Card-based Cryptographic Protocols with the Minimum Number of Rounds Using Private Operations

Hibiki Ono[1] and Yoshifumi Manabe[1][0000−0002−6312−257X]

Kogakuin University, Shinjuku,Tokyo 163–8677 Japan.
manabe@cc.kogakuin.ac.jp

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. Operations that a player executes in a place where the other players cannot see are called private operations. Using three private operations called private random bisection cuts, private reverse cuts, and private reveals, 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 shows 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. This paper then shows the relationship between the number of rounds and available private operations. Even if private reveal operations are not used, logical XOR can be executed with the minimum number of cards in two rounds. On the other hand, logical AND and copy operations can be executed with the minimum number of cards in three rounds without private reveal operations. Protocols that preserves an input are also shown.

Keywords: Multi-party secure computation · card-based cryptographic protocols · private operations · logical computations · copy · round.

1 Introduction

1.1 Motivation

Card-based cryptographic protocols [12, 24] were proposed in which physical cards are used instead of computers to securely calculate values. They can be used when computers cannot be used. den Boer [2] first showed a five-card protocol to securely calculate logical AND of two inputs. Since then, many protocols

have been proposed to calculate logical functions [13, 25] and specific computations such as millionaires' problem [17, 27, 33], voting [21, 26], random permutation [7, 9, 10], grouping [8], matching [16], proof of knowledge of a puzzle solution [3,5,36], and so on. This paper considers calculations of logical functions and the copy operation under the semi-honest model.

There are several types of protocols regards to the inputs and outputs of the computations. The first type is committed inputs [19], where the inputs are given as committed values. The players do not know the input values. The other type is non-committed inputs [15, 40], where players give their private inputs to the protocol using private input operations. The private input operations were also used in millionaires' problem [33]. Protocols with committed inputs are desirable, since they can be used for non-committed inputs: each player can give his private input value as a committed value.

Some protocols output their computation results as committed values [19]. The result is unknown to the players unless the players open the output cards. The other type of protocols [2] 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 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 the 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 millionaires' problem [27] and voting [26]. Using private operations, committed-input and committed-output logical AND, logical XOR, and copy protocols can be achieved

Table 1. Comparison of XOR protocols using private operations.

Article	# of rounds	# of cards	Preserving an input	Private reveal
[34]	3	4	No	Use
[34]	3	4	Yes	Use
[25]	2	4	No	Does not use
This paper §3	2	4	No	Use
This paper §4	2	4	No	Does not use
This paper §5	3	4	Yes	Use/Does not use

Table 2. Comparison of AND protocols using private operations.

Article	# of rounds	# of cards	Preserving an input	Private reveal
[34]	3	4	No	Use
[34]	3	6	Yes	Use
[34]	5	4	Yes	Use
[25]	2	6	No	Does not use
This paper §3	2	6	No	Use
This paper §4	3	4	No	Does not use
This paper §5	3	6	Yes	Use/Does not use
This paper §5	5	4	Yes	Does not use

with the minimum number of cards [34]. 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 works have been done for the protocols that do not use private operations [18]. 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 handing cards between players. Since each private operation is relatively simple, handing 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 AND, logical XOR, and copy protocols with the minimum number of rounds. The summary of results are shown in Table 1, 2, and 3. Note that the protocols in [25] needs one shuffle by each player, thus the actual execution time is larger than the ones in this paper though the number of rounds is the same. This paper then shows variations of the protocols that use a different set of private operations. 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. Such a protocol is called a protocol that preserves an input [29]. This paper shows the number of rounds of protocols that preserves an input.

In Section 2, basic notations, the private operations introduced in [34], and the definition of rounds are shown. Section 3 shows two round XOR, AND, and copy protocols. Section 4 shows the protocols that use a different set of private

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

Article	# of rounds	# of cards	Private reveal
[34]	3	4	Use
[25]	2	6	Does not use
This paper §3	2	6	Use
This paper §4	2	6	Does not use

operations. Section 5 shows protocols that preserve an input. Section 6 concludes the paper.

1.2 Related works

Many works have been done for calculating logical functions without private operations. den Boer [2] 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 [4, 28, 41], but they use more than six cards. Mizuki et al. [25] showed a logical AND protocol that uses six cards. It is proved that it is impossible to calculate logical AND with less than six cards when we use closed and uniform shuffles [11]. 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 [1, 12, 14, 32, 35]. Also, when Las Vegas protocols are allowed, logical AND protocols with five or four cards were shown [12, 22, 35].

For making a copy of input bit, Mizuki et al. showed a protocol with six cards [25]. A five-card protocol was shown that uses non-uniform shuffles [31].

Mizuki et al. [25] showed a logical XOR protocol that uses four cards, which is the minimum.

Several other protocols such as computations of many inputs [19, 30, 38], computing any boolean functions [13, 29, 39], two-bit output functions [6] were shown. Protocols using other types of cards were also shown [20, 23, 37].

2 Preliminaries

2.1 Basic notations

This section gives the notations and basic definitions of card-based protocols. This paper is based on a 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 is 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 $\underbrace{\quad}_{x}$. 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 $S = s_1, s_2, \ldots, s_n$, where s_i is i-th card of the sequence. $S = \underbrace{\quad}_{s_1}\underbrace{\quad}_{s_2}\underbrace{\quad}_{s_3}\ldots,\underbrace{\quad}_{s_n}$. 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.

2.2 Private operations

We show three private operations introduced in [34]: 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 $S_0 = \underbrace{\quad}_{x}$, the player's output $S_1 = \underbrace{\quad}_{x \oplus b}$, which is $\underbrace{\quad}_{x}$ or $\underbrace{\quad}_{\bar{x}}$.

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

A private reverse cut is the following operation on an even sequence $S_2 = s_1, s_2, \ldots, s_{2m}$ 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 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 a 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 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 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.

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

Protocol 1 *(AND protocol in [34])*

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

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

   and hands 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 at 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 handing cards between players and setting 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, say Alice, has some (or all) of the final outputs of the protocol. Since the number of rounds is one, handing cards 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 committed inputs.

2.4 Our results

The protocols in [34] are three rounds and use four cards. This paper shows 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 the minimum. Another advantage of these two-round protocols is that each player does not need to remember the random bit. In the protocols in [34], a player needs to remember the random bit until the player receives the cards again in order 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 he hands the cards to the other player. Note that Alice obtains the final result by the three-round protocols in [34] 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 two-round protocols with four card logical XOR and six card

logical AND with private operations have been implicitly shown by [25], this paper shows another type of protocols with fewer shuffles.

The above two-round protocols do not use private reverse cuts. Thus, there is a question of whether we can obtain protocols without another type of private operations. This paper answers this question also. We show protocols that do not use private reveals. There is a worry in using this primitive. A player might make a mistake to open cards that are not allowed and obtain private values. If private reveals are not executed at all, protections such as putting each card in an envelope can be done to prove that opening cards are not executed during 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. The number of rounds is three for the logical AND protocol without private reveals. Last, we show protocols that preserve an input.

3 XOR, AND, and Copy with the minimum number of rounds

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

The players are assumed to be semi-honest, that is, they honestly execute the protocol but they try to obtain secret values. The protocol is secure if the players obtain no information about input values and output values. There is no collusion between the two players executing one protocol together. The performance of protocols is evaluated by the number of cards and the number of rounds.

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

3.1 XOR protocol

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

1. *Alice executes a private random bisection cut on input* $S_0 = commit(x)$ *and* $S'_0 = commit(y)$ *using the same random bit* b*. Let the output be* $S_1 = commit(x')$ *and* $S'_1 = commit(y')$*, respectively. Note that* $x' = x \oplus b$ *and* $y' = y \oplus b$*. Alice hands* S_1 *and* S'_1 *to Bob.*
2. *Bob executes a private reveal on* $S_1 = commit(x')$*. Bob privately sets output*

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

Note that $commit(\bar{y'})$ *can be obtained by swapping the two cards of* $S'_1 = commit(y')$*.*

The protocol is two rounds.

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

Proof. Correctness: Alice hands $commit(x \oplus b)$ and $commit(y \oplus b)$ to Bob. Bob swaps the pair of $commit(y \oplus b)$ 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. □

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 [25]. Since one public shuffle can be changed to two private shuffles by each player, the minimum number of rounds is achieved also by their protocol. However, the protocol needs two shuffles, thus our new protocol is simple. Comparison of committed-input, committed-output XOR protocols using private operations are shown in Table 1.

3.2 AND protocol

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

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

Theorem 2. *The AND protocol is correct and secure.*

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

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

Bob outputs $commit(y)$ as S_2 when $(x', b) = (0, 1)$ or $(1, 0)$. Since $x' = x \oplus b$, these cases equal to $x = 1$. Bob outputs $commit(0)$ 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: The same as the XOR protocol. □

The number of rounds is the minimum. Mizuki et al. showed a six-card protocol with one public shuffle [25]. Since one public shuffle can be changed to two private shuffles by each player, the minimum number of rounds is achieved also 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)*

1. *Alice executes a private random bisection cut on $S_0 = commit(x)$. Let the output be $S_1 = commit(x')$. Alice sets S'_1 as m copies of $commit(b)$, where b is the bit selected in the random bisection cut. Note that $x' = x \oplus b$. Alice hands 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: The same as the XOR protocol.

Though the number of cards is increased, the number of rounds is the minimum. Comparison of COPY protocols(when $m = 2$) is shown in Table 3. Mizuki et al. showed a six-card protocol with one public shuffle [25]. Since one public shuffle can be changed to two private shuffles by each player, the minimum number of rounds is achieved also by their protocol. However, the protocol needs two shuffles, thus our new protocol is simple.

3.4 Any two-variable logical functions

Though this paper shows logical AND and logical XOR, any two-variable logical 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 [34].

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

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

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

where $f(1,y)$ and $f(0,y)$ are y, $\bar{y}$, 0, or 1. From [34], we just need to consider the cases 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

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

using one $commit(y)$ 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 $S_1 = commit(x')$ and selects $commit(f(1,y))$ if $x=1$. Bob selects $commit(f(0,y))$ if $x=0$.

Thus, any two-variable logical function can be calculated. □

3.5 n-variable logical functions

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

As another implementation with more number of cards, we show that any n-variable logical function can be calculated by the following protocol in two rounds, whose technique is similar to the one in [15]. Let f be any n-variable logical function.

Protocol 5 *(Protocol for any logical function with two rounds)*

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

Since $S_{x'_1,x'_2,\ldots,x'_n} = commit(f(b_1 \oplus x'_1, b_2 \oplus x'_2, \ldots, b_n \oplus x'_n)) = commit(f(x_1,x_2,\ldots,x_n))$, the output is correct. The security is the same as the one of the XOR protocol. The protocol is two-round. The number of cards is $2^{n+1}+2n$.

4 Protocols without private reveals

This section shows that the protocols in [34] that use the minimum number of cards can be executed without the private reveal operations with a more number of steps, but the same number of rounds. Since it is hard to prevent mistakes of privately opening cards that are not allowed, it would be better all reveal operations are publicly executed.

4.1 XOR protocol without private reveals

Protocol 6 *(XOR protocol without private reveals)*

1. *Alice executes a private random bisection cut on $S_0 = commit(x)$ and $S'_0 = commit(y)$ using the same random bit b. Let the output be $S_1 = commit(x')$ and $S'_1 = commit(y')$, respectively. Note that $x' = x \oplus b$ and $y' = y \oplus b$. Alice hands 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 $S_2 = commit(x'')$ and $S'_2 = commit(y'')$, respectively. $x'' = x \oplus b \oplus b'$ and $y'' = y \oplus b \oplus b'$ holds. Bob publicly opens S_2 and obtains value x''. Alice can see x''. Bob publicly sets*

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

S_3 is the final result.

The protocol is two rounds.

Theorem 5. *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 $x \oplus b \oplus b'$.

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. □

4.2 AND protocol without private reveals

Protocol 7 *(AND protocol without private reveals)*

1. *Alice executes a private random bisection cut on $S_0 = commit(x)$. Let the output be $S_1 = commit(x')$. Alice hands 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 output 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*

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

Bob then executes a private reverse cut on S_2 using the bit b' generated in the private random bisection cut. Let the output be S_3. Bob hands 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. Alice outputs the obtained sequence of S_4.*

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

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

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

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: 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. □

The number of rounds is three. Using the argument in Section 3.4, any two-variable logical function can also be calculated by four cards and three rounds without private reveals.

4.3 COPY protocol without private reveals

Protocol 8 *(COPY protocol without private reveals)*

1. *Alice executes a private random bisection cut on $S_0 = commit(x)$. Let the output be $S_1 = commit(x')$. Note that $x' = x \oplus b$. Alice sets S'_1 as m copies of $commit(b)$. Alice hands 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 and S'_2, respectively.*
   Bob publicly opens S_2 and obtains value x''. Alice can see the cards. 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 7. *The COPY protocol is correct and secure.*

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

Alice and Bob's security: The same as the XOR protocol without private reveals. □

4.4 n-variable logical functions without private reveals

Let f be any n-variable logical function.

Protocol 9 *(Protocol for any logical function without private reveals)*

1. *The same as Step 1 in Protocol 5.*
2. *Bob executes a private random bisection cut on* $commit(x'_i)(i = 1, 2, \ldots, n)$. *Note that one random bit* b'_i *is selected for each* $x'_i(i = 1, 2, \ldots, n)$. *Let* $commit(x''_i)$ $(i = 1, 2, \ldots, n)$ *be the obtained value.* $x''_i = x_i \oplus b_i \oplus b'_i(i = 1, 2, \ldots, n)$ *is satisfied. Bob privately relocates* $S_{a_1,a_2,\ldots,a_n}(a_i \in \{0, 1\}, i = 1, 2, \ldots, n)$ *so that* $S'_{a_1,a_2,\ldots,a_n} = S_{a_1\oplus b'_1,a_2\oplus b'_2,\ldots,a_n\oplus b'_n}(a_i \in \{0, 1\}, i = 1, 2, \ldots, n)$. *The cards satisfy that* $S'_{a_1,a_2,\ldots,a_n} = commit(f(a_1 \oplus b_1 \oplus b'_1, a_2 \oplus b_2 \oplus b'_2, \ldots, a_n \oplus b_n \oplus b'_n))$.
   Bob publicly reveals $commit(x''_i)$ *and obtains* $x''_i(i = 1, 2, \ldots, n)$. *Bob publicly selects* $S'_{x''_1,x''_2,\ldots,x''_n}$.

Since $S'_{x''_1,x''_2,\ldots,x''_n} = commit(f(x_1 \oplus b_1 \oplus b'_1 \oplus b_1 \oplus b'_1, x_2 \oplus b_2 \oplus b'_2 \oplus b_2 \oplus b'_2, \ldots, x_n \oplus b_n \oplus b'_n \oplus b_n \oplus b'_n)) = commit(f(x_1, x_2, \ldots, x_n))$, the output is correct. The security is the same as the XOR protocol. The protocol is two rounds. The number of cards is $2^{n+1} + 2n$.

All the other protocols that are shown in [34] can also be changed so as not to use the private reveal operations. The 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 $commit(x \oplus b)$ 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 then executes a private reverse cut on S using the bit b' and outputs S'. Alice executes a private reverse cut on S' and obtains the final result.

5 Protocols that preserve an input

In the above protocols to calculate logical 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 [29]. For the three-round XOR and AND protocols in [34], protocols that preserve an input were shown [34]. This paper uses the technique in [34] to obtain protocols to preserve an input.

First, consider XOR protocols in Section 3 and Section 4.

Protocol 10 *(XOR protocol that preserves an input)*

1,2 The same as Protocol 2 (or Protocol 6).
   At the end of Step 2, Bob sends back $S_1 = commit(x')$ *to Alice.*
3 Alice executes private reverse cut on S_1 *and obtains* $commit(x)$.

In the protocol in Section 3, since $commit(x')$ is unnecessary after Bob's private reveal, the cards can be sent back to Alice. Alice can recover $commit(x)$. The number of rounds is increased to three.

For the XOR protocols in Section 4, the same technique can be applied. After Bob publicly opens S_2 and obtains $x \oplus b \oplus b'$, he can recover $S_1 = commit(x')$ since he knows b'. The protocol is three rounds and uses four cards.

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

Protocol 11 *(AND protocol that preserves an input)*

1,2 The same as Protocol 3 (or Protocol 7).
At the end of Step 2, Bob sends back $S_1 = commit(x')$ to Alice.
3 Alice executes private reverse cut on S_1 and obtains $commit(x)$ (in Protocol 7, executed at the end of the protocol).

The number of rounds is three and the number of cards is six. For the protocol in Section 4, two new cards are necessary to send $commit(x')$ to Alice.

The same technique can be applied to n-variable protocol and $commit(x'_i)$ can be sent back to Alice.

As for the AND type protocol, another protocol that preserves an input without additional cards can be obtained. 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

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

thus the output value is $commit(\bar{x} \wedge f(1,y) \oplus x \wedge f(0,y))$. The output $f(x,y)$ can be written as $x \wedge f(1,y) \oplus \bar{x} \wedge f(0,y)$. Execute the above XOR protocol that preserves an input without private reveals 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 12 *(AND type protocol that preserves an input without private reveals)*

1,2 The same as Protocol 7.
3 After 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 10) for S_4 and S'_4. Let the preserved input, S_4, be the result. Obtain $commit(y)$ from the XOR result.

Thus, the protocol achieves preserving an input by four cards. 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.

6 Conclusion

This paper proposed round optimal card-based cryptographic protocols using private operations. Then this paper showed protocols without private reveal operations and several variant protocols. Further study includes round optimal protocols for the other fundamental problems.

References

1. Abe, Y., Hayashi, Y.i., Mizuki, T., Sone, H.: Five-card and protocol in committed format using only practical shuffles. In: Proc. of 5th ACM on ASIA Public-Key Cryptography Workshop(APKC 2018). pp. 3–8. ACM (2018)
2. den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Proc. of EUROCRYPT '89, LNCS Vol. 434. pp. 208–217 (1990)
3. 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: Proc. of 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2018), LNCS, Vol.11201. pp. 111–125 (2018)
4. Crépeau, C., Kilian, J.: Discreet solitary games. In: Proc. of 13th Crypto, LNCS Vol. 773. pp. 319–330 (1993)
5. Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for norinori. In: Proc. of 25th International Computing and Combinatorics Conference (COCOON 2019), LNCS Vol.11653. pp. 166–177 (2019)
6. 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)
7. 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)
8. 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-A**(9), 1512–1524 (2018)
9. Ibaraki, T., Manabe, Y.: A more efficient card-based protocol for generating a random permutation without fixed points. In: Proc. of 3rd International Conference on Mathematics and Computers in Sciences and in Industry (MCSI 2016). pp. 252–257 (2016)
10. 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)

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 23rd International Conference on the Theory and Applications of Cryptology and Information Security(ASIACRYPT2017), Part III, LNCS Vol. 10626. pp. 126–155 (2017)
12. Koch, A.: The landscape of optimal card-based protocols. IACR Cryptology ePrint Archive, Report 2018/951 (2018)
13. Koch, A., Walzer, S.: Private function evaluation with cards. IACR Cryptology ePrint Archive, Report 2018/1113 (2018)
14. 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)
15. Kurosawa, K., Shinozaki, T.: Compact card protocol. In: Proc. of 2017 Symposium on Cryptography and Information Security(SCIS 2017). pp. 1A2–6 (2017), (In Japanese)
16. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. IACR Cryptology ePrint Archive, Report 2015/1031 (2015)
17. Miyahara, D., Hayashi, Y.i., Mizuki, T., Sone, H.: Practical and easy-to-understand card-based implementation of yao's millionaire protocol. In: Proc. of 12th International Conference on Combinatorial Optimization and Applications (COCOA 2018), LNCS Vol. 11346. pp. 246–261 (2018)
18. Miyahara, D., Ueda, I., Hayashi, Y.i., Mizuki, T., Sone, H.: Analyzing execution time of card-based protocols. In: Proc. of 17th International Conference on Unconventional Computation and Natural Computation (UCNC 2018), LNCS Vol. 10867. pp. 145–158 (2018)
19. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theoretical Computer Science **622**, 34–44 (2016)
20. Mizuki, T.: Efficient and secure multiparty computations using a standard deck of playing cards. In: Proc. of Cryptology and Network Security (CANS 2016), LNCS Vol.10052 (2016)
21. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Proc. of International Conference on Unconventional Computing and Natural Computation (UCNC 2013), LNCS Vol. 7956. pp. 162–173 (2013)
22. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. Proc. of Asiacrypt 2012, LNCS Vol.7658 pp. 598–606 (2012)
23. 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)
24. 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-A**(1), 3–11 (2017)
25. 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)
26. 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)
27. Nakai, T., Tokushige, Y., Misawa, Y., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations.

In: Proc. of International Conference on Cryptology and Network Security(CANS 2016), LNCS Vol. 10052. pp. 500–517 (2016)
28. Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theoretical Computer Science **191**(1), 173–183 (1998)
29. 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)
30. 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-A**(6), 1145–1152 (2015)
31. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Five-card secure computations using unequal division shuffle. In: Proc. of 4th International Conference on Theory and Practice of Natural Computing(TNPC 2015), LNCS Vol. 9477. pp. 109–120 (2015)
32. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**(2), 361–371 (2018)
33. 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)
34. Ono, H., Manabe, Y.: Card-based cryptographic protocols with the minimum number of cards using private operations. In: Proc. of 11th International Symposium on Foundations & Practice of Security(FPS 2018), LNCS Vol. 11358. pp. 193–207 (2019)
35. 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)
36. Sasaki, T., Mizuki, T., Sone, H.: Card-Based Zero-Knowledge Proof for Sudoku. In: Proc. of 9th International Conference on Fun with Algorithms (FUN 2018). Leibniz International Proceedings in Informatics (LIPIcs), vol. 100, pp. 29:1–29:10 (2018)
37. Shinagawa, K., Mizuki, T.: Secure computation of any boolean function based on any deck of cards. In: Proc. of 13th International Frontiers in Algorithmics Workshop(FAW 2019), LNCS Vol. 11458. pp. 63–75 (2019)
38. 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 (2019)
39. Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any circuit. IACR Cryptology ePrint Archive **2019**, 380 (2019)
40. 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. 1A2–2 (2017), (In Japanese)
41. Stiglic, A.: Computations with a deck of cards. Theoretical Computer Science **259**(1), 671–678 (2001)
42. Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Proc. of 5th International Conference on Theory and Practice of Natural Computing (TPNC 2016), LNCS Vol. 10071. pp. 58–69 (2016)