Card-Based Cryptographic Logical Computations Using Private Operations

Hibiki Ono & Yoshifumi Manabe

New Generation Computing

ISSN 0288-3635

New Gener. Comput.
DOI 10.1007/s00354-020-00113-z

Springer

New Generation Computing
https://doi.org/10.1007/s00354-020-00113-z

Card-Based Cryptographic Logical Computations Using Private Operations

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

Received: 15 April 2020 / Accepted: 3 October 2020

Abstract
This paper proposes new card-based cryptographic protocols to calculate logic functions 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 AND, 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. Next, we consider protocols with malicious players.

Keywords Multi-party secure computation · Card-based cryptographic protocols · Private operations · Logical computations · Copy

Preliminary version of the paper was presented as Hibiki Ono and Yoshifumi Manabe: " Card-based Cryptographic Protocols with the Minimum Number of Cards Using Private Operations," Proc. of 11th International Symposium on Foundations & Practice of Security (FPS 2018) LNCS Vol. 11358, pp.193–207 (Apr. 2019).

✉ Yoshifumi Manabe
manabe@cc.kogakuin.ac.jp

[1] Faculty of Informatics, Kogakuin University, 1-24-2, Nishisinjuku, Shinjuku, Tokyo 163–8677, Japan

Published online: 31 October 2020

Introduction

Card-based cryptographic protocols [11, 26] 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 AND of two inputs. Since then, many protocols have been proposed to calculate logical functions [4, 5, 21, 24, 27, 30, 40, 44, 47] and specific computations such as computations on three inputs [32, 33, 43], calculation of symmetric functions [41], millionaires' problem [18, 29, 37], voting [22, 28, 51], random permutation [6, 8, 9], grouping [7], matching [17], ranking [48], zero-knowledge proof of puzzle solutions [3, 16, 19, 42] and so on. Most of the protocols assume a semi-honest model, that is, players obey the rule of the protocol but try to obtain secret values. This paper also assumes the semi-honest model in most sections. We discuss malicious players in Sect. 5.

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

First, randomization primitives have been discussed and then recently, private operations are considered. Many protocols use random bisection cuts [27], 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. [50] 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 the 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 [50], which is a normal shuffle operation in playing cards. Koch and Walzer [13] 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. Pile-shifting scramble [34] was proposed that achieves non-uniform shuffles with some special tools. These protocols can be executed in a public place, but they need additional special cards or tools.

A simple solution to execute private randomization is multiple player randomization, in which some operations are executed in a hidden place. 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. 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. Therefore, 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 [29]. The private operations used in the papers are similar to the primitives proposed in this paper, but it is not clear that the primitives can be used for the other protocols. Then private operations were used to calculate logical functions [15, 46]. 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 committed values. Thus, protocols that accept committed inputs are desirable. Another desirable property is committed output. If the output is given as a committed value, a 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 AND, 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, six was the minimum for finite-runtime copy [27], as shown in Table 1. The protocol in [36] uses 5 cards, but the number of steps of the protocol is unbounded. It is proved to be impossible to achieve a copy with four cards by the conventional model without private operations [10]. 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.

Table 1 Comparison of copy protocols in two type card model

Article	# of cards	Note
Crépeau et al. (1993) [4]	8	
Mizuki et al. (2009) [27]	6	
Nishimura et al. (2015) [35]	5	Uses unequal shuffle
Nishimura et al. (2018) [36]	5	Number of steps is not bounded
This paper	4	Uses private operations

Table 2 Comparison of AND protocols in the two type card model (except for (*))

Article	# of cards	Input	Output	Note
den Boer (1989) [2]	5	commit	non-commit	
Crépeau et al. (1993) [4]	10	commit	commit	(*) Four color cards
Niemi et al. (1998) [30]	12	commit	commit	
Stiglic (2001) [47]	8	commit	commit	
Mizuki et al. (2009) [27]	6	commit	commit	
Mizuki et al. (2012) [24]	4	commit	non-commit	
Koch et al. (2015) [14]	4	commit	commit	Non-uniform shuffle
Shirouchi et al. (2017) [46]	3	non-commit	non-commit	Uses private operations
Kurosawa et al. (2017) [15]	4	non-commit	commit	Uses private operations
Abe et al. (2018) [1]	5	commit	commit	Number of steps is not bounded
Ruangwises et al. (2019) [40]	4	commit	commit	Number of steps is not bounded
This paper	4	commit	commit	Uses private operations

The numbers of cards in committed-input, committed-output AND protocols are shown in Table 2. The protocol in [27] uses six cards. Though the protocol in [14] uses four 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 [40] does not terminate within a finite time. It is proved to be impossible to achieve finite-runtime AND with four cards by the conventional model without private operations [10, 12]. Our protocol uses four 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 [27], an input preserving (shown in Sect. 4.7) can be realized by our protocol without additional cards.

This paper then shows several variants of the protocols to calculate any logic functions that includes side effects of the calculations, parallel computations, and preserving an input.

Then, Sect. 5 discusses protocols when a malicious player exists. Last, Sect. 6 shows protocols that use asymmetric cards.

Table 3 Comparison of XOR protocols in the two type card model

Article	# of cards	Input	Output	Note
Crépeau et al. (1993) [4]	14	commit	commit	Four color cards
Mizuki et al. (2009) [27]	4	commit	commit	
Shirouchi et al. (2017) [46]	2	non-commit	commit	
Kurosawa et al. (2017) [15]	2	non-commit	commit	
This paper	4	commit	commit	Uses private operation Preserving an input is possible

An earlier version of this paper was presented and appeared as a conference paper [38]. The main difference is as follows. Execution examples are shown for the main protocols. This paper improved the discussion in Sect. 5. The detail of the cheat alert protocol is shown. The correctness proof of the protocol is newly written. In addition, a false alert prevention protocol is newly shown. Section 6 is a new section that proposes new protocols using asymmetric cards.

Preliminaries

This section gives the notation and basic definitions of card-based protocols. All Sections other than Section 6 are based on two type card model that is most commonly considered. 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.

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. $S = \underbrace{\boxed{?}}_{s_1}\underbrace{\boxed{?}}_{s_2}\underbrace{\boxed{?}}_{s_3}\cdots,\underbrace{\boxed{?}}_{s_n}$. 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. In this paper except for Sect. 5, 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.

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 $S_0 = s_1, s_2, \ldots, s_{2m}$. 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 does not disclose the bit b.

Note that the protocols in this paper use the operation only when $m = 1$ and $S_0 = \mathrm{commit}(x)$. 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 $\underbrace{\boxed{?}\boxed{?}}_{\bar{x}}$.

Note that a private random bisection cut is the same as the random bisection cut [27], but the operation is executed in a hidden place.

Primitive 2 (Private reverse cut, Private reverse selection) *A private reverse cut is the following operation on an even sequence $S_2 = s_1, s_2, \ldots, s_{2m}$ and a bit $b \in \{0, 1\}$ that is given to the player. The 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 player cannot see. The player must not disclose the bit b to the other players.

The difference between the private random bisection cut is that b is not newly selected by the player.

When a player executes a private reverse cut on

$S_2 = \underbrace{\boxed{?}\boxed{?}}\cdots\underbrace{\boxed{?}\boxed{?}\boxed{?}}\cdots\underbrace{\boxed{?}}$, the player outputs

$S_3 = \underbrace{\boxed{?}\boxed{?}}\cdots\underbrace{\boxed{?}\boxed{?}\boxed{?}}\cdots\underbrace{\boxed{?}}$ if $b = 0$ or

$S_3 = \underbrace{\boxed{?}\boxed{?}}_{s_{m+1}\ s_{m+2}}\cdots\underbrace{\boxed{?}\boxed{?}\boxed{?}}_{s_{2m}\ s_1\ s_2}\cdots\underbrace{\boxed{?}}_{s_m}$ if $b = 1$.

When a private reverse cut is executed using bit b to the sequence to which a private random bisection cut is executed using b, the private reverse cut can undo the private random bisection cut.

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 given to the player. The 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.

Primitive 3 (Private reveal) *A player privately opens a given committed bit. The player does not disclose the value to the other players.*

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

Though the player seems to obtain a secret value by a private reveal, it is avoided by the following procedure. Alice executes a private random bisection cut to commit(x). The result becomes to commit($x \oplus b$). When Bob executes a private reveal to commit($x \oplus b$), Bob has no information about x if b is randomly chosen 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 rounds [39].

The time complexity of protocols in the model without private operations was discussed [20]. The number of rounds was considered as the time complexity of the protocol with private operations. The definition of rounds is as follows [39]. The first round is (possibly parallel) local executions by each player using the cards initially given to each player. The first round begins from the initial state. 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 $i(>1)$-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 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. Since each operation is relatively simple, the dominating time to execute protocols with private operations is the time to sending cards between players and setting up so that the cards are not seen by the other players. Thus the number of rounds is the criteria to evaluate the time complexity of card-based protocols with private operations.

New Copy, AND, and XOR Protocols

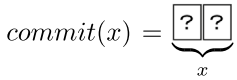
Using the private random bisection cuts, private reveals, and private reverse cuts, cory protocol, AND protocol, and XOR protocol with committed inputs and committed outputs can be realized with the minimum number of cards. All of these protocols are executed between two players, Alice and Bob. In Sect.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

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 sends* 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 sends 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 requires three rounds.

Example 1 (copy protocol) Suppose that $x = 1$. Input $commit(x) =$  is given.

Alice executes a private random bisection cut on commit(x). Alice randomly selects bit $b \in \{0, 1\}$. Let us suppose that $b = 1$. In this case, Alice outputs $commit(x') =$  to Bob. Bob privately opens commit(x'). Since $x = 1$ and $b = 1$,  .

Bob makes m copies of x' as  . Bob faces down this sequence and sends it to Alice.

Alice executes a private reverse cut on each pair of the cards. Since $b = 1$, all pairs are swapped. If the output is opened,  . Thus, the output is m copies of commit(x).

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. □

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.

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 sends* commit(x') *and* commit(y) *to Bob.*
2. *Bob executes a private reveal on* commit(x'). *Bob sets*

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

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

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 is leaked if the cards are opened. The protocol requires three rounds.

Example 2 (AND protocol)

Suppose that $x = 0$ and $y = 1$. Input $commit(x) = \underbrace{?\,?}_{x}$ and $commit(y) = \underbrace{?\,?}_{y}$ are given. Alice executes a private random bisection cut on commit(x). Alice randomly selects bit $b \in \{0, 1\}$. Let us suppose that $b = 0$. Alice sends $commit(x') = \underbrace{?\,?}_{x \oplus b}$ and commit(y) to Bob. Bob privately opens commit(x'). Since $x = 0$ and $b = 0$, $x' = \underbrace{\clubsuit\,\heartsuit}_{0}$. Thus, Bob sets $S_2 = \underbrace{\clubsuit\,\heartsuit}_{0}\underbrace{?\,?}_{y}$. Bob faces down the left cards and sends S_2 to Alice. Alice executes a private reverse selection on S_2. Since $b = 0$, the left two cards are selected. If the output, S_3, is opened, $S_3 = \underbrace{\clubsuit\,\heartsuit}_{0}$. Since $x \wedge y = 0$, the result is correct.

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.

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.

A comparison of AND protocols is shown in Table 2. Though Koch et al. [14] 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.

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 sends* commit(x') *and* commit(y) *to Bob.*
2. *Bob executes a private reveal on* commit(x'). *Bob sets*

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

and sends 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 requires three rounds.

Example 3 (XOR protocol) Suppose that $x = 1$ and $y = 0$. Input $commit(x) =$  and $commit(y) =$  are given. Alice executes a private random bisection cut on commit(x). Alice randomly selects bit $b \in \{0, 1\}$. Let us suppose that $b = 0$. Alice outputs $commit(x') =$  and commit(y) to Bob. Bob privately opens commit(x'). Since $x = 1$ and $b = 0$, $x' =$  . Thus, Bob sets $S_2 =$  . Bob sends S_2 to Alice.

Alice executes a private reverse cut on S_2. Since $b = 0$, the cards are unchanged. If the output, S_3, is opened, $S_3 =$  . Since $x \oplus y = 1$, the result is correct.

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.

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

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 [27]. an input preserving (shown in Sect. 4.7) can be realized without additional cards.

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

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

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

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 sends 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.

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

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

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 [27].

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

Preserving 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, the input preserving calculation is discussed [31].

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 in Theorem 4, commit(x') can be sent back to Alice and Alice can recover commit(x). This modified protocol needs six cards in total.

An input preserving calculation without increasing the number of cards can be executed for AND type protocols just like [31], which recovers commit(y). 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, preserving an input can be realized by 4 cards, which is the minimum. A comparison of input preserving AND type protocols is shown in Table 4.

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.

Table 4 Comparison of input preserving AND protocols in the two type card model

Article	# Of cards	Input	Output	Note
Nishida et al. (2015) [31]	6	commit	commit	
This paper	4	commit	commit	Uses private operations

Using the technique in [31] 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 stores the intermediate value to calculate f.
p_2 stores the intermediate value to calculate T_i.
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 [31] is six. 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 [15]. 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* $\mathrm{commit}(x_i)(i = 1, 2, \ldots, n)$. *Let the output be* $\mathrm{commit}(x'_i)(i = 1, 2, \ldots, n)$ *Note that one random bit* b_i *is selected for each* $x_i(i = 1, 2, \ldots, n)$. *Alice sends* $\mathrm{commit}(x'_i)(i = 1, 2, \ldots, n)$ *to Bob.*
2. *Bob executes a private reveal on* $\mathrm{commit}(x'_i)(i = 1, 2, \ldots, n)$. *Bob generates* 2^n *commitments* $S_{a_1,a_2,\ldots,a_n}(a_i \in \{0, 1\}, i = 1, 2, \ldots, n)$ *as* $S_{a_1,a_2,\ldots,a_n} = \mathrm{commit}(f(a_1 \oplus x'_1, a_2 \oplus x'_2, \ldots, a_n \oplus x'_n))$. *Bob sends these commitments to Alice.*
3. *Alice outputs* $S_{b_1,b_2,\ldots,b_n}$.

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

Note that in the model without private operations, a new protocol to calculate any logical functions using garbled circuits was proposed [44].

Improving Security

Although this paper assumes all players are semi-honest, some players might be malicious in real cases. In the model without private operations, analysis of mistakes [23] and prevention of revealing attacks [49] were considered. In the model with private operations, it is very hard to prevent malicious actions during 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. Let us consider the case with three players, Alice, Bob, and Carol. Bob acts as a watch person for Alice. Carol acts as a watch person for Bob. Alice acts as a watch person for Carol. All the players are in one room. One player goes out of the room for a while and does not watch the execution during the period.

Protocol 6 (Cheat alert AND protocol) *Input:* $\mathrm{commit}(x)$ *and* $\mathrm{commit}(y)$. *Output:* $\mathrm{commit}(x \wedge y)$.

1. *Carol goes out the room. Alice executes a private random bisection cut on* commit(x) *in front of Bob. Let* b_1 *be the random bit Alice selected. Bob knows* b_1. *The obtained data is* commit($x \oplus b_1$).
2. *Carol comes back to the room. Then Alice goes out of the room. Bob executes a private random bisection cut on* commit($x \oplus b_1$) *in front of Carol. Let* b_2 *be the random bit Bob selected. Carol knows* b_2. *The obtained data is* commit($x \oplus b_1 \oplus b_2$).
3. *Alice comes back to the room. Then Bob goes out of the room. Carol executes a private reveal on* commit($x \oplus b_1 \oplus b_2$) *in front of Alice. Let* $x' = x \oplus b_1 \oplus b_2$. *Carol sets*

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

   in front of Alice. Alice knows the value of $x \oplus b_1 \oplus b_2$.
4. *Bob comes back to the room. Then Carol goes out of the room. Alice executes a private reverse cut using the bit* b_1 *on* S_2 *in front of Bob. Let the obtained cards be* S_2'.
5. *Carol comes back to the room. Then Alice goes out of the room. Bob executes a private reverse selection using the bit* b_2 *on* S_2' *in front of Carol. Let the obtained cards be* S_3.
6. *Alice comes back to the room. Now Alice also knows that* S_3 *is the final output.*

Theorem 5 *The cheat alert AND protocol is correct, secure, and be able to alert misbehavior of players if there is no collusion of players.*

Proof About the misbehavior of players: At Step 1, Bob can verify that (1) Alice does not open the cards, (2) Alice really uses a random number generator to select her random bit, and (3) Alice honestly executes a private random bisection cut using the random bit.

At Step 2, Carol can verify that (1) Bob does not open the cards, (2) Bob really uses a random number generator to select his random bit, and (3) Bob honestly executes a private random bisection cut using the random bit.

At Step 3, Alice can verify that (1) Carol does not open the cards that are not allowed and (2) Carol honestly generates the committed cards using the value Carol opened.

At Step 4, Bob can verify that (1) Alice does not open the cards and (2) Alice honestly executes a private reverse cut using the bit generated at Step 1.

At Step 5, Carol can verify that (1) Bob does not open the cards and (2) Bob honestly executes a private reverse selection using the bit generated at Step 2.

By these verifications, the players can understand that the protocol is correctly executed if there is no alert.

About the security, Alice knows b_1 and $x \oplus b_1 \oplus b_2$. Bob knows b_1 and b_2. Carol knows b_2 and $x \oplus b_1 \oplus b_2$. Thus, every player has no knowledge about x if no collusion exists.

The correctness of the protocol: the same as the AND protocol. □

Similar protocols can be obtained for all the other protocols shown above.

Note that in the above three-player protocol, false alarms cannot be prevented. If Bob alerts that Alice misbehaved even if there is no misbehavior, there is no way for Carol to decide which of Alice and Bob is correct. In order to prevent this type of malicious behavior, the protocol needs to be executed more than three players. The protocol with more than three players $P_0, P_1, \ldots, P_{n-1} (n > 3)$ is as follows:

Protocol 7 (False alert prevention AND protocol) *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 verify the correctness of the current player.*

1. $P_j (j = 0, \ldots, n-2)$ *executes a private random bisection cut on* $\mathrm{commit}(x)$ *using random bit* b_j *when* $P_{j+i \mod n}$ *is out of the room, thus* $\mathrm{commit}(x \oplus_{j=0}^{n-2} b_j)$ *is obtained.*
2. P_{n-1} *executes a private reveal and sets* S_2 *when* $P_{n-1+i \mod n}$ *is out of the room.*
3. $P_j (j = 0, \ldots, n-2)$ *executes a private reverse cut using* b_j *when* $P_{j+i \mod n}$ *is out of the room.*
4. *After all the private reverse cuts are finished, the left pair is selected as the output.*

At any step of the protocol, one player executes some operation and $n - 2 (> 1)$ players watch the execution since one player is out of the room. If one malicious player P_j misbehaves some operation, $n - 2 (> 1)$ watching players alert, thus the misbehavior is detected by the majority. If a watching player falsely alert that P_j misbehaved, the other watching player(s) and P_j say that P_j is correct, thus the alert is detected false by the majority. The correctness of the protocol is just the same as the original AND protocol. In this execution, any player cannot obtain the value of the committed value because he does not have all information if no collusion exists.

Similar protocols can be obtained for all the other protocols shown above.

Asymmetric Card Protocols

When we use cards whose face is not symmetric, such as $\boxed{\clubsuit}$ and $\boxed{\clubsuit}$, but the back is symmetric, one bit data can be represented by one card as $\boxed{\clubsuit}_{=0}$ and $\boxed{\clubsuit}_{=1}$. Protocols with this type of card are first considered in [25] and then several protocols are shown in [45, 46]. For such an encoding method, a private random bisection cut on a committed bit is changed to upside-down the card according to the random bit. A private reverse cut and a private reverse selection on an even sequence are unchanged. A private reverse cut and a private reverse selection on a single card are changed as upside down the card according to the random bit selected in the private random bisection cut. Using these private operations, all protocols shown above work for the asymmetric cards. The number of cards used by this protocol is half of the two-type card protocols. Copy, AND, and XOR protocols are shown.

Protocol 8 (asymmetric card copy protocol) *Input:* commit(x). *Output: m copies of* commit(x).

1. *Alice randomly selects bit b. If b*=1, *Alice turns* commit(x) *upside down. Let the output be* commit(x'). *Note that* $x' = x \oplus b$. *Alice sends* 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 sends these cards, m copies of* commit(x'), *to Alice.*
3. *If* $b = 1$, *Alice turns each copy of* commit(x') *upside down.*

Example 4 (asymmetric card copy protocol) Suppose that $x = 0$. Input $commit(x) = \underbrace{\boxed{?}}_{x}$ is given. Alice randomly selects bit $b \in \{0, 1\}$. Let us suppose that $b = 1$. Alice turns commit(x) upside down and sends $commit(x') = \underbrace{\boxed{?}}_{x \oplus b}$ to Bob. Bob privately opens commit(x'). Since $x = 0$ and $b = 1$, $x' = \boxed{\clubsuit}$. Thus, Bob sets $\underbrace{\boxed{\clubsuit}\boxed{\clubsuit}\cdots\boxed{\clubsuit}}_{m \text{ copies of } 1}$. Bob faces down these cards and sends them to Alice.

Since $b = 1$, Alice turns the cards upside down. The output is m copies of commit(x).

Protocol 9 (asymmetric card AND protocol) *Input:* commit(x) *and* commit(y). *Output:* commit($x \wedge y$).

1. *Alice randomly selects bit b. If b*=1, *Alice turns* commit(x) *upside down. Let the output be* commit(x'). *Alice sends* commit(x') *and* commit(y) *to Bob.*
2. *Bob executes a private reveal on* commit(x'). *Bob sets*

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

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

Example 5 (asymmetric card AND protocol) Suppose that $x = 0$ and $y = 1$. Input $commit(x) = \underbrace{\boxed{?}}_{x}$ and $commit(y) = \underbrace{\boxed{?}}_{y}$ are given. Alice randomly selects bit $b \in \{0, 1\}$. Let us suppose that $b = 1$. Alice turns commit(x) upside down and sends $commit(x') = \underbrace{\boxed{?}}_{x \oplus b}$ and commit(y) to Bob. Bob privately opens commit(x'). Since

$x = 0$ and $b = 1$, $x' =$  . Thus, Bob sets $S_2 =$. Bob faces down the right card and sends S_2 to Alice.

Alice executes a private reverse selection on S_2. Since $b = 1$, the right card is selected. If the output, S_3, is opened, $S_3 =$  . Since $x \wedge y = 0$, the result is correct.

Protocol 10 (asymmetric card XOR protocol) *Input:* commit(x) *and* commit(y). *Output:* commit($x \oplus y$).

1. *Alice randomly selects bit b. If b*=1, *Alice turns* commit(x) *upside down. Let the output be* commit(x'). *Alice sends* commit(x') *and* commit(y) *to Bob.*
2. *Bob executes a private reveal on* commit(x'). *If* $x' = 1$, *Bob turns* commit(y) *upside down. Let the obtained card be* S_2. *Bob sends* S_2 *to Alice.*
3. *Alice executes a private reverse cut on* S_2 *using the bit b generated in the private random bisection cut. Let the obtained card be* S_3. *Alice outputs* S_3.

Example 6 (asymmetric card XOR protocol) Suppose that $x = 0$ and $y = 1$. Input $commit(x) =$  and $commit(y) =$ are given. Alice randomly selects bit $b \in \{0, 1\}$. Let us suppose that $b = 1$. Alice turns commit(x) upside down and sends $commit(x') =$  and commit(y) to Bob. Bob privately opens commit(x'). Since $x = 0$ and $b = 1$, $x' =$  . Thus, Bob turns commit(y) upside down and obtains S_2. Bob sends S_2 to Alice.

Since $b = 1$, Alice turns S_2 upside down and obtains S_3. If the output, S_3, is opened, $S_3 =$  . Since $x \oplus y = 1$, the result is correct.

Theorem 6 *Using asymmetric cards, Copy, AND, and XOR can be realized using two cards and in three rounds.*

Proof The correctness and security proofs are just the same as the ones for the two type card protocols. □

Conclusion

This paper proposed new card-based cryptographic protocols with the minimum number of cards using private operations. Though private operations are effective, there is a worry of the malicious actions during private operations. Thus, one of the most important open problems is detecting malicious actions when the number of players is two.

Acknowledgements The authors would like to thank anonymous referees who gave us valuable comments to improve this paper.

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. Crépeau, C., Kilian, J.: Discreet solitary games. In: Proc. of 13th Crypto, LNCS Vol. 773, pp. 319–330 (1993)
5. 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)
6. Hashimoto, Y., Nuida, K., Shinagawa, K., Inamura, M., Hanaoka, G.: Toward finite-runtime card-based protocol for generating hidden random permutation without fixed points. In: IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101-A**(9), 1503–1511 (2018)
7. 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)
8. 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)
9. 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)
10. 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)
11. Koch, A.: The landscape of optimal card-based protocols. IACR Cryptology ePrint Archive, Report 2018/951 (2018)
12. 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)
13. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. Cryptology ePrint Archive, Report 2017/423 (2017)
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. 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)
17. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. IACR Cryptology ePrint Archive, Report 2015/1031 (2015)
18. Miyahara, D., Hayashi, Y.I., Mizuki, T., Sone, H.: Practical card-based implementations of yao's millionaire protocol. Theor. Comput. Sci. **803**, 207–221 (2020)

19. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for kakuro. IEICE Trans. Fund. Electron. Commun. Comput. Sci. **102**(9), 1072–1078 (2019)
20. 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. Springer (2018)
21. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theor. Comput. Sci. **622**, 34–44 (2016)
22. 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)
23. Mizuki, T., Komano, Y.: Analysis of information leakage due to operative errors in card-based protocols. In: Proc. of 29th International Workshop on Combinatorial Algorithms(IWOCA 2019), LNCS Vol. 10979, pp. 250–262. Springer (2018)
24. 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)
25. 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)
26. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Trans. Fund. Electron. Commun. Comput. Sci. **100**(1), 3–11 (2017)
27. 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)
28. 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)
29. 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)
30. Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theor. Comput. Sci. **191**(1), 173–183 (1998)
31. 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)
32. 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)
33. 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)
34. 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)
35. 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)
36. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Comput. **22**(2), 361–371 (2018)
37. Ono, H., Manabe, Y.: Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In: Proc. of 13th Asia Joint Conference on Information Security(AsiaJCIS 2018), pp. 23–28 (2018)
38. 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 and Practice of Security(FPS 2018), LNCS Vol. 11358, pp. 193–207. Springer (2019)
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. 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)
41. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2*n* cards. arXiv preprint arXiv:1911.05994 (2019)
42. 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)
43. 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)
44. Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any circuit. Cryptology ePrint Archive, Report 2019/380 (2019). https://eprint.iacr.org/2019/380. Accessed 1 Apr 2020
45. Shinagawa, K., Nuida, K., Nishide, T., Hanaoka, G., Okamoto, E.: Committed and protocol using three cards with more handy shuffle. In: Proc. of International Symposium on Information Theory and Its Applications (ISITA 2016), pp. 700–702 (2016)
46. 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)
47. Stiglic, A.: Computations with a deck of cards. Theor. Comput. Sci. **259**(1), 671–678 (2001)
48. Takashima, K., Abe, Y., Sasaki, T., Miyahara, D., Shinagawa, K., Mizuki, T., Sone, H.: Card-based secure ranking computations. In: Proc. of 13th International Conference on Combinatorial Optimization and Applications (COCOA 2019), LNCS Vol. 11949, pp. 461–472. Springer (2019)
49. 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)
50. 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)
51. 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)