Secure Card-based Cryptographic Protocols Using Private Operations Against Malicious Players

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

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

Abstract. This paper shows new card-based cryptographic protocols using private operations that are secure against malicious players. 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 several private operations, calculations of two variable boolean functions and copy operations were realized with the minimum number of cards. Though the private operations are very powerful in card-based cryptographic protocols, there is a problem that it is very hard to prevent malicious actions during private operations. Though most card-based protocols are discussed in the semi-honest model, there might be cases when the semi-honest model is not enough. Thus, this paper shows new protocols that are secure against malicious players. We show logical XOR, logical AND, and copy protocols, since we can execute any logical computations with a combination of these protocols. We use envelopes as an additional tool that can be easily prepared and used by people.

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

1 Introduction

Card-based cryptographic protocols [6, 13, 28] were proposed in which physical cards are used instead of computers to securely calculate values. They can be used when computers cannot be used or users cannot trust the software on the computer. Also, the protocols are easy to understand, thus the protocols can be used to teach the basics of cryptography [4, 19, 23]. 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 realize primitives to calculate any logical functions [1, 12, 14, 16, 29, 33, 39, 40, 48, 49] and specific computations such as a specific class of logical functions [7, 24, 26, 34, 37, 41, 44, 47, 53], millionaires' problem [20,32,38], voting [25,31,35,54], random permutation [8,10,11], grouping [9], matching [19], ranking [51], proof of knowledge of a puzzle solution [3, 5, 18, 21, 22, 42, 43, 45], and so on. This paper considers calculations of logical functions

and a copy operation under the malicious model since any logical function can be realized with a combination of these calculations.

Operations that a player executes in a place where the other players cannot see are called private operations. These operations are considered to be executed under the table or in the back. Private operations are shown to be the most powerful primitives in card-based cryptographic protocols. They were first introduced to solve millionaires' problem [32]. Using three private operations shown later, committed-input and committed-output logical AND, logical XOR, and copy protocols can be achieved with the minimum number of cards [40]. Another class of private operations is private input operations that are used when a player inputs a private value [17, 38, 50]. These operations are not discussed in this paper since it is impossible to prevent false input from a malicious player. If the input values are honestly given, the players can use the protocols shown in this paper.

The biggest problem of protocols using private operations is malicious actions. Most of the card-based protocols assume the semi-honest model, in which the players obey the rule of the protocols but try to obtain private information. However, there are many cases when we must consider the malicious model. When we allow malicious actions, protocols using private operations are not secure. Since private operations are executed where the other player cannot see, any malicious operation is possible during the private operations, for example, watching the marks of face-down cards or changing the positions of cards.

One countermeasure to malicious actions is setting a watch person. When the protocols are executed by more than two players, it is possible to detect malicious actions by the following rule: whenever a player executes a private operation, another player watches the execution and reports incorrect behavior. The XOR, AND, and copy protocols can be executed securely against a malicious player when the protocols are executed by more than two players [40]. However, when the protocols are executed by two players, it is impossible to use the above method. If Bob watches Alice's private operations, Bob knows all operations, thus the relation between input data and output data is known to Bob. When the output card is opened, the secure input data are known to Bob using the relation between the input data and the output data.

Thus we need new protocols for the two-player case. Since Bob cannot watch Alice's private operations, some additional mechanism to prevent illegally watching the marks of face-down cards during private operations is necessary. This paper introduces envelopes to prevent illegally watching the marks of face-down cards. Cards that must not be seen are publicly put into an envelope. If the envelope is opened, it can be detected by anyone. Envelopes are used in [30] to realize cryptographic protocols that do not use physical cards. In card-based cryptographic protocols, envelopes are used in [8, 36, 44, 49] to realize some kind of shuffles that are not easy to execute by people.

This paper shows new card-based cryptographic protocols that are secure against malicious players using envelopes as an additional tool. The malicious actions during private operations are prevented by adding error-correction cards.

We show logical XOR, logical AND, and copy protocols since any logical functions can be obtained with a combination of these protocols.

As related works, protocols that use additional cards and prevent active attacks while a player executes a shuffle were shown [15]. Another type of active attack is inputting a false value that is not 0 or 1. A protocol to detect such injection attacks was discussed in [27]. Protocols that prevent revealing face-down cards were discussed in [52]. The protocol uses the technique of secret-sharing to prevent information leakage by opening some numbers of cards. The protocol cannot be applied to the problem discussed in this paper since a malicious player might reveal all cards. Another usage of private operations is realizing a public shuffle by multiple private shuffles [29]. Using the method, logical XOR, logical AND, and copy can be executed since there are no malicious actions in these private shuffles. Though the protocols are very simple, the private primitives used in the protocols is private shuffles. Preventing malicious actions for the new protocols that use private random bisection cuts and private reveals are not considered.

A protocol to detect malicious actions by executing two instances of a protocol and comparing the results was shown [46]. The protocol uses cases to prevent revealing face-down cards. The functionality of cases is just the same as the one of envelopes in this paper. The protocol uses twice as many cards as the original protocols and it is impossible to correct the malicious actions. This paper's protocols use fewer cards and can correct the result by malicious actions.

In Section 2, basic notations and the private operations introduced in [40] are shown. Section 3 shows XOR, AND, and copy protocols. Section 4 concludes the paper.

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, $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. Cards of the same marks cannot be distinguished. In addition, the back of both types of cards is $\boxed{?}$. It is impossible to determine the mark in the back of a given card of $\boxed{?}$.

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

All protocols are executed by two players, Alice and Bob. The players might be malicious, that is, they might not obey the rule of the protocols. There is no collusion between Alice and Bob, otherwise private input data can be easily revealed.

2.2 Private operations

We show three private operations introduced in [40]: private random bisection cuts, private reverse cuts, and private reveals.

Primitive 1 *(Private random bisection cut)*

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

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

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

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

We sometimes write the result of the random bisection cut using bit b to a sequence $S_1||S_2$(where $|S_1| = |S_2|$) as $swap(b, S_1||S_2)$. $swap(0, S_1||S_2) = S_1||S_2$ and $swap(1, S_1||S_2) = S_2||S_1$ are satisfied.

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

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

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

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

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

Note that in some protocols below, selecting left m cards is executed after a private reverse cut. The sequence of these two operations is called a private reverse selection. A private reverse selection is the following procedure on an even sequence $S_2 = s_1, s_2, \ldots, s_{2m}$ and a bit $b \in \{0, 1\}$. A player outputs

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

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

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

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

2.3 Space and time complexities

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

The number of rounds was proposed as a criterion to evaluate the time complexity of card-based protocols using private operations [39]. 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 its space complexity and time complexity.

Protocol 1 *(AND protocol in [40])*
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.*

The correctness of the protocol is shown in [40]. The number of cards is four, since the cards of $commit(x')$ is re-used to set $commit(0)$.

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.

2.4 Malicious actions during private operations

We show examples of cheats by a malicious player for the AND protocol shown in Protocol 1. In the first round, Alice may open the cards of $commit(x)$ and read the secret input value x. Alice might swap the two cards of $commit(x)$ and use $\bar{x}$ as the input value. In the second round, Bob might open the cards of $commit(y)$. Bob might set the cards incorrectly, for example, set

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

then the result becomes $x \vee y$ instead of $x \wedge y$. Bob can set any other card sequences to obtain other incorrect results. In the third round, Alice might execute a private reverse selection using a value $b' (\neq b)$. To make the protocol secure against malicious players, all of the above cheats must be prohibited or detected.

3 XOR, AND and copy under malicious model

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

3.1 Additional assumptions for preventing malicious actions

Throughout this paper, we assume that each input is given as a committed value. The output must also be given as a committed value so that the output can be used as an input to further computations. Though some multi-party secure calculation protocols assume that each player knows his/her private input, there are some cases when we cannot assume that. For example, suppose that x_1, x_2 are Alice's private input values and y_1, y_2 are Bob's private input values and they want to securely calculate $(x_1 \vee y_1) \wedge (x_2 \vee y_2)$. After $commit(x_1 \vee y_1)$ and $commit(x_2 \vee y_2)$ are calculated, they need to calculate logical AND of two secret

values. Thus, we need to calculate the logical functions of two committed inputs. If Alice knows an input value, she first commits her input and a committed input protocol can be used.

We add an assumption that for at least one input, say, x multiple copies of $commit(x)$ are given as input. The reason for this assumption is as follows. When a player, say, Alice is given $commit(x)$ and executes a private operation, there is no way for the other player to detect whether Alice maliciously executed swapping two cards of $commit(x)$ and made $commit(\bar{x})$. Since Bob does not know x, Bob cannot claim that $\bar{x}$ is used instead of x. To detect this type of malicious operation, another copy of $commit(x)$ must be given. Using the copy of $commit(x)$, Bob can detect that Alice used $commit(\bar{x})$ instead of $commit(x)$, as shown in the protocols in this paper. Note that a method to obtain multiple copies of inputs using envelopes is shown in section 3.4.

Next, we need to prevent malicious reveal of committed input values. In the following protocols, we use envelopes as an additional tool. The cards can be put into an envelope and sealed. Opening the envelope can be easily detected by anyone. Thus a malicious player cannot irregularly open envelopes during private operations because it is detected by the other player. It is impossible to distinguish two envelopes. No player can prepare the same envelopes in his/her pocket and exchange them for the envelopes used in the protocol. Such envelopes are used in some card-based protocols [8, 36, 44, 49].

We show some basic operations and notations related to the envelopes. The order of the cards put into an envelope is preserved when the cards are removed. For example, a card sequence S is put into an envelope, the output card sequence from the envelope must also be S. In the following protocols, two envelopes, the left and the right envelope are used and the following two types of insertions are applied. The first one is putting each card of commitments to the left and right envelope. For example, put the left cards of $commit(x)$ and $commit(y)$ into the left envelope and put the right cards of $commit(x)$ and $commit(y)$ into the right envelope. When the players remove the cards from the envelopes, $commit(x)$ and $commit(y)$ are obtained. We write the state of the two envelopes as $[commit(x), commit(y)]$. When we swap the left and right envelopes, the output cards become $commit(\bar{x})$ and $commit(\bar{y})$. Thus we write the state of the swapped envelopes as $[commit(\bar{x}), commit(\bar{y})]$.

The second one is putting the left card of $commit(x)$ and the two cards of $commit(y)$ to the left envelope, and putting the right card of $commit(x)$ and the two cards of $commit(z)$ to the right envelope. We write the state of the two envelopes as $[commit(x), commit(y)||commit(z)]$. When we swap the two envelopes, we can obtain $[commit(\bar{x}), commit(z)||commit(y)]$.

In this paper, private random bisection cuts are executed to these two envelopes. When Alice executes a private random bisection cut to the two envelopes that have $[commit(x), commit(y)]$, $[commit(x \oplus b), commit(y \oplus b)]$ is obtained. When Alice executes a private random bisection cut to the two envelopes that have $[commit(x), commit(y)||commit(z)]$,
$[commit(x \oplus b), swap(b, commit(y)||commit(z))]$ is obtained.

With the envelopes, the activities by a malicious player are as follows when the private primitives are private random bisection cuts, private reverse cuts, and private reveals on the envelopes.

Assumption 1 *(Operations by malicious players)*

- *When a malicious player executes a private operation, he/she can swap some envelopes even if it is not allowed in the protocol.*
- *When a malicious player executes a private random bisection cut to two sets of envelopes A and B using the same random bit, he/she can use different bits to A and B.*
- *When a malicious player executes a private reveal on envelope A, he/she can open another envelope B if it cannot be detected by the other player (for example, the number of cards in A and B are the same). Also, he/she might not place envelopes according to the opened cards.*
- *When a malicious player executes a private reverse cut using bit b, he/she might use $\overline{b}$ instead of b.*

3.2 XOR protocol

Protocol 2 *(XOR protocol)*
Input: two copies of $commit(x)$ and one copy of $commit(y)$.
Output: $commit(x \oplus y)$.

1. *Alice and Bob publicly put cards of one $commit(x)$ and $commit(y)$ into two envelopes. The left(right) cards of $commit(x)$ and $commit(y)$ are put into the left(right) envelope. The two envelopes have $[commit(x), commit(y)]$.*
   The remaining two cards of $commit(x)$ are put into two new envelopes so that the left(right) card is put into the left(right) envelope. The two envelopes have $[commit(x)]$.
   The envelopes that have $[commit(x)]$ and $[commit(x), commit(y)]$ are handed to Alice.
2. *Alice executes a private random bisection cut on $[commit(x)]$ and $[commit(x), commit(y)]$ using the same random bit b. Let the output be $[S_1]$ and $[S'_1, S''_1]$. $S_1 = commit(x \oplus b)$, $S'_1 = commit(x \oplus b)$, and $S''_1 = commit(y \oplus b)$. Alice hands $[S_1]$ and $[S'_1, S''_1]$ to Bob.*
3. *Bob first verifies that the envelopes are not opened. Then, Bob executes a private reveal on $[S_1 = commit(x')]$. Bob verifies that the numbers of cards in the envelopes are 1, otherwise Alice incorrectly handed envelopes. Bob privately swaps the two envelopes of $[S'_1, S''_1]$ if $x' = 1$, otherwise, does nothing. Bob makes the two envelopes public, which are denoted $[S'_2, S''_2]$.*
4. *Alice verifies that the envelopes are not opened. Alice and Bob open the envelopes together and obtain S'_2 and S''_2. They turns (that is, face-up) S'_2. If $S'_2 = 0$, S''_2 is the output of the protocol. If $S'_2 = 1$, swap the two cards of S''_2 and the result is the output of the protocol.*

The protocol is three rounds. The first round is the public execution by Alice and Bob. The second round is executed by Alice. The third round is executed by Bob. The last execution by Alice and Bob does not need handing cards or envelopes. Bob just makes the envelopes public and Bob can execute the operations in front of Alice. Thus no overhead is necessary for the public execution. Therefore, the number of rounds is considered to be three. The number of cards used in the protocol is six.

Theorem 1. *The output of the XOR protocol is correct even if Alice or Bob is malicious. The protocol does not reveal the input values to the players if no prohibited opening is executed.*

Proof. First, we show the correctness when both Alice and Bob are honest.

Alice hands $[S_1] = [commit(x \oplus b)]$ and $[S'_1, S''_1] = [commit(x \oplus b), commit(y \oplus b)]$ to Bob. Bob swaps the pair of $[S'_1, S''_1]$ if $x \oplus b = 1$. Thus $[S'_2, S''_2] = [commit((x \oplus b) \oplus (x \oplus b)), commit((y \oplus b) \oplus (x \oplus b))] = [commit(0), commit(x \oplus y)]$. Since $S'_2 = commit(0)$, S''_2 is not swapped and the output is $commit(x \oplus y)$. Therefore, the output is correct. The protocol is secure since Alice sees $S'_2 = 0$ and Bob sees $S'_2 = 0$ and $S_1 = x \oplus b$ but b is an unknown random value for Bob.

Next, consider the case when Alice is malicious and Bob is honest. If Alice opens an envelope during the private operation, Bob can detect the misbehavior. Next, consider the case when Alice does not execute the private random bisection cut correctly. Since the numbers of cards in $[S_1]$ and $[S'_1, S''_1]$ differs, the only cheat that cannot be detected by Bob is incorrectly swapping envelopes. Let b and b' be the random bits selected to swap the envelopes that have $[commit(x)]$ and $[commit(x), commit(y)]$, respectively. The output by Alice is $[S_1] = [commit(x \oplus b)]$ and $[S'_1, S''_1] = [commit(x \oplus b'), commit(y \oplus b')]$. After Bob opens $[S_1] = [commit(x \oplus b)]$, Bob swaps the envelopes if $x \oplus b = 1$, thus the result $[S'_2, S''_2] = [commit(x \oplus b' \oplus x \oplus b), commit(y \oplus b' \oplus x \oplus b)] = [commit(b \oplus b'), commit(y \oplus b' \oplus x \oplus b)]$. When the players open S'_2, they obtain no information about x since $S'_2 = commit(b \oplus b')$. In addition, if $b \oplus b' = 1$, the cards of S''_2 are swapped, thus the output is $commit(y \oplus b' \oplus x \oplus b \oplus (b \oplus b')) = commit(y \oplus x)$. The result is correct regardless of the selection of b and b'.

Next, consider the case Bob is also malicious. When Bob opens the envelopes of $[S'_1, S''_1]$, the cheat can be detected by Alice. Next, consider the case when Bob does not set the envelopes correctly. When Bob sees $x \oplus b$, Bob does not swap the envelopes correctly, that is, Bob selects some value $b''(\neq x \oplus b) \in \{0, 1\}$ and swaps the envelopes of $[S'_1, S''_1]$ using b''. If $b'' = x \oplus b$, the result is correct as shown above. Thus the only cheat selection of b'' is $b'' = \overline{x \oplus b} = x \oplus b \oplus 1$.

In this case, the result is $[S'_2, S''_2] = [commit(x \oplus b' \oplus b''), commit(y \oplus b' \oplus b'')] = [commit(b' \oplus b \oplus 1), commit(y \oplus b' \oplus x \oplus b \oplus 1)]$. When Alice and Bob open S'_2, they do not obtain information about x since the value is independent of x. If $b' \oplus b \oplus 1 = 1$, the two envelopes of S''_2 is swapped. The result is correct since the output is $commit(y \oplus b' \oplus x \oplus b \oplus 1 \oplus (b' \oplus b \oplus 1)) = commit(y \oplus x)$. □

Note that the protocol achieves an error-correction. Even if Alice and/or Bob make mistakes in swapping envelopes, the mistakes are automatically corrected as shown above.

3.3 AND protocol

Protocol 3 *(AND protocol)*
Input: two copies of $commit(x)$ *and one copy of* $commit(y)$.
Output: $commit(x \wedge y)$.

1. *Alice and Bob publicly put cards into two envelopes. The left card of* $commit(x)$ *and two new cards of* $commit(0)$ *are put into the left envelope. The right card of* $commit(x)$ *and the two cards of* $commit(y)$ *are put into the right envelope. The envelopes have* $[commit(x), commit(0)||commit(y)]$.
   The remaining two cards of $commit(x)$ *are put into two envelopes so that the left(right) card is put into the left(right) envelope. The envelopes have* $[commit(x)]$.
   The envelopes that have $[commit(x)]$ *and* $[commit(x), commit(0)||commit(y)]$ *are handed to Alice.*
2. *Alice executes a private random bisection cut on* $[commit(x)]$ *and* $[commit(x), commit(0)||commit(y)]$ *using the same random bit* b. *Let the output be* $[S_1]$ *and* $[S_1', S_1'']$. $S_1 = commit(x')$, *where* $x' = x \oplus b$. $S_1' = commit(x')$ *and* $S_1'' = swap(b, commit(0)||commit(y))$. *Alice hands* $[S_1]$ *and* $[S_1', S_1'']$ *to Bob.*
3. *Bob first verifies that the envelopes are not opened. Bob executes a private reveal on* $[S_1 = commit(x')]$. *Bob verifies that the numbers of cards in the envelopes are 1, otherwise Alice incorrectly handed the envelopes. Bob privately swaps two envelopes of* $[S_1', S_1'']$ *if* $x' = 1$, *otherwise, does nothing. Bob makes the two envelopes public, which are denoted* $[S_2', S_2'']$.
4. *Alice verifies that the envelopes that have* $[S_2', S_2'']$ *are not opened. Alice and Bob open the envelopes together and obtains* S_2' *and* S_2''. *They turn (that is, face-up)* S_2'. *If* $S_2' = 0$, *the left two cards of* S_2'' *is the output of the protocol. If* $S_2' = 1$, *the right two cards of* S_2'' *is the output of the protocol.*

The protocol is three rounds. The protocol uses eight cards since two new cards are used to set $commit(0)$.

Theorem 2. *The output of the AND protocol is correct even if Alice or Bob is malicious. The protocol does not reveal the input values to the players if no prohibited opening is executed.*

Proof. The desired output can be represented as follows.

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

First, we show the correctness when both Alice and Bob are honest.

Alice hands $[S_1] = [commit(x \oplus b)]$ and $[S'_1, S''_1] = [commit(x \oplus b), swap(b, commit(0)||commit(y))]$ to Bob. Bob swaps the pair of $[S'_1, S''_1]$ if $x \oplus b = 1$. Thus $[S'_2, S''_2] = [commit((x \oplus b) \oplus (x \oplus b)), swap(x \oplus b, swap(b, commit(0)||commit(y))] = [commit(0), swap(x, commit(0)||commit(y))]$. Thus the players select the left two cards of $swap(x, commit(0)||commit(y))$. The selected cards are $commit(y)$ if $x = 1$ and $commit(0)$ if $x = 0$. Thus, the output is correct.

The protocol is secure since Alice sees $S'_2 = 0$ and Bob sees $S'_2 = 0$ and $S_1 = x \oplus b$ but b is an unknown random value for Bob.

Next, consider the case when Alice is malicious and Bob is honest. If Alice opens an envelope during the private operation, Bob can detect the misbehavior. Next, consider the case when Alice does not execute the private random bisection cut correctly. Since the numbers of cards in the envelopes for $[S_1]$ and $[S'_1, S''_1]$ differs, the only cheat that cannot be detected by Bob is incorrectly swapping envelopes. Let b and b' be the random bits selected to swap the envelopes that have $[commit(x)]$ and $[commit(x), commit(0)||commit(y)]$, respectively. The output by Alice is $[commit(x \oplus b)]$ and $[commit(x \oplus b'), swap(b', commit(0)||commit(y))]$. After Bob opens $[commit(x \oplus b)]$, Bob swaps the envelopes if $x \oplus b = 1$, thus the result $[S'_2, S''_2] = [commit(x \oplus b' \oplus x \oplus b), swap(x \oplus b, swap(b', commit(0)||commit(y)))] = [commit(b \oplus b'), swap(x \oplus b \oplus b', commit(0)||commit(y))]$. When the players open S'_2, they obtain no information about x since $S'_2 = commit(b \oplus b')$. In addition, if $b \neq b'$, the right two cards of S''_2 are used as the output otherwise, the left two cards of S''_2 are used as the output. This is equivalent to execute $swap(b \oplus b', S''_2)$ and select the left two cards. Since $swap(b \oplus b', S''_2) = swap(b \oplus b', swap(x \oplus b \oplus b', commit(0)||commit(y))) = swap(x, commit(0)||commit(y))$, the output is $commit(0)$ if $x = 0$, otherwise the output is $commit(y)$. Therefore, the output is correct regardless of the selection of b and b'.

Next, consider the case Bob is also malicious. When Bob opens the envelopes of $[S'_1, S''_1]$, the cheat can be detected by Alice. Next, consider the case when Bob does not set the envelopes correctly. When Bob sees $x \oplus b$, Bob does not swap the envelopes correctly, that is, Bob selects some value $b''(\neq x \oplus b) \in \{0, 1\}$ and swaps the envelopes of $[S'_1, S''_1]$ using b''. When $b'' = x \oplus b$, the output is correct since it is the correct value. Thus the only cheat selection of b'' is $b'' = \overline{x \oplus b} = x \oplus b \oplus 1$.

In this case, the result is $[S'_2, S''_2] = [commit(x \oplus b' \oplus b''), swap(b'', swap(b, commit(0)||commit(y)))] = [commit(b \oplus b' \oplus 1), swap(x \oplus b \oplus b' \oplus 1, commit(0)||commit(y))]$. When Alice and Bob open S'_2, they do not obtain information about x since the value is independent of x.

In addition, if $b \oplus b' \oplus 1 = 1$, the right two cards of S''_2 are used as the output otherwise, the left two cards of S''_2 are used as the output. This is equivalent to execute $swap(b \oplus b' \oplus 1, S''_2)$ and select the left two cards. Since $swap(b \oplus b' \oplus 1, S''_2) = swap(b \oplus b' \oplus 1, swap(x \oplus b \oplus b' \oplus 1, commit(0)||commit(y))) = swap(x, commit(0)||commit(y))$, the output is $commit(0)$ if $x = 0$, otherwise the output is $commit(y)$. Therefore, the output is correct regardless of the selection of b and b'. □

Note that even if Alice and/or Bob make mistakes in swapping envelopes, the mistakes are automatically corrected as shown above.

3.4 COPY protocol

Next, we show a copy protocol. Multiple copies of output data of computation might be needed in some cases, for example, use the output result to a further computation. A method to obtain $m(>1)$ copies of the output is preparing m copies of $commit(y)$.

In the XOR protocol, at the first step of the protocol, they put cards into two envelopes so that $[commit(x), commit(y), commit(y), \ldots, commit(y)]$ is obtained. At the last step, S_2'' is m pairs of cards. When they need to swap the cards, each pair of S_2'' is swapped. Then we can obtain m copies of $commit(x \oplus y)$.

In the AND protocol, at the first step of the protocol, they put cards into two envelopes so that $[commit(x), (commit(0), \ldots, commit(0))||(commit(y), \ldots, commit(y))]$ is obtained, that is, put m copies of $commit(0)(commit(y))$ to the left(right) envelope. At the last step, if $S_2' = 0$, the output is the left m pairs of cards. Otherwise, the output is the right m pairs of cards.

We can obtain another protocol that directly increases the number of copies of input data using the XOR protocol. Two copies of $commit(x)$ are given as input. Execute the XOR protocol with two copies of $commit(x)$ and m copies of $commit(0)$. Then the players obtain m copies of $commit(x)$ as the output since $x \oplus 0 = x$.

Last, we show a method to obtain multiple copies of input x using two envelopes. For any number n, n  () cards are publicly put into the left(right) envelope and the envelopes are sealed. The two envelopes are given to the input player. The input player privately sets the two envelopes according to the private input value x. Then all players publicly open the seals of the envelopes and two piles of cards are obtained. When the players select one card from each of the piles, a copy of $commit(x)$ can be obtained, thus n copies of $commit(x)$ can be obtained.

When we calculate general logical functions using the above primitives, we need to prepare two copies of each input. Any number of copies of a value can be obtained by using the copy protocol at any time, if there are two copies of the value. Obtaining two copies of an output value can be realized by the above protocols, thus any logical functions can be calculated securely using these protocols.

4 Conclusion

This paper proposed new protocols using private operations that are secure against malicious players. We show logical XOR, logical AND, and copy protocols that use envelopes for an additional tool. Since the envelopes are a very powerful tool to restrict shuffle executions, malicious executions are corrected in the protocols.

We can consider weak tools for preventing illegal opening face-down cards, for example, seals on the marks of the cards. They cannot restrict shuffle executions. One of the open problems is considering secure protocols with such tools.

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. 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. Cheung, E., Hawthorne, C., Lee, P.: Cs 758 project: Secure computation with playing cards (2013), `http://cdchawthorne.com/writings/secure_playing_cards.pdf`
5. Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for norinori. In: Proc. of International Computing and Combinatorics Conference(COCOON 2019), LNCS Vol. 11653. pp. 166–177. Springer (2019)
6. Dvořák, P., Kouckỳ, M.: Barrington plays cards: The complexity of card-based protocols. arXiv preprint arXiv:2010.08445 (2020)
7. 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)
8. 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)
9. Hashimoto, Y., Shinagawa, K., Nuida, K., Inamura, M., Hanaoka, G.: Secure grouping protocol using a deck of cards. In: Proc. of 10th International Conference on Information Theoretic Security(ICITS 2017), LNCS Vol. 10681. pp. 135–152 (2017)
10. 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)
11. 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)
12. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Proc. of Asiacrypt 2017, Part III, LNCS Vol. 10626. pp. 126–155 (2017)
13. Koch, A.: The landscape of optimal card-based protocols. IACR Cryptology ePrint Archive, Report 2018/951 (2018)
14. Koch, A., Schrempp, M., Kirsten, M.: Card-based cryptography meets formal verification. In: Proc. of Asiacrypt 2019, LNCS Vol. 11921. pp. 488–517. Springer (2019)
15. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: 10th International Conference on Fun with Algorithms (FUN 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)

16. 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)
17. 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)
18. Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: A physical zkp for slitherlink: How to perform physical topology-preserving computation. In: Proc. of 15th International Conference on Information Security Practice and Experience(ISPEC 2019), LNCS Vol. 11879. pp. 135–151. Springer (2019)
19. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. IACR Cryptology ePrint Archive, Report 2015/1031 (2015)
20. Miyahara, D., Hayashi, Y.i., Mizuki, T., Sone, H.: Practical card-based implementations of yao's millionaire protocol. Theoretical Computer Science **803**, 207–221 (2020)
21. 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: 10th International Conference on Fun with Algorithms (FUN 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
22. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for kakuro. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **102**(9), 1072–1078 (2019)
23. Mizuki, T.: Applications of card-based cryptography to education. In: IEICE Technical Report ISEC2016-53. pp. 13–17 (2016), (In Japanese)
24. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theoretical Computer Science **622**, 34–44 (2016)
25. 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)
26. 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)
27. 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)
28. 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)
29. 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)
30. Moran, T., Naor, M.: Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In: Proc. of EUROCRYPT 2006, LNCS Vol. 4004. pp. 88–108. Springer (2006)
31. Nakai, T., Shirouchi, S., Iwamoto, M., Ohta, K.: Four cards are sufficient for a card-based three-input voting protocol utilizing private sends. In: Proc. of 10th International Conference on Information Theoretic Security (ICITS 2017), LNCS Vol. 10681. pp. 153–165 (2017)
32. 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)

33. 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)
34. 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)
35. Nishida, T., Mizuki, T., Sone, H.: Securely computing the three-input majority function with eight cards. In: 2nd International Conference on Theory and Practice of Natural Computing(TPNC 2013), LNCS Vol. 8273. pp. 193–204 (2013)
36. Nishimura, A., Hayashi, Y.i., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101**(9), 1494–1502 (2018)
37. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**(2), 361–371 (2018)
38. 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)
39. Ono, H., Manabe, Y.: Card-based cryptographic protocols with the minimum number of rounds using private operations. In: Proc. of 14th International Workshop on Data Privacy Management (DPM 2019) LNCS Vol. 11737. pp. 156–173 (2019)
40. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Generation Computing pp. 1–22 (2020)
41. 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)
42. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for numberlink. arXiv preprint arXiv:2002.01143 (2020)
43. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for ripple effect. arXiv preprint arXiv:2009.09983 (2020)
44. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. In: International Conference on Theory and Applications of Models of Computation. pp. 25–36. Springer (2020)
45. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for sudoku. Theoretical Computer Science (2020)
46. Shimizu, Y., Kishi, Y., Sasaki, T., Fujioka, A.: Card-based cryptographic protocols with private operations which can prevent malicious behaviors. In: IEICE Techinical Report ISEC2017-113. pp. 129–135 (2018), (In Japanese)
47. Shinagawa, K., Mizuki, T.: The six-card trick:secure computation of three-input equality. In: Proc. of 21st International Conference on Information Security and Cryptology (ICISC 2018), LNCS Vol. 11396. pp. 123–131 (2018)
48. Shinagawa, K., Mizuki, T.: Secure computation of any boolean function based on any deck of cards. In: Proc. of 13th International Workshop on Frontiers in Algorithmics (FAW 2019), LNCS Vol. 11458. pp. 63–75. Springer (2019)
49. Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any boolean circuit. Discrete Applied Mathematics **289**, 248–261 (2021)
50. 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)

51. Takashima, K., Abe, Y., Sasaki, T., Miyahara, D., Shinagawa, K., Mizuki, T., Sone, H.: Card-based protocols for secure ranking computations. Theoretical Computer Science **845**, 122–135 (2020)
52. Takashima, K., Miyahara, D., Mizuki, T., Sone, H.: Card-based protocol against actively revealing card attack. In: Proc. of 9th International Conference on Theory and Practice of Natural Computing(TPNC 2019), LNCS Vol. 11934. pp. 95–106. Springer (2019)
53. 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. pp. 2–8 (2020)
54. Watanabe, Y., Kuroki, Y., Suzuki, S., Koga, Y., Iwamoto, M., Ohta, K.: Card-based majority voting protocols with three inputs using three cards. In: 2018 International Symposium on Information Theory and Its Applications (ISITA). pp. 218–222. IEEE (2018)