Card-based Cryptographic Protocols for Three-input Functions Using Private Operations

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 card-based cryptographic protocols to calculate several Boolean functions using private operations under the semi-honest model. Private operations, introduced by Nakai et al. are the most powerful operations for card-based protocols. We showed that copy, logical AND, and logical XOR can be calculated with the minimum number of cards using three private operations, private random bisection cuts, private reverse cuts, and private reveals. This paper shows that by using these private operations, all of the following Boolean functions can be calculated without additional cards other than the input cards: (1) Any three input Boolean functions, (2) Half adder and full adder, and (3)Any n-input symmetric Boolean functions. The numbers of cards used in these protocols are smaller than the ones without private operations.

Keywords: card-based cryptographic protocols · multi-party secure computation · Boolean functions · half adder · symmetric functions

1 Introduction

Card-based cryptographic protocols [14, 30] have been proposed in which physical cards are used instead of computers to securely calculate values. They can be used when computers cannot be used or users cannot trust software in the computers. They can also be used to teach the foundation of cryptography [4, 26]. den Boer [2] first showed a five-card protocol to securely calculate logical AND of two inputs. Since then, many protocols have been proposed to calculate logical AND, logical XOR, and copy primitives to compute general Boolean functions [1, 5, 8, 13, 15, 18, 31, 32, 36, 37, 39, 42, 45, 55, 58] and specific computations such as a class of Boolean functions [22, 27, 29, 38, 46, 51], computation using garbled circuits [52], simulation of universal computation such as Turing machines [7, 16], millionaires' problem [23, 34, 40], voting [28, 35, 59, 60], random permutation [9,11,12,33], grouping [10], ranking [56], lottery [53], proof of knowledge of a puzzle solution [3, 6, 20, 24, 25, 43, 44, 47–49], and so on. This paper considers the calculation of Boolean functions under the semi-honest model.

The protocols are executed by two players, Alice and Bob. Though this paper and many other papers assume semi-honest model, malicious actions or mistakes

might occur in real cases. Preventing or detecting such actions were considered [17, 21, 42, 57].

There are several types of protocols regards to the inputs and outputs of the computations. The first type is committed inputs, where the inputs are given as committed values. The players do not know the input values. The other type is non-committed inputs, where players give their own private inputs to the protocol. Protocols with committed inputs are desirable since they can be used for non-committed inputs: each player can give his own private input value as a committed value. Some protocols output their computation results as committed values. The players do not know the output values. The other type of protocols output the result as a non-committed value, that is, the final result is obtained by opening some or all cards. Protocols with committed outputs are desirable since the committed output result can be used as an input to another computation. If further computation is unnecessary, the players just open the committed outputs and obtain the result. Thus, this paper discusses protocols with committed inputs and committed outputs.

This paper assumes the standard two-type card model, in which one bit data is represented by two cards. The detail is shown in Section 2.

Operations that a player executes where the other players cannot see are called private operations. They are considered to be executed under the table, in the back, and so on. They were first introduced by Nakai et al. to solve millionaires' problem [34]. Using private operations, committed-input and committed-output logical AND, logical XOR, and copy protocols can be achieved with four cards, that is, without additional cards other than the input (output) cards, with finite steps, and without non-uniform shuffles [42]. The AND protocol in [31] without private operations uses six cards. It is proved to be impossible to achieve finite-runtime AND with four cards by the model without private operations [13, 15]. As for the number of cards used for copy protocols, six was the minimum for finite-runtime copy [31] without private operations. It is proved to be impossible to achieve a copy with four cards by the model without private operations [13]. Thus private operations are effective in card-based protocols.

Another type of private operations, we call private input operations, were introduced to calculate Boolean functions with non-committed inputs [19, 54]. A player uses the private input operations to input his own private values to the protocol. The operations were used also in millionaires' problem [40], voting [59], and so on. Since this paper considers committed inputs, private input operations are not used.

Though the private operations are powerful, it is shown that we can calculate any n-input Boolean functions with four additional cards [42]. Thus the research question is obtaining the class of Boolean functions that can be calculated without additional cards using the private operations. This paper shows new card-based protocols using private operations to calculate (1) any three input Boolean functions and (2) half adder and full adder, and (3) any n-input symmetric Boolean functions. All of these protocols need no additional cards other than the input cards. Thus these protocols are optimal regards to the

number of cards. In [37, 38] two additional cards were necessary to calculate these functions without private operations.

In Section 2 basic definitions are shown. Section 3 shows the private operations introduced by [42]. Section 4 shows the sub-protocols shown in [42] that are used in this paper. Section 5 shows protocols to calculate three input Boolean functions. Section 6 shows protocols to calculate half and full adder, and n-input symmetric Boolean functions. Section 7 concludes the paper.

2 Preliminaries

This section gives the notations and basic definitions of card-based protocols. This paper is based on the standard two-type card model. In the two-type 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 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, $commit(\overline{x})$ can be obtained. Thus, NOT can be calculated without private operations.

A linearly ordered cards 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 multiple players. Throughout this paper, all players are semi-honest, that is, they obey the rule of the protocols, but try to obtain information x of $commit(x)$. There is no collusion among players executing one protocol together. No player wants any other player to obtain information of committed values.

The space complexity of card-based protocols is evaluated by the number of cards. The time complexity of card-based protocols using private operations is evaluated by the number of rounds [41]. The first round is (possibly parallel) local executions by each player using the cards initially given to each player, from the initial state to 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 $(i-1)$-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 all possible choices of the random values.

3 Private operations

We show three private operations introduced in [42], 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 uses the operation only 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 $\underbrace{\boxed{?}\boxed{?}}_{\overline{x}}$.

Note that a private random bisection cut is the same as the random bisection cut [31], 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* $b \in \{0, 1\}$*. 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 does not disclose b*.*

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

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

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

Next, we define a private reveal.

Primitive 3 *(Private reveal) A player privately opens a given committed bit. The player does 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 $x \oplus b$. 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.

4 Protocols for XOR, AND, Copy, and other Boolean functions

This section shows the sub-protocols presented in [41, 42] used in this paper's protocols. The correctness proof is shown in [41, 42].

4.1 XOR protocol

Protocol 1 *(XOR protocol) [41]*
Input: $commit(x)$ and $commit(y)$.
Output: $commit(x \oplus y)$.

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

The protocol is two rounds. Note that the protocol uses no cards other than the input cards.

4.2 AND protocol

Protocol 2 *(AND protocol) [42]*
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.* □

In Step 2, the cards of $commit(x')$ are re-used to set $commit(0)$. Thus the protocol uses no cards other than the input cards. The protocol is three rounds.

4.3 COPY protocol

Protocol 3 *(COPY protocol) [42]*
Input: $commit(x)$.
Output: m copies of $commit(x)$.

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

The protocol is three rounds. Note that the protocol does not need additional cards other than $2m$ output cards.

4.4 Any two-input Boolean functions

Though the previous subsection showed AND and XOR, any two-input Boolean functions can also be calculated by a similar protocol by three rounds and four cards [42]. Any two-input Boolean 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, $\overline{y}$, 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 $f(x, y) = \overline{x}$), 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 $\overline{y}$). This case is when $f(x, y) = y$ (or $f(x, y) = \overline{y}$), thus we do not need to calculate f by a two player protocol.

Next case is when $(f(1, y), f(0, y))$ is $(y, \overline{y})$ or $(\overline{y}, y)$. $(f(1, y), f(0, y)) = (\overline{y}, y)$ is $x \oplus y$ (XOR). $(f(1, y), f(0, y)) = (y, \overline{y})$ is $\overline{x \oplus y}$, thus this function can be calculated as follows: use 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 $\overline{y}$ and the other is 0 or 1. We can use the AND protocol and Bob sets

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

using one $commit(y)$ in the second step of the protocol.

Thus, any two-input Boolean function can be calculated without additional cards.

4.5 Parallel computations

The above two-input Boolean function calculations can be executed in parallel [42]. Consider the case when $commit(x)$ and $commit(y_i)(i = 1, 2, \ldots, n)$ are given and $commit(f_i(x, y_i))(i = 1, 2, \ldots, n)$ need to be calculated. They can be executed in parallel. Alice executes a private random bisection cut on $commit(x)$ and hands $commit(x')$ and $commit(y_i)(i = 1, 2, \ldots, n)$ to Bob. Bob sets $S_2^i(i = 1, 2, \ldots, n)$ using x' and $commit(y_i)$ according to 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.

Note that if f_i is calculated by an AND-type protocol, two new cards are necessary and the two cards of $commit(x')$ can be used. Thus, when at most one f_i is executed by an AND-type protocol and all the others are executed by XOR-type protocols, they can be executed in parallel without additional cards.

4.6 Preserving an input

In the above protocols to calculate Boolean functions, the input commitment values are lost. If the input is not lost, the input commitment can be used as an input to another calculation. Thus input preserving calculation is discussed [37, 42].

In the XOR protocol, $commit(x')$ is no more necessary after Bob sets S_2. Thus, Bob can send back $commit(x')$ to Alice. 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.

An input preserving calculation without increasing the number of cards can be executed for AND type protocols just like [37]. When we execute the AND type protocol, two cards are selected by Alice at the final step. The remaining two cards are used to recover an input value. The unused two cards' value is

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

thus the output is $commit((\overline{x} \wedge f(1,y)) \oplus (x \wedge f(0,y)))$.

Note that the function f satisfies that one of $(f(0,y),\ f(1,y))$ is y or $\overline{y}$ and the other is 0 or 1. Otherwise, we do not need to calculate f by the AND type two player protocol.

The output $f(x,y)$ can be represented as $(x \wedge f(1,y)) \oplus (\overline{x} \wedge f(0,y))$. Execute the above input preserving XOR protocol for these two output values so that the input $f(x,y)$ is preserved. The output of XOR protocol is $(\overline{x} \wedge f(1,y)) \oplus (x \wedge f(0,y)) \oplus (x \wedge f(1,y)) \oplus (\overline{x} \wedge f(0,y)) = f(1,y) \oplus f(0,y)$. Since one of $(f(0,y), f(1,y))$ is y or $\overline{y}$ and the other is 0 or 1, the output is y or $\overline{y}$ (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. In [37], two additional cards are necessary.

4.7 n-input Boolean functions

Since any 2-input Boolean function, NOT, and COPY can be executed, any n-input Boolean function can be calculated by the combination of the above protocols.

Using the technique in [37] and above input preserving Boolean function calculations, any n-input Boolean function can be calculated with $2n+4$ cards as follows [42].

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

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 $\overline{x_j}$. 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 Boolean function) [42]*
Input: commit$(x_i)(i = 1, 2, \ldots, n)$.
Output: commit$(f(x_1, x_2, \ldots, x_n))$.
The additional four cards (two pairs of cards) p_1 and p_2 are used as follows.
p_1: the intermediate value to calculate f is stored.
p_2: the intermediate value to calculate T_i is stored.
Execute the following steps for $i = 1, \ldots, k$.

1. *Copy v_1^i from the input x_1 to p_2.*
2. *For $j = 2, \ldots, n$, execute the following procedure: Apply the input-preserving AND protocol to p_2 and input x_j (If AND is taken between $\overline{x_j}$, 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. □

5 Protocols for three-input Boolean functions

This section shows protocols for three-input Boolean functions. [38] has shown that any three-input Boolean functions can be calculated with at most eight cards. We show that these functions can be calculated with six cards using private operations, that is, no additional cards are necessary other than the input cards.

The arguments to show the protocols with six cards are just the same as the one in [38]. The main difference is that AND-type functions can be calculated by four cards using the private operations.

There are $2^{2^3} = 256$ different functions with three inputs. However, some of these functions are equivalent by replacing variables and taking negations. NPN-classification [50] was considered to reduce the number of different functions considering the equivalence class of functions. The rules of NPN-classification are as follows.

1. Negation of input variables (Example: $x_i \leftrightarrow \overline{x_i}$).
2. Permutations of input variables (Example: $x_i \leftrightarrow x_j$).
3. Negation of the output ($f \leftrightarrow \overline{f}$).

For example, consider $f_1(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee x_3$. Several functions in the same equivalence class that includes f_1 are: $f_2 = (\overline{x_1} \wedge \overline{x_2}) \vee x_3$, $f_3 = (\overline{x_1} \wedge \overline{x_3}) \vee x_2$, $f_4 = \overline{f_3}$, and so on.

Input negation and output negation can be executed by card-based protocols without increasing the number of cards. They are executed by just swapping input cards or output cards. Permutations of input variables can also be executed without increasing the number of cards. They can be achieved by just changing the positions of the input values. Therefore, all functions in the same NPN equivalence class can be calculated with the same number of cards.

Theorem 1. *Any three input Boolean functions can be securely calculated without additional cards other than the input cards when we use private operations.*

Proof. When the number of inputs is 3, there are the following 14 NPN-representative functions [50]. (Note that x, y, and z are used to represent input variables.)

1. $NPN_1 = 1$
2. $NPN_2 = x$
3. $NPN_3 = x \vee y$
4. $NPN_4 = x \oplus y$
5. $NPN_5 = x \wedge y \wedge z$
6. $NPN_6 = (x \wedge y \wedge z) \vee (\overline{x} \wedge \overline{y} \wedge \overline{z})$
7. $NPN_7 = (x \wedge y) \vee (x \wedge z)$
8. $NPN_8 = (x \wedge y) \vee (\overline{x} \wedge \overline{y} \wedge z)$
9. $NPN_9 = (x \wedge y \wedge \overline{z}) \vee (x \wedge \overline{y} \wedge z) \vee (\overline{x} \wedge y \wedge z)$
10. $NPN_{10} = (x \wedge \overline{y} \wedge \overline{z}) \vee (\overline{x} \wedge y \wedge \overline{z}) \vee (\overline{x} \wedge \overline{y} \wedge z) \vee (x \wedge y \wedge z) = x \oplus y \oplus z$.
11. $NPN_{11} = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$

12. $NPN_{12} = (x \wedge \overline{z}) \vee (y \wedge z)$
13. $NPN_{13} = (x \wedge y \wedge z) \vee (x \wedge \overline{y} \wedge \overline{z})$
14. $NPN_{14} = (x \wedge y) \vee (x \wedge z) \vee (\overline{x} \wedge \overline{y} \wedge \overline{z})$

Among these 14 functions, NPN_1 - NPN_4 depend on less than three inputs. Since any two-variable function can be calculated without additional cards, these functions can be calculated with at most six cards.

We show a calculation protocol for each of the remaining functions.

For NPN_5, $x \wedge y$ can be calculated without additional cards. Then $x \wedge y \wedge z$ can be calculated without additional cards other than the input cards, $x \wedge y$ and z.

NPN_7 can be represented as $NPN_7 = x \wedge (y \vee z)$, thus this function can also be calculated without additional cards.

NPN_{10} can be calculated as $(x \oplus y) \oplus z$ without additional cards.

NPN_{13} can be represented as $NPN_{13} = x \wedge (\overline{y \oplus z})$, thus this function can also be calculated without additional cards.

NPN_{14} can be represented as $NPN_{14} = \overline{x} \oplus (y \vee z)$, thus this function can also be calculated without additional cards.

NPN_6 can be represented as $NPN_6 = (\overline{x \oplus y}) \wedge (\overline{x \oplus z})$. First, calculate $x \oplus y$ and $x \oplus z$ in parallel, where a private random bisection cut is executed to x. Then NOT is applied to each result. Next, calculate AND to these results.

NPN_8 can be represented as $NPN_8 = (\overline{x \oplus y}) \wedge (y \vee z)$. First, calculate $x \oplus y$ and $y \vee z$ in parallel, where a private random bisection cut is executed to y. Then NOT is applied to the first result. Next, calculate AND to these results.

NPN_9 can be represented as $NPN_9 = (\overline{x \oplus y \oplus z}) \wedge (x \vee z)$. First, calculate $x \oplus y$ with preserving input x. Next, calculate $(x \oplus y) \oplus z$ and $x \vee z$ in parallel, where a private random bisection cut is executed to z. Then NOT is applied to the first result. Next, calculate AND to these results.

NPN_{12} can be calculated as follows. First, calculate $x \wedge \overline{z}$ with preserving input z. Next, calculate $y \wedge z$. Then, calculate OR to these results.

NPN_{11} can be represented as

$$NPN_{11} = \begin{cases} z & \text{if } x \oplus y = 1 \\ x & \text{if } x \oplus y = 0 \end{cases}$$

This function can be calculated as follows. First, calculate $x \oplus y$ with preserving input x. Thus, x, z, and $x \oplus y$ are obtained. Then, modify the AND-type protocol as follows.

1. Alice executes private random bisection cut on $x \oplus y$. The obtained value is $x \oplus y \oplus b$, where b is the random value.
2. Bob executes private reveal on $x \oplus y \oplus b$. Bob sets
$$S_2 = \begin{cases} commit(z)||commit(x) \text{ if } x \oplus y \oplus b = 1 \\ commit(x)||commit(z) \text{ if } x \oplus y \oplus b = 0 \end{cases}$$
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 output is $commit(z)$ if ($x \oplus y \oplus b = 1$ and $b = 0$) or ($x \oplus y \oplus b = 0$ and $b = 1$). The case equals to $x \oplus y = 1$. The output is $commit(x)$ if ($x \oplus y \oplus b = 1$ and $b = 1$) or ($x \oplus y \oplus b = 0$ and $b = 0$). The case equals to $x \oplus y = 0$. Thus the result is correct. Therefore, NPN_{11} can also be calculated without additional cards. □

6 Half adder and full adder, and symmetric functions

This section first shows a realization of half adder and full adder.

The input and output of the secure half adder are as follows:

- Input: $commit(x)$ and $commit(y)$
- Output: $S = commit(x \oplus y)$ and $C = commit(x \wedge y)$

The half adder is realized by the following steps, whose idea is just the same as the one in [37].

1. Execute XOR protocol with preserving input x. Thus x and $x \oplus y$ are obtained.
2. Obtain $\overline{x \oplus y}$ by swapping the two cards of $x \oplus y$.
3. Execute AND protocol to x and $\overline{x \oplus y}$ with preserving input $\overline{x \oplus y}$. Thus $\overline{x \oplus y}$ and $x \wedge \overline{(x \oplus y)} = x \wedge y$ are obtained.
4. Obtain $x \oplus y$ by swapping the two cards of $\overline{x \oplus y}$.

No additional cards are necessary other than the four input cards. The protocol in [37] needs two additional cards, thus the number of cards is reduced by our protocol.

The input and output of the secure full adder are as follows:

- Input: $commit(x), commit(y)$, and $commit(C_I)$
- Output: $S = commit(x \oplus y \oplus C_I)$, $C_O = commit((x \wedge y) \vee (x \wedge C_I) \vee (y \wedge C_I))$

Since the half adder can be calculated without additional cards, the full adder can also be calculated without additional cards by the following protocol.

1. Add x and y using the half adder. The outputs are $x \oplus y$ and $x \wedge y$.
2. Add C_I to the result $x \oplus y$ using the half adder. The outputs are $x \oplus y \oplus C_I$ and $C_I \wedge (x \oplus y)$.
3. Execute OR protocol to $C_I \wedge (x \oplus y)$ and $x \wedge y$. Since $(C_I \wedge (x \oplus y)) \vee (x \wedge y) = (x \wedge y) \vee (x \wedge C_I) \vee (y \wedge C_I)$, the carry C_O is obtained.

Using the half adder and full adder, calculation of symmetric function can be done by the technique in [37]. n-input symmetric function $f(x_1, x_2, \ldots, x_n)$ depends only on the number of variables such that $x_i = 1$. Let $Y = \sum_{i=1}^{n} x_i$. Then the function f can be written as $f(x_1, x_2, \ldots, x_n) = g(Y)$. When Y is given by a binary representation, $Y = y_k y_{k-1} \ldots y_1$, g can be written as $g(y_1, y_2, \ldots, y_k)$, where $k = \lfloor \log n \rfloor + 1$.

Given input $x_1, x_2, \ldots, x_n$, first obtain the sum of these inputs using the half adder and full adder protocols without additional cards. The sum is obtained as $y_1, y_2, \ldots, y_k$. Then, calculate g using y_is. When $n \leq 7$, $k \leq 3$, thus any three input Boolean function g can be calculated without additional cards. When $n \geq 8$, Y is represented with $k = \lfloor \log n \rfloor + 1$ bits. Since $n - k \geq 4$, at least 8 input cards are unused after y_is are calculated. Any Boolean function can be calculated with four additional cards, thus g can be calculated without additional cards other than the input cards.

Theorem 2. *Any symmetric Boolean function can be securely calculated without additional cards other than the input cards when we use private operations.*

7 Conclusion

This paper showed card-based cryptographic protocols to calculate three input Boolean functions, half adder, full adder, and symmetric functions using private operations. One of the important open problems is obtaining another class of Boolean functions that can be calculated without additional cards using private operations.

Acknowledgements The authors would like to thank anonymous referees for their careful reading of our manuscript and their many insightful comments and suggestions.

References

1. Abe, Y., Hayashi, Y.i., Mizuki, T., Sone, H.: Five-card and computations in committed format using only uniform cyclic shuffles. New Generation Computing **39**(1), 97–114 (2021)
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. Crépeau, C., Kilian, J.: Discreet solitary games. In: Proc. of 13th Crypto, LNCS Vol. 773. pp. 319–330 (1993)
6. Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for norinori. In: Proc. of 25th International Computing and Combinatorics Conference(COCOON 2019), LNCS Vol. 11653. pp. 166–177. Springer (2019)
7. Dvořák, P., Koucký, M.: Barrington plays cards: The complexity of card-based protocols. arXiv preprint arXiv:2010.08445 (2020)

8. 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)
9. 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)
10. Hashimoto, Y., Shinagawa, K., Nuida, K., Inamura, M., Hanaoka, G.: Secure grouping protocol using a deck of cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101**(9), 1512–1524 (2018)
11. 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)
12. 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)
13. 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)
14. Koch, A.: The landscape of optimal card-based protocols. IACR Cryptology ePrint Archive, Report 2018/951 (2018)
15. Koch, A., Schrempp, M., Kirsten, M.: Card-based cryptography meets formal verification. New Generation Computing **39**(1), 115–158 (2021)
16. Koch, A., Walzer, S.: Private function evaluation with cards. Cryptology ePrint Archive, Report 2018/1113 (2018), `https://eprint.iacr.org/2018/1113`
17. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: Proc. of 10th International Conference on Fun with Algorithms (FUN 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
18. 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)
19. 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)
20. 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)
21. Manabe, Y., Ono, H.: Secure card-based cryptographic protocols using private operations against malicious players. In: Proc. of 13th International Conference on Information Technology and Communications Security(SecITC 2020), LNCS Vol. 12596. pp. 55–70. Springer (2020)
22. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. IACR Cryptology ePrint Archive, Report 2015/1031 (2015)
23. 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)

24. 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: Proc. of 10th International Conference on Fun with Algorithms (FUN 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
25. 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)
26. Mizuki, T.: Applications of card-based cryptography to education. In: IEICE Technical Report ISEC2016-53. pp. 13–17 (2016), (In Japanese)
27. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theoretical Computer Science **622**, 34–44 (2016)
28. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Proc. of 12th International Conference on Unconventional Computing and Natural Computation (UCNC 2013), LNCS Vol. 7956. pp. 162–173 (2013)
29. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Proc. of Asiacrypt 2012, LNCS Vol.7658. pp. 598–606 (2012)
30. 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)
31. 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)
32. Mizuki, T., Uchiike, F., Sone, H.: Securely computing xor with 10 cards. Australasian Journal of Combinatorics **36**, 279–293 (2006)
33. Murata, S., Miyahara, D., Mizuki, T., Sone, H.: Efficient generation of a card-based uniformly distributed random derangement. In: Proc. of 15th International Workshop on Algorithms and Computation (WALCOM 2021), LNCS Vol. 12635. pp. 78–89. Springer International Publishing, Cham (2021)
34. Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: How to solve millionaires' problem with two kinds of cards. New Generation Computing **39**(1), 73–96 (2021)
35. 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)
36. Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theoretical Computer Science **191**(1), 173–183 (1998)
37. 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)
38. 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)
39. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**(2), 361–371 (2018)
40. 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)

41. 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)
42. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Generation Computing **39**(1), 19–40 (2021)
43. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Physical zero-knowledge proof for suguru puzzle. In: Proc. of 22th International Symposium on Stabilizing, Safety, and Security of Distributed Systems(SSS 2020), LNCS Vol. 12514. pp. 235–247. Springer (2020)
44. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Interactive physical zkp for connectivity:applications to nurikabe and hitori. In: Proc. of 17th International Conference on Computability in Europe(CiE 2021), LNCS (2021)
45. 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)
46. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. In: Proc. of 16th International Conference on Theory and Applications of Models of Computation(TAMC 2020), LNCS Vol. 12337. pp. 25–36. Springer (2020)
47. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for numberlink puzzle and k vertex-disjoint paths problem. New Generation Computing **39**(1), 3–17 (2021)
48. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for ripple effect. In: Proc. of 15th International Workshop on Algorithms and Computation (WALCOM 2021), LNCS Vol. 12635. pp. 296–307. Springer International Publishing, Cham (2021)
49. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for sudoku. Theoretical Computer Science **839**, 135–142 (2020)
50. Sasao, T., Butler, J.T.: Progress in Applications of Boolean Functions. Morgan and Claypool Publishers (2010)
51. 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)
52. 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)
53. Shinoda, Y., Miyahara, D., Shinagawa, K., Mizuki, T., Sone, H.: Card-based covert lottery. In: Proc. of 13th International Conference on Information Technology and Communications Security(SecITC 2020), LNCS Vol. 12596. pp. 257–270. Springer (2020)
54. 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)
55. Stiglic, A.: Computations with a deck of cards. Theoretical Computer Science **259**(1), 671–678 (2001)
56. 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)
57. Takashima, K., Miyahara, D., Mizuki, T., Sone, H.: Actively revealing card attack on card-based protocols. Natural Computing pp. 1–14 (2021)
58. Toyoda, K., Miyahara, D., Mizuki, T., Sone, H.: Six-card finite-runtime xor protocol with only random cut. In: Proc. of the 7th ACM Workshop on ASIA Public-Key Cryptography. pp. 2–8 (2020)

59. Watanabe, Y., Kuroki, Y., Suzuki, S., Koga, Y., Iwamoto, M., Ohta, K.: Card-based majority voting protocols with three inputs using three cards. In: Proc. of 2018 International Symposium on Information Theory and Its Applications (ISITA). pp. 218–222. IEEE (2018)
60. Yasunaga, K.: Practical card-based protocol for three-input majority. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E103.A**(11), 1296–1298 (2020). https://doi.org/10.1587/transfun.2020EAL2025