random cut is executed when $m = 1$ and $S_0 = commit(x)$, given $S_0 = \underbrace{\boxed{?}\boxed{?}}_{x}$, The player's output $S_1 = \underbrace{\boxed{?}\boxed{?}}_{x \oplus b}$, which is $\underbrace{\boxed{?}\boxed{?}}_{x}$ or $\underbrace{\boxed{?}\boxed{?}}_{\overline{x}}$.

Note that a private random bisection cut is the same as the random bisection cut[8], but the operation is executed in a hidden place.

Primitive 2. *(Private reverse cut, Private reverse selection)*

A private reverse cut is the following operation on an even sequence $S_2 = s_1, s_2, \ldots, s_{2m}$ *and a bit* $b \in \{0, 1\}$*. A player outputs*

$$S_3 = \begin{cases} S_2 & \text{if } b = 0 \\ s_{m+1}, s_{m+2}, \ldots, s_{2m}, s_1, s_2, \ldots, s_m & \text{if } b = 1 \end{cases}$$

The player executes this operation in a place where the other players cannot see. The player must not disclose b*.*

Note that the bit b is not newly selected by the player. This is the difference between the primitive in Primitive 1, where a random bit must be newly selected by the player.

Note that in some protocols below, selecting left m cards is executed after a private reverse cut. The sequence of these two operations is called a private reverse

selection. A private reverse selection is the following procedure on an even sequence $S_2 = s_1, s_2, \ldots, s_{2m}$ and a bit $b \in \{0, 1\}$. A player outputs

$$S_3 = \begin{cases} s_1, s_2, \ldots, s_m & \text{if } b = 0 \\ s_{m+1}, s_{m+2}, \ldots, s_{2m} & \text{if } b = 1 \end{cases}$$

Primitive 3. *(Private reveal) A player privately opens a given committed bit. The player must not disclose the obtained value.*

Using the obtained value, the player privately sets a sequence of cards.

Consider the case when Alice executes a private random bisection cut on $commit(x)$ and Bob executes a private reveal on the bit. Since the committed bit is randomized by the bit b selected by Alice, the opened bit is $x \oplus b$. Even if Bob privately opens the cards, Bob obtains no information about x if b is randomly selected and not disclosed by Alice. Bob must not disclose the obtained value. If Bob discloses the obtained value to Alice, Alice knows the value of the committed bit.

2.3 Opaque commitment pair

An opaque commitment pair is defined as a useful situation for to design a secure protocol using a standard deck of cards [66]. It is a pair of commitments whose bases are unknown to a player. Let us consider the following two commitments using cards i, j, i' and j'. The left (right) commitment has value x (y), respectively, but it is unknown that (1) the left (right) commitment is made using i and j (i' and j'), respectively, or (2) the left (right) commitment is made using i' and j' (i and j), respectively. Such a pair of commitments is called an opaque commitment pair and written as $commit(x)^{\{i,j\},\{i',j'\}} || commit(y)^{\{i,j\},\{i',j'\}}$.

The protocols in this paper use a little different kind of pair, called semi-opaque commitment pair. A player thinks a pair is an opaque commitment pair but another player knows the bases of the commitments. Let us consider the case when a protocol is executed by Alice and Bob. Bob privately makes the pair of commitments with the knowledge of x and y. For example, Bob randomly selects a bit $b \in \{0, 1\}$ and

$$S = \begin{cases} commit(x)^{\{i,j\}} || commit(y)^{\{i',j'\}} & \text{if } b = 0 \\ commit(x)^{\{i',j'\}} || commit(y)^{\{i,j\}} & \text{if } b = 1 \end{cases}$$

then $S = commit(x)^{\{i,j\},\{i',j'\}} || commit(y)^{\{i,j\},\{i',j'\}}$ for Alice. Such a pair is called a semi-opaque commitment pair and written as $commit(x)^{\{i,j\},\{i',j'\}|Alice} || commit(y)^{\{i,j\},\{i',j'\}|Alice}$, where the name(s) of the players who think the pair is a opaque commitment pair is written. Note that a name is not written does not mean the player knows the bases of the commitments. For example, the above example says nothing about whether Bob knows the bases or not. Note that the name of the player is written with the initial when it is not ambiguous.

2.4 Space and time complexities

The space complexity of card-based protocols is evaluated by the number of cards. Minimizing the number of cards is discussed in many works.

The number of rounds was proposed as a criterion to evaluate the time complexity of card-based protocols using private operations [13]. The first round begins from the initial state. The first round is (possibly parallel) local executions by each player using the cards initially given to each player. It ends at the instant when no further local execution is possible without receiving cards from another player. The local executions in each round include sending cards to some other players but do not include receiving cards. The result of every private execution is known to the player. For example, shuffling whose result is unknown to the player himself is not executed. Since the private operations are executed in a place where the other players cannot see, it is hard to force the player to execute such operations whose result is unknown to the player. The $i(>1)$-th round begins with receiving all the cards sent during the $(i-1)$-th round. Each player executes local executions using the received cards and the cards left to the player at the end of the $(i-1)$-th round. Each player executes local executions until no further local execution is possible without receiving cards from another player. The number of rounds of a protocol is the maximum number of rounds necessary to output the result among all possible inputs and random values. If the local execution needs many operations, for example, $O(n)$ operations where n is the size of the problem, we might need another criterion to consider the cost of local executions.

Let us show an example of a protocol execution, its space complexity, and time complexity with the conventional two-color card model. In the two-color card model, there are two kinds of marks, $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. One-bit data is represented by two cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$.

Protocol 1. *(two-color model AND protocol in [12])*
Input: $commit(x)$ *and* $commit(y)$.
Output: $commit(x \wedge y)$.

1. *Alice executes a private random bisection cut on* $commit(x)$. *Let the output be* $commit(x')$. *Alice sends* $commit(x')$ *and* $commit(y)$ *to Bob.*
2. *Bob executes a private reveal on* $commit(x')$. *Bob privately sets*

$$S_2 = \begin{cases} commit(y)||commit(0) & \text{if } x' = 1 \\ commit(0)||commit(y) & \text{if } x' = 0 \end{cases}$$

   and sends S_2 *to Alice.*
3. *Alice executes a private reverse selection on* S_2 *using the bit* b *generated in the private random bisection cut. Let the obtained sequence be* S_3. *Alice outputs* S_3.

The AND protocol realizes the following equation.

$$x \wedge y = \begin{cases} y & \text{if } x = 1 \\ 0 & \text{if } x = 0 \end{cases}$$

The correctness of the protocol is shown in [12]. The number of cards is four since the cards of $commit(x')$ are re-used to set $commit(0)$.

Let us consider the time complexity of the protocol. The first round ends at the instant when Alice sends $commit(x')$ and $commit(y)$ to Bob. The second round begins with receiving the cards by Bob. The second round ends at the instant when Bob sends S_2 to Alice. The third round begins with receiving the cards by Alice. The number of rounds of this protocol is three.

Since each operation is relatively simple, the dominating time to execute protocols with private operations is the time to send cards between players and set up so that the cards are not seen by the other players. Thus the number of rounds is the criterion to evaluate the time complexity of card-based protocols with private operations.

2.5 Problems with a standard deck of cards

The above AND protocol cannot be executed as it is with a standard deck of cards.

The protocol uses the property that all $\boxed{\heartsuit}$ cards ($\boxed{\clubsuit}$ cards) are indistinguishable. Even if the final cards are opened to see the result, it is impossible to know that the opened cards are the cards of $commit(y)$ or $commit(0)$. If it is possible to detect the above information, the value of x is known to the players.

First, let us consider a simple encoding using a standard deck of a playing card that heart and diamond cards mean $\boxed{\heartsuit}$ and all club and spade cards mean $\boxed{\clubsuit}$. With this simple encoding, let us consider the case when the aces of diamond and spade are used to set $commit(x)$ and the aces of heart and club are used to set $commit(y)$.

Suppose that $x = 1$ and $y = 0$. In this case, the result is $commit(y)$, thus the result is correct since $y = 0$. In Step 2 of the protocol, aces of diamond and spade are re-used to set $commit(0)$. Since $x = 1$, the result is $commit(y)$. When the cards are opened to see the result, the cards are the aces of heart and club. The players can know that y is selected as the output, thus x must be 1. The execution reveals the information of inputs from the cards used to set the input commitments.

Next, consider the case when the encoding rule $\boxed{\mathbf{i}}\boxed{\mathbf{j}} = 0, \boxed{\mathbf{j}}\boxed{\mathbf{i}} = 1$ if $i < j$ is used to the standard deck of playing cards. Suppose again that $x = 1$ and $y = 0$. When two inputs are given as $commit(x)^{\{1,2\}}$ and $commit(y)^{\{3,4\}}$, $commit(0)$ and $commit(y)$ are set as $commit(0)^{\{1,2\}}$ and $commit(y)^{\{3,4\}}$, respectively in Step 2. Since $x = 1$, the result is $commit(y)^{\{3,4\}}$. When the cards are opened to see the result, the cards are 3 and 4. The players can know that y is selected as the output, thus x must be 1. This execution also reveals the information of inputs from the base of the commitments.

When we design a protocol with a standard deck of cards, we must consider the information leakage from the base of the commitment.

3 AND, XOR, and copy with a standard deck of cards

This section shows our new protocols for AND, and copy operation with the minimum number of cards using private operations. We also show XOR protocol using private operations to show the minimum number of cards can also be achieved using private

operations. Before we show the protocols, we show subroutines to change the base of a given commitment.

3.1 Base-fixed Protocols

A base change protocol changes the base of a commitment without changing the value of the commitment. A base change protocol is also shown in [66], but the protocol uses a public shuffle, thus we show a new protocol that uses private operations.

Protocol 2. *(Base change protocol)*
Input: $commit(x)^{\{1,2\}}$ and two new cards 3 and 4.
Output: $commit(x)^{\{3,4\}}$.

1. *Bob executes a private random bisection cut on $commit(x)^{\{1,2\}}$. Let $b \in \{0,1\}$ be the bit Bob selected. The result is $S_1 = commit(x \oplus b)^{\{1,2\}}$. Bob sends S_1 to Alice.*
2. *Alice executes a private reveal on S_1. Alice sees $x \oplus b$. Alice makes $S_2 = commit(x \oplus b)^{\{3,4\}}$ and sends S_2 to Bob.*
3. *Bob executes a private reverse cut using b on S_2. The result is $commit(x)^{\{3,4\}}$.*

The protocol is three rounds. The security of the protocol is as follows. When Alice sees the cards in Step 2, the value is $x \oplus b$. Since b is a random value unknown to Alice, Alice has no information about x by the reveal. Bob sees no open cards, thus Bob has no information about x. Note again that Bob must not disclose b to Alice.

Next, we show a base-fixed protocol that changes a semi-opaque commitment pair to a standard commitment. Note that base-fixed protocols that do not use private operations were shown in [66, 79].

Protocol 3. *(Base-fixed protocol)*
Input: $commit(x)^{\{1,2\},\{3,4\}|A} || commit(y)^{\{1,2\},\{3,4\}|A}$.
(Note: y is a private value that must not be known to the players)
Output: $commit(x)^{\{1,2\}}$.

1. *Bob executes a private random bisection cut on both pairs using two distinct random bits $br_1, br_2 \in \{0,1\}$. The result $S_1 = commit(x \oplus br_1)^{\{1,2\},\{3,4\}|A} || commit(y \oplus br_2)^{\{1,2\},\{3,4\}|A}$. Bob sends S_1 to Alice.*
2. *Alice executes a private reveal on S_1. Alice sees $x \oplus br_1$ and $y \oplus br_2$. If the base of the left pair is $\{1,2\}$, Alice just faces down the left pair and the cards, S_2, are the result. Otherwise, the base of the right pair is $\{1,2\}$. Alice makes $S_2 = commit(x \oplus br_1)^{\{1,2\}}$ using the right cards. Alice sends S_2 to Bob.*
3. *Bob executes a private reverse cut using br_1 on S_2. The result is $commit(x)^{\{1,2\}}$.*

In this protocol, Alice knows the bases of the input commitments. The protocol can be used only when this information leakage does not cause a security problem, for example, the bases are randomly set by Bob. The example case is as follows. Initially, Bob knows the relation between the bases and the private input values. If the result is opened and the base becomes public, Bob knows the private input value from the base of the result. Thus, Bob first randomizes the relation between the bases and the values. Since Bob changed the bases, Bob still knows the relation between the bases and the values, but Alice cannot know the relation because of the randomization by Bob. Thus, when Alice privately opens the cards, Alice knows no information from the

bases of the cards. Alice privately opens the cards and fixes the base of the output. When the base is fixed, Bob cannot know the private input value from the base of the commitment. Therefore, when the final output is opened, no information about private input value is known to the players from the base.

The security of the input value x is just the same as the base change protocol.

3.2 AND protocol

In the following AND, copy, and XOR protocols, the bases of the output commitments are fixed to avoid information leakage from the bases when the outputs are opened.

Protocol 4. *(AND protocol)*
Input: $commit(x)^{\{1,2\}}$ *and* $commit(y)^{\{3,4\}}$.
Output: $commit(x \wedge y)^{\{1,2\}}$.

1. *Alice executes a private random bisection cut on* $commit(x)^{\{1,2\}}$ *using random bit* a_1. *Alice sends the results,* $S_1 = commit(x \oplus a_1)^{\{1,2\}}$ *and* $S_2 = commit(y)^{\{3,4\}}$ *to Bob.*
2. *Bob executes a private reveal on* S_1. *Bob sees* $x \oplus a_1$. *Bob privately sets*

$$S_{3,0} = \begin{cases} commit(0)^{\{1,2\}} || commit(y)^{\{3,4\}} & \text{if } x \oplus a_1 = 0 \\ commit(y)^{\{3,4\}} || commit(0)^{\{1,2\}} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Bob sends $S_{3,0}$ *to Alice.*
3. *Alice executes private random bisection cuts on each of pairs in* $S_{3,0}$ *using two distinct random bits* a_2 *and* a_3. *Let the result be* $S_{3,1}$.

$$S_{3,1} = \begin{cases} commit(0 \oplus a_2)^{\{1,2\}} || commit(y \oplus a_3)^{\{3,4\}} & \text{if } x \oplus a_1 = 0 \\ commit(y \oplus a_2)^{\{3,4\}} || commit(0 \oplus a_3)^{\{1,2\}} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Alice sends $S_{3,1}$ *to Bob.*
4. *Bob randomly selects bit* $b_1 \in \{0, 1\}$. *Bob reveals* $S_{3,1}$ *and exchanges the bases of the two commitments if* $b_1 = 1$. *Let the result be* $S_{3,2}$.

$$S_{3,2} = \begin{cases} commit(0 \oplus a_2)^{\{1,2\},\{3,4\}|A} || commit(y \oplus a_3)^{\{1,2\},\{3,4\}|A} & \text{if } x \oplus a_1 = 0 \\ commit(y \oplus a_2)^{\{1,2\},\{3,4\}|A} || commit(0 \oplus a_3)^{\{1,2\},\{3,4\}|A} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Bob sends $S_{3,2}$ *to Alice.*
5. *Alice executes private reverse cuts on the two pairs of* $S_{3,2}$ *using* a_2 *and* a_3, *respectively. Let the result be* S_4.

$$S_4 = \begin{cases} commit(0)^{\{1,2\},\{3,4\}|A} || commit(y)^{\{1,2\},\{3,4\}|A} & \text{if } x \oplus a_1 = 0 \\ commit(y)^{\{1,2\},\{3,4\}|A} || commit(0)^{\{1,2\},\{3,4\}|A} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Alice then executes a private reverse selection on S_4 *using* a_1. *Let* S_5 *be the result and the remaining two cards be* S_6. *The result* $S_5 = commit(y)^{\{1,2\},\{3,4\}|A}$ *if* ($a_1 = 0$ *and* $x \oplus a_1 = 1$) *or* ($a_1 = 1$ *and* $x \oplus a_1 = 0$). *The condition equals* $x = 1$.

$S_5 = commit(0)^{\{1,2\},\{3,4\}|A}$ *if* $(a_1 = 0$ *and* $x \oplus a_1 = 0)$ *or* $(a_1 = 1$ *and* $x \oplus a_1 = 1)$. *The condition equals* $x = 0$. *Thus,*

$$S_5 = \begin{cases} commit(y)^{\{1,2\},\{3,4\}|A} & \text{if } x = 1 \\ commit(0)^{\{1,2\},\{3,4\}|A} & \text{if } x = 0 \end{cases}$$

$$= commit(x \wedge y)^{\{1,2\},\{3,4\}|A}$$

Alice sends S_5 *and* S_6 *to Bob.*

6. *Bob and Alice execute Protocol 3 (Base-fixed protocol) to* $S_5||S_6$. *Then they obtain* $commit(x \wedge y)^{\{1,2\}}$.

Note that this protocol has been changed from the proceedings version [80]. In the proceedings version, Bob first executes a base randomization and after that, Bob reveals randomized x and sets the result just like Step 2 in this paper's protocol. Thus, Alice's randomization of value y was necessary before Bob's base randomization. In this paper's protocol, Bob's reveal and setting the result is first executed and then Bob's base randomization is executed. By changing the order, Alice does not need to randomize y in Step 1 and the protocol becomes close to the original two-color model protocol [12].

The protocol is eight rounds. The number of cards is four. Since four cards are necessary to input x and y, the number of cards is the minimum. The correctness of the output value is shown in the protocol, thus we show the security.

Theorem 1. *The AND protocol is secure.*

Proof. First, we show the security for Bob. Though Bob sees cards in Step 2, the cards, $S_1 = commit(x \oplus a_1)^{\{1,2\}}$ are randomized by a_1. Thus Bob obtains no information about x. Bob then sees the cards of $S_{3,1}$ in Step 4. Since Bob knows $x \oplus a_1$, he knows $S_{3,1}$ was $commit(0)||commit(y)$ or $commit(y)||commit(0)$ before the randomization by Alice. However, the opened values are randomized by a_2 and a_3. Thus, Bob obtains no information about y from the opened values.

Alice sees cards in Step 2 of the base-fixed protocol. In Step 4 after the base randomization by Bob,

$$S_{3,2} = \begin{cases} commit(0 \oplus a_2)^{\{1,2\}}||commit(y \oplus a_3)^{\{3,4\}} & \text{if } x \oplus a_1 = 0 \text{ and } b_1 = 0 \\ commit(0 \oplus a_2)^{\{3,4\}}||commit(y \oplus a_3)^{\{1,2\}} & \text{if } x \oplus a_1 = 0 \text{ and } b_1 = 1 \\ commit(y \oplus a_2)^{\{3,4\}}||commit(0 \oplus a_3)^{\{1,2\}} & \text{if } x \oplus a_1 = 1 \text{ and } b_1 = 0 \\ commit(y \oplus a_2)^{\{1,2\}}||commit(0 \oplus a_3)^{\{3,4\}} & \text{if } x \oplus a_1 = 1 \text{ and } b_1 = 1 \end{cases}$$

Thus, after the reverse cuts in Step 5,

$$S_4 = \begin{cases} commit(0)^{\{1,2\}}||commit(y)^{\{3,4\}} & \text{if } x \oplus a_1 = 0 \text{ and } b_1 = 0 \\ commit(0)^{\{3,4\}}||commit(y)^{\{1,2\}} & \text{if } x \oplus a_1 = 0 \text{ and } b_1 = 1 \\ commit(y)^{\{3,4\}}||commit(0)^{\{1,2\}} & \text{if } x \oplus a_1 = 1 \text{ and } b_1 = 0 \\ commit(y)^{\{1,2\}}||commit(0)^{\{3,4\}} & \text{if } x \oplus a_1 = 1 \text{ and } b_1 = 1 \end{cases}$$

After the private reverse selection by Alice,

$$S_5 = \begin{cases} commit(0)^{\{1,2\}} & \text{if } x = 0 \text{ and } b_1 = 0 \\ commit(0)^{\{3,4\}} & \text{if } x = 0 \text{ and } b_1 = 1 \\ commit(y)^{\{3,4\}} & \text{if } x = 1 \text{ and } b_1 = 0 \\ commit(y)^{\{1,2\}} & \text{if } x = 1 \text{ and } b_1 = 1 \end{cases}$$

Similarly,

$$S_6 = \begin{cases} commit(y)^{\{3,4\}} & \text{if } x = 0 \text{ and } b_1 = 0 \\ commit(y)^{\{1,2\}} & \text{if } x = 0 \text{ and } b_1 = 1 \\ commit(0)^{\{1,2\}} & \text{if } x = 1 \text{ and } b_1 = 0 \\ commit(0)^{\{3,4\}} & \text{if } x = 1 \text{ and } b_1 = 1 \end{cases}$$

These values are then randomized using br_1 and br_2 in Step 1 of the base-fixed protocol. Thus, Alice sees the randomized cards of $S_5||S_6$, which are

$$\begin{cases} commit(br_1)^{\{1,2\}}||commit(y \oplus br_2)^{\{3,4\}} & \text{if } x = 0 \text{ and } b_1 = 0 \\ commit(br_1)^{\{3,4\}}||commit(y \oplus br_2)^{\{1,2\}} & \text{if } x = 0 \text{ and } b_1 = 1 \\ commit(y \oplus br_1)^{\{3,4\}}||commit(br_2)^{\{1,2\}} & \text{if } x = 1 \text{ and } b_1 = 0 \\ commit(y \oplus br_1)^{\{1,2\}}||commit(br_2)^{\{3,4\}} & \text{if } x = 1 \text{ and } b_1 = 1 \end{cases}$$

Therefore, Alice sees

$$\begin{cases} commit(0)^{\{1,2\}}||commit(0)^{\{3,4\}} \\ \text{if } (x = 0 \wedge b_1 = 0 \wedge br_1 = 0 \wedge y \oplus br_2 = 0) \vee (x = 1 \wedge b_1 = 1 \wedge y \oplus br_1 = 0 \wedge br_2 = 0) \\ commit(0)^{\{1,2\}}||commit(1)^{\{3,4\}} \\ \text{if } (x = 0 \wedge b_1 = 0 \wedge br_1 = 0 \wedge y \oplus br_2 = 1) \vee (x = 1 \wedge b_1 = 1 \wedge y \oplus br_1 = 0 \wedge br_2 = 1) \\ commit(1)^{\{1,2\}}||commit(0)^{\{3,4\}} \\ \text{if } (x = 0 \wedge b_1 = 0 \wedge br_1 = 1 \wedge y \oplus br_2 = 0) \vee (x = 1 \wedge b_1 = 1 \wedge y \oplus br_1 = 1 \wedge br_2 = 0) \\ commit(1)^{\{1,2\}}||commit(1)^{\{3,4\}} \\ \text{if } (x = 0 \wedge b_1 = 0 \wedge br_1 = 1 \wedge y \oplus br_2 = 1) \vee (x = 1 \wedge b_1 = 1 \wedge y \oplus br_1 = 1 \wedge br_2 = 1) \\ commit(0)^{\{3,4\}}||commit(0)^{\{1,2\}} \\ \text{if } (x = 0 \wedge b_1 = 1 \wedge br_1 = 0 \wedge y \oplus br_2 = 0) \vee (x = 1 \wedge b_1 = 0 \wedge y \oplus br_1 = 0 \wedge br_2 = 0) \\ commit(0)^{\{3,4\}}||commit(1)^{\{1,2\}} \\ \text{if } (x = 0 \wedge b_1 = 1 \wedge br_1 = 0 \wedge y \oplus br_2 = 1) \vee (x = 1 \wedge b_1 = 0 \wedge y \oplus br_1 = 0 \wedge br_2 = 1) \\ commit(1)^{\{3,4\}}||commit(0)^{\{1,2\}} \\ \text{if } (x = 0 \wedge b_1 = 1 \wedge br_1 = 1 \wedge y \oplus br_2 = 0) \vee (x = 1 \wedge b_1 = 0 \wedge y \oplus br_1 = 1 \wedge br_2 = 0) \\ commit(1)^{\{3,4\}}||commit(1)^{\{1,2\}} \\ \text{if } (x = 0 \wedge b_1 = 1 \wedge br_1 = 1 \wedge y \oplus br_2 = 1) \vee (x = 1 \wedge b_1 = 0 \wedge y \oplus br_1 = 1 \wedge br_2 = 1) \end{cases}$$

Let $P_{ij}(i \in \{0,1\}, j \in \{0,1\})$ be the probability when $x = i$ and $y = j$. The probabilities $P(br_1 = 0), P(br_1 = 1), P(br_2 = 0), P(br_2 = 1), P(b_1 = 0)$, and $P(b_1 = 1)$ are $1/2$, thus the probabilities when Alice sees $commit(v)^{\{i,i+1\}}||commit(w)^{\{4-i,5-i\}}(v, w \in \{0,1\}, i \in \{1,3\})$ are the same value $(P_{00} + P_{01} + P_{10} + P_{11})/8$. Thus, Alice obtains no information from the cards she sees. □

The comparison of AND protocols is shown in Table 1.

Note that by the discussion in [12], any two-variable Boolean function can be computed by a protocol similar to Protocol 4.

3.3 Copy protocol

Next, we show a new copy protocol. Note that the protocol is essentially the same as the one in [12] for the two-color card model. The number of cards is the minimum.

Protocol 5. *(Copy protocol)*

Input: $commit(x)^{\{1,2\}}$ *and two new cards 3 and 4.*

Output: $commit(x)^{\{1,2\}}$ *and* $commit(x)^{\{3,4\}}$

1. *Alice executes a private random bisection cut on* $commit(x)^{\{1,2\}}$*. Let* b *be the random bit Alice selects. Alice sends the result,* $commit(x \oplus b)^{\{1,2\}}$*, to Bob.*
2. *Bob executes a private reveal on* $commit(x \oplus b)^{\{1,2\}}$ *and sees* $x \oplus b$*. Bob privately makes* $commit(x \oplus b)^{\{3,4\}}$*. Bob sends* $commit(x \oplus b)^{\{1,2\}}$ *and* $commit(x \oplus b)^{\{3,4\}}$ *to Alice.*
3. *Alice executes a private reverse cut on each of the pairs using* b*. The result is* $commit(x)^{\{1,2\}}$ *and* $commit(x)^{\{3,4\}}$*.*

The protocol is three rounds.

Theorem 2. *The copy protocol is secure.*

Proof. Since Alice sees no open cards, Alice obtains no information about the input value. Though Bob sees $x \oplus b$, input x is randomized by b and Bob obtains no information about x. □

The comparison of copy protocols is shown in Table 2.

3.4 XOR protocol

Though the minimum number of cards is already achieved in [66], the protocol uses public shuffles. We show a new protocol that uses private operations. The protocol is essentially the same as the one in [13] for the two-color card model.

Protocol 6. *(XOR protocol)*

Input: $commit(x)^{\{1,2\}}$ *and* $commit(y)^{\{3,4\}}$*.*

Output: $commit(x \oplus y)^{\{1,2\}}$*.*

1. *Alice executes a private random bisection cut on* $commit(x)^{\{1,2\}}$ *and* $commit(y)^{\{3,4\}}$ *using the same random bit* $b \in \{0, 1\}$*. The result is* $commit(x \oplus b)^{\{1,2\}}$ *and* $commit(y \oplus b)^{\{3,4\}}$*. Alice sends these cards to Bob.*
2. *Bob executes a private reveal on* $commit(y \oplus b)^{\{3,4\}}$*. Bob sees* $y \oplus b$*. Bob executes a private reverse cut on* $commit(x \oplus b)^{\{1,2\}}$ *using* $y \oplus b$*. The result is* $commit((x \oplus b) \oplus (y \oplus b))^{\{1,2\}} = commit(x \oplus y)^{\{1,2\}}$*.*

The protocol is two rounds. The protocol uses four cards. Since any protocol needs four cards to input x and y, the number of cards is the minimum.

Note that if Bob sends $commit(y \oplus b)^{\{3,4\}}$ in Step 2 to Alice and Alice executes a private reverse cut using b, an input $commit(y)^{\{3,4\}}$ can be obtained without additional cards. This protocol is called an input preserving XOR and it is used in Section 3.6.

Theorem 3. *The XOR protocol is secure.*

Proof. Since Alice sees no open cards, Alice obtains no information about the input values. Though Bob sees $y \oplus b$, input y is randomized by b and Bob obtains no information about y. □

The comparison of XOR protocols is shown in Table 3.

3.5 Difference between AND and XOR protocols

The AND protocol (Protocol 4) is much more complicated compared to the XOR protocol (Protocol 6) and the copy protocol (protocol 5). The reason is the result of XOR does not change after the randomization of input values, but the result of AND changes the value.

Some randomization of input values is necessary for any card-based protocols to hide the relation between the input values and the output values. After a randomization, the input value x is changed to $x \oplus b$ for some random bit $b \in \{0, 1\}$. Since $(x \oplus b) \oplus (y \oplus b) = x \oplus y$ is satisfied, thus the randomization by b is canceled by the XOR calculation of two randomized input values. This is the reason the XOR protocol is simple. On the other hand, AND does not have the property, that is, $(x \oplus b) \wedge (y \oplus b) \neq x \wedge y$. Thus, additional steps are necessary to obtain the result using randomized input values and the AND protocol is more complicated than the XOR protocol.

For the protocols using a standard deck of cards, the above fact leads the information leakage from the base of the cards. In the XOR protocol (Protocol 6), the base of output is always the same as the one used in the input $commit(x)$. Just swapping the two cards (or doing nothing) can output the result, thus the base is not changed during the execution. In the copy protocol (Protocol 5), the bases of the outputs are the same as the one used in the input $commit(x)$ and the new cards. The base of the output $commit(x)$ is unchanged and the base of the new copy of x is the base of the new cards. Thus, there is no information leakage from the bases of the cards if the cards are opened to see the value. Therefore, complicated steps to hide the bases of the cards are unnecessary in these protocols.

On the other hand, the AND protocol must output $commit(y)$ if $x = 1$ and $commit(0)$ if $x = 0$. The base of $commit(y)$ is $\{3, 4\}$. The base of $commit(0)$ differs from the base of $commit(y)$ (in order to minimize the number of cards, the base of $commit(0)$ is $\{1, 2\}$). If the protocol just outputs $commit(y)^{\{3,4\}}$ or $commit(0)^{\{1,2\}}$, the players know the value of x from the base of the output when the final result is opened. Thus, the base of the output must be fixed to $\{1, 2\}$ in either case and complicated steps are necessary in Protocol 4.

3.6 Any Boolean function

We show two kinds of protocols to compute any n-variable Boolean function. The first one uses many cards but the number of rounds is constant. The second one uses fewer cards but needs many rounds. Let $f(x_1, x_2, \ldots, x_n)$ be an n-variable Boolean function.

Protocol 7. *(Protocol for any n-variable Boolean function (1))*
Input: $commit(x_i)^{\{2i-1,2i\}}(i = 1, 2, \ldots, n)$.
Output: $commit(f(x_1, x_2, \ldots, x_n))^{\{1,2\}}$.

1. *Alice executes a private random bisection cut on* $commit(x_i)^{\{2i-1,2i\}}(i = 1, 2, \ldots, n)$. *Let the output be* $commit(x'_i)^{\{2i-1,2i\}}(i = 1, 2, \ldots, n)$. *Note that one distinct random bit* b_i *is selected for each* $x_i(i = 1, 2, \ldots, n)$. $x'_i = x_i \oplus b_i(i = 1, 2, \ldots, n)$. *Alice sends* $commit(x'_i)^{\{2i-1,2i\}}(i = 1, 2, \ldots, n)$ *to Bob.*
2. *Bob executes a private reveal on* $commit(x'_i)^{\{2i-1,2i\}}(i = 1, 2, \ldots, n)$. *Bob selects a random bit* $b \in \{0, 1\}$. *Bob privately makes* 2^n *commitments* $S_{a_1,a_2,\ldots,a_n}$ $(a_i \in \{0, 1\}, i = 1, 2, \ldots, n)$ *as* $S_{a_1,a_2,\ldots,a_n} = commit(f(a_1 \oplus x'_1, a_2 \oplus x'_2, \ldots, a_n \oplus x'_n) \oplus b)$ *using card* $3, 4, \ldots, 2^{n+1} + 1, 2^{n+1} + 2$. *Note that the cards used to set each commitment are randomly selected by Bob. Bob executes a private random bisection cut on* $commit(\cdot)^{\{1,2\}}$ *to erase the value. Bob sends these commitments to Alice.*
3. *Alice privately reveals* $S_{b_1,b_2,\ldots,b_n}$. *Alice sees* $f(b_1 \oplus x'_1, b_2 \oplus x'_2, \ldots, b_n \oplus x'_n) \oplus b = f(x_1, x_2, \ldots, x_n) \oplus b$, *since* $x'_i = x_i \oplus b_i(i = 1, 2, \ldots, n)$. *Alice privately makes* $S = commit(f(x_1, x_2, \ldots, x_n) \oplus b)^{\{1,2\}}$ *and sends* S *to Bob.*
4. *Bob executes a private reverse cut using* b *on* S. *The result is* $commit(f(x_1, x_2, \ldots, x_n))^{\{1,2\}}$. *Bob outputs the result.*

Note that Bob can re-use cards of $3, 4, \ldots, 2n - 1$, and $2n$ to set $S_{a_1,a_2,\ldots,a_n}$. The protocol uses $2^{n+1} + 2$ cards. The number of rounds is four.

Theorem 4. *Protocol 7 is secure.*

Proof. Bob sees $x'_i = x_i \oplus b_i$, but the input x_i is randomized by b_i and Bob obtains no information about x_i. Alice sees $f(x_1, x_2, \ldots, x_n) \oplus b$, but the value is randomized by b and Alice obtains no information about $f(x_1, x_2, \ldots, x_n)$. Alice obtains no information from the base of the commitment since the base is randomly selected by Bob. □

The main idea of the other protocol is the same as the one in [12] for the two-color card model, which uses an input preserving AND protocol. After the AND protocol, the unused pair of cards has $g = \bar{x} \wedge y$ [12]. Let $h = x \wedge y$. The last step of the AND protocol (the first step of the base-fixed protocol) is changed so that Alice sets $commit(h \oplus br_1)^{\{1,2\}}$ and $commit(g \oplus br_2)^{\{3,4\}}$. By the private reverse cut by Bob, Bob obtains $commit(h)^{\{1,2\}}$ and $commit(g)^{\{3,4\}}$. Execute the input preserving XOR protocol to g and h so that h is preserved. The output $g \oplus h = x \wedge y \oplus \bar{x} \wedge y = y$, thus we can obtain $commit(x \wedge y)^{\{1,2\}}$ and $commit(y)^{\{3,4\}}$. Therefore, one input value can be preserved without additional cards by the AND protocol.

Any Boolean function $f(x_1, x_2, \ldots, x_n)$ can be represented as follows:
$f(x_1, x_2, \ldots, x_n) = \bar{x_1} \wedge \bar{x_2} \wedge \cdots \bar{x_n} \wedge f(0, 0, \ldots, 0) \oplus x_1 \wedge \bar{x_2} \wedge \cdots \bar{x_n} \wedge f(1, 0, \ldots, 0) \oplus \bar{x_1} \wedge x_2 \wedge \cdots \bar{x_n} \wedge f(0, 1, \ldots, 0) \oplus \cdots \oplus x_1 \wedge x_2 \wedge \cdots x_n \wedge f(1, 1, \ldots, 1)$.

Since the terms with $f(i_1, i_2, \ldots, i_n) = 0$ can be removed, this function f can be written as $f = \bigoplus_{i=1}^{k} v_1^i \wedge v_2^i \wedge \cdots \wedge v_n^i$, where $v_j^i = x_j$ or $\bar{x_j}$. Let us write $T_i = v_1^i \wedge v_2^i \wedge \cdots \wedge v_n^i$. The number of terms $k(< 2^n)$ depends on f.

Protocol 8. *(Protocol for any n-variable Boolean function (2))*

Input: $commit(x_i)^{\{2i+3,2i+4\}} (i = 1, 2, \ldots, n)$.

Output: $commit(f(x_1, x_2, \ldots, x_n))^{\{1,2\}}$.

The additional four cards (two pairs of cards) 1,2,3, and 4 are used as follows.

1 and 2 store the intermediate value to compute f.

3 and 4 store the intermediate value to compute T_i.

Execute the following steps for $i = 1, 2, \ldots, k$.

1. *Copy v_1^i from the input $commit(x_1)$ as $commit(v_1^i)^{\{3,4\}}$. (Note that if v_1^i is $\bar{x_1}$, NOT is taken after the copy).*
2. *For $j = 2, \ldots, n$, execute the following procedure: Execute the input preserving AND protocol to $commit(\cdot)^{\{3,4\}}$ and $commit(v_j^i)$ so that input $commit(v_j^i)$ is preserved. The result is stored as $commit(\cdot)^{\{3,4\}}$. (Note that if v_j^i is $\bar{x_j}$, NOT is taken before the AND protocol, and NOT is taken again for the preserved input.)*

   At the end of this step, T_i is obtained as $commit(v_1^i \wedge v_2^i \wedge \cdots \wedge v_n^i)^{\{3,4\}}$.
3. *If $i = 1$, copy $commit(\cdot)^{\{3,4\}}$ to $commit(\cdot)^{\{1,2\}}$. If $i > 1$, apply the XOR protocol between $commit(\cdot)^{\{3,4\}}$ and $commit(\cdot)^{\{1,2\}}$. The result is stored as $commit(\cdot)^{\{1,2\}}$.*

At the end of the protocol, $commit(f(x_1, x_2, \ldots x_n))^{\{1,2\}}$ is obtained.

The comparison of protocols to compute any n-variable Boolean function is shown in Table 4.

The number of additional cards in [14] with a standard deck of cards is 8. Thus the number of additional cards is reduced using private operations.

4 Private input protocols

This section shows protocols when each player has his/her secret value and they securely input the value into the protocol. The protocols for the two-color model were shown in [72, 73, 81]. The main idea of these protocols is as follows: when Alice has input x, Bob has input y, and they calculate $f(x, y)$, Alice prepares the pairs $commit(f(x, 0))||commit(f(x, 1))$. Bob privately selects the one with his input value y from the two pairs, then they can obtain $commit(f(x, y))$.

Using the idea, we can obtain the following protocols for any two-variable function $f(x, y)$ that includes $x \wedge y$. Note that a procedure to fix the base of the output is necessary to prevent information leakage from the bases of the cards. In the following protocol, fixing the base can be executed before the selection of the output, since Bob knows which pairs are the final output. This reduces the number of rounds.

Protocol 9. *(Private input AND-type protocol)*

Input: Alice has x. Bob has y.

Output: $commit(f(x, y))^{\{1,2\}}$.

1. *Alice selects two random bits $a_1, a_2 \in \{0, 1\}$ and makes $S_1 = commit(f(x, 0) \oplus a_1)^{\{1,2\}}||commit(f(x, 1) \oplus a_2)^{\{3,4\}}$ and sends S_1 to Bob.*

2. *Bob does nothing if $y = 0$. Otherwise, Bob privately reveals S_1 and swaps the bases of the two pairs, that is, the result is $commit(f(x,0) \oplus a_1)^{\{3,4\}} || commit(f(x,1) \oplus a_2)^{\{1,2\}}$. By the step, the base of $commit(f(x,y))$ becomes $\{1,2\}$. The result $S_2 = commit(f(x,0) \oplus a_1)^{\{1,2\},\{3,4\}|A} || commit(f(x,1) \oplus a_2)^{\{1,2\}\{3,4\}|A}$. Bob sends S_2 to Alice.*
3. *Alice executes private reverse cuts on each pair of cards in S_2 using a_1 and a_2. The result $S_3 = commit(f(x,0))^{\{1,2\},\{3,4\}|A} || commit(f(x,1))^{\{1,2\}\{3,4\}|A}$ Alice sends S_3 to Bob.*
4. *Bob executes a private reverse selection on S_3 using y. The result is $S_4 = commit(f(x,y))^{\{1,2\}}$.*

The protocol is four rounds. The security for Alice is obvious since Alice sees no open cards. Though Bob sees some cards, the values are randomized thus Bob has no information about a and $f(x,y)$. From the bases of the cards, Bob obtains no information since the sequence of the bases (whether $\{1,2\},\{3,4\}$ or $\{3,4\},\{1,2\}$) is decided by Bob.

For XOR, a more simple protocol can be obtained, since executing a base-fixed protocol is unnecessary.

Protocol 10. *(Private input XOR protocol)*
Input: Alice has x. Bob has y.
Output: $commit(x \oplus y)^{\{1,2\}}$.

1. *Alice makes $S_1 = commit(x)^{\{1,2\}}$ and sends S_1 to Bob.*
2. *Bob executes a private reverse cut using y, that is, the left and the right are swapped if $y = 1$ and do nothing if $y = 0$. The result $S_2 = commit(x \oplus y)^{\{1,2\}}$.*

The protocol is two rounds. Both players see no cards, thus the security is obvious.

5 Asymmetric card protocol

A standard deck of cards has some number of asymmetric cards (Ace, 3, 5, 6, 7, and 9 of spade, heart, and club). If the back of the cards is symmetric, one-bit commitment can be represented by one card such as  =0 and  =1. For each asymmetric card, Alice and Bob decide which is the upward or the downward in advance.

Protocols with asymmetric cards are first considered in [60] and then several protocols are shown in [12, 73–78]. These works consider the case when there are many cards with the same asymmetric mark. Thus, this is the first protocol that uses distinct asymmetric cards.

In this section, distinct asymmetric cards are numbered as $1, 2, \ldots, m$ and symmetric cards are not used.  =0 and  =1. Commitment of x using card i is written as $commit(x)^{\{i\}}$.

For such an encoding method, a private random bisection cut on a committed bit is changed to upside down the card according to the random bit. A private reverse cut and a private reverse selection on an even sequence are unchanged. A private reverse cut and a private reverse selection on a single card are changed upside down the card according to the bit. Using these private operations, all protocols shown above work

for the asymmetric cards. The number of cards used by this protocol is half of the symmetric card protocols.

First, we show AND, XOR, and copy protocols whose inputs are given in a committed manner.

Protocol 11. *(Asymmetric card base-fixed protocol)*
Input: $commit(x)^{\{1\},\{2\}|A}||commit(y)^{\{1\},\{2\}|A}$.
(Note: y is a private value that must not be known to the players)
Output: $commit(x)^{\{1\}}$.

1. *Bob selects two random bisection bits* $br_1, br_2 \in \{0,1\}$. *If* $br_1 = 1$, *Bob turns upside down* $commit(x)$. *If* $br_2 = 1$, *Bob turns upside down* $commit(y)$. *The result* $S_1 = commit(x \oplus br_1)^{\{1\},\{2\}|A}||commit(y \oplus br_2)^{\{1\},\{2\}|A}$. *Bob sends* S_1 *to Alice.*
2. *Alice executes a private reveal on* S_1. *Alice sees* $x \oplus br_1$ *and* $y \oplus br_2$. *If the base of the left card is* $\{1\}$, *Alice just faces down the left card and the card,* S_2, *is the result. Otherwise, the base of the right card is* $\{1\}$. *Alice makes* $S_2 = commit(x \oplus br_1)^{\{1\}}$ *using the right card. Alice sends* S_2 *to Bob.*
3. *Bob turns upside down* S_2 *if* $br_1 = 1$. *The result is* $commit(x)^{\{1\}}$.

The protocol is three rounds.

Protocol 12. *(Asymmetric card AND protocol)*
Input: $commit(x)^{\{1\}}$ *and* $commit(y)^{\{2\}}$.
Output: $commit(x \wedge y)^{\{1\}}$.

1. *Alice selects a random bit* a_1. *Alice turns upside down* $commit(x)^{\{1\}}$ *if* $a_1 = 1$. *Alice sends the result,* $S_1 = commit(x \oplus a_1)^{\{1\}}$ *and* $S_2 = commit(y)^{\{2\}}$ *to Bob.*
2. *Bob executes a private reveal on* S_1. *Bob sees* $x \oplus a_1$. *Bob privately sets*

$$S_{3,0} = \begin{cases} commit(0)^{\{1\}}||commit(y)^{\{2\}} & \text{if } x \oplus a_1 = 0 \\ commit(y)^{\{2\}}||commit(0)^{\{1\}} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Bob sends $S_{3,0}$ *to Alice.*
3. *Alice selects two random bits* a_2 *and* a_3. *Alice turns upside down the left card of* $S_{3,0}$ *if* $a_2 = 1$. *Alice turns upside down the right card of* $S_{3,0}$ *if* $a_3 = 1$. *Let the result be* $S_{3,1}$.

$$S_{3,1} = \begin{cases} commit(0 \oplus a_2)^{\{1\}}||commit(y \oplus a_3)^{\{2\}} & \text{if } x \oplus a_1 = 0 \\ commit(y \oplus a_2)^{\{2\}}||commit(0 \oplus a_3)^{\{1\}} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Alice sends $S_{3,1}$ *to Bob.*
4. *Bob randomly selects bit* $b_1 \in \{0,1\}$. *Bob reveals* $S_{3,1}$ *and exchanges the bases of the two commitments if* $b_1 = 1$. *Let the result be* $S_{3,2}$.

$$S_{3,2} = \begin{cases} commit(0 \oplus a_2)^{\{1\},\{2\}|A}||commit(y \oplus a_3)^{\{1\},\{2\}|A} & \text{if } x \oplus a_1 = 0 \\ commit(y \oplus a_2)^{\{1\},\{2\}|A}||commit(0 \oplus a_3)^{\{1\},\{2\}|A} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Bob sends $S_{3,2}$ *to Alice.*

5. *Alice turns upside down the left card of $S_{3,2}$ if $a_2 = 1$. Alice turns upside down the right card of $S_{3,2}$ if $a_3 = 1$. Let the result be S_4.*

$$S_4 = \begin{cases} commit(0)^{\{1\},\{2\}|A} || commit(y)^{\{1\},\{2\}|A} & \text{if } x \oplus a_1 = 0 \\ commit(y)^{\{1\},\{2\}|A} || commit(0)^{\{1\},\{2\}|A} & \text{if } x \oplus a_1 = 1 \end{cases}$$

Alice then executes a private reverse selection on S_4 using a_1. Let S_5 be the result and the remaining card be S_6. The result $S_5 = commit(y)^{\{1\},\{2\}|A}$ if ($a_1 = 0$ and $x \oplus a_1 = 1$) or ($a_1 = 1$ and $x \oplus a_1 = 0$). The condition equals $x = 1$.

$S_5 = commit(0)^{\{1\},\{2\}|A}$ if ($a_1 = 0$ and $x \oplus a_1 = 0$) or ($a_1 = 1$ and $x \oplus a_1 = 1$). The condition equals $x = 0$. Thus,

$$S_5 = \begin{cases} commit(y)^{\{1\},\{2\}|A} & \text{if } x = 1 \\ commit(0)^{\{1\},\{2\}|A} & \text{if } x = 0 \end{cases}$$

$$= commit(x \wedge y)^{\{1\},\{2\}|A}$$

Alice sends S_5 and S_6 to Bob.

6. *Bob and Alice execute an asymmetric card base-fixed protocol to $S_5||S_6$. Then they obtain $commit(x \wedge y)^{\{1\}}$.*

The eight-round protocol uses two cards.

Protocol 13. *(Asymmetric card XOR protocol)*
Input: $commit(x)^{\{1\}}$ and $commit(y)^{\{2\}}$.
Output: $commit(x \oplus y)^{\{1\}}$.

1. *Alice selects random bit b. Alice turns down $commit(x)^{\{1\}}$ and $commit(y)^{\{2\}}$ if $b = 1$. The result is $commit(x \oplus b)^{\{1\}}$ and $commit(y \oplus b)^{\{2\}}$. Alice sends these cards to Bob.*
2. *Bob executes a private reveal on $commit(y \oplus b)^{\{2\}}$. Bob sees $y \oplus b$. Bob turns upside down $commit(x \oplus b)^{\{1\}}$ if $y \oplus b = 1$. The result is $commit((x \oplus b) \oplus (y \oplus b))^{\{1\}} = commit(x \oplus y)^{\{1\}}$.*

The two-round protocol uses two cards.

Protocol 14. *(asymmetric card Copy protocol)*
Input: $commit(x)^{\{1\}}$ and new card 2.
Output: $commit(x)^{\{1\}}$ and $commit(x)^{\{2\}}$.

1. *Alice selects a random bit $b \in \{0, 1\}$. If $b = 1$, Alice turns upside down $commit(x)^{\{1\}}$. Alice sends the result, $commit(x \oplus b)^{\{1\}}$, to Bob.*
2. *Bob executes a private reveal on $commit(x \oplus b)^{\{1\}}$ and sees $x \oplus b$. Bob privately makes $commit(x \oplus b)^{\{2\}}$. Bob sends $commit(x \oplus b)^{\{1\}}$ and $commit(x \oplus b)^{\{2\}}$ to Alice.*
3. *Alice turns upside down each card if $b = 1$. The result is $commit(x)^{\{1\}}$ and $commit(x)^{\{2\}}$.*

The three-round protocol uses two cards.

Next, we show private input protocols.

Protocol 15. *(Asymmetric card private input AND-type protocol)*
Input: Alice has x. Bob has y.
Output: $commit(f(x,y))^{\{1\}}$.

1. *Alice selects two random bits $a_1, a_2 \in \{0,1\}$ and makes $S_1 = commit(f(x,0) \oplus a_1)^{\{1\}} || commit(f(x,1) \oplus a_2)^{\{2\}}$ and sends S_1 to Bob.*
2. *Bob does nothing if $y = 0$. Otherwise, Bob privately reveals S_1 and swaps the bases of the two cards, that is, the result is $commit(f(x,0) \oplus a_1)^{\{2\}} || commit(f(x,1) \oplus a_2)^{\{1\}}$. By the step, the base of $commit(f(x,y))$ becomes $\{1\}$. The result $S_2 = commit(f(x,0) \oplus a_1)^{\{1\},\{2\}|A} || commit(f(x,1) \oplus a_2)^{\{1\}\{2\}|A}$. Bob sends S_2 to Alice.*
3. *Alice turns upside down the left (right) card if $a_1 = 1 (a_2 = 1)$, respectively. The result $S_3 = commit(f(x,0))^{\{1\},\{2\}|A} || commit(f(x,1))^{\{1\}\{2\}|A}$ Alice sends S_3 to Bob.*
4. *Bob executes private reverse cut on S_3 using y. The result is $S_4 = commit(f(x,y))^{\{1\}}$.*

The four-round protocol uses two cards.

Protocol 16. *(Asymmetric card private input XOR protocol)*
Input: Alice has x. Bob has y.
Output: $commit(x \oplus y)^{\{1\}}$.

1. *Alice makes $S_1 = commit(x)^{\{1\}}$ and sends S_1 to Bob.*
2. *Bob turns upside down S_1 if $y = 1$. The result $S_2 = commit(x \oplus y)^{\{1\}}$.*

The two-round protocol uses one card.

The correctness and security proofs are just the same as the ones for the standard deck of symmetric card protocols.

6 Conclusion

This paper showed AND, XOR, and copy protocols that use a standard deck of cards. The number of cards used by the protocols is the minimum. The results show the effectiveness of private operations. One of the remaining problems is obtaining protocols when a player is malicious.

Acknowledgements

The authors would like to thank anonymous reviewers for their valuable comments to improve the manuscript.

Declarations

The first author is a board member of this special issue.

No funding was received for conducting this study.

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] 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)

[2] Koch, A.: The landscape of optimal card-based protocols. Mathematical Cryptology **1**(2), 115–131 (2021)

[3] Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security **13**(1), 15–23 (2014)

[4] Cheung, E., Hawthorne, C., Lee, P.: CS 758 Project: Secure Computation with Playing Cards. http://cdchawthorne.com/writings/secure_playing_cards.pdf (2013)

[5] Mizuki, T.: Applications of card-based cryptography to education. In: IEICE Techinical Report ISEC2016-53, pp. 13–17 (2016). (In Japanese)

[6] Hanaoka, G., Iwamoto, M., Watanabe, Y., Mizuki, T., Abe, Y., Shinagawa, K., Arai, M., Yanai, N.: Physical and visual cryptography to accelerate social implementation of advanced cryptographic technologies. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 214–228 (2023). (In Japanese)

[7] Boer, B.: More efficient match-making and satisfiability the five card trick. In: Proc. of EUROCRYPT '89, LNCS Vol. 434, pp. 208–217 (1990)

[8] 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)

[9] 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)

[10] 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)

[11] 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)

[12] Ono, H., Manabe, Y.: Card-based cryptographic logical computations using

private operations. New Generation Computing **39**(1), 19–40 (2021)

[13] Ono, H., Manabe, Y.: Minimum round card-based cryptographic protocols using private operations. Cryptography **5**(3) (2021)

[14] 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 (2019). Springer

[15] 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)

[16] 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)

[17] 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)

[18] Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theoretical Computer Science **622**, 34–44 (2016)

[19] Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**(2), 361–371 (2018)

[20] 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)

[21] 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)

[22] 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)

[23] Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. Theoretical Computer Science **887**, 99–110 (2021)

[24] 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)

[25] 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)

[26] 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. APKC '21, pp. 13–22. Association for Computing Machinery, New York, NY, USA (2021)

[27] 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 (2021). Springer

[28] Marcedone, A., Wen, Z., Shi, E.: Secure Dating with Four or Fewer Cards. IACR Cryptology ePrint Archive, Report 2015/1031 (2015)

[29] 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 (2021). Springer

[30] Abe, Y., Mizuki, T., Sone, H.: Committed-format and protocol using only random cuts. Natural Computing, 1–7 (2021)

[31] 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)

[32] Shikata, H., Miyahara, D., Mizuki, T.: Few-helping-card protocols for some wider class of symmetric boolean functions with arbitrary ranges. In: Proceedings of the 10th ACM International Workshop on ASIA Public-Key Cryptography (APKC), pp. 33–41 (2023)

[33] Shikata, H., Toyoda, K., Miyahara, D., Mizuki, T.: Card-minimal protocols for symmetric boolean functions of more than seven inputs. In: Seidl, H., Liu, Z., Pasareanu, C.S. (eds.) Proc. of 18th International Conference on Theoretical Aspects of Computing (ICTAC 2022) LNCS Vol. 13572, pp. 388–406. Springer, Cham (2022)

[34] Yoshida, T., Tanaka, K., Nakabayashi, K., Chida, E., Mizuki, T.: Upper bounds on the number of shuffles for two-helping-card multi-input and protocols. In: Proc. of 22nd International Conference on Cryptology and Network Security(CANS 2023), LNCS Vol. 14342, pp. 211–231 (2023). Springer

[35] Dvořák, P., Kouckỳ, M.: Barrington Plays Cards: The Complexity of Card-based Protocols. arXiv preprint arXiv:2010.08445 (2020)

[36] Koch, A., Walzer, S.: Private function evaluation with cards. New Generation Computing **40**(1), 115–147 (2022)

[37] 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)

[38] 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)

[39] 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)

[40] Nuida, K.: Efficient card-based millionaires' protocols via non-binary input encoding. In: Proc. of 18th Internatioal Workshop on Security(IWSEC 2023), LNCS Vol. 14128, pp. 237–254. Springer, Cham (2023)

[41] 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)

[42] 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)

[43] 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)

[44] 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 (2018). IEEE

[45] 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 (2021). Springer

[46] 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)

[47] 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)

[48] ABE, Y., NAKAI, T., WATANABE, Y., IWAMOTO, M., OHTA, K.: A computationally efficient card-based majority voting protocol with fewer cards in the private model. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E106.A**(3), 315–324 (2023) https://doi.org/10.1587/transfun.2022CIP0021

[49] 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)

[50] 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)

[51] 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)

[52] 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, Cham (2021)

[53] 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)

[54] 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 (2023). Springer

[55] Manabe, Y., Shinagawa, K.: Free-xor in card-based garbled circuits. In: Proc. of 22nd International Conference on Cryptology and Network Security (CANS 2023), LNCS Vol. 14342, pp. 232–248. Springer, ??? (2023)

[56] 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)

[57] 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)

[58] 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 (2020). Springer

[59] Takashima, K., Miyahara, D., Mizuki, T., Sone, H.: Actively revealing card attack on card-based protocols. Natural Computing **21**(4), 615–628 (2022)

[60] Mizuki, T., Shizuya, H.: Practical card-based cryptography. In: Proc. of 7th International Conference on Fun with Algorithms(FUN2014), LNCS Vol. 8496, pp. 313–324 (2014)

[61] Mizuki, T., Komano, Y.: Information leakage due to operative errors in card-based protocols. Information and Computation **285**, 104910 (2022)

[62] Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: Proc. of 10th International Conference on Fun with Algorithms (FUN 2020) (2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik

[63] Manabe, Y., Ono, H.: Card-based cryptographic protocols with malicious players using private operations. New Generation Computing **40**(1), 67–93 (2022)

[64] Morooka, T., Manabe, Y., Shinagawa, K.: Malicious player card-based cryptographic protocols with a standard deck of cards using private operations. In: Proc. of 18th International Conference on Information Security Practice and Experience (ISPEC 2023), LNCS Vol. 14341, pp. 332–346. Springer, ??? (2023)

[65] Niemi, V., Renvall, A.: Solitaire zero-knowledge. Fundamenta Informaticae **38**(1, 2), 181–188 (1999)

[66] Mizuki, T.: Efficient and secure multiparty computations using a standard deck of playing cards. In: Proc. of 15th International Conference on Cryptology and Network Security(CANS 2016), LNCS Vol.10052, pp. 484–499 (2016). Springer

[67] Koch, A., Schrempp, M., Kirsten, M.: Card-based cryptography meets formal verification. New Generation Computing **39**(1), 115–158 (2021)

[68] Koyama, H., Miyahara, D., Mizuki, T., Sone, H.: A secure three-input and protocol with a standard deck of minimal cards. In: Santhanam, R., Musatov, D. (eds.) Proc. of 16th International Computer Science Symposium in Russia (CSR 2021), LNCS Vol. 12730, pp. 242–256. Springer, Cham (2021)

[69] Miyahara, D., Mizuki, T.: Secure computations through checking suits of playing

cards. In: Proc. of International Workshop on Frontiers in Algorithmic Wisdom (IJTCS-FAW 2022), LNCS Vol. 13461, pp. 110–128 (2022). Springer

[70] Ruangwises, S.: Two standard decks of playing cards are sufficient for a zkp for sudoku. New Generation Computing **40**(1), 49–65 (2022)

[71] Ruangwises, S., Itoh, T.: Physical zkp for makaro using a standard deck of cards. In: Proc. of 17th International Conference on Theory and Applications of Models of Computation (TAMC 2022), LNCS Vol. 13571, pp. 43–54 (2022). Springer

[72] Kurosawa, K., Shinozaki, T.: Compact card protocol. In: Proc. of 2017 Symposium on Cryptography and Information Security(SCIS 2017), pp. 1–26 (2017). (In Japanese)

[73] Shirouchi, S., Nakai, T., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for logic gates utilizing private permutations. In: Proc. of 2017 Symposium on Cryptography and Information Security(SCIS 2017), pp. 1–22 (2017). (In Japanese)

[74] Shinagawa, K., Nuida, K., Nishide, T., Hanaoka, G., Okamoto, E.: Committed and protocol using three cards with more handy shuffle. In: Proc. of 2016 International Symposium on Information Theory and Its Applications (ISITA 2016), pp. 700–702 (2016)

[75] Suga, Y.: A classification for commutative three-element semigroups with local xor structure and its implementability of card-based protocols. In: 2023 International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan), pp. 543–544 (2023). IEEE

[76] Suga, Y.: Poster: A card-based protocol that lets you know how close two parties are in their opinions (agree/disagree) by using a four-point likert scale. In: International Conference on Applied Cryptography and Network Security, pp. 716–721 (2023). Springer

[77] Suga, Y.: Security considerations for the fourth data over non-committed 3-valued card-based protocols. In: 2023 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC), pp. 1–4 (2023). IEEE

[78] Suga, Y.: How to implement non-committed card protocols to realize and operations satisfying the three-valued logics. In: 2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW), pp. 370–374 (2022). IEEE

[79] Koch, A.: Cryptographic protocols from physical assumptions. PhD thesis, Karlsruhe Institute of Technology, Germany (2019)

[80] 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, pp. 256–274 (2021). Springer

[81] Manabe, Y.: Survey: Card-based cryptographic protocols to calculate primitives of boolean functions. International Journal of Computer & Software Engineering **27**(1), 178 (2022)