Manabe, Int J Comput Softw Eng 2022, 7: 178
https://doi.org/10.15344/2456-4451/2022/178

International Journal of
Computer & Software Engineering

Original Article Open Access

Card-based Cryptographic Protocols to Calculate Primitives of Boolean Functions: Survey

Yoshifumi Manabe*
Faculty of Informatics, Kogakuin University 1-24-2, Nishi-Shinjuku, Shinjuku, Tokyo, 163-8677 Japan

Abstract

This paper surveys fundamental results on card-based cryptographic protocols. They are multi-party secure computation that is widely considered in cryptography. Physical cards are used in card-based cryptographic protocols instead of computers. They can be used when computers cannot be used or users cannot trust the software on the computer. This paper discusses protocols to calculate basic primitives to calculate any boolean functions. Logical AND, logical XOR, and copy protocols are considered, since we can execute any boolean computations with a combination of these protocols. Recently, private operations are introduced, in which operations are executed where the other players cannot see. Minimizing the number of cards is achieved by simple protocols using private operations.

Publication History:

Received: May 10, 2022
Accepted: June 09, 2022
Published: June 11, 2022

Keywords:

Multi-party secure computation, Card-based cryptographic, protocols, Private operations, Boolean functions, Copy

Introduction

Let us consider the case when Alice and Bob decide whether they have a date or not. If Alice and Bob love each other, it is all right for them to know the fact. If Alice loves Bob but Bob does not love Alice, Alice might not want to reveal that Alice loves Bob. In this case, both players just know that they do not have a date without knowing each player's decision. This is a secure computation problem that is widely discussed in cryptography, but they are not experts in cryptography. How to solve the problem?

Card-based cryptographic protocols [3, 22] were proposed in which physical cards are used instead of computers to securely calculate values. They can be used when computers cannot be used or users cannot trust the software on the computer. For example, when a user inputs private data on a computer, it is very hard to prove that the private data are discarded after the computation is finished. If a user cannot trust the software, she might hesitate to use the software on the computer. In such cases, card-based cryptographic protocols can be used. After the computation is finished, the cards are shuffled and the private input data are discarded. Also, the protocols are easy to understand, thus the protocols can be used to teach the basics of cryptography [4, 18]. den Boer [3] first showed a 5 card protocol to securely calculate the logical AND of two Yoshifumi Manabe inputs. Since then, many protocols have been proposed. They are clssified into four groups. The first type is fundamental primitives to calculate any boolean functions. The second type is to calculate complicated boolean functions, for example, half adder and full adder [12, 28]. The third type is to calculate a complicated function to solve some specific problems, for example, voting and millionaires' problem [15, 20, 24, 31]. The last type is card-based zero-knowledge proofs of puzzle solutions, such as Sudoku [34, 35, 38].

This paper surveys the first type of protocol. We show logical AND, logical XOR, and copy protocols since any boolean functions can be realized by a combination of these protocols. We show some important protocols among the proposed protocols, which includes new protocols using private operations.

In Section 2, basic notations and operations are shown. Section 3 shows the five-card trick shown by den Boer [3]. Section 4, 5, and 6show logical AND, logical XOR, and copy protocols. Section 7 shows protocols using standard playing cards. Section 8 concludes the paper.

Preliminaries

This section gives the notations and basic definitions of card-based protocols. Most of the results until Section 6 are based on a two-color card model. In Section 7, another type of card is used. In the two-color card model, there are two kinds of marks, $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. Cards of the same marks cannot be distinguished. In addition, the back of both types of cards is $\boxed{?}$. It is impossible to determine the mark on the back of a given card of $\boxed{?}$.

One bit data is represented by two cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit}$ = 0, and $\boxed{\heartsuit}\boxed{\clubsuit}$ = 1. One pair of cards that represents one bit $x \in \{0, 1\}$, whose face is down, is called a commitment of x, and denoted as commit (x). It is written as $\underbrace{\boxed{?}\boxed{?}}_{x}$.

Note that when these two cards are swapped, commit($\bar{x}$) can be obtained. Thus, logical negation can be easily calculated.

A set of cards placed in a row is called a sequence of cards. A sequence of cards S whose length is n is denoted as $S = s_1, s_2, \ldots, s_n$, where s_i is i-th card of the sequence. $S = \underbrace{\boxed{?}}_{s_1}\underbrace{\boxed{?}}_{s_2}\underbrace{\boxed{?}}_{s_3}\ldots,\underbrace{\boxed{?}}_{s_n}$.
A sequence whose length is even is called an even sequence. $S_1||S_2$ is a concatenation of sequence S_1 and S_2.

All protocols are executed by two players, Alice and Bob. The players are semi-honest, that is, they obey the rule of the protocol, but they try to obtain secret values.

Next, we discuss the inputs and outputs of the protocols. Most protocols have committed inputs, that is, the inputs are given to the players in a committed manner. The players do not know the input values and they might try to obtain the input values during the protocol

***Corresponding Author:** Prof. Yoshifumi Manabe, Faculty of Informatics, Kogakuin University, Shinjuku, Tokyo 163–8677, Japan, E-mail: manabe@cc.kogakuin.ac.jp

Citation: Manabe Y (2022) Survey: Card-based Cryptographic Protocols to Calculate Primitives of Boolean Functions. Int J Comput Softw Eng 7: 178. doi: https://doi.org/10.15344/2456-4451/2022/178

execution. The other types of protocols consider the case when each player inputs his/her input value that must be hidden from the other player. They are called non-commit input protocols. Note that committed-input protocols can be used when the players input their own values. Each player makes a commitment of his/her input in advance and the values are used as inputs. Thus, committed-input protocols are desirable.

Most protocols output the result in a committed manner. They are called committed-output protocols. On the other hand, several protocols terminate the protocol by opening the cards and obtaining the result from the sequence of the opened cards. Such protocols are called non-commit output protocols. Committed-output protocols are desirable since the committed output can be used as an input for further calculations.

Next, we show operations on the cards. Opening a card is turning a face-down card into a face-up, thus the players can see the mark on the card. Face-down a card is turning a face-up card to face-down. Rearrangement is a permutation of a sequence of cards, that is, the position of a given sequence of the cards is changed.

A shuffle is executed on a sequence of card S. Its parameter is (Π, $\mathcal{F}$), where Π is a set of permutations on S and $\mathcal{F}$ is a probability distribution on Π. For a given sequence S, each permutation $\pi \in \Pi$ is selected by the probability distribution $\mathcal{F}$ and π is applied to S. If π is applied on $S = s_1, s_2, \ldots, s_n$, the result is $s_{\pi^{-1}}(1), s_{\pi^{-1}}(2), \ldots, s_{\pi^{-1}}(n)$. Since π is selected from Π, the result is not deterministic. Non-deterministic execution is necessary for card-based protocols. If all operations are deterministic, the relation between the committed input value and the (committed) output value is known to the players. When the (committed) output cards are opened to see the final result, the secure input data is known to the players using the relation between the input and the output. Thus non-deterministic execution is necessary to hide the secure input values.

We show examples of shuffles used in the protocols shown below. A random cut is a cyclic permutation. When $S = s_1, s_2, s_3, s_4, s_5$, the result of a random cut is $S_1 = s_1, s_2, s_3, s_4, s_5$, $S_2 = s_2, s_3, s_4, s_5, s_1$, $S_3 = s_3, s_4, s_5, s_1, s_2$, $S_4 = s_4, s_5, s_1, s_2, s_3$, or $S_5 = s_5, s_1, s_2, s_3, s_4$. The probability of obtaining each result is $1/|S|$.

A random bisection cut is swapping the left half and the right half of a given even sequence. When $S = s_1, s_2, s_3, s_4, s_5, s_6$, the result of a random bisection cut is $s_1, s_2, s_3, s_4, s_5, s_6$ or $s_4, s_5, s_6, s_1, s_2, s_3$. The probability of obtaining each result is 1/2.

A shuffle is uniform if $\mathcal{F}$ is a uniform distribution, that is, $\pi \in \Pi$ is selected uniformly at random. A shuffle is closed if multiple executions of a shuffle are also the same shuffle. Non-uniform shuffles are not desirable since such nonuniform randomizations are difficult to execute by human hands. Using some additional cards or tools, protocols to execute non-uniform shuffles were shown [17, 29, 37, 40]. Closed shuffles are desirable since each one of Alice and Bob can execute one instance of the shuffle to obtain one shuffle result. Even if Alice and Bob are not honest and each player knows the result of each shuffle, the final result of the two shuffles is unknown to the players if there is no collusion between Alice and Bob. The random cut and the random bisection cut shown above are uniform and closed.

Last, the efficiency of the protocol is evaluated by the number of cards used by the protocol. It corresponds to the space complexity of programs. Note that any two-variable committed-input boolean function protocol needs at least 4 cards since the committed inputs need 4 cards.

The number of shuffles is used to evaluate the time complexity of the protocols since the other operations are simple [16].

Five-card Trick

This section shows the five-card trick shown by den Boer [3].

Protocol 1 (Five-card trick) [3]
Input: commit(x) and commit(y).
Output: $x \wedge y$.

1. *Input commit (x) and commit (y) are set as Figure 1a.*
2. *The center card is set face down and the two cards of commit (x) are swapped as Figure 1b.*
3. *Execute a random cut on the sequence of the cards as Figure 1c. The result is one of the five sequences shown below the cards.*
4. *Open all the cards. If there is three sequence of ♡, the output value is 1 (one of the sequences in Figure 1(d), otherwise the output value is 0 (one of the sequences in Figure 1(e).*

The correctness of the protocol is as follows. If $x \wedge y = 1$, there is a three sequence of ♡ after Step 2. By the random cut, the result becomes one of the sequences in Figure 1d. If $x \wedge y = 0$, there is no three sequence of after Step 2. Thus, the result becomes one of the sequences in Figure 1e.

The security of input values is as follows. If the output $x \wedge y = 1$, both players know that $x = 1$ and $y = 1$. The players can know the inputs from the result in this case. On the other hand, if $x \wedge y = 0$, it is impossible to know whether $(x, y) = (0, 0), (0, 1)$, or $(1, 0)$ from the output, since these card sequences are equivalent after a random cut.

Note that the five-card trick is a non-commit output protocol. Mizuki, Kumamoto, and Sone [21] showed that committed-input non-commit output AND protocol can be achieved using 4 cards, that is the minimum. Committed-output protocols are shown in the next section.

AND protocols

This section shows logical AND protocols. After the five-card trick, Crépeau and Kilian [5], Niemi and Renvall [26], and Stiglic [39] proposed committed-input committed-output protocols, but the number of cards is not so small. Mizuki and Sone [23] showed a committed-input committed-output protocol with 6 cards using a random bisection cut, which is a new kind of shuffle.

Protocol 2 *(AND protocol using random bisection cuts) [23]*
Input: commit (x) and commit (y).
Output: commit($x \wedge y$).

1. *Input commit (x) and commit(y) are set as Figure 2a.*
2. *The positions of the cards are changed as Figure 2b.*
3. *Execute a random bisection cut on the sequence of the cards. The result can be written as follows: select a random bit $b \in \{0, 1\}$, that is unknown to the players. If $b = 0$, there is no change in the order of the cards. If $b = 1$, the left half and the right half are swapped as Figure 2c.*
4. *Change the sequence of the cards as Figure 2d.*
5. *Open the left two cards. If the sequence is ♣ ♡, the center two cards are commit($x \wedge y$). Otherwise, the right two cards are commit ($x \wedge y$), as Figure 2e.*

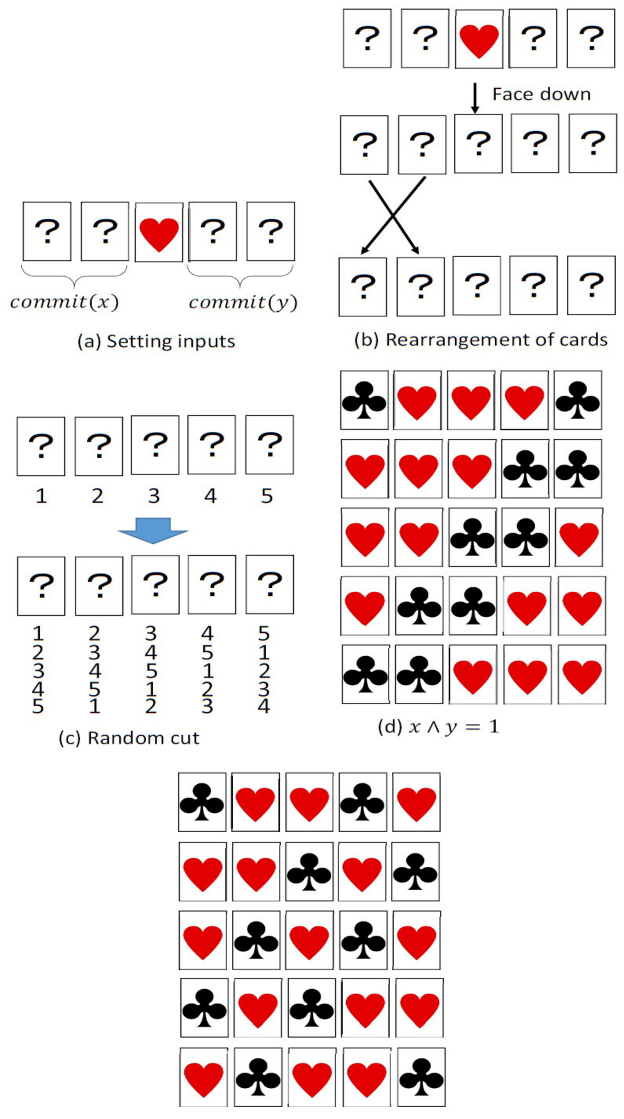

Figure 1: Five-card trick.

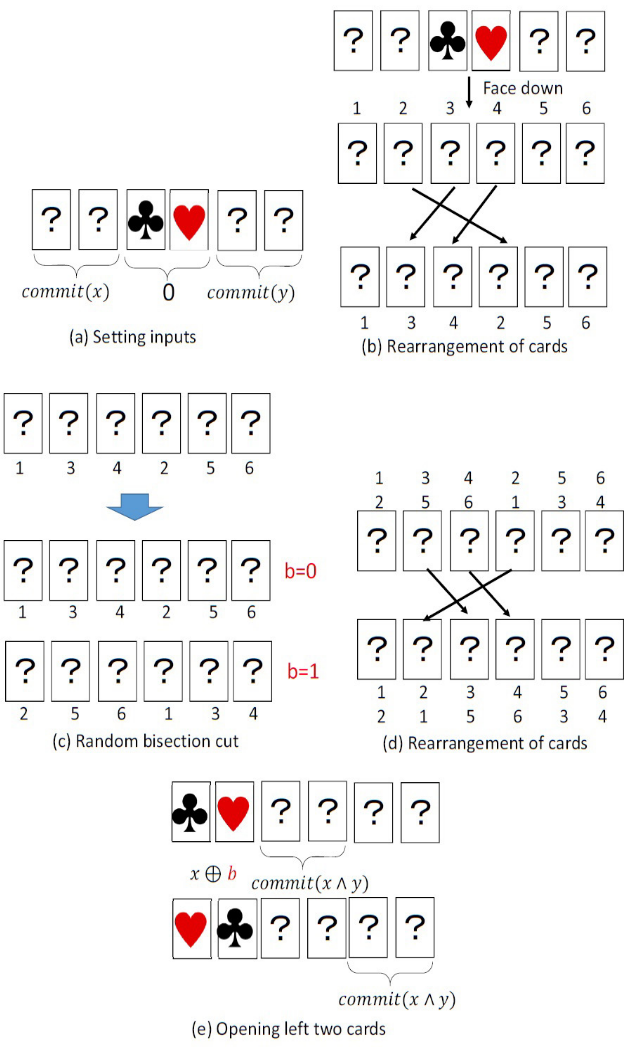

Figure 2: AND protocol using random bisection cuts.

The protocol is secure since the opened two cards are $x \oplus b$ and b is an unknown random value. Thus, x is unknown to the players. The protocol uses 6 cards. After the protocol, many works are done to reduce the number of cards. Koch, Walzer, and Härtel [9] showed a protocol with 4 cards but the protocol is 'Las Vegas' protocol, that is, the execution might not terminate forever. They showed another protocol that always terminates within a finite runtime and uses 5 cards, but the shuffles used in the protocol are not uniform. Koch [7] showed 4 card Las Vegas protocol with uniform shuffles. Ruangwises and Itoh [36] showed 5 card finite runtime protocol with uniform shuffles. Abe, Hayashi, Mizuki, and Sone [1] showed 5 card Las Vegas protocol with uniform and closed shuffles. Abe, Mizuki, and Sone [2] showed 6 card finite runtime protocol that uses only random cuts.

About the lower bound of the number of cards, Kastner, Koch,Walzer, Miyahara, Hayashi, Mizuki, and Sone [6] showed that 5 card finite runtime protocol is impossible if the protocol uses only uniform and closed shuffles. Thus the above Mizuki-Sone protocol [23] is one of the optimal finite runtime protocols among the ones using uniform and closed shuffles. Kastner et al. [6] also showed that at least 5 cards are necessary if the protocol uses only uniform and closed shuffles and does not use a 'restart from the beginning again' step in the protocol. There is a gap between the lower bound and the upper bound for this type of protocol since only 6 card protocols are known.

When the inputs are not committed, the number of cards can be decreased. Consider the case when Alice has input x that must be secret to Bob and Bob has input y that must be secret to Alice. They can use private input operations, which are executed to input secret values in a place that the other player cannot see. The operations can be executed under the table or in the back. Marcedone, Wen, and Shi [14] showed a non-commit input non-commit output protocol that calculates logical AND with 3 cards. Kurosawa and Shinozaki [11] showed a non-commit input non-commit output protocol with 2 cards. They also showed 4 card non-commit input and committed-output protocol with four cards, shown below.

Protocol 3 *(AND protocol using private input operations) [11]*
Input: Alice's input x and Bob's input y.
Output: commit(x ^ y).

1. *Alice privately makes two committed pair, S_0 = commit(x ^ 0) = commit (0) and S_1 = commit(x ^ 1) = commit(x). S_0 and S_1 are handed to Bob.*
2. *Bob privately selects S_y and outputs the pair as the result.*

Note that any two-variable function f can be calculated by the same idea that S_0 = commit($f(x, 0)$) and S_1 = commit($f(x, 1)$). The private operations are very powerful. Thus the question is whether we can obtain committed-input and committed-output protocol using private operations. Ono and Manabe [32] showed 4 card committed-input, committed-output protocol using the following three private operations.

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

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

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

Note that when $m = 1$ and $S_0 = commit(x)$, $S_0 = \underbrace{\boxed{?}\boxed{?}}_{x}$ and the player's output $S_1 = \underbrace{\boxed{?}\boxed{?}}_{x \oplus b}$. A private random bisection cut is the same as the random bisection cut [23], but the operation is executed in a hidden place.

Primitive 2 (*Private reverse cut*)
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 player.

The difference between the private random bisection cut is that b is not newly selected by the player. A private reverse cut can undo a private random bisection cut using the same bit b.

Next, a private reveal is shown.

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

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.

Protocol 4 *(AND protocol using private operations) [32]*

Input: commit(x) and commit(y).

Output: commit(x ^ 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*
$$S_2 = \begin{cases} commit(y) \| commit(0) \ldots. if\ x' = 1 \\ commit(y) \| commit(0) \ldots. if\ x' = 0 \end{cases}$$
and 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. Alice then selects the left pair. Let the obtained sequence be S_3. Alice outputs S_3.*

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.

The security of the protocol is as follows: 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 comparison of the AND protocols is shown in Table 1. Note that the private random bisection cut can be considered as a kind of shuffles. Thus, in the table, it is counted as the number of shuffles. However, it is more simple than random bisection cuts, since Alice knows b.

XOR Protocols

Though any boolean function can be realized by logical AND and NOT, logical XOR protocols are considered because the protocol is more simple than the AND protocols.

Crépeau and Kilian [5] showed 14 card committed-input committed-output XOR protocol. Mizuki and Sone [23] showed 4 card committed-input committed-output XOR protocol, whose number of cards is the minimum.

Protocol 5 *(XOR protocol using random bisection cuts) [23]*
Input: commit(x) and commit(y).
Output: commit($x \oplus y$).

1. *Input commit(x) and commit(y) are set as Figure 3a.*
2. *The positions of the cards are changed as Figure 3b.*
3. *Execute a random bisection cut on the sequence of the cards. The result can be written as follows: select a random bit $b \in \{0, 1\}$, that is unknown to the players. If b = 0, there is no change in the order of the cards. If b = 1, the left half and the right half are swapped as Figure 3c.*
4. *Change the sequence of the cards as Figure 3d.*
5. *Open the left two cards. If the sequence is $\boxed{♣}\boxed{♡}$, the right two cards are commit($x \oplus y$). Otherwise, the right two cards are commit ($\overline{x \oplus y}$), as Figure 3e.*

Nakai, Shirouchi, Tokushige, Iwamoto, and Ohta [25] showed 2 card non-commit input non-commit output protocol. Kurosawa and Shinozaki [11] achieved a non-commit input non-commit output protocol with 2 cards. They also showed 2 card non-commit input committed-output protocol.

Article	# of cards	Input commit	Output commit	Finite	Shuffle (or private shuffle)			Note
					Uniform	Closed	# of shuffles	
[3]	5	yes	no	yes	yes	yes	1	Only random cuts
[5]	10	yes	yes	no	yes	yes	8	Only random cuts, Four color cards
[26]	12	yes	yes	no	yes	yes	7.5	Only random cuts
[39]	8	yes	yes	no	yes	yes	2	Only random cuts
[23]	6	yes	yes	yes	yes	yes	1	
[9]	4	yes	yes	no	no	yes	8	
[9]	5	yes	yes	yes	no	no	14/3	
[7]	4	yes	yes	no	yes	no	8	
[36]	4	yes	yes	no	yes	no	10	
[36]	5	yes	yes	yes	yes	no	14/3	
[1]	5	yes	yes	no	yes	yes	4.5	
[2]	6	yes	yes	yes	yes	yes	2	Only random cuts
[21]	4	yes	no	yes	yes	yes	2	
[14]	3	no	no	yes	–	–	0	Use private operations
[11]	2	no	no	yes	–	–	0	Use private operations
[11]	4	no	yes	yes	–	–	0	Use private operations
[32]	4	yes	yes	yes	yes	–	1	Use private operations

Table 1: Comparison of AND protocols.

Protocol 6 (*XOR protocol using private input operations) [11]*

Input: Alice's input x and Bob's input y.

Output: commit($x \oplus y$).

1. *Alice privately makes S_0 = commit (x) and hands S_0 to Bob.*
2. *Bob privately swaps the two cards of S_0 if y = 1. Otherwise, Bob does nothing. Bob outputs the result.*

Using private operations, Ono and Manabe [33] showed 4 card committed-input committed-output protocol shown below.

Protocol 7 *(XOR protocol using private operations) [33]*

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 comparison of XOR protocols is shown in Table 2.

Copy protocols

To calculate complicated boolean functions, multiple copies of a value might be necessary to input a value to many boolean circuits. Thus, a copy protocol is necessary. Many protocols consider obtaining any n ($\geq$ 2) committed copies of a given committed input, but this paper considers obtaining two copies of inputs for the comparison. Note that non-commit input copy protocols are not considered since obtaining multiple copies of input is easy if a player knows a private input value.

Crépeau and Kilian [5] showed 8 card copy protocol. Mizuki and Sone [23] showed 6 card copy protocol, shown below.

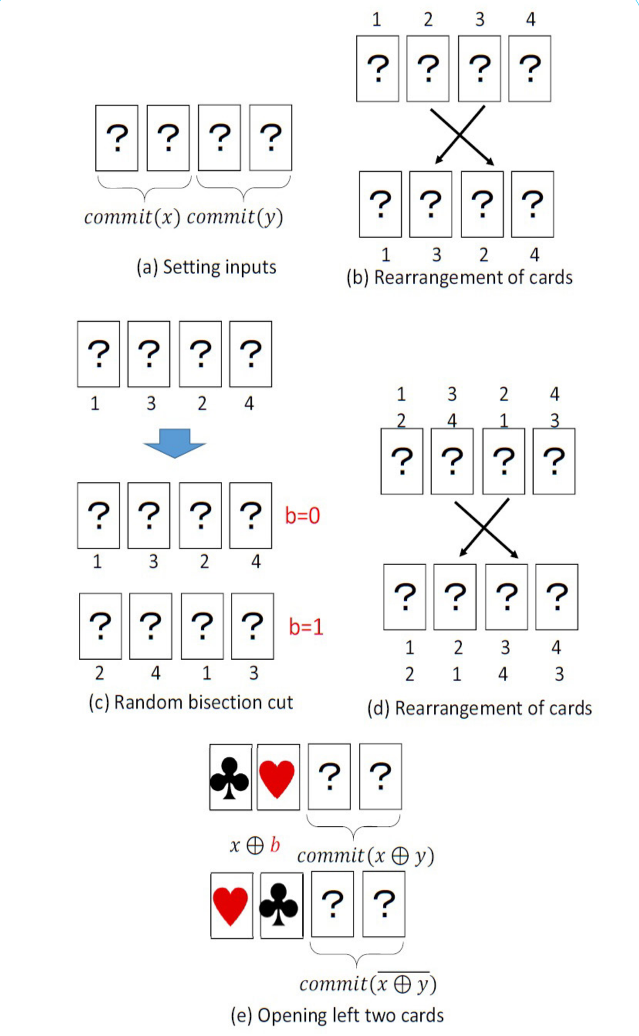

Figure 3: XOR protocol using random bisection cuts.

Article	# of cards	Input commit	Output commit	Finite	Shuffle (or private shuffle)			Note
					Uniform	Closed	# of shuffles	
[5]	14	yes	yes	no	yes	yes	6	Only random cuts, Four color cards
[23]	4	yes	yes	yes	yes	yes	1	-
[25]	2	yes	no	yes	-	-	0	Use private operations
[11]	2	no	no	yes	-	-	0	Use private operations
[11]	2	no	yes	yes	-	-	0	Use private operations
[33]	4	yes	yes	yes	yes	yes	1	Use private operations

Table 2: Comparison of XOR protocols.

Protocol 8 (copy protocol using random bisection cuts) [23]

Input: commit (x).

Output: two copies of commit(x).

1. *Input commit(x) and two copies of commit(0) are set as Figure 4a.*
2. *The positions of the cards are changed as Figure 4b.*
3. *Execute a random bisection cut on the sequence of the cards. The result can be written as follows: select a random bit b ∈ {0, 1}, that is unknown to the players. If b = 0, there is no change in the order of the cards. If b = 1, the left half and the right half are swapped as Figure 4c.*
4. *Change the sequence of the cards as Figure 4d.*
5. *Open the left two cards. If the sequence is ♣ ♡, the remaining pairs are commit(x). Otherwise, the remaining pairs are commit($\bar{x}$), as Figure 4e*

Nishimura, Nishida, Hayashi, Mizuki, and Sone [30] showed 5 card Las Vegas protocol that uses unequal shuffles. Koyama, Toyoda, Miyahara and Mizuki [10] showed 6 card Las Vegas protocol that uses only random cuts. Kastner, Koch, Walzer, Miyahara, Hayashi, Mizuki, and Sone [6] showed lower bounds of copy protocols without private operations. It is proved that any protocol needs at least 5 cards, thus the protocol in [30] is optimal. It is proved that any finite runtime protocol needs at least 6 cards, thus the protocol in [23] is optimal.

Ono and Manabe [32] showed a protocol with the minimum number of cards using private operations.

Protocol 9 (copy protocol with private operations) [32]

Input: commit (x).

Output: 2 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⊕ b. Alice sends commit(x') to Bob.*
2. *Bob executes a private reveal on commit(x') and obtains x'. Bob makes 2 copies of x'. Bob faces down these cards. Bob sends these cards, 2 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 comparison of copy protocols is shown in Table 3.

Protocols with standard deck of cards

The protocols shown above used a special kind of card. There might be cases such cards are not easy to obtain. Thus, protocols using playing cards are considered.

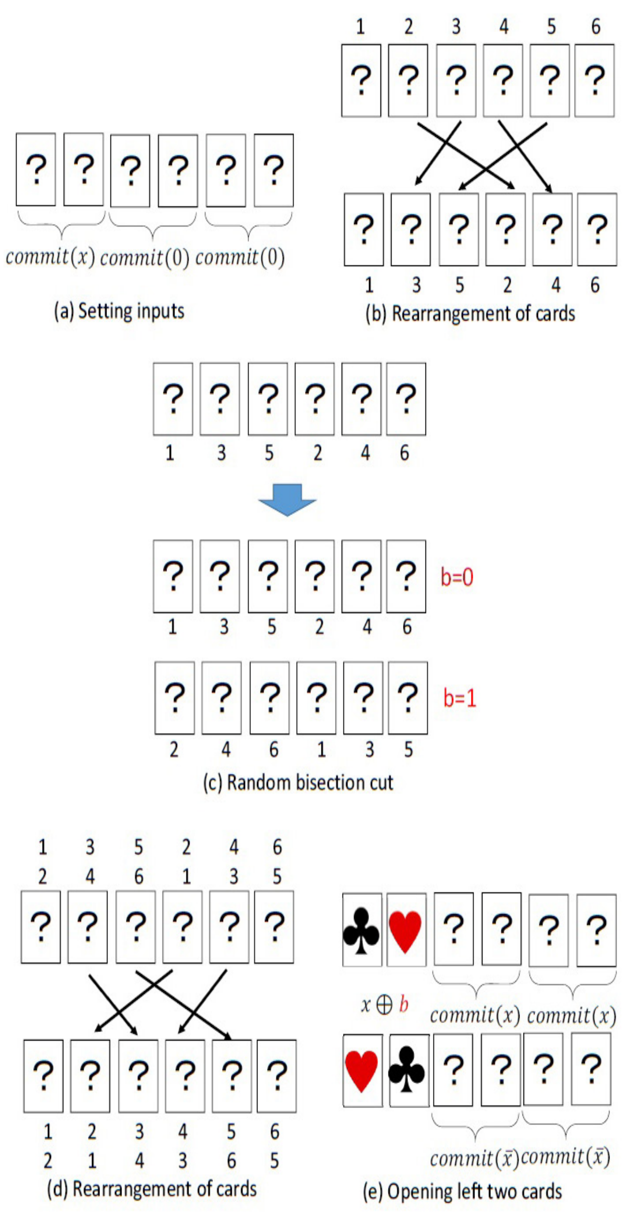
commit(x) commit(0) commit(0)
(a) Setting inputs
1 2 3 4 5 6
1 3 5 2 4 6
(b) Rearrangement of cards
b=0
b=1
2 4 6 1 3 5
(c) Random bisection cut
(d) Rearrangement of cards
x ⊕ b
commit(x) commit(x)
commit($\bar{x}$)commit($\bar{x}$)
(e) Opening left two cards

Figure 4: Copy protocol using random bisection cuts.

Article	# of cards	Finite	Shuffle (or private shuffle)			Note
			Uniform	Closed	# of shuffles	
[5]	8	yes	yes	yes	2	Only random cuts
[23]	6	yes	yes	yes	1	
[30]	5	no	no	no	2	
[10]	6	no	yes	yes	3	Only random cuts
[32]	4	yes	yes	-	1	Use private operations

Table 3: Comparison of COPY protocols.

The model of a standard deck of cards is as follows. A deck of playing cards consists of 52 distinct mark cards, which are named from 1 to 52. The number of each card (for example, 1 is the ace of spade and 52 is the king of club) is common knowledge among the players. The back of all cards is the same $\boxed{?}$. It is impossible to determine the mark on the back of a given card of $\boxed{?}$.

One bit data is represented by two cards as follows: $\boxed{i}\,\boxed{j} = 0$ and $\boxed{j}\,\boxed{i} = 1$ if i < j. The base of a commitment is the pair of cards used for the commitment. If card i and j(i < j) are used to set commit(x) (That is, set $\boxed{i}\,\boxed{j}$ if x = 0 and set $\boxed{j}\,\boxed{i}$ if x = 1), the commitment is written as commit $(x)^{\{i,j\}}$ and written as $\underbrace{\boxed{?}\,\boxed{?}}_{(x)^{\{i,j\}}}$.

We need to consider information leakage from the base of a commitment. For example, Ono and Manabe's AND protocol in the two-color card model is based on the following equation.

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

If the protocol is executed as it is with input $commit(x)^{\{1,2\}}$ and $commit(y)^{\{3,4\}}$, the output is $commit(y)^{\{3,4\}}$ when $x = 1$. Thus, if the cards are opened to see the result, the cards are 3 and 4. The players can know that $x = 1$. To make the protocol secure, we need to avoid information leakage from the base.

Niemi and Renvall [27] first showed protocols using playing cards. They showed AND, XOR, and copy protocols. Mizuki [19] showed efficient AND, XOR, and copy protocols using random bisection cuts. Koyama, Toyoda, Miyahara, and Mizuki [10] showed XOR and copy protocols that use only random cuts. Koch, Schrempp, and Kirsten [8] showed the lower bound of finite runtime AND protocol without private operations is five, thus there is a gap between the lower bound and the upper bound.

When we use private operations, Manabe and Ono [13] showed 4 card AND, XOR, and copy protocols. This result also shows the power of private operations. The comparisons of the protocols are shown in Tables 4, 5, and 6.

Article	# of cards	Finite	Shuffle (or private shuffle)			Note
			Uniform	Closed	# of shuffles	
[27]	5	no	yes	yes	7.5	Only random cuts
[19]	8	yes	yes	yes	4	
[8]	4	no	yes	yes	6	Only random cuts
[13]	4	yes	yes	no	5	Use private operations

Table 4: Comparison of committed-input committed-output AND protocols with a standard deck of cards.

Article	# of cards	Finite	Shuffle (or private shuffle)			Note
			Uniform	Closed	# of shuffles	
[27]	4	no	yes	yes	7	Only random cuts
[19]	4	yes	yes	yes	1	
[10]	4	yes	yes	yes	1	Only random cuts
[13]	4	yes	yes	-	1	Use private operations

Table 5: Comparison of committed-input committed-output XOR protocols with a standard deck of cards.

Article	# of cards	Finite	Shuffle (or private shuffle)			Note
			Uniform	Closed	# of shuffles	
[27]	6	yes	yes	yes	5.5	Only random cuts
[19]	6	no	yes	yes	1	
[10]	6	yes	yes	yes	3	Only random cuts
[13]	4	no	yes	-	1	Use private operations

Table 6: Comparison of copy protocols with a standard deck of cards

Conclusion

This paper showed a survey on the card-based cryptographic protocols. This paper showed protocols of fundamental boolean functions, logical AND, logical XOR, and copy. The results show the effectiveness of private operations. Other than the fundamental functions, there are many works for calculating complicated boolean functions such as half and full adders. In addition, there are works for solving specific applications, for example, voting and millionaires' problem, and so on. Efficient calculation of such complicated functions is one of the open problems.

Competing Interests

The authors declare that they have no competing interests.

References

1. Abe Y, Hayashi YI, Mizuki T, Sone H (2021) Five-Card AND Computations in Committed Format Using Only Uniform Cyclic Shuffles. New Gener Comput 39: 97-114.
2. Abe Y, Mizuki T, Sone H (2021) Committed-format AND protocol using only random cuts. Nat Comput 20: 639–645.
3. den Boer B (1990) More efficient match-making and satisfiability the five card trick. In: Proc. of EUROCRYPT '89, LNCS 434: 208–217.
4. Cheung E, Hawthorne C, Lee P (2013) Cs 758 project: Secure computation with playing cards.
5. Crépeau C, Kilian J (1993) Discreet Solitary Games. In: Stinson, D.R. (eds) Advances in Cryptology - CRYPTO' 93. CRYPTO 1993. Lecture Notes in Computer Science, vol 773. Springer, Berlin, Heidelberg.
6. Kastner J, Koch A, Walzer S, Miyahara D, Hayashi Y, et al. (2017) The Minimum Number of Cards in Practical Card-Based Protocols. In: Takagi, T., Peyrin, T. (eds) Advances in Cryptology – ASIACRYPT 2017. ASIACRYPT 2017. Lecture Notes in Computer Science(), vol 10626. Springer, Cham.
7. Koch A (2018) The landscape of optimal card-based protocols. IACR Cryptology ePrint Archive, Report 2018/951.
8. Koch A, Schrempp M, Kirsten M (2021) Card-based cryptography meets formal verification. New Gen Comput 39: 115-158.
9. Koch A, Walzer S, Härtel K (2015) Card-based cryptographic protocols using a minimal number of cards. In: Proc. of Asiacrypt 2015, LNCS Vol. 9452. pp. 783-807.
10. Koyama H, Toyoda K, Miyahara D, Mizuki T (2021) New card-based copy protocols using only random cuts. APKC '21: Proceedings of the 8th ACM on ASIA Public-Key Cryptography Workshop, May 2021, pp. 13-22.
11. Kurosawa K, Shinozaki T (2017) Compact card protocol. In: Proc. of 2017 Symposium on Cryptography and Information Security (SCIS 2017). pp. 1A2–1A6. (In Japanese)
12. Manabe Y, Ono H (2021) Combinatorial Algorithms: 32nd International Workshop, IWOCA 2021, Ottawa, ON, Canada, July 5-7, 2021, Proceedings, Jul 2021 pp. 469-484.
13. Manabe Y, Ono H (2021) Theoretical Aspects of Computing – ICTAC 2021: 18th International Colloquium, Virtual Event, Nur-Sultan, Kazakhstan, September 8-10, 2021, Proceedings, pp. 256-274.
14. Marcedone A, Wen Z, Shi E (2015) Secure dating with four or fewer cards. IACR Cryptology ePrint Archive, Report 2015/1031.
15. Miyahara D, Hayashi YI, Mizuki T, Sone H (2020) Practical card-based implementations of yao's millionaire protocol. Theoretical Computer Science 803: 207-221.
16. Miyahara D, Ueda I, Hayashi YI, Mizuki T, Sone H (2020) Evaluating card-based protocols in terms of execution time. Int J Inf Secur 20: 729-740.
17. Miyamoto K, Shinagawa K (2022) Graph automorphism shuffles from pile-scramble shuffles. New Gen Comput 40: 199–223.
18. Mizuki T (2016) Applications of card-based cryptography to education. In: IEICE Techinical Report ISEC2016-53. pp. 13-17. (In Japanese)
19. Mizuki T (2016) Efficient and Secure Multiparty Computations Using a Standard Deck of Playing Cards. In: Foresti, S., Persiano, G. (eds) Cryptology and Network Security. CANS 2016. Lecture Notes in Computer Science(), vol 10052. Springer, Cham.
20. Mizuki T, Asiedu IK, Sone H (2013) Voting with a Logarithmic Number of Cards. In: Mauri, G., Dennunzio, A., Manzoni, L., Porreca, A.E. (eds) Unconventional Computation and Natural Computation. UCNC 2013. Lecture Notes in Computer Science, vol 7956. Springer, Berlin, Heidelberg.
21. Mizuki T, Kumamoto M, Sone H (2012) The five-card trick can be done with four cards. In: Proc. of Asiacrypt 2012, LNCS Vol.7658. pp. 598–606.
22. Mizuki T, Shizuya H (2017) Computational model of card-based cryptographic protocols and its applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 100: 3-11.
23. Mizuki T, Sone H (2009) Six-Card Secure AND and Four-Card Secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds) Frontiers in Algorithmics. FAW 2009. Lecture Notes in Computer Science, vol 5598. Springer, Berlin, Heidelberg.
24. Nakai T, Misawa Y, Tokushige Y, Iwamoto M, Ohta K (2021) How to solve millionaires' problem with two kinds of cards. New Generation Computing 39: 73–96.
25. Nakai T, Shirouchi S, Tokushige Y, Iwamoto M, Ohta K (2022) Secure computation for threshold functions with physical cards: Power of private permutations. New Gen Comput 40: 95–113.
26. Niemi V, Renvall A (1998) Secure multiparty computations without computers. Theoretical Computer Science 191: 173-183.
27. Niemi V, Renvall A (1999) Solitaire zero-knowledge. Fundamenta Informaticae 38: 181-188.
28. Nishida T, Hayashi Y, Mizuki T, Sone H ((2015) Securely computing three-input functions with eight cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 98: 1145-1152.
29. Nishimura A, Hayashi YI, Mizuki T, Sone H (2018) Pile-shifting scramble for cardbased protocols. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 101: 1494-1502.
30. Nishimura A, Nishida T, Hayashi Y, Mizuki T, Sone H (2018) Card-based protocols using unequal division shuffles. Soft Comput 22: 361–371.
31. Ono H, Manabe Y (2018) 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
32. Ono H, Manabe Y (2021) Card-based cryptographic logical computations using private operations. New Gen Comput 39: 19-40.
33. Ono H, Manabe Y (2021) Minimum round card-based cryptographic protocols using private operations. Cryptography 5: 17.
34. Robert L, Miyahara D, Lafourcade P, Mizuki T (2022) Card-based zkp for connectivity: Applications to nurikabe, hitori, and heyawake. New Gener Comput 40: 149–171.
35. Ruangwises S (2022) Two standard decks of playing cards are sufficient for a zkp for sudoku. New Gener Comput 40: 49-65.
36. Ruangwises S, Itoh T (2019) And protocols using only uniform shuffles. In: Proc. of 14th International Computer Science Symposium in Russia(CSR 2019), LNCS Vol.11532. pp. 349–358.
37. Saito T, Miyahara D, Abe Y, Mizuki T, Shizuya H (2020) How to implement a non-uniform or non-closed shuffle. Theory and Practice of Natural Computing: 9th International Conference, TPNC 2020, Taoyuan, Taiwan, December 7–9, 2020, Proceedings, pp.107-118.
38. Sasaki T, Miyahara D, Mizuki T, Sone H (2020) Efficient card-based zero-knowledge proof for sudoku. Theoretical Computer Science 839: 135-142.
39. Stiglic A (2001) Computations with a deck of cards. Theoretical Computer Science 259: 671-678.
40. Ueda I, Miyahara D, Nishimura A, Hayashi YI, Mizuki T, et al. (2020) Secure implementations of a random bisection cut. Int J Inf Secur 19: 445-452.