



Article

Minimum Round Card-Based Cryptographic Protocols Using Private Operations [†]

Hibiki Ono and Yoshifumi Manabe *

Faculty of Informatics, Kogakuin University, Shinjuku, Tokyo 163-8677, Japan; j114026@g.kogakuin.jp

* Correspondence: manabe@cc.kogakuin.ac.jp

[†] This paper is an extended version of our paper published in 14th International Workshop on Data Privacy Management (DPM 2019), Luxembourg, September 26–27, LNCS, Springer, 2019; Volume 11737, pp. 156–173.

Abstract: This paper shows new card-based cryptographic protocols with the minimum number of rounds, using private operations under the semi-honest model. Physical cards are used in card-based cryptographic protocols instead of computers to achieve secure multiparty computation. Operations that a player executes in a place where the other players cannot see are called private operations. Using three private operations—private random bisection cuts, private reverse cuts, and private reveals—the calculations of two variable Boolean functions and copy operations were realized with the minimum number of cards. Though the number of cards has been discussed, the efficiency of these protocols has not been discussed. This paper defines the number of rounds to evaluate the efficiency of the protocols, using private operations. Most of the meaningful calculations using private operations need at least two rounds. This paper presents a new two-round committed-input, committed-output logical XOR protocol, using four cards. Then, we show new two-round committed-input, committed-output logical AND and copy protocols, using six cards. Even if private reveal operations are not used, logical XOR, logical AND, and copy operations can be executed with the minimum number of rounds. Protocols for general n -variable Boolean functions and protocols that preserve an input are also shown. Lastly, protocols with asymmetric cards are shown.

Keywords: multiparty secure computation; card-based cryptographic protocols; private operations; logical computations; copy; round



Citation: Ono, H.; Manabe, Y. Minimum Round Card-Based Cryptographic Protocols Using Private Operations. *Cryptography* **2021**, *5*, 17. <https://doi.org/10.3390/cryptography5030017>

Academic Editor: Siamak F. Shahandashti

Received: 14 April 2021

Accepted: 11 July 2021

Published: 13 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

1.1. Motivation

Card-based cryptographic protocols [1–3] have been proposed in which physical cards are used instead of computers to securely calculate values. They can be used when computers cannot be used or when users cannot trust cryptographic software in the computers. The protocols can be used to teach the basics of cryptography [4,5]. den Boer [6] first showed a five-card protocol to securely calculate the logical AND of two inputs. Since then, many protocols have been proposed to calculate Boolean functions [7–9] and specific computations, such as the millionaires' problem [10–12], realizing Turing machines [13,14], voting [15–18], random permutation [19–22], grouping [23], ranking [24], lottery [25], proof of knowledge of a puzzle solution [26–34], and so on. This paper considers calculations of Boolean functions and the copy operation under the semi-honest model.

There are several types of protocols with regard to the inputs and outputs of the computations. The first type is committed inputs [7], where the inputs are given as committed values. The players do not know the input values. The other type is non-committed inputs [35,36], where players give their private inputs to the protocol, using private input operations. The private input operations were also used in the millionaires' problem [11]. Protocols with committed inputs are desirable since they can be used for non-committed inputs: each player can give their private input value as a committed value.

Some protocols output their computation results as committed values [7]. The result is unknown to the players unless the players open the output cards. The other type of protocols [6,37] output the result as a non-committed value, that is, the final result is obtained only by opening cards. Protocols with committed outputs are desirable since the committed output result can be used as an input to another computation. If further calculations are unnecessary, the players just open the committed outputs and obtain the result. Thus, this paper discusses protocols with committed inputs and committed outputs.

An example of a calculation with committed inputs is a matching service between men and women. The matching service provider does not allow direct communication between the clients until the matching is over. A client, Anne, receives information about a candidate, Bruce, from her agent, Alice. Anne sends the reply of acceptance/rejection to Alice, but Anne does not want the matching service provider agents to know the reply. Bruce also receives information about Anne from his agent Bob. Bruce sends his reply of acceptance/rejection to Bob, but Bruce does not want the matching service provider agents to know the reply. Alice and Bob must calculate whether the matching is successful or not without knowing the inputs. In this case, a calculation with committed inputs is necessary. To prevent malicious activities by the players, Anne observes all the actions executed by Alice. Bruce observes all the actions executed by Bob. If a player executes some action that is not allowed, the observing person can point out the misbehavior. Thus, Alice and Bob become semi-honest players. Note that Anne (Bruce) cannot observe Bob's (Alice's) actions. If a person observes both players' actions, the person can know a secret value.

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 so that the operations cannot be seen by the other players. Private operations are shown to be the most powerful primitives in card-based cryptographic protocols. They were first introduced to solve the millionaires' problem [10]. Using private operations, committed-input and committed-output logical AND, logical XOR, and copy protocols can be achieved with the minimum number of cards [9]. Thus, this paper considers protocols using private operations.

The number of cards is the space complexity of the card-based protocols. Thus, the time complexity must also be evaluated. Some studies have been done for protocols that do not use private operations [38]. As for the protocols using private operations, the number of rounds, defined in Section 2, is the most appropriate criterion to evaluate the time complexity. Roughly speaking, the number of rounds counts the number of sending cards between players. Since each private operation is relatively simple, sending cards between players and setting up so that the cards are not seen by the other players is the dominating time to execute private operations. Thus, this paper discusses the number of rounds of card-based protocols, using private operations.

This paper shows logical XOR, logical AND, and copy protocols with the minimum number of rounds. The summary of results are shown in Tables 1–3. Note that the protocols in [7] need one shuffle by each player; thus, the actual execution time is larger than that in this paper, though the number of rounds is the same.

This paper then shows variations of the protocols that do not use private reveals. Since a private reveal obtains private values, mistakes of private reveals are fatal for security. Thus, it would be better if every reveal operation is publicly executed and verified by multiple players. Thus, we consider protocols that do not use private reveal operations. We show that we can obtain protocols without increasing the number of rounds or cards, even if we cannot use private reveals.

Next, we show protocols that preserve an input. In usual logical AND protocols, the input bits are lost. If one of the inputs is not lost, the input bit can be used for further computations [39]. This paper shows the number of rounds of protocols that preserve an input. Lastly, this paper shows that the protocols can be executed using asymmetric cards.

Table 1. Comparison of XOR protocols, using private operations.

Article	# of Rounds	# of Cards	Preserving an Input	Private Reveal
[9]	3	4	No	Use
[9]	3	4	Yes	Use
[7]	2	4	No	Does not use
Protocol 2	2	4	No	Use
Protocol 6	2	4	No	Does not use
Protocol 12	3	4	Yes	Use
Protocol 13	3	4	Yes	Does not use

Table 2. Comparison of AND protocols, using private operations.

Article	# of Rounds	# of Cards	Preserving an Input	Private Reveal
[9]	3	4	No	Use
[9]	3	6	Yes	Use
[9]	5	4	Yes	Use
[7]	2	6	No	Does not use
[39]	3	6	Yes	Does not use
Protocol 3	2	6	No	Use
Protocol 7	3	4	No	Does not use
Protocol 8	2	6	No	Does not use
Protocol 14	3	6	Yes	Use
Protocol 15	3	6	Yes	Does not use
Protocol 16	5	4	Yes	Does not use

Table 3. Comparison of copy protocols ($m = 2$), using private operations.

Article	# of Rounds	# of Cards	Private Reveal
[9]	3	4	Use
[7]	2	6	Does not use
Protocol 4	2	6	Use
Protocol 9	3	4	Does not use
Protocol 10	2	6	Does not use

In Section 2, basic notations, the private operations introduced in [9], and the definition of the rounds are shown. Section 3 shows two-round XOR, AND, and copy protocols. Section 4 shows the protocols that do not use private reveals. Section 5 shows protocols that preserve an input. Section 6 shows parallel execution of the protocols. Section 7 shows protocols with asymmetric cards. Section 8 concludes the paper.

1.2. Related Works

Many studies have been done for calculating Boolean functions without private operations; den Boer [6] first showed a five-card protocol to securely calculate logical AND of two inputs. Since then, several protocols to calculate logical AND of two committed inputs have been shown [40–42], but they use more than six cards. Mizuki et al. [7] showed a logical AND protocol that uses six cards. It was proved that it is impossible to calculate

logical AND with less than six cards when we use closed and uniform shuffles [43]. When it is allowed to use a special kind of shuffle that is not closed or uniform, the minimum number of cards of logical AND protocols is decreased to five [8,44,45]. In addition, when Las Vegas protocols are allowed, logical AND protocols with five or four cards were shown [2,37,46].

For making copies of an input bit, Mizuki et al. showed a protocol with six cards [7]. A five-card protocol was shown that uses non-uniform shuffles [47].





Mizuki et al. [7] showed a logical XOR protocol that uses four cards, which is the minimum. An XOR protocol that uses random cuts was shown [48].



Several other protocols, such as computations of many inputs [49–53], computing any Boolean functions [13,14,39,54], and two-bit output functions [55], were shown. Protocols using other types of cards were also shown [56–63].


2. Preliminaries

2.1. Basic Notations

This section gives the notations and basic definitions of card-based protocols. Most of this paper is based on a standard two-color card model.

In the two-color card model, there are two kinds of marks:  and . Cards of the same marks cannot be distinguished. In addition, the back of both types of cards is . It is impossible to determine the mark in the back of a given card of .

One bit data are represented by two cards as follows:  = 0 and  = 1.

One pair of cards that represents one bit $x \in \{0, 1\}$, whose face is down, is called a commitment of x , and denoted as $commit(x)$. It is written as follows: . Note that

when these two cards are swapped, $commit(\bar{x})$ can be obtained. Thus, logical negation can be calculated 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 $S = s_1, s_2, \dots, s_n$, where s_i is the i -th card of the sequence.

$S = \underbrace{\boxed{?}}_{s_1} \underbrace{\boxed{?}}_{s_2} \underbrace{\boxed{?}}_{s_3} \dots \underbrace{\boxed{?}}_{s_n}$. 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 multiple players. Throughout this paper, all players are semi-honest, that is, they obey the rule of the protocols but try to obtain information x of $commit(x)$. There is no collusion among players executing one protocol together. No player wants any other player to obtain information on committed values. Since the private operations are executed in a place where the other players cannot see, malicious actions might be easily executed, hence the semi-honest model might not be inappropriate in some cases. To consider malicious actions, two techniques are considered. The first technique executes more than two players [9]. For example, three players—Alice, Bob, and Carol—execute one protocol together. Alice watches Bob’s private operations, Bob watches Carol’s private operations, and Carol watches Alice’s private operations. If a player executes a malicious action, it is detected by the corresponding watching player. The method can be easily introduced to the protocols shown in this paper.

The second technique uses envelopes as an additional tool [64]. The technique is used when the number of players is two. The cards are put into envelopes and sealed in a public place. When the seals are illegally opened during a private operation, they can be detected by the other player. Using the envelopes, all malicious private actions during AND, XOR, and copy protocols are detected or automatically corrected [64]. The technique can also be applied to this paper’s protocols. Thus, this paper assumes the semi-honest model to show the fundamental protocols.

In Section 7, protocols with asymmetric cards are written. When we use cards whose face is not symmetric, such as  and , but the back is symmetric, one-bit data can be

represented by one card as $\spadesuit = 0$ and $\heartsuit = 1$. Protocols with this type of card are first considered in [56] and then several protocols are shown in [9,36,60]. This paper shows minimum round protocols using private operations.

2.2. Private Operations

We show three private operations introduced in [9] for the two-color model: 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 $S_0 = s_1, s_2, \dots, s_{2m}$. A player selects a random bit $b \in \{0, 1\}$ and outputs the following:

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

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 $S_0 = \text{commit}(x)$, 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 $\underbrace{\boxed{?} \boxed{?}}_{\bar{x}}$.

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

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

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

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 Definition 1, where a random bit must be newly selected by the player.

Note that in many 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 $S_2 = s_1, s_2, \dots, s_{2m}$ and a bit $b \in \{0, 1\}$. A player outputs the following:

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

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 $\text{commit}(x)$ and Bob executes a private reveal on the bit. Since the committed bit is randomized by the bit b selected by Alice, the revealed bit is $x \oplus b$. 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. Definition of Round

The space complexity of card-based protocols is evaluated by the number of cards. We define the number of rounds as a criterion to evaluate the time complexity of card-based protocols, using private operations. 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 for which the result is unknown to the player themselves 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.

Let us show an example of a protocol execution and the number of rounds.

Protocol 1. (AND protocol in [9])

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 the following:

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

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 first round ends at the instant when Alice sends $commit(x')$ and $commit(y)$ to Bob. The second round begins at receiving the cards by Bob. The second round ends at the instant when Bob sends S_2 to Alice. The third round begins upon 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.

The minimum number of rounds of most protocols is two. Suppose that the number of rounds is one. Suppose that a player, for example, Alice, has some (or all) of the final outputs of the protocol. Since the number of rounds is one, sending cards between players is not executed. Thus, all the operations to obtain Alice's outputs are executed by Alice. Thus, Alice knows the relation between the committed inputs and Alice's outputs. If the output cards are faced up to know the results, Alice knows the private input values. Therefore, most protocols need at least two rounds for the privacy of the committed inputs.

2.4. Our Results

The protocols in [9] are three rounds and use four cards. We show a two-round logical XOR protocol, using four cards. Then, we show two-round logical AND and copy protocols, using six cards. Though the number of cards is increased, the number of rounds is minimal. Another advantage of these two-round protocols is that each player does not need to remember the random bit. In the protocols in [9], a player needs to remember the random bit until the player receives the cards again to execute a private reverse cut. If a player replies late, the other player must remember the random bit for a very long time. If a player executes many instances of the protocols with many players in parallel, it is hard for the player to remember so many random values. In the two-round protocols, the first

player can exit from the protocol after she sends the cards to the other player. Note that Alice obtains the final result by the three-round protocols in [9] but Bob obtains the final result by the two-round protocols in this paper. These protocols can be used only if this change is acceptable by both players. Note that the two-round protocols with the four card logical XOR, six card logical AND, and copy with private operations are implicitly shown by [7] since one public shuffle used in the paper can be realized by two private shuffles by the two players. This paper shows another type of protocol with fewer shuffles.

The above two-round protocols do not use private reverse cuts, using a remembered bit. Thus, there is a question of whether we can obtain protocols without another type of private operation. This paper also answers this question. We show protocols that do not use private reveals. There is a concern in using this primitive: a player might make the mistake of opening cards that are not allowed and obtaining private values. Since no other player sees the operation, it is hard to detect or prevent such a mistake. If private reveals are not executed at all, protections, such as putting each card in an envelope, can be put in place to prove that incorrectly opening cards is not executed during the private operations. Thus, it would be better if all reveals are publicly executed. Even if we do not use private reveals, the number of rounds is unchanged for logical XOR and copy protocols. For AND and copy, both the (1) two-round and six card protocol and (2) three-round and four card protocol can be obtained, even without private reveals.

Next, we show protocols that preserve an input. In most protocols, input values are lost at the end of the protocol. If an input is not lost, the input can be used for further computations, using the same input value. Thus, protocols that preserve an input are considered [39]. Protocols that preserve an input are shown.

Last, we show protocols with asymmetric cards. By using asymmetric cards, the numbers of cards are halved for all protocols.

3. XOR, AND, and Copy with the Minimum Number of Rounds

This section shows our new two-round protocols for XOR, AND, and copy.

These protocols do not use private reverse cuts, using the remembered random bit. Thus, the first player, Alice, does not need to remember the random bit b after she sends the cards to the other player.

3.1. XOR Protocol

Protocol 2. (XOR protocol with the minimum number of rounds)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \oplus y)$.

1. Alice executes a private random bisection cut on input $S_0 = \text{commit}(x)$ and $S'_0 = \text{commit}(y)$, using the same random bit b . Let the output be $S_1 = \text{commit}(x')$ and $S'_1 = \text{commit}(y')$, respectively. Note that $x' = x \oplus b$ and $y' = y \oplus b$. Alice sends S_1 and S'_1 to Bob.
2. Bob executes a private reveal on $S_1 = \text{commit}(x')$. Bob executes a private reverse cut on S'_1 using x' . Let the result be S_2 . Bob outputs S_2 .

The protocol is two rounds.

Theorem 1. *The XOR protocol is correct and secure. It uses the minimum number of cards.*

Proof. Correctness: Alice sends $\text{commit}(x \oplus b)$ and $\text{commit}(y \oplus b)$ to Bob. Bob swaps the pair of $\text{commit}(y \oplus b)$ if $x \oplus b = 1$. Thus, the output S_2 is $(y \oplus b) \oplus (x \oplus b) = x \oplus y$. Therefore, the output is correct.

Alice and Bob's security: Alice sees no open cards. Thus, Alice obtains no information. Bob sees $x \oplus b$. Since b is a random value that Bob does not know, Bob obtains no information about x .

The number of cards: At least four cards are necessary for any protocol to input x and y . This protocol uses no additional cards other than the input cards. \square

Note that though the same bit b is used to randomize x and y , it is not a security problem because $y \oplus b$ is not opened.

The number of rounds is the minimum. Mizuki et al. showed a four-card protocol with one public shuffle [7]. Since one public shuffle can be changed to two private shuffles by each player, the minimum number of rounds is also achieved by their protocol. However, the protocol needs two shuffles; thus, our new protocol is more simple. A comparison of committed-input, committed-output XOR protocols using private operations is shown in Table 1.

3.2. AND Protocol

Protocol 3. (AND protocol with the minimum number of rounds)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \wedge y)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$ and $S'_0 = \text{commit}(0) || \text{commit}(y)$ using the same random bit b . Two new cards are used to set $\text{commit}(0)$. Let the output be $S_1 = \text{commit}(x')$ and S'_1 , respectively. Note the following:

$$S'_1 = \begin{cases} \text{commit}(y) || \text{commit}(0) & \text{if } b = 1 \\ \text{commit}(0) || \text{commit}(y) & \text{if } b = 0 \end{cases}$$

Alice sends S_1 and S'_1 to Bob.

2. Bob executes a private reveal on S_1 . Bob executes a private reverse selection on S'_1 using x' . Let the selected cards be S_2 . Bob outputs S_2 as the result.

The protocol is two rounds. The protocol uses six cards since two new cards are used to set $\text{commit}(0)$.

Theorem 2. The AND protocol is correct and secure.

Proof. Correctness: The desired output can be represented as follows.

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

Bob outputs $\text{commit}(y)$ as S_2 when $(x', b) = (0, 1)$ or $(1, 0)$. Since $x' = x \oplus b$, these cases equal to $x = 1$. Bob outputs $\text{commit}(0)$ as S_2 when $(x', b) = (0, 0)$ or $(1, 1)$. Since $x' = x \oplus b$, these cases equal to $x = 0$. Thus, the output is correct.

Alice and Bob's security is the same as in the XOR protocol. \square

The number of rounds is the minimum. Mizuki et al. showed a six-card protocol with one public shuffle [7]. Since one public shuffle can be changed to two private shuffles by each player, the minimum number of rounds is also achieved by their protocol. However, the protocol needs two shuffles, thus our new protocol is simple. Comparison of committed-input, committed-output logical AND protocols using private operations are shown in Table 2.

3.3. Copy Protocol

Next, we show a new copy protocol with the minimum number of rounds.

Protocol 4. (Copy protocol with the minimum number of rounds)

Input: $\text{commit}(x)$.

Output: m copies of $\text{commit}(x)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$. Let the output be $S_1 = \text{commit}(x')$. Alice sets S'_1 as m copies of $\text{commit}(b)$, where b is the bit selected in the random bisection cut. Note that $x' = x \oplus b$. Alice sends S_1 and S'_1 to Bob.

2. Bob executes a private reveal on S_1 and obtains x' . Bob executes a private reverse cut on each pair of S'_1 using x' . Let the result be S_2 . Bob outputs S_2 .

The protocol is two rounds. The protocol uses $2m + 2$ cards.

Theorem 3. *The copy protocol is correct and secure.*

Proof. Correctness: Since Bob obtains $x' = x \oplus b$, the output is $b \oplus (x \oplus b) = x$. Alice and Bob's security is the same as in the XOR protocol. \square

Though the number of cards is increased, the number of rounds is the minimum. A comparison of the copy protocols (when $m = 2$) is shown in Table 3. Mizuki et al. showed a six-card protocol with one public shuffle [7]. Since one public shuffle can be changed to two private shuffles by each player, the minimum number of rounds is also achieved by their protocol. However, the protocol needs two shuffles; thus, our new protocol is more simple.

3.4. Any Two-Variable Boolean Functions

Though this paper shows logical AND and logical XOR, any two-variable Boolean functions can also be calculated by a similar protocol. Though the protocol differs, the idea of the construction is similar to the one for the three-round protocol in [9].

Theorem 4. *Any two-variable Boolean function can be securely calculated in two rounds and at most six cards.*

Proof. Any two-variable Boolean function $f(x, y)$ can be written as follows:

$$f(x, y) = \begin{cases} f(1, y) & \text{if } x = 1 \\ f(0, y) & \text{if } x = 0 \end{cases}$$

where $f(1, y)$ and $f(0, y)$ are $y, \bar{y}, 0$, or 1 . However, we need to consider the case when one of $f(1, y)$ and $f(0, y)$ is y or \bar{y} and the other is 0 , or 1 . The reason, written in [9] is as follows. First, consider the case when both of $f(1, y)$ and $f(0, y)$ are 0 or 1 . $(f(1, y), f(0, y)) = (0, 0)$ (or $(1, 1)$) means that $f(x, y) = 0$ (or $f(x, y) = 1$), thus we do not need to calculate f . $(f(1, y), f(0, y)) = (1, 0)$ (or $(0, 1)$) means the $f(x, y) = x$ (or $f(x, y) = \bar{x}$); thus, we do not need to calculate f by a two player protocol.

Next, consider the case when both of $(f(1, y), f(0, y))$ are y (or \bar{y}). This case is when $f(x, y) = y$ (or $f(x, y) = \bar{y}$); thus, we do not need to calculate f by a two-player protocol.

The next case is when $(f(1, y), f(0, y))$ is (y, \bar{y}) or (\bar{y}, y) . $(f(1, y), f(0, y)) = (\bar{y}, y)$ is $x \oplus y$ (XOR). $(f(1, y), f(0, y)) = (y, \bar{y})$ is $\bar{x} \oplus \bar{y}$; thus, this function can be calculated as follows: execute the XOR protocol and NOT is taken to the output. Thus, this function can also be calculated.

The remaining case is when one of $(f(1, y), f(0, y))$ is y or \bar{y} and the other is 0 or 1 . We can modify the first step of the AND protocol and Alice sets as follows:

$$S'_1 = \begin{cases} \text{commit}(f(1, y)) || \text{commit}(f(0, y)) & \text{if } b = 1 \\ \text{commit}(f(0, y)) || \text{commit}(f(1, y)) & \text{if } b = 0 \end{cases}$$

using one $\text{commit}(y)$ and two new cards, since one of $(f(1, y), f(0, y))$ is y or \bar{y} and the other is 0 or 1 . Bob executes a private reveal on $S_1 = \text{commit}(x')$ and selects the left(right) pair if $x' = 0$ ($x' = 1$). Thus, Bob selects $\text{commit}(f(1, y))$ if $x = 1$. Bob selects $\text{commit}(f(0, y))$ if $x = 0$.

Thus, any two-variable Boolean function can be calculated with, at most, six cards and in two rounds. \square

3.5. n -Variable Boolean Functions

Since two-variable Boolean functions, logical negation, and a copy can be executed, any n -variable Boolean function can be calculated by the combination of the above protocols.

As another implementation with more cards, we show that any n -variable Boolean function can be calculated by the following protocol in two rounds, whose technique is similar to the one in [35]. Let f be any n -variable logical function.

Protocol 5. (Protocol for any Boolean function with two rounds)

Input: $\text{commit}(x_i) (i = 1, 2, \dots, n)$.

Output: $\text{commit}(f(x_1, x_2, \dots, x_n))$.

1. Alice executes a private random bisection cut on $\text{commit}(x_i) (i = 1, 2, \dots, n)$. Let the results be $\text{commit}(x'_i) (i = 1, 2, \dots, n)$. $x'_i = x_i \oplus b_i (i = 1, 2, \dots, n)$. Note that one random bit b_i is selected for each $x_i (i = 1, 2, \dots, n)$. Alice generates 2^n commitment $S_{a_1, a_2, \dots, a_n} (a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ as $S_{a_1, a_2, \dots, a_n} = \text{commit}(f(a_1 \oplus b_1, a_2 \oplus b_2, \dots, a_n \oplus b_n))$. Alice sends $\text{commit}(x'_i) (i = 1, 2, \dots, n)$ and $S_{a_1, a_2, \dots, a_n} (a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ to Bob.
2. Bob executes a private reveal on $\text{commit}(x'_i) (i = 1, 2, \dots, n)$. Bob outputs $S_{x'_1, x'_2, \dots, x'_n}$.

The protocol is two-round. The number of cards is $2^{n+1} + 2n$.

Theorem 5. Protocol 5 is correct and secure.

Proof. Correctness: Since $S_{x'_1, x'_2, \dots, x'_n} = \text{commit}(f(b_1 \oplus x'_1, b_2 \oplus x'_2, \dots, b_n \oplus x'_n)) = \text{commit}(f(x_1, x_2, \dots, x_n))$, the output is correct.

Alice and Bob's security is the same as in the XOR protocol. \square

4. Protocols without Private Reveals

This section shows that the above protocols can be executed without the private reveal operations. Since it is hard to prevent mistakes of privately revealing cards that are not allowed, it would be better for all reveal operations to be publicly executed. The general conversion rule is as follows: When Bob executes a private reveal and set a sequence S in the original protocol, Bob executes a private random bisection cut to $\text{commit}(x \oplus b)$ instead. Let b' be the random bit selected by Bob. Then, Bob publicly opens the committed bit and publicly sets a sequence S by the original rule. Bob (and Alice) then executes private reverse cuts to undo the randomization by b and b' .

4.1. XOR Protocol without Private Reveals

Protocol 6. (XOR protocol without private reveals)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \oplus y)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$ and $S'_0 = \text{commit}(y)$ using the same random bit b . Let the output be $S_1 = \text{commit}(x')$ and $S'_1 = \text{commit}(y')$, respectively. Note that $x' = x \oplus b$ and $y' = y \oplus b$. Alice sends S_1 and S'_1 to Bob.
2. Bob executes a private random bisection cut on S_1 and S'_1 using a private bit b' . Let the output be $S_2 = \text{commit}(x'')$ and $S'_2 = \text{commit}(y'')$, respectively. $x'' = x \oplus b \oplus b'$ and $y'' = y \oplus b \oplus b'$ hold. Bob publicly opens S_2 and obtains x'' . Alice can see x'' . Bob publicly sets the following:

$$S_3 = \begin{cases} \text{commit}(\bar{y}'') & \text{if } x'' = 1 \\ \text{commit}(y'') & \text{if } x'' = 0 \end{cases}$$

S_3 is the final result.

The protocol is two rounds.

Theorem 6. *The XOR protocol is correct and secure. It uses the minimum number of cards.*

Proof. Correctness: Bob obtains $commit(x \oplus b \oplus b')$ and $commit(y \oplus b \oplus b')$. Bob sets S_3 as $y'' \oplus x'' = y \oplus b \oplus b' \oplus x \oplus b \oplus b' = x \oplus y$. Thus, the result is correct.

Alice and Bob's security: After Bob executes a private random bisection cut on S_1 , the obtained value $commit(x'') = commit(x \oplus b \oplus b')$. Even if this value is opened, no player can obtain the value of x since Alice knows b and $x \oplus b \oplus b'$ and Bob knows b' and $x \oplus b \oplus b'$.

At least four cards are necessary for any protocol to input x and y . This protocol uses no additional cards other than the input cards. \square

4.2. AND Protocol without Private Reveals

We can consider two kinds of protocols: (1) three rounds and four cards (the minimum number of cards); and (2) two rounds (the minimum number of rounds) and six cards.

Protocol 7. *(AND protocol without private reveals (1))*

Input: $commit(x)$ and $commit(y)$.

Output: $commit(x \wedge y)$.

1. Alice executes a private random bisection cut on $S_0 = commit(x)$. Let the result be $S_1 = commit(x')$. Alice sends S_1 and $S'_0 = commit(y)$ to Bob.
2. Bob executes a private random bisection cut on S_1 , using a private bit b' . Let the result be $S'_1 = commit(x'')$. $x'' = x \oplus b \oplus b'$ holds. Bob publicly opens S'_1 and obtains value x'' . Alice can see x'' . Bob publicly sets the following:

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

Bob then executes a private reverse cut on S_2 using the bit b' generated in the private random bisection cut. Let the result be S_3 . Bob sends S_3 to Alice.

3. Alice executes a private reverse selection on S_3 using the bit b generated in the private random bisection cut. Let the result be S_4 . Alice outputs S_4 .

The number of rounds is three.

Theorem 7. *Protocol 7 is correct, secure, and uses the minimum number of cards.*

Proof. Correctness: The desired output can be represented by Equation (1). When Bob obtains $x'' = 1$, $commit(y)||commit(0)$ is set as S_2 . When Bob obtains $x'' = 0$, $commit(0)||commit(y)$ is set as S_2 . Since Bob executes a private reverse cut on S_2 , $commit(y)||commit(0)$ is given to Alice when $(x'', b') = (1, 0)$ or $(0, 1)$. Since $x'' = x \oplus b \oplus b'$, these cases equal to $x \oplus b = 1$. $commit(0)||commit(y)$ is given to Alice when $(x'', b') = (1, 1)$ or $(0, 0)$. These cases equal to $x \oplus b = 0$.

Thus Alice's output is $commit(y)$ if $(x \oplus b, b) = (1, 0)$ or $(0, 1)$. These cases equal to $x = 1$. Alice's output is $commit(0)$ if $(x \oplus b, b) = (1, 1)$ or $(0, 0)$. These cases equal to $x = 0$. Therefore, the output is correct.

Alice and Bob's security is the same as the XOR protocol without private reveals.

The number of cards: Any committed input protocol needs at least four cards to input. When Bob sets S_2 , the cards used for $commit(x'')$ can be re-used to set $commit(0)$. Thus, the total number of cards is four and the minimum. \square

Protocol 8. *(AND protocol without private reveals (2))*

Input: $commit(x)$ and $commit(y)$.

Output: $commit(x \wedge y)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$ and $S'_0 = \text{commit}(0) || \text{commit}(y)$ using the same random bit b . Two new cards are used to set $\text{commit}(0)$. Let the output be $S_1 = \text{commit}(x')$ and S'_1 , respectively. Note the following:

$$S'_1 = \begin{cases} \text{commit}(y) || \text{commit}(0) & \text{if } b = 1 \\ \text{commit}(0) || \text{commit}(y) & \text{if } b = 0 \end{cases}$$

Alice sends S_1 and S'_1 to Bob.

2. Bob executes a private random bisection cut on S_1 and S'_1 using the same random bit b' . Let the result be S_2 and S'_2 , respectively. Note that $S_2 = \text{commit}(x \oplus b \oplus b')$. Bob publicly opens cards of S_2 and obtains $x'' = x \oplus b \oplus b'$. Alice can see x'' . Bob publicly selects the left pair of S'_2 if $x'' = 0$, otherwise selects the right pair of S'_2 . Bob outputs the pair as the result.

The protocol is two rounds. The protocol uses six cards since two new cards are used to set $\text{commit}(0)$.

Theorem 8. Protocol 8 is correct and secure.

Proof.

$$S'_2 = \begin{cases} \text{commit}(y) || \text{commit}(0) & \text{if } b \oplus b' = 1 \\ \text{commit}(0) || \text{commit}(y) & \text{if } b \oplus b' = 0 \end{cases}$$

Thus Bob outputs $\text{commit}(y)$ if $(b \oplus b', x \oplus b \oplus b') = (1, 0)$ or $(0, 1)$. These cases equal to $x = 1$. Bob outputs $\text{commit}(0)$ if $(b \oplus b', x \oplus b \oplus b') = (1, 1)$ or $(0, 0)$. These cases equal to $x = 0$. Therefore, the output is correct.

Alice and Bob's security is the same as the XOR protocol without private reveals. \square

Using the argument in Section 3.4, any two-variable Boolean function can also be calculated without private reveals by (1) three rounds and four cards, and (2) two rounds and six cards.

4.3. Copy Protocol without Private Reveals

Similar to the AND protocol, we can consider two kinds of copy protocols: (1) three rounds and $2m$ cards (the minimum number of cards); and (2) two rounds (the minimum number of rounds) and $2m + 2$ cards.

Protocol 9. (Copy protocol without private reveals (1))

Input: $\text{commit}(x)$.

Output: m copies of $\text{commit}(x)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$. Let the result be $S_1 = \text{commit}(x')$. Note that $x' = x \oplus b$. Alice sends S_1 to Bob.
2. Bob executes a private random bisection cut on S_1 using a private random bit b' . Let the result be $S_2 = \text{commit}(x'')$. Note that $x'' = x \oplus b \oplus b'$. Bob publicly opens S_2 and obtains x'' . Alice can see x'' . Bob publicly sets m pairs of cards of x'' . Bob faces down the cards. Let the cards be S_3 . Bob executes a private reverse cut on each pair of S_3 using b' . Let the result be S'_3 . Bob sends S'_3 to Alice.
3. Alice executes a private reverse cut on each pair of S'_3 using b . Alice outputs the pairs.

The protocol is three rounds. The protocol uses $2m$ cards since the cards of S_2 are reused to set S_3 .

Theorem 9. Protocol 9 is correct and secure.

Proof. Correctness: Bob makes copies of $\text{commit}(x'')$. Since $x'' = x \oplus b \oplus b'$, after the private reverse cuts by Bob and Alice, the cards are $\text{commit}(x'' \oplus b' \oplus b) = \text{commit}(x \oplus b \oplus b' \oplus b' \oplus b) = \text{commit}(x)$. Thus, the result is correct.

Alice and Bob's security is the same as the XOR protocol without private reveals. \square

Protocol 10. (COPY protocol without private reveals (2))

Input: $\text{commit}(x)$.

Output: m copies of $\text{commit}(x)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$. Let the result be $S_1 = \text{commit}(x')$. Note that $x' = x \oplus b$. Alice privately sets S'_1 as m copies of $\text{commit}(b)$. Alice sends S_1 and S'_1 to Bob.
2. Bob executes a private random bisection cut on S_1 and each pair of S'_1 using a private random bit b' . Let the output be $S_2 = \text{commit}(x'')$ and S'_2 , respectively. Bob publicly opens S_2 and obtains x'' . Alice can see x'' . Bob publicly swaps each pair of S'_2 if $x'' = 1$. Otherwise, Bob does nothing. Let the result be S_3 . Bob outputs S_3 .

The protocol is two rounds. The protocol uses $2m + 2$ cards.

Theorem 10. Protocol 10 is correct and secure.

Proof. Correctness: Bob obtains $\text{commit}(x'')$ and $\text{commit}(b \oplus b')$, where $x'' = x \oplus b \oplus b'$. Bob sets S_3 as $\text{commit}(b \oplus b' \oplus x'') = \text{commit}(b \oplus b' \oplus x \oplus b \oplus b') = \text{commit}(x)$. Thus, the result is correct.

Alice and Bob's security is the same as the XOR protocol without private reveals. \square

4.4. n -Variable Boolean Functions without Private Reveals

Let f be an n -variable Boolean function.

Protocol 11. (Protocol for n -variable Boolean function without private reveal)

Input: $\text{commit}(x_i) (i = 1, 2, \dots, n)$.

Output: $\text{commit}(f(x_1, x_2, \dots, x_n))$.

1. Alice executes a private random bisection cut on $\text{commit}(x_i) (i = 1, 2, \dots, n)$. Let the results be $\text{commit}(x'_i) (i = 1, 2, \dots, n)$. $x'_i = x_i \oplus b_i (i = 1, 2, \dots, n)$. Note that one random bit b_i is selected for each $x_i (i = 1, 2, \dots, n)$. Alice generates 2^n commitment $S_{a_1, a_2, \dots, a_n} (a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ as $S_{a_1, a_2, \dots, a_n} = \text{commit}(f(a_1 \oplus b_1, a_2 \oplus b_2, \dots, a_n \oplus b_n))$. Alice sends $\text{commit}(x'_i) (i = 1, 2, \dots, n)$ and $S_{a_1, a_2, \dots, a_n} (a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ to Bob.
2. Bob executes a private random bisection cut on $\text{commit}(x'_i) (i = 1, 2, \dots, n)$. Note that one random bit b'_i is selected for each $x'_i (i = 1, 2, \dots, n)$. Let $\text{commit}(x''_i) (i = 1, 2, \dots, n)$ be the obtained value. $x''_i = x_i \oplus b_i \oplus b'_i (i = 1, 2, \dots, n)$ is satisfied. Bob privately relocates $S_{a_1, a_2, \dots, a_n} (a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ so that $S'_{a_1, a_2, \dots, a_n} = S_{a_1 \oplus b'_1, a_2 \oplus b'_2, \dots, a_n \oplus b'_n} (a_i \in \{0, 1\}, i = 1, 2, \dots, n)$. The cards satisfy $S'_{a_1, a_2, \dots, a_n} = \text{commit}(f(a_1 \oplus b_1 \oplus b'_1, a_2 \oplus b_2 \oplus b'_2, \dots, a_n \oplus b_n \oplus b'_n))$. Bob publicly reveals $\text{commit}(x''_i) (i = 1, 2, \dots, n)$. Alice can see $x''_i (i = 1, 2, \dots, n)$. Bob publicly selects $S'_{x''_1, x''_2, \dots, x''_n}$.

The protocol is two rounds. The number of cards is $2^{n+1} + 2n$.

Theorem 11. Protocol 11 is correct and secure.

Proof. Correctness: Since $S'_{x''_1, x''_2, \dots, x''_n} = \text{commit}(f(x_1 \oplus b_1 \oplus b'_1 \oplus b_1 \oplus b'_1, x_2 \oplus b_2 \oplus b'_2 \oplus b_2 \oplus b'_2, \dots, x_n \oplus b_n \oplus b'_n \oplus b_n \oplus b'_n)) = \text{commit}(f(x_1, x_2, \dots, x_n))$, the output is correct.

The security of Alice and Bob is the same as the XOR protocol without private reveals. \square

5. Protocols that Preserve an Input

In the above protocols to calculate Boolean functions, the input commitment values are lost. If an input is not lost, the input commitment can be used as an input to another

calculation. Thus, protocols that preserve an input are discussed [39]. For the three-round XOR and AND protocols in [9], protocols that preserve an input were shown [9].

First, consider XOR protocols in Sections 3 and 4.

Protocol 12. (XOR protocol that preserves an input)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \oplus y)$ and $\text{commit}(x)$.

1. Alice executes a private random bisection cut on input $S_0 = \text{commit}(x)$ and $S'_0 = \text{commit}(y)$ using the same random bit b . Let the output be $S_1 = \text{commit}(x')$ and $S'_1 = \text{commit}(y')$, respectively. Note that $x' = x \oplus b$ and $y' = y \oplus b$. Alice sends S_1 and S'_1 to Bob.
2. Bob executes a private reveal on $S_1 = \text{commit}(x')$. Bob executes a private reverse cut on S'_1 , using x' . Let the result be S_2 . Bob outputs S_2 . Bob sends back $S_1 = \text{commit}(x')$ to Alice.
3. Alice executes a private reverse cut on S_1 using b and obtains $\text{commit}(x)$.

In Protocol 2, since $\text{commit}(x')$ is unnecessary after Bob's private reveal, the cards can be sent back to Alice. Alice can recover $\text{commit}(x)$. The number of rounds is increased to three.

Theorem 12. Protocol 12 is correct and secure.

Proof. Correctness: From the correctness of Protocol 2, $\text{commit}(x \oplus y)$ is obtained. $\text{commit}(x)$ can be obtained since $x \oplus b \oplus b = x$.

The security of Alice and Bob is the same as the XOR protocol without input preserving. \square

Protocol 13. (XOR protocol that preserves an input without private reveals)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \oplus y)$ and $\text{commit}(x)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$ and $S'_0 = \text{commit}(y)$ using the same random bit b . Let the output be $S_1 = \text{commit}(x')$ and $S'_1 = \text{commit}(y')$, respectively. Note that $x' = x \oplus b$ and $y' = y \oplus b$. Alice sends S_1 and S'_1 to Bob.
2. Bob executes a private random bisection cut on S_1 and S'_1 using a private bit b' . Let the output be $S_2 = \text{commit}(x'')$ and $S'_2 = \text{commit}(y'')$, respectively. $x'' = x \oplus b \oplus b'$ and $y'' = y \oplus b \oplus b'$ hold. Bob publicly opens S_2 and obtains x'' . Alice can see x'' . Bob publicly sets the following:

$$S_3 = \begin{cases} \text{commit}(\bar{y}'') & \text{if } x'' = 1 \\ \text{commit}(y'') & \text{if } x'' = 0 \end{cases}$$

S_3 is the output. Bob sends back $S_1 = \text{commit}(x')$ to Alice.

3. Alice executes a private reverse cut on S_1 using b and obtains $\text{commit}(x)$.

Theorem 13. Protocol 13 is correct and secure.

Proof. Correctness: From the correctness of Protocol 6, $\text{commit}(x \oplus y)$ is obtained. After Bob publicly reveals S_2 and obtains $x \oplus b \oplus b'$, he can privately recover $\text{commit}(x')$ since he knows b' . Thus, Bob can send back $\text{commit}(x')$ to Alice. $\text{commit}(x)$ can be obtained since $x \oplus b \oplus b = x$.

The security of Alice and Bob is the same as the XOR protocol without input preserving. \square

The protocol is three rounds and uses four cards.

Similarly, the AND protocols in Sections 3 and 4 can be modified to a three-round protocol to preserve an input.

Protocol 14. (AND protocol that preserves an input)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \wedge y)$ and $\text{commit}(x)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$ and $S'_0 = \text{commit}(0) || \text{commit}(y)$ using the same random bit b . Two new cards are used to set $\text{commit}(0)$. Let the output be $S_1 = \text{commit}(x')$ and S'_1 , respectively. Note the following:

$$S'_1 = \begin{cases} \text{commit}(y) || \text{commit}(0) & \text{if } b = 1 \\ \text{commit}(0) || \text{commit}(y) & \text{if } b = 0 \end{cases}$$

Alice sends S_1 and S'_1 to Bob.

2. Bob executes a private reveal on S_1 . Bob executes a private reverse selection on S'_1 using x' . Let the selected cards be S_2 . Bob outputs S_2 as the result. Bob sends back $S_1 = \text{commit}(x')$ to Alice.
3. Alice executes a private reverse cut on S_1 using b and obtains $\text{commit}(x)$.

The number of rounds is three and the number of cards is six. Note that though the characteristics of the Protocol 14 are the same as the protocol in [9], the steps differ: for example, Alice obtains the output in [9] and Bob obtains the output in Protocol 14.

Theorem 14. Protocol 14 is correct and secure.

Proof. Correctness: From the correctness of Protocol 3, $\text{commit}(x \wedge y)$ is obtained. After Bob privately reveals $S_1 = \text{commit}(x')$, he can send back $\text{commit}(x')$ to Alice. $\text{commit}(x)$ can be obtained since $x \oplus b \oplus b = x$.

The security of Alice and Bob is the same as the AND protocol without input preserving. \square

Protocol 15. (AND protocol that preserves an input without private reveals)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \wedge y)$ and $\text{commit}(x)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$. Let the result be $S_1 = \text{commit}(x')$. Alice sends S_1 and $S'_0 = \text{commit}(y)$ to Bob.
2. Bob executes a private random bisection cut on S_1 using a private bit b' . Let the result be $S'_1 = \text{commit}(x'')$. $x'' = x \oplus b \oplus b'$ holds. Bob publicly opens S'_1 and obtains value x'' . Alice can see x'' . Bob publicly sets the following:

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

Bob then executes a private reverse cut on S_2 using the bit b' generated in the private random bisection cut. Let the result be S_3 . Bob sends S_3 and S_1 to Alice.

3. Alice executes a private reverse selection on S_3 using the bit b generated in the private random bisection cut. Let the result be S_4 . Alice outputs S_4 . Alice executes a private reverse cut on S_1 using b and obtains $\text{commit}(x)$.

The number of rounds is three and the number of cards is six.

Theorem 15. Protocol 15 is correct and secure.

Proof. Correctness: From the correctness of Protocol 7, $\text{commit}(x \wedge y)$ is obtained. After Bob publicly reveals S'_1 and obtains $x \oplus b \oplus b'$, he can privately recover $\text{commit}(x')$ since he knows b' . Thus, Bob can send back $\text{commit}(x')$ to Alice. $\text{commit}(x)$ can be obtained since $x \oplus b \oplus b = x$.

The security of Alice and Bob is the same as the AND protocol without input preserving. \square

As for the AND type protocol, to calculate $f(x, y)$, another protocol that preserves an input without additional cards can be obtained, using the technique in [39]. Note that the function f satisfies that one of $(f(0, y), f(1, y))$ is y or \bar{y} and the other is 0 or 1. Otherwise, we do not need to calculate f by the AND type two player protocol. When we execute the four-card AND type protocol without private reveals, two cards are selected by Alice at the final step. The remaining two cards are not used, but they also output some values. The unused two cards' value is the following:

$$\begin{cases} f(0, y) & \text{if } x = 1 \\ f(1, y) & \text{if } x = 0 \end{cases}$$

Thus, the output value is $\text{commit}(\bar{x} \wedge f(1, y) \oplus x \wedge f(0, y))$. The output $f(x, y)$ can be written as $x \wedge f(1, y) \oplus \bar{x} \wedge f(0, y)$. We execute the above XOR protocol that preserves an input without private reveal for these two output values so that $f(x, y)$ is preserved. The output of XOR protocol is $\bar{x} \wedge f(1, y) \oplus x \wedge f(0, y) \oplus x \wedge f(1, y) \oplus \bar{x} \wedge f(0, y) = f(1, y) \oplus f(0, y)$. Since one of $(f(0, y), f(1, y))$ is y or \bar{y} and the other is 0 or 1, the output is y or \bar{y} (depending on f). Thus, input y can be recovered without additional cards.

Protocol 16. (AND type protocol that preserves an input without private reveals)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(f(x, y))$ and $\text{commit}(y)$.

1. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$. Let the result be $S_1 = \text{commit}(x')$. Alice sends S_1 and $S'_0 = \text{commit}(y)$ to Bob.
2. Bob executes a private random bisection cut on S_1 , using a private bit b' . Let the result be $S'_1 = \text{commit}(x'')$. $x'' = x \oplus b \oplus b'$ holds. Bob publicly opens S'_1 and obtains value x'' . Alice can see x'' . Bob publicly sets the following:

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

Bob then executes a private reverse cut on S_2 using the bit b' generated in the private random bisection cut. Let the result be S_3 . Bob sends S_3 to Alice.

3. Alice executes a private reverse selection on S_3 using the bit b generated in the private random bisection cut. Let the result be S_4 . Alice outputs S_4 . Let S'_4 be the cards that are not selected.
4. Alice and Bob execute the XOR protocol that preserves an input without private reveals (Protocol 12) for S_4 and S'_4 . Let the preserved input, S_4 , be the result. We obtain $\text{commit}(y)$ from the XOR result.

Thus, the protocol achieves preserving an input by four cards. The protocol does not use private reveals. The AND type protocol needs three rounds and the XOR protocol that preserves an input needs three rounds. The last round of the AND type protocol and the first round of the XOR protocol are executed by Alice, thus they can be done in one round. Therefore, the total number of rounds is five.

Theorem 16. Protocol 16 is correct and secure.

Proof. Correctness: From the correctness of Protocol 7, $\text{commit}(f(x, y))$ is obtained as S_4 . Since $S'_4 = \text{commit}(\bar{x} \wedge f(1, y) \oplus x \wedge f(0, y))$, the output of the input preserving XOR is $f(x, y) \oplus \bar{x} \wedge f(1, y) \oplus x \wedge f(0, y) = f(0, y) \oplus f(1, y)$, which is y or \bar{y} . The input $f(x, y)$ is preserved, thus the output can be $\text{commit}(y)$ and $\text{commit}(f(x, y))$

The security of Alice and Bob comes from the security of the AND protocol and the XOR protocol with input preserving. \square

Lastly, we show n -variable Boolean function protocols that preserve inputs.

Protocol 17. (Protocol for n -variable Boolean function that preserve inputs)

Input: $\text{commit}(x_i)(i = 1, 2, \dots, n)$.

Output: $\text{commit}(f(x_1, x_2, \dots, x_n))$ and $\text{commit}(x_i)(i = 1, 2, \dots, n)$.

1. Alice executes a private random bisection cut on $\text{commit}(x_i)(i = 1, 2, \dots, n)$. Let the results be $\text{commit}(x'_i)(i = 1, 2, \dots, n)$. $x'_i = x_i \oplus b_i(i = 1, 2, \dots, n)$. Note that one random bit b_i is selected for each $x_i(i = 1, 2, \dots, n)$. Alice generates 2^n commitment $S_{a_1, a_2, \dots, a_n}(a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ as $S_{a_1, a_2, \dots, a_n} = \text{commit}(f(a_1 \oplus b_1, a_2 \oplus b_2, \dots, a_n \oplus b_n))$. Alice sends $\text{commit}(x'_i)(i = 1, 2, \dots, n)$ and $S_{a_1, a_2, \dots, a_n}(a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ to Bob.
2. Bob executes a private reveal on $\text{commit}(x'_i)(i = 1, 2, \dots, n)$. Bob outputs $S_{x'_1, x'_2, \dots, x'_n}$. Bob sends back $\text{commit}(x'_i)(i = 1, 2, \dots, n)$ to Alice.
3. Alice executes a private reverse cut on $\text{commit}(x'_i)$ using $b_i(i = 1, 2, \dots, n)$. Alice obtains $\text{commit}(x_i)(i = 1, 2, \dots, n)$.

The protocol is three rounds.

Theorem 17. Protocol 17 is correct and secure.

Proof. Correctness: From the correctness of Protocol 5, $\text{commit}(f(x_1, x_2, \dots, x_n))$ is obtained. After Bob privately reveals $\text{commit}(x'_i)(i = 1, 2, \dots, n)$, he can send back $\text{commit}(x'_i)(i = 1, 2, \dots, n)$ to Alice. $\text{commit}(x_i)(i = 1, 2, \dots, n)$ can be obtained since $x_i \oplus b_i \oplus b_i = x_i(i = 1, 2, \dots, n)$.

The security of Alice and Bob is the same as the n -variable Boolean function protocol without input preserving. \square

Protocol 18. (Protocol for n -variable Boolean function that preserves inputs without private reveals)

Input: $\text{commit}(x_i)(i = 1, 2, \dots, n)$.

Output: $\text{commit}(f(x_1, x_2, \dots, x_n))$ and $\text{commit}(x_i)(i = 1, 2, \dots, n)$.

1. Alice executes a private random bisection cut on $\text{commit}(x_i)(i = 1, 2, \dots, n)$. Let the results be $\text{commit}(x'_i)(i = 1, 2, \dots, n)$. $x'_i = x_i \oplus b_i(i = 1, 2, \dots, n)$. Note that one random bit b_i is selected for each $x_i(i = 1, 2, \dots, n)$. Alice generates 2^n commitment $S_{a_1, a_2, \dots, a_n}(a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ as $S_{a_1, a_2, \dots, a_n} = \text{commit}(f(a_1 \oplus b_1, a_2 \oplus b_2, \dots, a_n \oplus b_n))$. Alice sends $\text{commit}(x'_i)(i = 1, 2, \dots, n)$ and $S_{a_1, a_2, \dots, a_n}(a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ to Bob.
2. Bob executes a private random bisection cut on $\text{commit}(x'_i)(i = 1, 2, \dots, n)$. Note that one random bit b'_i is selected for each $x'_i(i = 1, 2, \dots, n)$. Let $\text{commit}(x''_i)(i = 1, 2, \dots, n)$ be the obtained value. $x''_i = x_i \oplus b_i \oplus b'_i(i = 1, 2, \dots, n)$ is satisfied. Bob privately relocates $S_{a_1, a_2, \dots, a_n}(a_i \in \{0, 1\}, i = 1, 2, \dots, n)$ so that $S'_{a_1, a_2, \dots, a_n} = S_{a_1 \oplus b'_1, a_2 \oplus b'_2, \dots, a_n \oplus b'_n}(a_i \in \{0, 1\}, i = 1, 2, \dots, n)$. The cards satisfy that $S'_{a_1, a_2, \dots, a_n} = \text{commit}(f(a_1 \oplus b_1 \oplus b'_1, a_2 \oplus b_2 \oplus b'_2, \dots, a_n \oplus b_n \oplus b'_n))$. Bob publicly reveals $\text{commit}(x''_i)$ and obtains $x''_i(i = 1, 2, \dots, n)$. Alice can see $x''_i(i = 1, 2, \dots, n)$. Bob publicly selects $S'_{x''_1, x''_2, \dots, x''_n}$. Bob privately sets $\text{commit}(x'_i \oplus b'_i)(i = 1, 2, \dots, n)$. Note that $x''_i \oplus b'_i = x_i \oplus b_i(i = 1, 2, \dots, n)$. Bob sends these pairs to Alice.
3. Alice privately executes a private reverse cut on $\text{commit}(x_i \oplus b_i)$ using b_i for each $i(i = 1, 2, \dots, n)$. Alice obtains $\text{commit}(x_i)(i = 1, 2, \dots, n)$.

The protocol is three rounds.

Theorem 18. Protocol 18 is correct and secure.

Proof. Correctness: From the correctness of Protocol 11, $\text{commit}(f(x_1, x_2, \dots, x_n))$ is obtained. After Bob publicly reveals $\text{commit}(x''_i)(i = 1, 2, \dots, n)$, he can send back $\text{commit}(x'_i)(i = 1, 2, \dots, n)$ to Alice. $\text{commit}(x_i)(i = 1, 2, \dots, n)$ can be obtained since $x_i \oplus b_i \oplus b_i = x_i(i = 1, 2, \dots, n)$.

The security of Alice and Bob is the same as the n -variable Boolean function protocol without input preserving. \square

6. Parallel Computations

The XOR protocol and AND-type protocol shown above can be executed in parallel, using the technique in [9]. Consider the case when $\text{commit}(x)$ and $\text{commit}(y_i)(i = 1, 2, \dots, n)$ are given and $\text{commit}(f_i(x, y_i))(i = 1, 2, \dots, n)$ need to be calculated.

For example, we show the case of the minimum round protocols in Protocol 2 and 3. Alice executes a private random bisection cut on $S_0 = \text{commit}(x)$ and $S_i = \text{commit}(f_i(0, y)) \parallel \text{commit}(f_i(1, y))(i = 1, 2, \dots, n)$ (Note that if $f_i(x, y_i) = x \oplus y_i$, $S_i = \text{commit}(y_i)$) using the same random bit b . Let the results be $S'_i(i = 0, 1, 2, \dots, n)$. Alice sends $S'_i(i = 0, 1, 2, \dots, n)$ to Bob. Bob executes a private reveal on S'_0 and obtains $x' = x \oplus b$. Bob executes a private reverse selection (or a private reverse cut) on S'_i using x' and obtains $\text{commit}(f_i(x, y_i))(i = 1, 2, \dots, n)$.

By the procedure, $\text{commit}(f_i(x, y_i))(i = 1, 2, \dots, n)$ are simultaneously obtained in two rounds. Parallel computations can also be considered for the other protocols.

7. Asymmetric Card Protocols

This section shows protocols when we use asymmetric cards shown in Section 2. one bit data can be represented by one card as $\boxed{\clubsuit} = 0$ and $\boxed{\spadesuit} = 1$.

For such an encoding method, a private random bisection cut on a committed bit is changed to turning the card upside-down, 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 to turning the card upside down, according to the given bit. Using these private operations, all protocols shown above work for the asymmetric cards. The numbers of cards used by these protocols are half of the two-color card protocols. For example, XOR, AND, and copy protocols with the minimum number of rounds are shown below.

Protocol 19. (asymmetric card XOR protocol with the minimum number of rounds)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \oplus y)$.

1. Alice randomly selects bit b . If $b=1$, Alice turns $\text{commit}(x)$ and $\text{commit}(y)$ upside down, otherwise does nothing. Let the result be S_1 and S'_1 , respectively. Note that $S_1 = \text{commit}(x')$ and $S'_1 = \text{commit}(y')$, where $x' = x \oplus b$ and $y' = y \oplus b$. Alice sends S_1 and S'_1 to Bob.
2. Bob executes a private reveal on S_1 and obtains x' . If $x' = 1$, Bob privately turns S'_1 upside down and otherwise does nothing. Let the result be S_2 . Bob outputs S_2 .

The protocol is two rounds. The protocol uses two cards.

Protocol 20. (asymmetric card AND protocol with the minimum number of rounds)

Input: $\text{commit}(x)$ and $\text{commit}(y)$.

Output: $\text{commit}(x \wedge y)$.

1. Alice executes a private random bisection cut on $\text{commit}(0) \parallel \text{commit}(y)$ using a random bit b . Let the result be S'_1 . If $b=1$, Alice turns $\text{commit}(x)$ upside down, and otherwise does nothing. Let the output be $S_1 = \text{commit}(x')$. Note that $x' = x \oplus b$. Alice sends S_1 and S'_1 to Bob.
2. Bob executes a private reveal on S_1 . Bob executes a private reverse selection on S'_1 , using x' . Let the selected card be S_2 . Bob outputs S_2 as the result.

The protocol is two rounds. The protocol uses three cards.

Protocol 21. (asymmetric card copy protocol with the minimum number of rounds)

Input: $\text{commit}(x)$.

Output: m copies of $\text{commit}(x)$.

1. Alice randomly selects bit b . If $b=1$, Alice privately turns $\text{commit}(x)$ upside down, otherwise does nothing. Let the result be $S_1 = \text{commit}(x')$. Note that $x' = x \oplus b$. Alice privately sets S'_1 as m copies of $\text{commit}(b)$. Alice sends S_1 and S'_1 to Bob.
2. Bob executes a private reveal on S_1 and obtains x' . Bob privately upside down all cards of S'_1 if $x' = 1$, otherwise does nothing. Let the results be S_2 . Bob outputs S_2 .

The protocol is two rounds. The protocol uses $m + 1$ cards.

Theorem 19. Using asymmetric cards, XOR, AND, and a copy can be realized in two rounds.

Proof. The correctness and security proofs are the same as those for the two-color card protocols. \square

8. Conclusions

This paper proposes round optimal card-based cryptographic protocols, using private operations. We showed protocols without private reveal operations and several variant protocols. This paper contributes the following new results, compared to the conference version: (1) new XOR and copy protocols without private reveals; (2) correctness and security proofs, given to all protocols; (3) parallel computation of the AND/XOR type protocols; and (4) protocols with asymmetric cards.

Further study will include round optimal protocols for the other fundamental problems.

Author Contributions: Basic idea of private operations, H.O.; minimizing the number of rounds, Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mizuki, T.; Shizuya, H. Computational Model of Card-Based Cryptographic Protocols and Its Applications. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2017**, *100*, 3–11. [CrossRef]
2. Koch, A. *The Landscape of Optimal Card-based Protocols*; Report 2018/951; IACR Cryptology ePrint Archive; 2018.
3. Mizuki, T.; Shizuya, H. A formalization of card-based cryptographic protocols via abstract machine. *Int. J. Inf. Secur.* **2014**, *13*, 15–23. [CrossRef]
4. Cheung, E.; Hawthorne, C.; Lee, P. CS 758 Project: Secure Computation with Playing Cards. 2013. Available online: http://cdchawthorne.com/writings/secure_playing_cards.pdf (accessed on 1 April 2021).
5. Mizuki, T. *Applications of Card-Based Cryptography to Education*; IEICE Technical Report ISEC2016-53; IEICE: Tokyo, Japan, 2016; pp. 13–17. (In Japanese)
6. den Boer, B. More efficient match-making and satisfiability the five card trick. In Proceedings of the EUROCRYPT '89, LNCS, Houthalen, Belgium, 10–13 April 1989; Volume 434, pp. 208–217.
7. Mizuki, T.; Sone, H. Six-Card Secure AND and Four-Card Secure XOR. In Proceedings of the 3rd International Workshop on Frontiers in Algorithms (FAW 2009), LNCS, Hefei, China, 20–23 June 2009; Volume 5598, pp. 358–369.
8. Koch, A.; Walzer, S.; Härtel, K. Card-based cryptographic protocols using a minimal number of cards. In Proceedings of the Asiacrypt 2015, LNCS, Auckland, New Zealand, 29 November–3 December 2015; Volume 9452, pp. 783–807.
9. Ono, H.; Manabe, Y. Card-Based Cryptographic Logical Computations Using Private Operations. *New Gener. Comput.* **2021**, *39*, 19–40. [CrossRef]
10. Nakai, T.; Misawa, Y.; Tokushige, Y.; Iwamoto, M.; Ohta, K. How to Solve Millionaires' Problem with Two Kinds of Cards. *New Gener. Comput.* **2021**, *39*, 73–96. [CrossRef]
11. Ono, H.; Manabe, Y. Efficient Card-based Cryptographic Protocols for the Millionaires' Problem Using Private Input Operations. In Proceedings of the 13th Asia Joint Conference on Information Security (AsiaJCIS 2018), Guilin, China, 8–9 August 2018; pp. 23–28.

12. Miyahara, D.; Hayashi, Y.I.; Mizuki, T.; Sone, H. Practical card-based implementations of Yao's millionaire protocol. *Theor. Comput. Sci.* **2020**, *803*, 207–221. [[CrossRef](#)]
13. Dvořák, P.; Koucký, M. Barrington Plays Cards: The Complexity of Card-based Protocols. *arXiv* **2020**, arXiv:2010.08445.
14. Koch, A.; Walzer, S. Private Function Evaluation with Cards. Cryptology ePrint Archive, Report 2018/1113, 2018. Available online: <https://eprint.iacr.org/2018/1113> (accessed on 1 April 2021).
15. Mizuki, T.; Asiedu, I.K.; Sone, H. Voting with a logarithmic number of cards. In Proceedings of the 12th International Conference on Unconventional Computing and Natural Computation (UCNC 2013), LNCS, Milan, Italy, 1–5 July 2013; Volume 7956, pp. 162–173.
16. Nakai, T.; Shirouchi, S.; Iwamoto, M.; Ohta, K. Four Cards Are Sufficient for a Card-Based Three-Input Voting Protocol Utilizing Private sends. In Proceedings of the 10th International Conference on Information Theoretic Security (ICITS 2017), LNCS, Hong Kong, China, 29 November–2 December 2017; Volume 10681, pp. 153–165.
17. Nishida, T.; Mizuki, T.; Sone, H. Securely computing the three-input majority function with eight cards. In Proceedings of the 2nd International Conference on Theory and Practice of Natural Computing (TPNC 2013), LNCS, Cáceres, Spain, 3–5 December 2013; Volume 8273, pp. 193–204.
18. Watanabe, Y.; Kuroki, Y.; Suzuki, S.; Koga, Y.; Iwamoto, M.; Ohta, K. Card-based majority voting protocols with three inputs using three cards. In Proceedings of the 2018 International Symposium on Information Theory and Its Applications (ISITA), Singapore, 28–31 October 2018; pp. 218–222.
19. Ibaraki, T.; Manabe, Y. A More Efficient Card-Based Protocol for Generating a Random Permutation without Fixed Points. In Proceedings of the 3rd International Conference on Mathematics and Computers in Sciences and in Industry (MCSI 2016), Chania, Greece, 27–29 August 2016; pp. 252–257.
20. Ishikawa, R.; Chida, E.; Mizuki, T. Efficient card-based protocols for generating a hidden random permutation without fixed points. In Proceedings of the 14th International Conference on Unconventional Computation and Natural Computation (UCNC 2015), LNCS, Auckland, New Zealand, 30 August–3 September 2015; Volume 9252, pp. 215–226.
21. 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 Trans. Fundam. Electron. Commun. Comput. Sci.* **2018**, *101-A*, 1503–1511. [[CrossRef](#)]
22. Murata, S.; Miyahara, D.; Mizuki, T.; Sone, H. Efficient Generation of a Card-Based Uniformly Distributed Random Derangement. In Proceedings of the 15th International Workshop on Algorithms and Computation (WALCOM 2021), LNCS, Yangon, Myanmar, 28 February–2 March 2021; Springer International Publishing: Cham, Switzerland, 2021; Volume 12635, pp. 78–89.
23. Hashimoto, Y.; Shinagawa, K.; Nuida, K.; Inamura, M.; Hanaoka, G. Secure grouping protocol using a deck of cards. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2018**, *101*, 1512–1524. [[CrossRef](#)]
24. Takashima, K.; Abe, Y.; Sasaki, T.; Miyahara, D.; Shinagawa, K.; Mizuki, T.; Sone, H. Card-based protocols for secure ranking computations. *Theor. Comput. Sci.* **2020**, *845*, 122–135. [[CrossRef](#)]
25. Shinoda, Y.; Miyahara, D.; Shinagawa, K.; Mizuki, T.; Sone, H. Card-Based Covert Lottery. In Proceedings of the 13th International Conference on Information Technology and Communications Security (SecITC 2020), LNCS, Bucharest, Romania, 19–20 November 2020; Springer: Cham, Switzerland, 2020; Volume 12596, pp. 257–270.
26. Sasaki, T.; Miyahara, D.; Mizuki, T.; Sone, H. Efficient card-based zero-knowledge proof for sudoku. *Theor. Comput. Sci.* **2020**, *839*, 135–142. [[CrossRef](#)]
27. Bultel, X.; Dreier, J.; Dumas, J.G.; Lafourcade, P.; Miyahara, D.; Mizuki, T.; Nagao, A.; Sasaki, T.; Shinagawa, K.; Sone, H. Physical Zero-Knowledge Proof for Makaro. In Proceedings of the 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2018), LNCS, Tokyo, Japan, 4–7 November 2018; Volume 11201, pp. 111–125.
28. Miyahara, D.; Sasaki, T.; Mizuki, T.; Sone, H. Card-Based Physical Zero-Knowledge Proof for Kakuro. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2019**, *102*, 1072–1078. [[CrossRef](#)]
29. Dumas, J.G.; Lafourcade, P.; Miyahara, D.; Mizuki, T.; Sasaki, T.; Sone, H. Interactive physical zero-knowledge proof for Norinori. In Proceedings of the 25th International Computing and Combinatorics Conference (COCOON 2019), LNCS, Xi'an, China, 29–31 July 2019; Springer: Cham, Switzerland, 2019; Volume 11653, pp. 166–177.
30. Lafourcade, P.; Miyahara, D.; Mizuki, T.; Sasaki, T.; Sone, H. A Physical ZKP for Slitherlink: How to Perform Physical Topology-Preserving Computation. In Proceedings of the 15th International Conference on Information Security Practice and Experience (ISPEC 2019), LNCS, Kuala Lumpur, Malaysia, 26–28 November 2019; Springer: Cham, Switzerland, 2019; Volume 11879, pp. 135–151.
31. Ruangwises, S.; Itoh, T. Physical Zero-Knowledge Proof for Numberlink Puzzle and k Vertex-Disjoint Paths Problem. *New Gener. Comput.* **2021**, *39*, 3–17. [[CrossRef](#)]
32. Miyahara, D.; Robert, L.; Lafourcade, P.; Takeshige, S.; Mizuki, T.; Shinagawa, K.; Nagao, A.; Sone, H. Card-based ZKP protocols for Takuzu and Juosan. In *Proceedings of the 10th International Conference on Fun with Algorithms (FUN 2020)*; LIPICs Vol. 157; Schloss Dagstuhl-Leibniz-Zentrum für Informatik: Saarbrücken, Germany, 2020.
33. Ruangwises, S.; Itoh, T. Physical Zero-Knowledge Proof for Ripple Effect. In Proceedings of the 15th International Workshop on Algorithms and Computation (WALCOM 2021), LNCS, Yangon, Myanmar, 28 February–2 March 2021; Springer International Publishing: Cham, Switzerland, 2021; Volume 12635, pp. 296–307.

34. Robert, L.; Miyahara, D.; Lafourcade, P.; Mizuki, T. Interactive Physical ZKP for Connectivity: Applications to Nurikabe and Hitori. In *Proceedings of the 17th International Conference on Computability in Europe (CiE 2021), Virtual Event*; LNCS; Springer International Publishing: Cham, Switzerland, 2021; Volume 12813, pp. 373–384.
35. Kurosawa, K.; Shinozaki, T. Compact Card Protocol. In *Proceedings of the 2017 Symposium on Cryptography and Information Security (SCIS 2017), Okinawa, Japan, 24–27 January 2017*; IEICE: Tokyo, Japan, 2017. 1A2–6 (In Japanese)
36. Shirouchi, S.; Nakai, T.; Iwamoto, M.; Ohta, K. Efficient Card-based Cryptographic Protocols for Logic Gates Utilizing Private Permutations. In *Proceedings of the 2017 Symposium on Cryptography and Information Security (SCIS 2017), Okinawa, Japan, 24–27 January 2017*; IEICE: Tokyo, Japan, 2017. 1A2–2 (In Japanese)
37. Mizuki, T.; Kumamoto, M.; Sone, H. The five-card trick can be done with four cards. In *Proceedings of the Asiacrypt 2012, LNCS, Beijing, China, 2–6 December 2012*; Volume 7658, pp. 598–606.
38. Miyahara, D.; Ueda, I.; Hayashi, Y.i.; Mizuki, T.; Sone, H. Evaluating card-based protocols in terms of execution time. *Int. J. Inf. Secur.* **2020**, 1–12. [[CrossRef](#)]
39. Nishida, T.; Hayashi, Y.; Mizuki, T.; Sone, H. Card-based protocols for any boolean function. In *Proceedings of the 15th International Conference on Theory and Applications of Models of Computation (TAMC 2015), LNCS, Singapore, 18–20 May 2015*; Volume 9076, pp. 110–121.
40. Crépeau, C.; Kilian, J. Discreet solitary games. In *Proceedings of the 13th Crypto, LNCS, Santa Barbara, CA, USA, 22–26 August 1993*; Volume 773, pp. 319–330.
41. Niemi, V.; Renvall, A. Secure multiparty computations without computers. *Theor. Comput. Sci.* **1998**, 191, 173–183. [[CrossRef](#)]
42. Stiglic, A. Computations with a deck of cards. *Theor. Comput. Sci.* **2001**, 259, 671–678. [[CrossRef](#)]
43. 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 *Proceedings of the Asiacrypt 2017, Part III, LNCS, Hong Kong, China, 3–7 December 2017*; Volume 10626, pp. 126–155.
44. Ruangwises, S.; Itoh, T. AND Protocols Using Only Uniform Shuffles. In *Proceedings of the 14th International Computer Science Symposium in Russia (CSR 2019), LNCS, Novosibirsk, Russia, 1–5 July 2019*; Volume 11532, pp. 349–358.
45. Nishimura, A.; Nishida, T.; Hayashi, Y.; Mizuki, T.; Sone, H. Card-based protocols using unequal division shuffles. *Soft Comput.* **2018**, 22, 361–371. [[CrossRef](#)]
46. Abe, Y.; Hayashi, Y.i.; Mizuki, T.; Sone, H. Five-Card AND Computations in Committed Format Using Only Uniform Cyclic Shuffles. *New Gener. Comput.* **2021**, 39, 97–114. [[CrossRef](#)]
47. Nishimura, A.; Nishida, T.; Hayashi, Y.; Mizuki, T.; Sone, H. Five-card secure computations using unequal division shuffle. In *Proceedings of the 4th International Conference on Theory and Practice of Natural Computing (TPNC 2015), LNCS, Mieres, Spain, 15–16 December 2015*; Volume 9477, pp. 109–120.
48. Toyoda, K.; Miyahara, D.; Mizuki, T.; Sone, H. Six-Card Finite-Runtime XOR Protocol with Only Random Cut. In *Proceedings of the 7th ACM Workshop on ASIA Public-Key Cryptography, Taipei, Taiwan, 6 October 2020*; pp. 2–8.
49. Mizuki, T. Card-based protocols for securely computing the conjunction of multiple variables. *Theor. Comput. Sci.* **2016**, 622, 34–44. [[CrossRef](#)]
50. Nishida, T.; Hayashi, Y.; Mizuki, T.; Sone, H. Securely computing three-input functions with eight cards. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2015**, 98, 1145–1152. [[CrossRef](#)]
51. Shinagawa, K.; Mizuki, T. The Six-Card Trick: Secure Computation of Three-Input Equality. In *Proceedings of the 21st International Conference on Information Security and Cryptology (ICISC 2018), LNCS, Seoul, Korea, 28–30 November 2018*; Volume 11396, pp. 123–131.
52. Yasunaga, K. Practical Card-Based Protocol for Three-Input Majority. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2020**, E103.A, 1296–1298. [[CrossRef](#)]
53. Manabe, Y.; Ono, H. Card-based Cryptographic Protocols for Three-input Functions Using Private Operations. In *Proceedings of the 32nd International Workshop on Combinatorial Algorithms (IWOCA 2021), LNCS, Ottawa, ON, Canada, 5–7 July 2021*; Springer: Cham, Switzerland, 2021; Volume 12757, pp. 469–484.
54. Shinagawa, K.; Nuida, K. A single shuffle is enough for secure card-based computation of any Boolean circuit. *Discret. Appl. Math.* **2021**, 289, 248–261. [[CrossRef](#)]
55. 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 *Proceedings of the Second International Conference on Cryptology and Malicious Security (Mycrypt 2016), LNCS, Kuala Lumpur, Malaysia, 1–2 December 2016*; Volume 10311, pp. 193–211.
56. Mizuki, T.; Shizuya, H. Practical card-based cryptography. In *Proceedings of the 7th International Conference on Fun with Algorithms (FUN2014), LNCS, Lipari Island, Sicily, Italy, 1–3 July 2014*; Volume 8496, pp. 313–324.
57. Mizuki, T. Efficient and secure multiparty computations using a standard deck of playing cards. In *Proceedings of the 15th International Conference on Cryptology and Network Security (CANS 2016), LNCS, Milan, Italy, 14–16 November 2016*; Springer: Cham, Switzerland, 2016; Volume 10052, pp. 484–499.
58. Shinagawa, K.; Mizuki, T. Secure computation of any boolean function based on any deck of cards. In *Proceedings of the 13th International Workshop on Frontiers in Algorithmics (FAW 2019), LNCS, Sanya, China, 29 April–3 May 2019*; Springer: Cham, Switzerland, 2019; Volume 11458, pp. 63–75.
59. Niemi, V.; Renvall, A. Solitaire zero-knowledge. *Fundam. Informaticae* **1999**, 38, 181–188. [[CrossRef](#)]

60. Shinagawa, K.; Nuida, K.; Nishide, T.; Hanaoka, G.; Okamoto, E. Committed AND protocol using three cards with more handy shuffle. In Proceedings of the 2016 International Symposium on Information Theory and Its Applications (ISITA 2016), Monterey, CA, USA, 30 October– 2 November 2016; IEICE: Tokyo, Japan, 2016; pp. 700–702.
61. Manabe, Y.; Ono, H. Card-based Cryptographic Protocols with a Standard Deck of Cards Using Private Operations. In Proceedings of the 18th International Colloquium on Theoretical Aspects of Computing (ICTAC 2021), LNCS, Nur-Sultan, Kazakhstan, 6–10 September 2021; Springer: Cham, Switzerland, 2021; Volume 12819.
62. Koyama, H.; Miyahara, D.; Mizuki, T.; Sone, H. A Secure Three-Input AND Protocol with a Standard Deck of Minimal Cards. In Proceedings of the 16th International Computer Science Symposium in Russia (CSR 2021), LNCS, Sochi, Russia, 28 June–2 July 2021; Springer International Publishing: Cham, Switzerland, 2021; Volume 12730, pp. 242–256.
63. Koch, A.; Schrempf, M.; Kirsten, M. Card-based cryptography meets formal verification. *New Gener. Comput.* **2021**, *39*, 115–158. [[CrossRef](#)]
64. Manabe, Y.; Ono, H. Secure Card-Based Cryptographic Protocols Using Private Operations Against Malicious Players. In Proceedings of the 13th International Conference on Information Technology and Communications Security (SecITC 2020), LNCS, Bucharest, Romania, 19–20 November 2020; Springer: Cham, Switzerland, 2020; Volume 12596, pp. 55–70.