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

Hibiki Ono[1] and Yoshifumi Manabe[1]

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

Abstract. This paper proposes new card-based cryptographic protocols with the minimum number of cards using private operations under the semi-honest model. Though various card-based cryptographic protocols were shown, the minimum number of cards used in the protocol has not been achieved yet for many problems. Operations executed by a player where the other players cannot see are called private operations. Private operations have been introduced in some protocols to solve a particular problem or to input private values. However, the effectiveness of introducing private operations to the calculation of general logic functions has not been considered. This paper introduces three new private operations: private random bisection cuts, private reverse cuts, and private reveals. With these three new operations, we show that all of logical and, logical xor, and copy protocols are achieved with the minimum number of cards by simple three round protocols. This paper, then shows a protocol to calculate any logical functions using these private operations.

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

1 Introduction

Card-based cryptographic protocols [9, 17] have been proposed in which physical cards are used instead of computers to securely calculate values. 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 [3, 4, 14, 16, 18, 21, 27, 29] and specific computations such as computations on three inputs [23, 24], millionaires' problem [20, 26], voting [15, 19], random permutation [6, 7], grouping [5], matching [13] and so on.

Private randomization is the most important primitive in these card-based protocols. Many recent protocols use random bisection cuts [18], which randomly execute swapping two decks of cards or not swapping. If the random value used in the randomization is disclosed, the secret input

value is known to the players. There are two types of randomization: single player randomization and multiple player randomization. For the single player randomization, the player must not know the random value he selected. Ueda et al. [30] proposed several methods that can be done in front of people, but no one can know the random value. However, if a person privately brings a high-speed video camera, he might able to know the random value by analyzing the image. Currently, the size of high-speed video cameras is too large to privately bring without getting caught, but the size might become smaller in a near future. In the case, the randomization in a public place becomes difficult. By introducing additional cards, a random bisection cut can be executed using a random cut [30]. Koch and Walzer [10] proposed a protocol for a player to execute a private permutation that is unknown to the other players, but the player can prove that he really executed an allowed permutation. The protocol can be executed in a public place, but it needs additional special cards.

A simple solution to execute a private randomization is a multiple player randomization, in which some operations are executed in a hidden place. In order to execute a private random bisection cut, Alice executes a random bisection cut in a place where Bob cannot see (under the table, or in the back, etc). Then, Bob executes a random bisection cut in a place where Alice cannot see. The result is unknown to either player. Note that the number of players can be arbitrarily increased. In order to know the random value, a person needs to know all of the values the players used. Such an operation that is done where the other players cannot see is called a private operation. So we have a natural question: if we introduce some private operations other than the random bisection cut, can we have effective card-based cryptographic protocols to calculate logical functions?

Private operations have been first introduced to solve millionaires' problem [20, 26]. The private operations used in the papers are similar to the primitives proposed in this paper, but the operations were embedded into the millionaires' protocol, thus it is not clear that the primitives can be used to the other protocols. Then private operations were used to calculate logical functions [12, 28]. These papers discussed a private operation that sets each player's private inputs. Though the number of cards used in these protocols is less than the ones in the conventional protocols, these protocols cannot be used for general cases when the players do not know the inputs, that is, the inputs are given as committed values. Protocols with committed inputs can also be used for the cases when each player knows his input values by setting his private inputs as commit-

ted values. Thus protocols that accept committed inputs are desirable. Another desirable property is committed output. If the output is given as a committed value, further private calculation can be done using the output value.

This paper considers card-based protocols with committed inputs and committed outputs using private operations under the semi-honest model. This paper introduces three private operations: private random bisection cuts, private reverse cuts, and private reveals. This paper shows protocols which execute logical and, logical xor, and copy with four cards, which is the minimum. We also show protocols that calculate any logical functions.

As for the number of cards used for copy protocols, 6 was the minimum for finite-runtime copy [18], as shown in Table 1. The protocol in [25] uses 5 cards, but the number of steps of the protocol is unbounded. It is proved to be impossible to achieve copy with 4 cards by the conventional model without private operations [8]. In their model, each card sequence has a probability to occur. Using the probabilities, players are prohibited to open a card that reveals secret information. Such arguments lead to the impossibility results. On the other hand, if private operations are executed, one card sequence can have two different probabilities: the one Alice knows and the other one Bob knows. Bob is allowed to privately open a card that does not reveal secret information to Bob.

The numbers of cards in committed-input, committed-output AND protocols are shown in Table 2. The protocol in [18] uses 6 cards. Though the protocol in [11] uses 4 cards, the protocol uses a non-uniform shuffle, which obtains one result by the probability of 1/3 and the other result by the probability of 2/3. Such a non-uniform shuffle is difficult to achieve without some special tools. Another four-card protocol with uniform shuffles [27] does not terminate within a finite time. It is proved to be impossible to achieve finite-runtime AND with 4 cards by the conventional model without private operations [8]. Our protocol uses 4 cards, which is the minimum, and it is easy to execute.

The number of cards in XOR protocols is shown in Table 3. Though the number of cards is the same in our protocol and [18], an input preserving (shown in Section 4.7) can be realized by our protocol without additional cards.

2 Preliminaries

This section gives the notation and basic definitions of card-based protocols. This paper is based on two type card model. In the model, there

are two kinds of marks, $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. Cards of the same marks cannot be distinguished. In addition, the back of both types of cards is $\boxed{?}$. It is impossible to determine the mark in the back of a given card with $\boxed{?}$. One bit of data is represented by two cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$. One pair of cards that represents one bit $x \in \{0, 1\}$, whose face is down, is called a commitment of x, and denoted as $commit(x)$. It is written as $\underbrace{\boxed{?}\boxed{?}}_{x}$. Note that when these two cards are swapped, $commit(\bar{x})$ can be obtained. Thus, NOT can be calculated without private operations.

A linearly ordered cards are 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 the i-th card of the sequence. $S = \underbrace{\boxed{?}}_{s_1}\underbrace{\boxed{?}}_{s_2}\underbrace{\boxed{?}}_{s_3}\ldots,\underbrace{\boxed{?}}_{s_n}$. A sequence whose length is even is called an even sequence. $S_1||S_2$ is a concatenation of sequence S_1 and S_2.

All protocols are executed by multiple players. Throughout of 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 of committed values.

3 Private operations

We introduce three private operations: 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 $S_0 = s_1, s_2, \ldots, s_{2m}$. Alice selects a random bit $b \in \{0, 1\}$ and outputs

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

Alice executes this operation in a place where Bob cannot see. Alice does not disclose the bit b. □

Note that the protocols in this paper use the operation only when $m = 1$ and $S_0 = commit(x)$. Given $S_0 = \underbrace{\boxed{?}\boxed{?}}_{x}$, Alice's output $S_1 = \underbrace{\boxed{?}\boxed{?}}_{x \oplus b}$, which is $\underbrace{\boxed{?}\boxed{?}}_{x}$ or $\underbrace{\boxed{?}\boxed{?}}_{\bar{x}}$.

Note that a private random bisection cut is exactly the same as the random bisection cut [18], but the operation is done in a hidden place.

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

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

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

Alice executes this operation in a place where Bob cannot see. b *is the secret value that only Alice knows. Alice does not disclose* b*.*

Note that in many protocols below, the left m *cards are selected after a private reverse cut. The sequence of these two operations is called a private reverse selection. A private reverse selection is the following procedure on an even sequence* $S_2 = s_1, s_2, \ldots, s_{2m}$ *and the bit* $b \in \{0, 1\}$*, which is selected by Alice during the private random bisection cut. Alice's output*

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

Next, we define a private reveal. Consider the case Alice executes a private random bisection cut on $commit(x)$. A private reveal, executed by Bob, is as follows.

Primitive 3 *(Private reveal)*

Bob privately opens a given committed bit. Since the committed bit is randomized by the bit b *selected by Alice, the opened bit is* $x \oplus b$*.* □

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

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.

Card-based protocols are evaluated by the following criteria.

- The number of cards used in the protocol.
- The number of operations executed in the protocol.
- The number of communications: the number of times when cards are handed between players.

4 New copy, logical and, and logical xor protocols

Using the private random bisection cuts, private reveals, and private reverse cuts, COPY protocol, AND protocol, and XOR protocol with committed inputs and committed outputs can be realized with the minimum number of cards. All of these protocols are executed between two players, Alice and Bob. In section 5, the number of players is increased in order to improve security.

4.1 COPY protocol

Protocol 1 *(COPY protocol)*
Input: commit(x)*. Output:* m *copies of commit*(x)*.*

1. *Alice executes a private random bisection cut on commit*(x)*. Let the output be commit*(x')*. Note that* $x' = x \oplus b$*. Alice hands commit*(x') *to Bob.*
2. *Bob executes a private reveal on commit*(x') *and obtains* x'*. Bob makes* m *copies of* x'*. Bob faces down these cards. Bob hands these cards,* m *copies of commit*(x')*, to Alice.*
3. *Alice executes a private reverse cut to each copy of commit*(x') *using the bit* b *Alice generated in the private random bisection cut. Alice outputs these copies.* □

The protocol is three rounds. The number of communications between players is two.

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

(Proof) Correctness: If $b = 0$, Bob sees x and makes m copies of x. Alice does nothing at the private reverse cut, thus m copies of x are obtained. If $b = 1$, Bob sees $\bar{x}$ and makes m copies of $\bar{x}$. Alice swaps each copy of $\bar{x}$, thus m copies of x are obtained.

Alice's security: Alice sees no opened cards, thus Alice obtains no information about x.

Bob's security: When Bob privately opens *commit*(x'), $x' = x \oplus b$, thus Bob obtains no information about x if b is randomly selected and not disclosed.

The number of cards: In order to obtain m copies of a commitment, at least $2m$ cards are necessary. The protocol is executed with $2m$ cards, thus the number of cards is the minimum. □

Table 1. Comparison of COPY protocols

Article	# of cards	Note
[3]	8	
[18]	6	
[25]	5	Number of steps is not bounded
This paper	4	Use private operations

Comparison of COPY protocols(when $m = 2$) is shown in Table 1. This protocol is the first protocol that achieves the minimum number of cards.

4.2 AND protocol

Logical AND can also be executed with the minimum number of cards.

Protocol 2 *(AND protocol)*
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 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.* □

Note that the two cards that were not selected by Alice at the last step of the protocol, must be discarded. Since the unused cards have some information on x and y, information about input values are leaked if the cards are opened. The protocol is three rounds. The number of communications between players is two.

Theorem 2. *The AND protocol is correct and secure. It 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 given to Alice. When Bob obtains $x' = 0$, $commit(0)||commit(y)$ is given to Alice. Thus Alice's output is $commit(y)$ if $(x', b) = (1, 0)$ or $(0, 1)$. Since $x' = x \oplus b$, these cases equal to $x = 1$.

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

Alice and Bob's security: The same as the COPY protocol.

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

A careful discussion is necessary when a player knows the value x of given $commit(x)$, for example, x is the player's private input value.

First, consider the case when Bob knows x. When Bob executes a private reveal on $commit(x \oplus b)$, Bob knows the bit b Alice selected. This scenario is not a security problem. Bob knows b, thus he knows whether the final output is $commit(0)$ or $commit(y)$ in advance. However, since

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

it is not new information for Bob who already knows x.

Note that if $x = 0$ and Bob wants to know y, Bob can replace $commit(y)$ with two new cards (that Bob hidden in his pocket), and sends
$commit(0)||commit(0)$ to Alice. The result is still correct and Bob can privately open $commit(y)$ afterwards. In order to prevent this type of attack, marking currently using cards or a watch person (discussed in Section 5) is necessary.

Next, consider the case when Alice knows x. Alice knows $x' = x \oplus b$. Thus Alice knows whether the final output is $commit(0)$ or $commit(y)$ in advance, but it is not new information for Alice. Note that if $x = 0$ and Alice wants to know y, Alice can replace $commit(y)$ with two new cards during the execution. Prevention of this type of attack is just the same as the one for the case of Bob.

A similar discussion can be done for the other protocols shown in this paper.

A comparison of AND protocols is shown in Table 2. Though Koch et al. [11] showed a finite step protocol with the minimum number of cards, their protocol must use a non-uniform shuffle, which is not easy to realize.

Table 2. Comparison of AND protocols

Article	# of cards	Input	Output	Note
[2]	5	commit	non-commit	
[3]	10	commit	commit	Four color cards
[21]	12	commit	commit	
[29]	8	commit	commit	
[18]	6	commit	commit	
[1]	5	commit	commit	Number of steps is not bounded
[11]	4	commit	commit	Non-uniform shuffle
[27]	4	commit	commit	Number of steps is not bounded
[16]	4	commit	non-commit	
[28]	3	non-commit	non-commit	Use private operations
[12]	4	non-commit	commit	Use private operations
This paper	4	commit	commit	Use private operations

4.3 XOR protocol

Protocol 3 *(XOR protocol)*
Input: commit(x) *and commit*(y)*. Output: commit*$(x \oplus y)$*.*

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*

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

and hands S_2 *to Alice. Note that commit*$(\bar{y})$ *can be obtained by swapping the two cards of commit*(y)*.*
3. *Alice executes a private reverse cut 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 protocol is three rounds. The number of communications between players is two.

Theorem 3. *The XOR protocol is correct and secure. It uses the minimum number of cards.*

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

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

Table 3. Comparison of XOR protocols

Article	# of cards	Input	Output	Note
[3]	14	commit	commit	Four color cards
[18]	4	commit	commit	
[28]	2	non-commit	commit	
[12]	2	non-commit	commit	
This paper	4	commit	commit	Use private operation Preserving an input is possible

When $x' = 1$, $commit(\bar{y})$ is given to Alice. When $x' = 0$, $commit(y)$ is given to Alice. Thus, Alice's output is $commit(\bar{y})$ if $(x', b) = (1, 0)$ or $(0, 1)$. Since $x' = x \oplus b$, these cases equal to $x = 1$.

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

Alice and Bob's security: The same as the COPY protocol.

The number of cards: At least four cards are necessary for any protocol to input $commit(x)$ and $commit(y)$. This protocol uses no additional cards other than the input cards. □

A comparison of XOR protocols is shown in Table 3. Though the minimum number of cards is already realized by [18]. an input preserving (shown in Section 4.7) can be realized without additional cards.

4.4 Any logical functions

Though this paper shows AND and XOR, any two-variable logical functions can also be calculated by a similar protocol.

Theorem 4. *Any two-variable logical function can be securely calculated in three rounds and four 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.

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 $\overline{x \oplus y}$, 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 modify the second step of AND protocol, so that Bob sets

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

using one $commit(y)$ and the two cards used for $commit(x')$. Then, Alice obtains $commit(f(1,y))$ if $x = 1$ and $commit(f(0,y))$ if $x = 0$ by the private reverse selection.

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

In [18] without private operations, two additional cards are required to calculate any two-variable logical function.

4.5 Parallel computations

The above two-variable logical function calculations can be executed in parallel. Consider the case when $commit(x)$ and $commit(y_i)(i = 1, 2, \ldots, n)$ are given and $commit(f_i(x, y_i))(i = 1, 2, \ldots, n)$ need to be calculated. They can be executed in parallel. Alice executes a private random bisection cut on $commit(x)$ and hands $commit(x')$ and $commit(y_i)(i = 1, 2, \ldots, n)$ to Bob. Bob sets $S_2^i(i = 1, 2, \ldots, n)$ using x', $commit(y_i)$, and f_i. Alice executes a private reverse cut or a private reverse selection on each of $S_2^i(i = 1, 2, \ldots, n)$ using the bit b selected at the private random bisection cut. By the procedure, $commit(f_i(x, y_i))(i = 1, 2, \ldots, n)$ are simultaneously obtained.

4.6 Side effects

When we execute the AND protocol, 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

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

Table 4. Comparison of input preserving AND protocols

Article	# of cards	Input	Output	Note
[22]	6	commit	commit	
This paper	4	commit	commit	Use private operations

thus the output is $commit(\bar{x} \wedge y)$. The cards can be used as a side effect just like the six-card AND protocol in [18].

Generally, for a function f that is calculated by AND type protocol shown in Theorem 4, the side-effect output is $commit(\bar{x} \wedge f(1,y) \oplus x \wedge f(0,y))$.

4.7 Preserving input

In the above protocols to calculate logical functions, the input commitment values are lost. If the input is not lost, the input commitment can be used as an input to another calculation. Thus, the input preserving calculation is discussed [22].

In the XOR protocol, $commit(x')$ is no more necessary after Bob sets S_2. Thus, Bob can send back $commit(x')$ to Alice when Bob sends S_2. Then, Alice can recover $commit(x)$ using the private reverse cut. In this modified protocol, the output is $commit(x \oplus y)$ and $commit(x)$ without additional cards or rounds.

As for the AND type protocol, $commit(x')$ can be sent back to Alice and Alice can recover $commit(x)$. This modified protocol needs 6 cards in total.

An input preserving calculation without increasing the number of cards can be executed for AND type protocols just like [22]. 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. At the end of the protocol, the side-effect output is $\bar{x} \wedge f(1,y) \oplus x \wedge f(0,y)$. The output $f(x,y)$ can be represented as $x \wedge f(1,y) \oplus \bar{x} \wedge f(0,y)$. Execute the above input preserving XOR protocol for these two output values so that $f(x,y)$ is recovered. 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. Thus, input preserving can be realized by 4 cards, which is the minimum. Comparison of input preserving AND type protocols is shown in Table 4.

4.8 n-variable logical functions

Since any 2-variable logical function, $\bar{x}$, and COPY can be executed, any n-variable logical function can be calculated by the combination of the above protocols.

Using the technique in [22] and above input preserving logical function calculations, any n-variable logical function can be calculated with $2n+4$ cards as follows.

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

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

Protocol 4 *(Protocol for any logical function (1))*

Input: commit$(x_i)(i = 1, 2, \ldots, n)$. *Output: commit*$(f(x_1, x_2, \ldots, x_n))$.

The additional four cards (two pairs of cards) p_1 *and* p_2 *are used as follows.*

p_1*: the intermediate value to calculate* f *is stored.*

p_2*: the intermediate value to calculate* T_i *is stored.*

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

1. *Copy* v_1^i *from the input* x_1 *to* p_2.
2. *For* $j = 2, \ldots, n$, *execute the following procedure: Apply the input-preserving AND protocol to* p_2 *and input* x_j *(If AND is taken between* $\bar{x_j}$*, first execute NOT to the input, then apply the AND protocol, and return the input to* x_j *again.)*

   At the end of this step, T_i *is obtained at* p_2.
3. *If* $i = 1$, *move* p_2 *to* p_1. *If* $i > 1$, *apply the XOR protocol between* p_1 *and* p_2. *The result is stored to* p_1.

At the end of the protocol, $f(x_1, x_2, \ldots x_n)$ *is obtained at* p_1. □

The number of additional cards in [22] is 6. Thus our protocol reduces the number of cards. The number of rounds is $O(2^n)$.

As another implementation with a larger number of cards, we show that any n-variable logical function can be calculated by the following protocol, whose technique is similar to the one in [12]. Let f be any n-variable logical function.

Protocol 5 *(Protocol for any logical function (2))*
Input: commit$(x_i)(i = 1, 2, \ldots, n)$. *Output: commit*$(f(x_1, x_2, \ldots, x_n))$.

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

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

5 Improving security

Although this paper assumes all players are semi-honest, some players might be malicious in real cases. It is very hard to prevent malicious actions when a player executes a private operation. One countermeasure to deal with a malicious player is setting one watch person to each player.

The watch person for Alice watches the execution by Alice and verifies that (1) Alice does not open the cards, (2) Alice really uses a random number generator (for example, coin-flipping) to select her random bit, (3) Alice honestly executes a private random bisection cut using the random bit, and (4) Alice honestly executes a private reverse cut or private reverse selection using the bit generated in (2).

The watch person for Bob watches the execution by Bob and verifies that (1) Bob does not open the cards that are not allowed and (2) Bob honestly generates the committed cards using the value Bob privately opened.

Note that the watch persons must not disclose the values they watch.

When the number of players is more than two, each player can simultaneously act as a player and a watch person. Suppose that player $P_0, P_1, \ldots, P_{n-1}(n > 2)$ execute the AND protocol together and no collusion exists. Select some random number $i(0 < i < n)$. One player is out of the room when each player executes a round of the protocol. All the other players in the room watch the execution by the current player and

verifies the correctness of the current player. $P_j(j = 0, \ldots, n-2)$ executes a private random bisection cut on $commit(x)$ using random bit b_j when $P_{j+i \mod n}$ is out of the room, thus $commit(x \oplus_{j=0}^{n-2} b_j)$ is obtained. P_{n-1} executes a private reveal and sets S_2 when $P_{n-1+i \mod n}$ is out of the room. Then, $P_j(j = 0, \ldots, n-2)$ executes a private reverse cut using b_j when $P_{j+i \mod n}$ is out of the room. When all the private reverse cuts are finished, the left pair is selected as the output. In this execution, any player cannot obtain the value of the committed value because he does not have all information if no collusion exists. Note that the number of watch persons can be arbitrary changed.

6 Conclusion

This paper proposed new card-based cryptographic protocols with the minimum number of cards using private operations. Though the private operations are effective, the protocols cannot be used when a malicious player exists. How to prevent active attacks using some protocol techniques just like the zero-knowledge proofs is an open problem.

References

1. Abe, Y., Hayashi, Y., Mizuki, T., Sone, H.: Five-card and protocol in committed format using only practical shuffles. In: Proc. of 5th ACM International Workshop on Asia Public-Key Cryptography (APKC 2018). pp. 3–8 (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. Crépeau, C., Kilian, J.: Discreet solitary games. In: Proc. of 13th Crypto, LNCS Vol. 773. pp. 319–330 (1993)
4. 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 Mycrypt 2016, LNCS Vol. 10311. pp. 193–211 (2017)
5. Hashimoto, Y., Shinagawa, K., Nuida, K., Inamura, M., Hanaoka, G.: Secure grouping protocol using a deck of cards. In: Proc. of ICITS 2017, LNCS Vol. 10681. pp. 135–152 (2017)
6. Ibaraki, T., Manabe, Y.: A more efficient card-based protocol for generating a random permutation without fixed points. In: Proc. of 3rd Int. Conf. on Mathematics and Computers in Sciences and in Industry (MCSI 2016). pp. 252–257 (2016)
7. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Proc. of UCNC 2015, LNCS Vol. 9252. pp. 215–226 (2015)
8. 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 ASIACRYPT2017, Part III, LNCS Vol. 10626. pp. 126–155 (2017)
9. Koch, A.: The landscape of optimal card-based protocols. IACR Cryptology ePrint Archive, Report 2018/951 (2018)

10. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. Cryptology ePrint Archive, Report 2017/423 (2017)
11. 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)
12. Kurosawa, K., Shinozaki, T.: Compact card protocol. In: Proc. of SCIS 2017. pp. 1A2–6 (2017), (In Japanese)
13. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. IACR Cryptology ePrint Archive, Report 2015/1031 (2015)
14. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theoretical Computer Science **622**, 34–44 (2016)
15. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Proc. of UCNC 2013, LNCS Vol. 7956. pp. 162–173 (2013)
16. 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)
17. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **100**(1), 3–11 (2017)
18. Mizuki, T., Sone, H.: Six-card secure and and four-card secure xor. In: Proc. of FAW 2009, LNCS Vol. 5598. pp. 358–369 (2009)
19. 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 ICITS 2017, LNCS Vol. 10681. pp. 153–165 (2017)
20. Nakai, T., Tokushige, Y., Misawa, Y., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations. In: Proc. of CANS 2016, LNCS Vol. 10052. pp. 500–517 (2016)
21. Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theoretical Computer Science **191**(1), 173–183 (1998)
22. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any boolean function. In: Proc. of TAMC 2015, LNCS Vol. 9076. pp. 110–121 (2015)
23. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Securely computing three-input functions with eight cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **98**(6), 1145–1152 (2015)
24. Nishida, T., Mizuki, T., Sone, H.: Securely computing the three-input majority function with eight cards. In: Proc. of TPNC 2013, LNCS Vol. 8273. pp. 193–204 (2013)
25. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**(2), 361–371 (2018)
26. 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)
27. Ruangwises, S., Itoh, T.: And protocols using only uniform shuffles. arXiv preprint arXiv:1810.00769 (2018)
28. Shirouchi, S., Nakai, T., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for logic gates utilizing private permutations. In: Proc. of SCIS 2017. pp. 1A2–2 (2017), (In Japanese)
29. Stiglic, A.: Computations with a deck of cards. Theoretical Computer Science **259**(1), 671–678 (2001)
30. Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Proc. of TPNC 2016, LNCS Vol. 10071. pp. 58–69 (2016)