Secure Implementations of a Random Bisection Cut*

Itaru Ueda† Daiki Miyahara† Akihiro Nishimura† Yu-ichi Hayashi‡
Takaaki Mizuki† Hideaki Sone†

Abstract

By using a deck of cards, it is possible to realize a secure multiparty computation. In particular, since a new shuffling operation, called a random bisection cut, was devised in 2009, many efficient card-based protocols have been designed. The random bisection cut functions in the following manner. A sequence of cards is bisected, and the two halves are shuffled. This results in two possible cases depending on whether the two halves of the card sequence are swapped. As only two possibilities exist when a random bisection cut is performed, it has been suggested that information regarding the outcome of the shuffle could sometimes be leaked visually. Thus, in this paper we propose some methods for securely implementing a random bisection cut without leaking such information.

1 Introduction

It is known that by using a deck of cards, we can realize secure multiparty computations. For example, consider a secure AND computation of bits $a \in \{0,1\}$ and $b \in \{0,1\}$, i.e., assume that we only want to know the value of $a \wedge b$. By utilizing a black card $\boxed{\clubsuit}$ and a red card $\boxed{\heartsuit}$, we can represent the value of a bit as follows:

$$\boxed{\clubsuit}\boxed{\heartsuit} = 0,\ \boxed{\heartsuit}\boxed{\clubsuit} = 1.$$

According to this encoding, each of the input bits a and b can be represented using two face-down cards of different colors:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\ \underbrace{\boxed{?}\boxed{?}}_{b}.$$

A pair of face-down cards (such as in the above example) is called *a commitment*. That is, the two cards on the left constitute a commitment to a, and the two cards on the right constitute a commitment to b. As in this example, the cards we use are either black cards $\boxed{\clubsuit}$ or red cards $\boxed{\heartsuit}$, whose backs are assumed to be identical $\boxed{?}$. As shown in Table 1, many protocols have been designed to perform a secure AND computation, from among which we introduce the Mizuki–Sone AND protocol [13]. Given commitments to inputs a and b along with two additional cards $\boxed{\clubsuit}\boxed{\heartsuit}$, the protocol works as follows.

1. A commitment to 0 is placed between the two input commitments:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{b} \ \rightarrow\ \underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{b}.$$

*An earlier version of this study was presented at 5th International Conference on the Theory and Practice of Natural Computing, TPNC 2016, Japan, December 12–13, 2016, and appeared in Proc. TPNC 2016, Lecture Notes in Computer Science, Springer International Publishing, vol. 10071, pp. 58–69, 2016 [22]. This is a post-peer-review, pre-copyedit version of an article published in International Journal of Information Security. The final authenticated version is available online at: https://dx.doi.org/10.1007/s10207-019-00463-w.

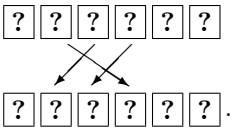
†Tohoku University

‡Nara Institute of Science and Technology

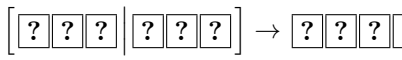
Table 1: Some committed-format AND protocols (RC: Random Cut, RBC: Random Bisection Cut)

	# of colors	# of cards	Type of shuffle	Avg. # of trials
[3]	4	10	RC	6
[17]	2	12	RC	2.5
[21]	2	8	RC	2
[13]	2	6	RBC	1
[1]	2	5	RC,RBC	5

2. The sequence order is then rearranged as follows:

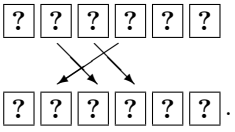


3. A *random bisection cut* is applied as follows:

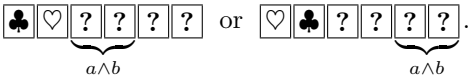


A random bisection cut is a shuffling operation that bisects a sequence of cards and swaps the two halves randomly. Therefore, the shuffle results in two possible cases, depending on whether the two halves are swapped, each with a probability of 1/2.

4. The sequence order is rearranged as follows:

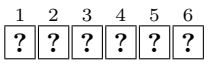


5. The two left-most cards are turned over, and we are able to obtain a commitment to $a \wedge b$ as follows:

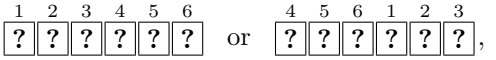


Although we omitted an explanation regarding the correctness and secrecy of this protocol, one can confirm that the protocol outputs a commitment to $a \wedge b$ by using six cards after one execution of the random bisection cut [13]. (A protocol such as this that outputs commitments is called a *committed-format protocol*.)

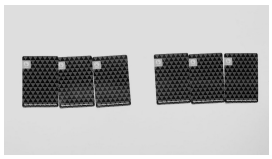
In practice, humans can perform a random bisection cut by shuffling the two halves after bisecting a given sequence of cards, as illustrated in Figure 1. Thus, given a sequence of six cards



a random bisection cut results in



with a probability of 1/2 for each possibility, where the numbers attached to the cards are for the sake of convenience.



(a) Bisection of a sequence of cards

(b) Shuffling of the two halves

Figure 1: Execution of a random bisection cut

Following the computational model formalized in the studies [7, 11], this random bisection cut can be described as follows:

$$(\mathsf{shuffle}, \{\mathsf{id}, (1\,4)(2\,5)(3\,6)\}),$$

where id represents the identity permutation, and an expression, such as $(1\,4)$, represents a cyclic permutation. Therefore, id indicates that the two halves are not swapped, and permutation $(1\,4)(2\,5)(3\,6)$ indicates that the two halves are swapped.

Historically, random bisection cuts first appeared when a six-card AND protocol was designed in 2009 [13]. Even before this design, some committed-format AND protocols had been designed. These earlier protocols employed *random cut* as a shuffling operation, as shown in Table 1. A random cut refers to a cyclic shuffling operation. For example, given eight face-down cards

$$\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array},$$

a random cut results in one of the following eight cases, each with a probability of 1/8:

$$\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array}$$

$$\begin{array}{cccccccc} 2 & 3 & 4 & 5 & 6 & 7 & 8 & 1 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array}$$

$$\vdots$$

$$\begin{array}{cccccccc} 8 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array}.$$

Therefore, by following the computational model in the studies [7, 11], we can similarly describe the random cut as

$$(\mathsf{shuffle}, \{\mathsf{id}, \pi, \pi^2, \pi^3, \pi^4, \pi^5, \pi^6, \pi^7\}),$$

where $\pi = (8\,7\,6\,5\,4\,3\,2\,1)$.

As seen in Table 1, committed-format AND computations have become more efficient by virtue of the introduction of the random bisection cut in 2009. This introduction also provided the additional benefit of improving the efficiency of non-committed-format AND computations and committed-format XOR computations, as detailed in Table 2. In addition, other efficient protocols have been designed using random bisection cuts [6, 8, 9, 18, 19].

As explained earlier, *card-based protocols* are intended in practice to be executed by humans who actually desire to perform secure multiparty computations by using a real deck of cards. Hence, when we execute a card-based protocol, it is expected that all players gather at the same physical location, and perform operations, such as shuffles, in public, as in the case of ordinary card games [12] [1].

To implement a random cut in such a situation, it is sufficient that each player cuts a sequence of face-down cards in turn until all players are satisfied with the result. Indeed, in practice, it is relatively

[1] It should be noted that recent work (e.g., [15, 16]) considers the use of private actions by players to design efficient protocols.

Table 2: Some other protocols

	# of colors	# of cards	Type of shuffle	Avg. # of trials
◦ *Non-committed-format AND Protocols*				
[2]	2	5	RC	1
[10]	2	4	RBC	1
◦ *Committed-format XOR Protocols*				
[3]	4	14	RC	6
[14]	2	10	RC	2
[13]	2	4	RBC	1

Figure 2: Each half is placed in an envelope

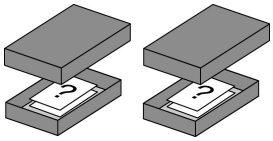


Figure 3: Each half is placed in a box

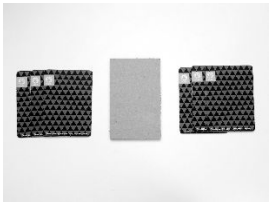
easy to implement a random cut such that nobody is able to determine the result of the shuffle at all. We will discuss this further in Section 5.

When players operate a random bisection cut, if they are not familiar with playing cards and have difficulty in shuffling the two halves such that each half stays together, as in Figure 1(b), then they may secure each half using paper clips or envelopes [10, 13]. By using these auxiliary tools, we are able to fix each of the two halves together, as shown in Figure 2. Following this, it suffices to swap the two bundles of cards randomly. However, the result of the shuffle could be revealed when we execute a random bisection cut in public because there are only two possibilities, i.e., the two halves of the card sequence are swapped or not swapped. That is, someone may count how many times the two bundles are swapped. To avoid such a leak of information, one solution is that each player shuffles the two bundles behind his/her back or under a table, so that other players cannot see whether the two bundles are swapped. In this case, it may be preferable to use envelopes or boxes (as illustrated in Figures 2 and 3) rather than paper clips to avoid malicious actions[2]. Nevertheless, it is desirable for all actions to be performed in front of all players and/or third parties publicly. Therefore, in Sections 2, 3, and 4, we present implementations of random bisection cuts, where every action can be performed in public.

The remainder of this paper is organized as follows. In Section 2, we present some methods for implementing a random bisection cut by using auxiliary tools. In Section 3, we propose a method to reduce the execution of a random bisection cut to the execution of random cuts using dummy cards. In Section 4, we propose another method to implement a random bisection cut without relying on dummy cards. In Section 5, we discuss secure implementations of the random cut.

An earlier version of this study was presented and appeared as an LNCS (Lecture Notes in Computer Science) paper [22]. The present paper is its extension and provides a more secure way for performing the "spinning throw" (which is an implementation of the random bisection cut) and proposes yet another implementation reducing the random bisection cut to the random cut so that neither dummy cards nor trials are needed. Sections 2.2 and 4 are devoted to these new results.

[2]Envelopes or boxes can be also used for implementing other types of shuffling operations ([4–6, 20]).



(a) Separator and two halves

(b) One half is placed on the separator

(c) The pile consisting of one half and the separator is flipped

(d) The other half is placed on the pile

Figure 4: Setup for spinning throw

2 Executing a Random Bisection Cut Using Auxiliary Tools

In this section, we provide methods for implementing a random bisection cut by using auxiliary tools that consist of everyday objects.

2.1 Use of a Separator Card and Rubber Band

In this subsection, we present a novel method of performing a random bisection cut by using a separator card with a rubber band. Both sides of the separator (as shown in the middle of Figure 4(a)) must be indistinguishable.

The method works as follows. First, a sequence of cards is bisected, with one half placed on the separator, as shown in Figure 4(b). Second, the pile consisting of one half and the separator is turned over, as shown in Figure 4(c)[3]. Third, the other half is placed on the pile, as shown in Figure 4(d), and these are fixed together by using a rubber band to prevent the cards from scattering. Next, the pile is thrown in a spinning manner (as illustrated in Figure 5). We call this action a *spinning throw*. After the pile is caught, we are completely unsure of which half is on the top. Finally, the rubber band is removed, and the actions described in Figure 4 are undone in the reverse order from Figures 4(d) to 4(a). In this manner, we can conduct a random bisection cut securely.

2.2 More Secure Implementation by Using a Ball

During execution of the spinning throw shown in Figure 5, the outcome cannot be traced by human eyesight. For checking the security of this shuffle, we recorded a video[4] of a spinning throw and confirmed that we could not determine the result of the shuffle by watching the video even in slow-motion.

Nevertheless, someone may assume that if we use an enterprise high-speed camera, the result of this shuffle might possibly be revealed. To address this issue, we considered the use of a curving polystyrene foam ball, as shown in Figure 6.

The procedure of using the ball device is as follows. After banding the pile by using a rubber band and a separator, it is placed in one half of a curving ball, as illustrated in Figure 6(a), covered with the

[3]The separator prevents information regarding the color of cards from being leaked.

[4]The camera we used was SONY FDR-AX40 and the video was recorded in 4K resolution and 60 fps.

(a) Hold the pile of cards

(b) Throw the pile like a coin

Figure 5: A spinning throw

other half (Figure 6(b)), and the halves are then taped together (Figure 6(c)) so as not to be separated. Finally, the ball is thrown in the air in a spinning manner, as illustrated in Figure 7. In this case, the pile spins inside the ball, and is therefore shuffled out of sight of everyone present. Consequently, a random bisection cut is implemented perfectly.

3 Execution of a Random Bisection Cut by Using Dummy Cards

In this section, we propose a method for reducing the execution of a random bisection cut to the execution of random cuts by using dummy cards.

Hereafter, we assume that we want to apply a random bisection cut to a sequence of $2n$ cards, where $n \geq 2$:

$$\Big[\underbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}_{n \text{ cards}}\Big|\underbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}_{n \text{ cards}}\Big].$$

Formally, it cam be defined as

$$(\mathsf{shuffle}, \{\mathsf{id}, (1\ n{+}1)(2\ n{+}2)\cdots(n\ 2n)\}),$$

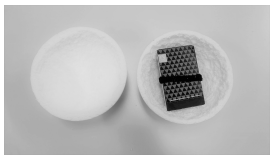
following the card-based computational model [7, 11].

As dummy cards, we use cards with backs as $\boxed{?}$ and faces other than $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, namely $\boxed{\diamond}$ or $\boxed{\spadesuit}$. Specifically, we use $2\lceil s/2\rceil$ $\boxed{\diamond}$ and $2\lfloor s/2\rfloor$ $\boxed{\spadesuit}$ cards, where $s \geq 2$. That is, we have a total of $2s$ additional cards.

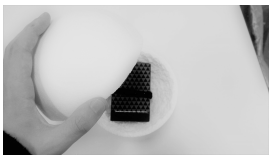
By using such dummy cards, we are able to implement a random bisection cut as follows.

1. Place dummy cards with their faces down, as follows:

$$\overbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}^{\text{dummy cards}}\underbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}_{n \text{ cards}}\overbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}^{\text{dummy cards}}\underbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}_{n \text{ cards}},$$



(a) The pile is placed into one half of the curving ball



(b) Cover the pile with the other

(c) The ball's halves are taped together

Figure 6: Making a ball device to execute a spinning throw securely

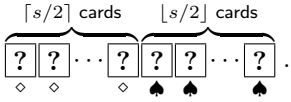
(a) Holding the ball

(b) Throwing the ball

(c) Catching the ball

Figure 7: The scene of throwing the ball device in the air with a spin

where the dummy cards are arranged as

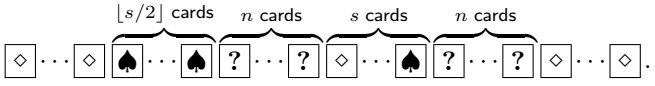


2. Apply a random cut:

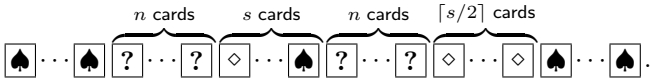
$$\langle \boxed{?}\boxed{?}\cdots\boxed{?}\boxed{?}\boxed{?}\cdots\boxed{?}\boxed{?}\boxed{?}\cdots\boxed{?}\boxed{?}\boxed{?}\cdots\boxed{?} \rangle .$$

3. Turn over the left-most card.

   (a) If the face-up card is $\boxed{\diamond}$, then turn over cards in the forward (the right-hand) direction until $\lfloor s/2 \rfloor$ $\boxed{\spadesuit}$ cards appear. Now that the positions of all of the dummy cards have been determined, all of them can be removed:



(b) If the face-up card is $\boxed{\spadesuit}$, then turn over cards in the backward direction (aside from cyclic rotations) until $\lceil s/2 \rceil$ $\boxed{\diamond}$ cards appear. After determining the positions of all of the dummy cards, all of them can be removed:



(c) If the face-up card is $\boxed{\clubsuit}$ or $\boxed{\heartsuit}$, then turn it over again and return to Step 2.

In this manner, after all the dummy cards have been removed, a random bisection cut is completed.[5]

In Step 3, the probability that either (a) or (b) occurs is $s/(n+s)$. Therefore, we are able to implement a random bisection cut by using $2s$ dummy cards after an average of $(n+s)/s$ executions of the random cut.

This method of discarding dummy cards was first devised by Crépeau and Kilian [3], when they proposed some random permutation generating protocols. We adopted their idea in this study.

Regarding the parameter s, a trade-off exists between the number of required cards and the average number of executions of the random cut. For example, if we want to implement the Mizuki–Sone six-card AND protocol [13] with an average number of two random cuts, then we require six additional dummy cards. This requires more cards than Stiglic's eight-card AND protocol [21] (although the Mizuki–Sone six-card AND protocol might have the advantage that its correctness is simpler to understand).

4 Utilizing Vertical Asymmetricity of the Backs of Cards

In Section 3, we required additional types of cards to reduce the execution of a random bisection cut to the execution of random cuts.

In this section, we present another method to implement a random bisection cut without relying on dummy cards. To this end, we exploited the vertical asymmetricity of the backs of cards $\boxed{?}$. As the back is asymmetric, it can be seen as either $\boxed{?}$ or $\boxed{¿}$, depending on its position.

4.1 Reduction to a Random Cut

The method of reducing to a random cut is quite simple, described as follows.

1. The first card of each half is rotated 180°:

$$\boxed{¿}\boxed{?}\cdots\boxed{?}\quad\boxed{¿}\boxed{?}\cdots\boxed{?}.$$

2. Apply a random cut:

$$\langle\,\boxed{¿}\boxed{?}\cdots\boxed{?}\boxed{¿}\boxed{?}\cdots\boxed{?}\,\rangle$$
$$\to\ \boxed{?}\cdots\boxed{¿}\cdots\boxed{?}\boxed{?}\cdots\boxed{¿}\cdots\boxed{?}.$$

Then, cyclically shift the sequence with the first card as $\boxed{¿}$:

$$\boxed{¿}\boxed{?}\cdots\boxed{?}\boxed{¿}\boxed{?}\cdots\boxed{?}.$$

3. Rotate the two of $\boxed{¿}$ again:

$$\boxed{¿}\boxed{?}\cdots\boxed{?}\boxed{¿}\boxed{?}\cdots\boxed{?}$$
$$\to\ \boxed{?}\boxed{?}\cdots\boxed{?}\boxed{?}\boxed{?}\cdots\boxed{?}.$$

In this manner, by executing one random cut, we are able to implement a random bisection cut with no dummy card.

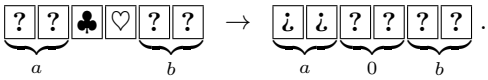
For example, the Mizuki–Sone six-card AND protocol [13] can be implemented using one random cut, as described in the following subsection.

[5]Note that we need two types of dummy cards ($\boxed{\diamond}$, $\boxed{\spadesuit}$) to determine their exact positions.

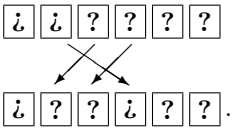
4.2 Six-Card AND Protocol with a Random Cut

We rewrite the six-card AND protocol [13] by using the method presented in Section 4.1.

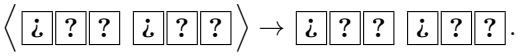
1. The two cards of a commitment to a are placed upside down:



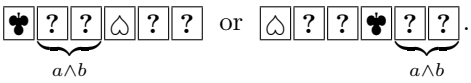
2. The order sequence is rearranged as follows:



3. A random cut (rather than a random bisection cut) is applied, and then the card(s) is cyclically shifted so that the first card is $\boxed{¿}$:



4. Two of $\boxed{¿}$ are turned over:



4.3 Application to Pile-Shifting Scrambles

We can extend the method described in Section 4.1 to implement a more general shuffle, called the *pile-shifting scramble*. In a pile-shifting scramble, a card sequence of n cards (such that $n \mod m = 0$) is divided into m piles, and a random cut is applied to the sequence of the piles without changing the order of the cards inside each plie. Therefore, a random bisection cut is a special case of pile-shifting scrambles, i.e., it corresponds to $m = 2$. One can easily have a reduction of a pile-shifting scramble to a random cut based on the idea that the first card in each pile is marked by placing it upside down.

In our method, we must apply a random cut to cards with asymmetric backs, and hence information regarding the result of the shuffle could be leaked more easily than with cards that have identical backs.

By considering the aforementioned, we discuss secure implementations of the random cut in the next section.

5 Secrecy of Implementations of the Random Cut

In Sections 3 and 4, we proposed some methods for reducing the execution of a random bisection cut to the execution of random cuts. In general, it is believed that a random cut can be securely implemented by humans. To support this belief, we discuss the secure implementation of a random cut by shuffling a real deck of cards.

As a random cut consists of a cyclic shuffle, its simple implementation proceeds as in Figure 8: some cards (or a card) are taken from the top of the pile, and then moved to the bottom of the pile (this is called a *cut*). At every cut, we should change the number of cards to be moved. For such an implementation, some people can trace the move of cards.

Thus, we require an alternative secure implementation of the random cut. The point of strength of shuffle secureness is the visual observation of the number of cards moved at every cut, and summing up of all the numbers. Hence, the key is to ensure that it is impossible for people to count the number of cards moved during every cut.

Figure 8: Simple execution of a random cut

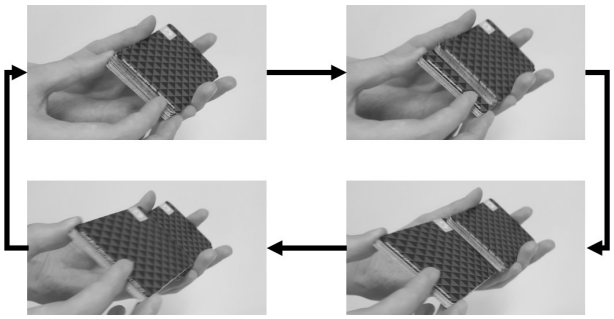


Figure 9: Execution of a "Hindu cut"; take a random number of cards from the bottom, move them to the top of the pile, and repeat this.

One concept is to move cards from the bottom to the top of the pile instead of moving them from the top to the bottom when executing a cut operation. As such, the recognition of the number of cards that have been moved becomes difficult. Moreover, if the positions of the cards are out of alignment, as shown in Figure 8, then it is possible to easily recognize the number of cards moved. Therefore, we should ensure that cards are not out of alignment when we execute cut operations.

Based on this concept, we found that a variation of the so-called Hindu shuffle (shown in Figure 9) is effective for preventing the cut operation from being revealed (we call this the *Hindu cut*).

We have experimentally demonstrated the security of this Hindu cut in the previous paper [22]. A summary of the experience is as follows. We requested 72 participants (who were non-specialists) to watch a video depicting the execution of the Hindu cut to the sequence of eight cards containing two non-identical back sides. As a result, 64 participants told us that they had not been able to track the move of the shuffle. Regarding the remaining 8 participants, to rule out wild guesses, we asked them to watch four more videos, and consequently, none of them was able to answer correctly for all of the five video. Refer to [22] for the details.

6 Conclusion

The random bisection cut has played an important role in improving card-based protocols. However, implementation issues have not been previously discussed. Therefore, in this paper, we proposed some novel methods for implementing the random bisection cut and demonstrated that it could be implemented in practice. Users can choose one from our several methods proposed in Sections 2, 3, and 4, depending on the availability of auxiliary tools and the patterns of backs of available cards.

Acknowledgements

We would like to offer our special thanks to Kohei Yamaguchi, who provided an excellent implementation of the random bisection cut, the spinning throw, as introduced in Section 2.1. We thank the anonymous

referees, whose comments have helped us to improve the presentation of the paper.

Compliance with Ethical Standards

Funding

This work was supported by JSPS KAKENHI Grant Number JP17K00001.

Conflict of Interest

The authors declare that they have no conflict of interest.

Ethical approval

This article does not contain any new studies with human participants or animals performed by any of the authors.

References

[1] Abe, Y., Hayashi, Y., Mizuki, T., Sone, H.: Five-card and protocol in committed format using only practical shuffles. In: Proceedings of the 5th ACM on ASIA Public-Key Cryptography Workshop. pp. 3–8. APKC '18, ACM, New York, NY, USA (2018)

[2] den Boer, B.: More efficient match-making and satisfiability: the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology — EUROCRYPT '89, Lecture Notes in Computer Science, vol. 434, pp. 208–217. Springer Berlin Heidelberg (1990)

[3] Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) Advances in Cryptology — CRYPTO '93, Lecture Notes in Computer Science, vol. 773, pp. 319–330. Springer Berlin Heidelberg (1994)

[4] Hashimoto, Y., Shinagawa, K., Nuida, K., Inamura, M., Hanaoka, G.: Secure grouping protocol using a deck of cards. In: Shikata, J. (ed.) Information Theoretic Security. Lecture Notes in Computer Science, vol. 10681, pp. 135–152. Springer International Publishing, Cham (2017)

[5] Ibaraki, T., Manabe, Y.: A more efficient card-based protocol for generating a random permutation without fixed points. In: Mathematics and Computers in Sciences and in Industry (MCSI), 2016 Third International Conference on. pp. 252–257. IEEE (2016)

[6] Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Calude, C.S., Dinneen, M.J. (eds.) Unconventional Computation and Natural Computation, Lecture Notes in Computer Science, vol. 9252, pp. 215–226. Springer International Publishing (2015)

[7] Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J. (eds.) Advances in Cryptology – ASIACRYPT 2015, Lecture Notes in Computer Science, vol. 9452, pp. 783–807. Springer Berlin Heidelberg (2015)

[8] Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theoretical Computer Science 622, 34–44 (2016)

[9] Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Mauri, G., Dennunzio, A., Manzoni, L., Porreca, A.E. (eds.) Unconventional Computation and Natural Computation, Lecture Notes in Computer Science, vol. 7956, pp. 162–173. Springer Berlin Heidelberg (2013)

[10] Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Wang, X., Sako, K. (eds.) Advances in Cryptology — ASIACRYPT 2012, Lecture Notes in Computer Science, vol. 7658, pp. 598–606. Springer Berlin Heidelberg (2012)

[11] Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security 13(1), 15–23 (2014)

[12] Mizuki, T., Shizuya, H.: Practical card-based cryptography. In: Ferro, A., Luccio, F., Widmayer, P. (eds.) Fun with Algorithms, Lecture Notes in Computer Science, vol. 8496, pp. 313–324. Springer International Publishing (2014)

[13] Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) Frontiers in Algorithmics, Lecture Notes in Computer Science, vol. 5598, pp. 358–369. Springer Berlin Heidelberg (2009)

[14] Mizuki, T., Uchiike, F., Sone, H.: Securely computing XOR with 10 cards. The Australasian Journal of Combinatorics 36, 279–293 (2006)

[15] Nakai, T., Shirouchi, S., Iwamoto, M., Ohta, K.: Four cards are sufficient for a card-based three-input voting protocol utilizing private permutations. In: Shikata, J. (ed.) Information Theoretic Security. Lecture Notes in Computer Science, vol. 10681, pp. 153–165. Springer International Publishing, Cham (2017)

[16] Nakai, T., Tokushige, Y., Misawa, Y., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations. In: Foresti, S., Persiano, G. (eds.) Cryptology and Network Security. Lecture Notes in Computer Science, vol. 10052, pp. 500–517. Springer International Publishing, Cham (2016)

[17] Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theoretical Computer Science 191(1–2), 173–183 (1998)

[18] Nishida, T., Hayashi, Y., Mizuki, T., Hideaki, S.: Securely computing three-input functions with eight cards. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences 98(6), 1145–1152 (2015)

[19] Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any boolean function. In: Jain, R., Jain, S., Stephan, F. (eds.) Theory and Applications of Models of Computation, Lecture Notes in Computer Science, vol. 9076, pp. 110–121. Springer International Publishing (2015)

[20] Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing 22(2), 361–371 (Jan 2018)

[21] Stiglic, A.: Computations with a deck of cards. Theoretical Computer Science 259(1–2), 671–678 (2001)

[22] Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Martín-Vide, C., Mizuki, T., Vega-Rodríguez, M.A. (eds.) Theory and Practice of Natural Computing. Lecture Notes in Computer Science, vol. 10071, pp. 58–69. Springer International Publishing, Cham (2016)