A More Efficient Card-Based Protocol for Generating a Random Permutation Without Fixed Points

Takuya Ibaraki and Yoshifumi Manabe
Faculty of Informatics
Kogakuin University
1-24-2, Nishi-Shinjuku, Shinjuku, Tokyo, 163-8677 Japan
Email: em16001@ns.kogakuin.ac.jp manabe@cc.kogakuin.ac.jp

***Abstract*—Many works have been done for secure computation of functions. Most of them assume computation on computers. The protocols are difficult for the people who has no knowledge on cryptography. Therefore, secure computation using cards was considered. Since nothing other than the cards are used, the protocols are easy to understand for the people who has no knowledge on cryptography. This paper presents a card-based protocol that generates a random hidden permutation. This problem is applicable to a case when players exchange gifts. The protocol must not output a permutation with a fixed point, which means that every player does not receive the gift he/she prepared. Ishikawa et. al proposed a protocol with two-color cards using random shuffles and a proof of having no fixed points. However, the success probability of the protocol is not so high. Therefore, more efficient random permutation protocol is required. In this paper, we use cyclic shuffles in addition to the random shuffles to lower the possibility of fixed points. We show the success probability of obtaining a permutation without fixed points by our protocol is better than the one by the existing protocol.**

I. Introduction

Many works have been done for secure computation of functions. Most of them assume computation on computers.

In contrast, cryptographic protocols using physical cards are considered [1,2,3,4,5,6,7,8,9]. The cards considered to use are two-color cards [1,3,4,5,8], four-color cards [2], and the special cards [6,7]. In this paper, we use two-color cards. Since nothing other than the cards are used, the protocols are easy to understand for the people who has no knowledge on cryptography. There is a feature that all participants can confirm the correctness of the protocol.

Secure computing with cards begins with den Boer's Five-Card Trick in 1989[5]. The protocol uses five two-color cards and executes secret computation of AND of two bits. Then, basic operations such as AND, OR, XOR, and COPY have been devised [3,5]. Various application protocols were also proposed, such as secure voting protocol [7] and secure three-input majority computation protocol [8]. This paper considers protocols that generate a random permutation without a fixed point [1]. The number of cards used by the protocol in [1] is $2n\lceil \log n \rceil + 6$, where n is the number of players. In the protocol, a random permutation is generated and then the existence of a fixed point is checked. If a fixed point is found, a random permutation is generated again. Thus the complexity of the protocol depends on the success probability of each random permutation generation. However, the probability is not evaluated in the paper. We use cyclic shuffles in addition to the random shuffles to lower the possibility of fixed points. Our protocol uses $4n\lceil \log n \rceil + 6$ cards.

In Section 2, the encoding of bits by cards used in this paper is stated. Then the fundamental protocols used in our protocol are stated. In Section 3, an existing protocol for random permutation [1] is stated. Section 4 describes our protocol. In Section 5, we show the success probabilities and the amount of computation of the protocols. Finally, we mark the future challenges and summarize in Section 6.

II. Card-Based Protocols

A. Semi-Honest Model

Throughout this paper, we assume that the players are semi-honest. All players take actions in accordance with the protocol, but try to obtain the other players' secret data. In the secure computing protocols with cards, operations such as shuffling are carried out while the other players are watching. Therefore, this assumption is a natural one.

B. Encoding Rule

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

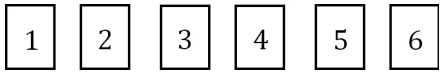
We commonly use the bit encoding such that $\boxed{\spadesuit}\boxed{\heartsuit}$ = 0 and $\boxed{\heartsuit}\boxed{\spadesuit}$ = 1. We call this encoded bit commitment bit. For a given bit encoding, NOT of the bit can be obtained by replacing the left and right of the cards. We use a notation when $a = \boxed{\spadesuit}$, $\bar{a} = \boxed{\heartsuit}$.

C. Random Bisection Cut

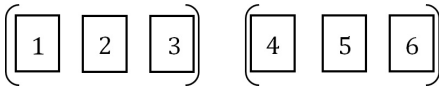
Random bisection cut is a shuffling method for obtaining an efficient protocol [3]. Random bisection cut can be carried out to any even number of cards. We show a case of six cards.

978-1-5090-0973-2/16 $31.00 © 2016 IEEE
DOI 10.1109/MCSI.2016.23

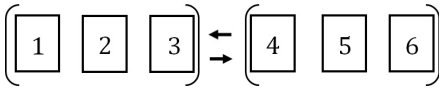
1. Arrange the six cards as follows:



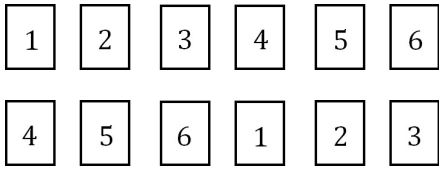
2. Divide in half in the middle:



3. Shuffle the left and right of cards:



4. Either output is obtained uniformly randomly:



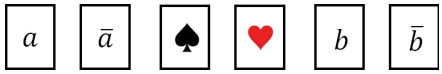
D. Pile-Scramble Shuffle

Pile-scramble shuffle is a random permutation to piles of the cards. The order of cards in each pile must be unchanged though the shuffle. This shuffle is executed by all players. If only one player executes the shuffle, the player may be possible to know the result. The pile-scramble shuffle can be easily implemented by people using rubber bands, clips, or something like that.

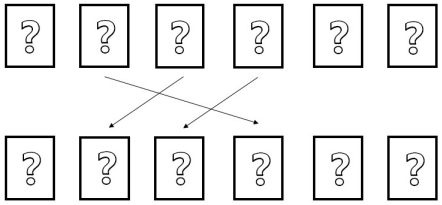
E. AND Protocol

Given a pair of commitment bits $a, b \in \{0, 1\}$, the known AND protocol [3] generates a comitment bit of a and b, as follows:

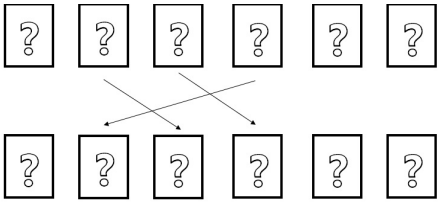
1. Arrange the commitment bits a, b and a commitment bit of 0 as follows:



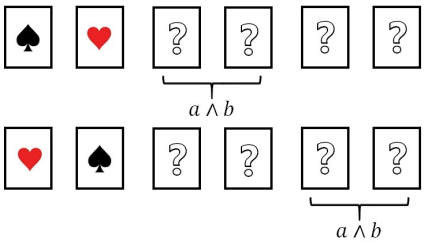
2. Rearrange the order as follows:



3. Execute the random bisection cut.
4. Rearrange the order as follows:



5. Reveal the two cards from the left. We get the result as follows according to the open cards.

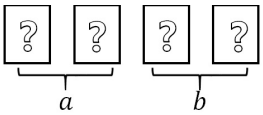


Thus, we obtain a commitment of $a \wedge b$ in both cases. It is proved that no information about a or b can be obtained by the open cards.

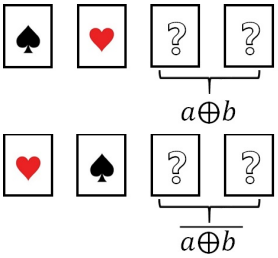
F. XOR Protocol

Given a pair of commitment bits $a, b \in \{0, 1\}$, the known XOR protocol [3] generates a commitment bit of $a \oplus b$, as follows:

1. Arrange the commitment bits a, b as follows:



2. Swap the second and the third cards.
3. Execute the random bisection cut.
4. Swap the second and the third cards.
5. Reveal the left two cards. We get the result as follows according to the open cards.

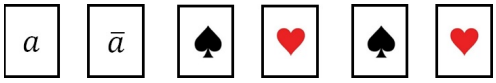


Thus, we obtain a commitment of $a \oplus b$ in both cases. It is proved that no information about a or b can be obtained by the open cards.

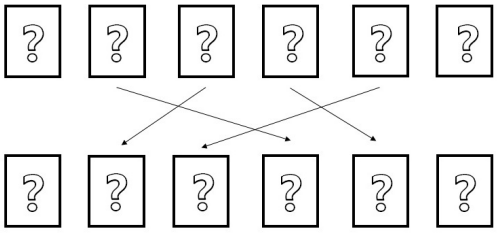
G. Copy Protocol

Given a commitment of a bit a together with four additional cards, the known copy protocol [3] generates two copied commitments of a, as follows:

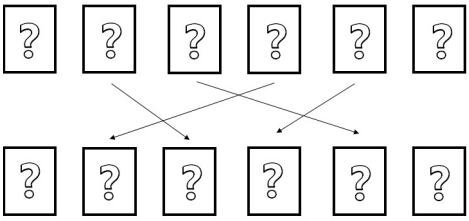
1. Arrange the commitment bit a and commitment bits of 0 as follows:



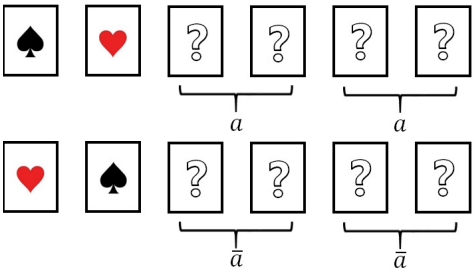
2. Rearrange the order as follows:



3. Execute the random bisection cut.
4. Rearrange the order as follows:



5. Reveal the two cards from the left. We get the result as follows according to the open cards.

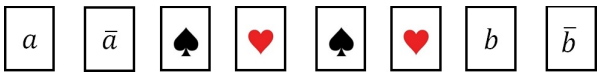


Thus, we obtain two copies of the commitment bit a. In the case of $\overline{a}$, it can be easily converted into a by replacing the right and left of the cards. It is proved that no information about a can be obtained by the open cards.

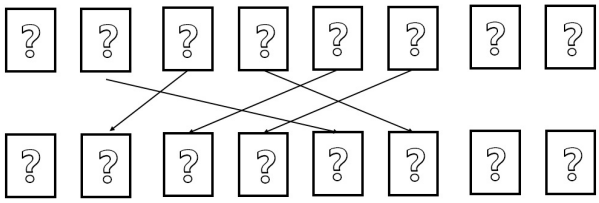
H. One-input-preserving AND Protocol

Given a pair of commitment bits $a, b \in \{0, 1\}$ together with four additional cards, the known One-input-preserving AND protocol [1] generates an AND commitment and a commitment of a, as follows:

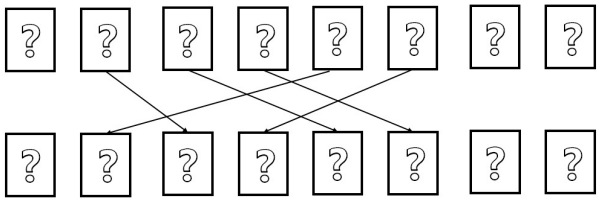
1. Arrange the commitment bits a,b and commitment bits of 0 as follows:



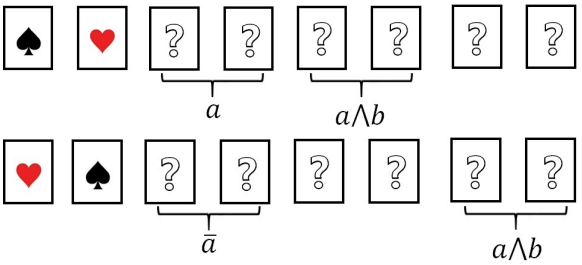
2. Rearrange the order as follows:



3. Execute the random bisection cut.
4. Rearrange the order as follows:



5. Reveal the two cards from the left. We get the result as follows according to the open cards.



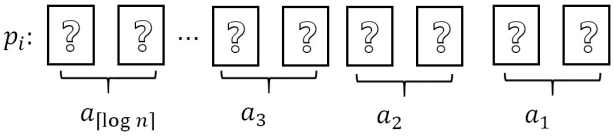
Thus, we obtain commitments of both $a \wedge b$ and a in both cases. It is proved that no information about a or b can be obtained by the open cards.

III. Known Method

There are n players $p_1, p_2, \ldots, p_n$ who would like to exchange their gifts. Gifts must not come back to himself/herself. Even if a player p_i accidentally knows the gift to p_i came from player p_j, p_i must not be able to guess the player who obtains the gift from any player other than p_j. Therefore, in order to achieve the property, a random permutation without a fixed point is necessary.

A. How to Check Fixed Points

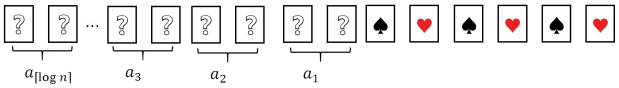
Each player has a pile of $\lceil log\ n \rceil$ commitments. After a pile-scramble shuffle, each player needs to check whether a fixed point exists. Player p_i has a card string obtained by a pile-scramble shuffle. It is a pile of $\lceil log\ n \rceil$ commitments:



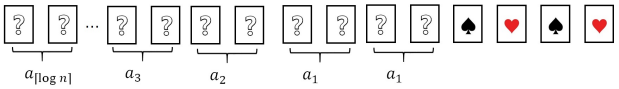
b is used to identify each player. It is used to check whether a fixed point exists. b is generated by each player as follows (i is the player number):

$$(i-1)_{10} = (b_{\lceil log\ n \rceil} ... b_2 b_1)_2$$

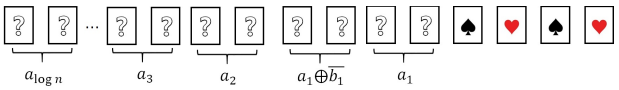
1. Arrange $\lceil log\ n \rceil$ input commitments and six additional cards as follows:



2. Copy the commitment of a_1 using the copy protocol [3]:



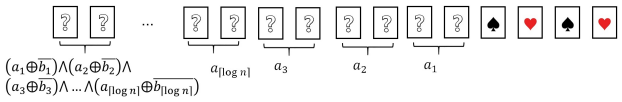
3. Apply NOT to b_1. Furthermore, apply the XOR protocol to one of the two a_1's :



4. Repeat the following steps from $i = 2$ to $i = \lceil \log n \rceil$.

4.1 Apply NOT to b_i. Furthermore, apply the XOR protocol to a_i.

4.2 Apply the one-input-preserving AND protocol so that $(a_1 \oplus \overline{b_1}) \wedge ... \wedge (a_i \oplus \overline{b_i})$ and $(a_i \oplus \overline{b_i})$ are obtained. Furthermore, obtain a_i by applying the XOR protocol to $(a_i \oplus \overline{b_i})$ and $\overline{b_i}$.

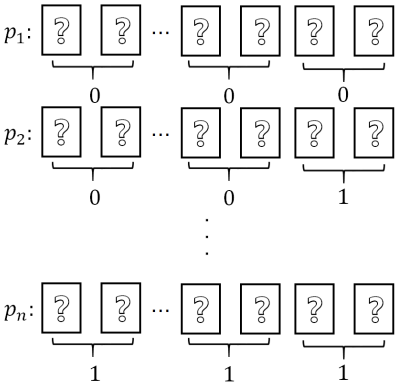


5. Reveal the commitment of $(a_1 \oplus \overline{b_1}) \wedge ... \wedge (a_{\lceil log\ n \rceil} \oplus \overline{b_{\lceil log\ n \rceil}})$. If the value is 1, then this is a fixed point. Otherwise, it is a not fixed point. Throughout the protocol, the original input $a_1, \ldots, a_{\lceil log\ n \rceil}$ are kept.

B. The existing random permutation protocol

Suppose the number of players be n. A random permutation can be obtained using $2n\lceil \log n \rceil + 6$ cards.

1. Each player has a tuple of cards that is the commitments of bits of $(i-1)_2$:



2. Each row is regarded as a pile. Apply the pile-scramble shuffle to the n piles.

3. Each player executes the protocol in 3.A to check the existence of a fixed point, where the input a is set as the cards obtained by the shuffle.

4. If a player finds a fixed point, all players go back to step 2. Otherwise, the protocol terminates.

IV. Our New Protocol

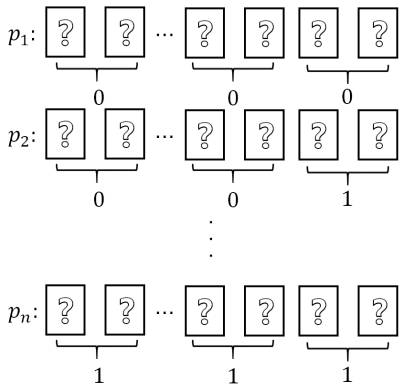
The success probability of the protocol in [1] has not been evaluated. If the probability of having a fixed point is large, the protocol needs many rounds until obtaining a permutation without a fixed point and the complexity is large. We introduce a player ID and cyclic shuffle to improve the success probability and the amount of computation of the protocol.

A. Cyclic Shuffle

Define n card piles $a = (a_0, ..., a_{n-1})$. Cyclic shuffle obtains a cyclic permutation of the piles. For example, the result of two cyclic shifts to a is $(a_2, a_3, \ldots, a_{n-1}, a_0, a_1)$. Player $p_i (1 \leq i \leq n)$ performs cyclic shift r_i times. Note that r_i is not known to any other players. In total, cyclic shifts are done $r = r_1 + ... + r_n (mod\ n)$ times and r is a uniform random number that is unknown to any player. For example, suppose $n = 6$. The result of cyclic shift is one of the following six cases:$(a_0, a_1, a_2, a_3, a_4, a_5)$,$(a_1, a_2, a_3, a_4, a_5, a_0)$, $(a_2, a_3, a_4, a_5, a_0, a_1)$,$(a_3, a_4, a_5, a_0, a_1, a_2)$,$(a_4, a_5, a_0, a_1, a_2, a_3)$,$(a_5, a_0, a_1, a_2, a_3, a_4)$. The probability of each occurrence is $1/6$.

B. Our Main Protocol

1. Each player has a tuple of cards that is the commitments of bits of $(i-1)_2$:

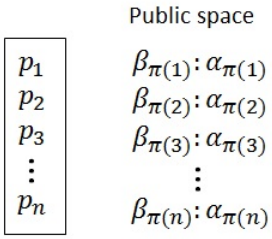


Each row of the cards is called gift ID and denoted α_i.

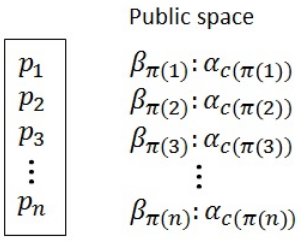
2. Generate a player ID. The player ID generated by p_i is denoted $\beta_i = (b_{\lceil log\ n \rceil}, ..., b_1)$, where

$$(i-1)_{10} = (b_{\lceil log\ n \rceil}, ..., b_2 b_1)_2.$$

3. Apply the pile-scramble shuffle to the pairs of player ID and gift ID. The shuffle is executed in a public space. π is a permutation function of $\{1, \ldots, n\} \to \{1, 2, \ldots, n\}$.



4. Divide the player ID and gift ID. Apply the cyclic shuffle only to the gift IDs. c is a cyclic shuffle of $\{1, \ldots, n\} \to \{1, 2, \ldots, n\}$.



5. A representative player executes the fixed point check protocol in 3.1 to the pair of $(\beta_{\pi(1)}, \alpha_{c(\pi(1))})$. In order to execute the check, the player ID , $\beta_{\pi(1)}$ needs to be copied. If the representative player finds a fixed point, all players go back to step 4.

6. Apply the pile-scramble shuffle to the pairs of the player ID and gift ID. Reveal the player ID of every pair. Let i-th pair be $(\beta_{x_i}, \alpha_{y_i})$. If $\beta_{x_i} = \beta_j$, p_j obtains the gift α_{y_i}.

The main differences between the proposed protocol and the known protocol are introduction of cyclic shuffles and using the pair of player ID and gift ID. In order to use the player ID, the increase of the number of cards is $2n\lceil \log n \rceil$. In addition, the player ID needs to be copied in step 5. It does not increase the number of cards, since we can use the six cards that are added in step 3.A for the additional cards in the copy protocol and the copy can be executed bit by bit. However, the number of executions of the copy protocol has increased $\lceil \log n \rceil$ times. Because of the nature of the cyclic shuffle, either of the following two cases occur: (1) no player has a fixed point or (2) all players have a fixed point. Thus, it is sufficient to check the existence of a fixed point to only one pair of player ID and gift ID. Therefore, the number of executions of fixed point check protocol decreases. If a cyclic shift is performed without a random permutation, player p_i knows which gift is given to which player from the gift ID received by p_i. The sequence of a random permutation and a cyclic shift is not secure either, because the player IDs are open at the end of the protocol and each player knows the order of gift IDs is a cyclic shift of current order of player IDs. Our protocol executes a random permutation before and after the cyclic shuffle. Therefore, it is impossible to guess the other player's obtained gift ID from the player's obtained gift ID. Thus, this protocol is secure.

V. Evaluation of the Protocol

This section compares the proposed protocol and the known protocol by the probability of the success and the number of shuffles. In this paper, the computation cost of one cyclic shuffle and one random permutation is considered to be the same.

A. Known Protocol

The number of different permutation π such that $\pi(i) \neq i$ for every $i (1 \leq i \leq n)$ is represented as follows:

$$X_{(n)} = \sum_{m=2}^{n} \frac{n!}{m!} \cdot (-1)^m$$

Since the number of all possible permutations of n items is $n!$, the success probability of the existing protocol (executing one random permutation) is

$$X = \sum_{m=2}^{n} \frac{1}{m!} \cdot (-1)^m$$

B. Our Protocol

The number of different cyclic shuffle c such that $c(i) \neq i$ for every $i (1 \leq i \leq n)$ is represented as follows:

$$Y_{(n)} = n - 1$$

Since the number of different cyclic shuffle is n, the success probability of our protocol (executing one random permutation) is

$$Y = (n-1)/n$$

C. Evaluation

The main difference between the known protocol and our protocol is the part of shuffle. Therefore, the evaluation is carried out only for the shuffle part. In the existing protocol, the random permutation is executed once in each round at

step $3.A$. If fixed points are found after each of $m-1$ random permutations but not found after the m-th random permutation, the total number of permutations is m. The proposed protocol needs to execute the random permutation twice, thus the number of total shuffles is $m+2$ if fixed points are found after each of $m-1$ cyclic shuffles but not found after the m-th cyclic shuffle. Therefore, in order to compare the success probabilities of the two protocols for the same computation costs, we need to compare executing three rounds by the known protocol and one round by the proposed protocol. The probabilities when the number of players n is 3, 4, 5, 6, and 7 are shown in Table 1.

TABLE I
SUCCESS PROBABILITY OF THE TWO PROTOCOLS

number of players	3	4	5	6	7
[1]within one round	0.333	0.375	0.367	0.368	0.368
[1]within two rounds	0.556	0.601	0.599	0.600	0.600
[1]within three rounds	0.704	0.755	0.746	0.748	0.747
ours one round	0.666	0.750	0.800	0.833	0.857

In Table 1, we need to compare the probability of success within three rounds by the known protocol and one round by our protocol. If the number of players is small $(n < 5)$ the success probabilities of the known protocol is better. However, if the number of players $n \geq 5$, the success probabilities of our protocol is better. Next we compare 2 rounds by our protocol and 4 rounds by the known protocol.

TABLE II
SUCCESS PROBABILITY OF WITHIN TWO ROUNDS BY OUR PROTOCOL

number of people	3	4	5	6	7
[1]within four rounds	0.802	0.847	0.840	0.841	0.840
ours within two rounds	0.888	0.934	0.960	0.972	0.980

From Table 2, if our protocol was carried out twice, even if the number of players is three, our protocol is better. Next, we compare the computation costs other than the permutations of the two protocols. Since random bisection cut is a dominating procedure for each of the primitive protocols, we evaluate the number of random bisection cuts. NOT protocol does not execute a random bisection cut. The other protocols such as AND, copy, etc. execute one random bisection cut. Therefore, the cost of these protocols are treated as equivalent. For one fixed point check, AND, XOR, One-input-preserving AND, and Copy protocol are executed $3\lceil \log n \rceil - 1$ times. In the known protocol, all players need to execute one fixed point check. Thus the total number of random bisection cuts is $n(3\lceil \log n \rceil - 1)$. In our protocol, a representative player executes checking the existence of fixed points, thus the number of executions of random bisection cuts is $3\lceil \log n \rceil - 1$. Furthermore, in order to save the player ID, the copy protocol is executed $\lceil \log n \rceil$ times. The total number of executions of random bisection cuts when our prococol is executed once is $4(\lceil \log n \rceil - 1) \leq n(3\lceil \log n \rceil - 1)$, by the known protocol. Thus for the computation cost in one round, our protocol is better. About the number of rounds, our protocol tends to be better when the number of player is large. Thus our protocol is also better in the point of computation costs.

VI. CONCLUSION

In this paper, we introduced the cyclic shuffle and player ID to the card based protocol that calculates a random permutation without fixed points. The number of cards is $4n\lceil \log n \rceil + 6$. In the cace of $n \geq 5$, the success probability of our protocol is better than the known protocol for the same number of shuffles. Because of the nature of the cyclic shuffle, just one player executes the ckecking of existence of fxed points, thus the computation cost is also better than the known protocol.

The proposed protocol cannot suppress the worst-case execution times. Future challenges is the propose of the protocol that is guaranteed to finish in constant rounds.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 26330019.

REFERENCES

[1] Rie Ishikawa, Eikoh Chida, and Takaaki Mizuki: "Efficient Card-based Protocols for Generating a Hidden Random Permutation without Fixed Points." Unconventional Computation and Natural Computation (UCNC 2015), Lecture Notes in Computer Science, Springer-Verlag, vol.9252, pp.215-226, 2015.

[2] Claude Crépeau, Joe Kilian: "Discreet Solitary Games." CRYPTO 1993, LNCS 773, pp. 319-330, 1994.

[3] Takaaki Mizuki and Hideaki Sone: "Six-Card Secure AND and Four-Card Secure XOR." Frontiers in Algorithmics (FAW 2009), Lecture Notes in Computer Science, Springer-Verlag, vol.5598, pp.358-369, 2009.

[4] B. den Boer: "More efficient match-making and satisability:the five card trick."Proc. EUROCRYPT '89, Lecture Notes in Computer Science, vol. 434, pp. 208-217, Springer-Verlag, 1990.

[5] Takaaki Mizuki, Michihito Kumamoto, and Hideaki Sone: "The Five-Card Trick Can Be Done with Four Cards." ASIACRYPT 2012, Lecture Notes in Computer Science, Springer-Verlag, vol.7658, pp.598-606, 2012.

[6] Kazumasa Shinagawa, Naoki Kanayama, Koji Nuida, and Takashi Nishide: "Card-Based Cryptographic Protocol using Polarization Plates." Computer Security Symposium 2014.(In Japanese)

[7] Kazumasa Shinagawa, Takaaki Mizuki, Koji Nuida, Naoki Kanayama, Takashi Nishide, and Eiji Okamoto: "Secure Computation Using Regular Polygon Cards." Symposium on Cryptography and Information Security 2015:(In Japanese)

[8] Takuya Nishida, Yu-ichi Hayashi, Takkaki Mizuki, and Hideaki Sone: "Secure Three-Input Majority Computation Using a Deck of Cards." Computer Security Symposium 2013.(In Japanese)

[9] Takaaki Mizuki: "Secure Multi-Party Computations Using a Deck of Cards." IEICE Fundamentals Review Vol.9 No.3 2015.(In Japanese)