PAPER Special Section on Cryptography and Information Security

A Computationally Efficient Card-Based Majority Voting Protocol with Fewer Cards in the Private Model

Yoshiki ABE[†,††a)], ***Nonmember***, **Takeshi NAKAI**[†††*], **Yohei WATANABE**[†,††], ***Members***, **Mitsugu IWAMOTO**[†], ***Senior Member***, ***and*** **Kazuo OHTA**[†,††], ***Fellow***

SUMMARY Card-based cryptography realizes secure multiparty computation using physical cards. In 2018, Watanabe et al. proposed a card-based three-input majority voting protocol using three cards. In a card-based cryptographic protocol with n-bit inputs, it is known that a protocol using *shuffles* requires at least $2n$ cards. In contrast, as Watanabe et al.'s protocol, a protocol using *private permutations* can be constructed with fewer cards than the lower bounds above. Moreover, an n-input protocol using private permutations would not even require n cards in principle since a private permutation depending on an input can represent the input without using additional cards. However, there are only a few protocols with fewer than n cards. Recently, Abe et al. extended Watanabe et al.'s protocol and proposed an n-input majority voting protocol with n cards and $n + \lfloor n/2 \rfloor + 1$ private permutations. This paper proposes an n-input majority voting protocol with $\lceil n/2 \rceil + 1$ cards and $2n - 1$ private permutations, which is also obtained by extending Watanabe et al.'s protocol. Compared with Abe et al.'s protocol, although the number of private permutations increases by about $n/2$, the number of cards is reduced by about $n/2$. In addition, unlike Abe et al.'s protocol, our protocol includes Watanabe et al.'s protocol as a special case where $n = 3$.

key words: *multiparty computation, n-input majority voting, card-based protocol*

1. Introduction

1.1 Background

Secure multiparty computation [1] aims to compute a function of players' inputs keeping the inputs hidden from each other. *Card-based cryptography* [2], [3] is an implementation of secure multiparty computation using physical tools like playing cards. In this paper, we use two types of cards represented by ♣ and ♡, which have an identical back. Card-based cryptographic protocols are classified into two types by the operations used in the protocols. One is the protocols using operations called *shuffles* [2], [4]. All operations in shuffle-based protocols are performed in a public place. The other is the protocols using operations called *private permutations*. In private-permutation-based protocols, each player can perform operations at a private space, where other players cannot see the operations performed [5], [6].

In shuffle-based protocols, two cards are used for each one-bit input because of the convenience of several operations, e.g., negation. For example, a binary input is encoded as $0 \mapsto$ ♣♡ and $1 \mapsto$ ♡♣[**]. In this encoding, we can easily get the negated bit by swapping the two cards, which can be performed without knowing the bit value. Under the encoding using two cards, at least $2n$ cards are required for computing an n-input Boolean function to represent inputs.

In contrast, in private-permutation-based protocols, an input of each player can be represented by not only placing down their cards but also operating face-down cards on the place depending on the input. For example, a player represents a binary input by swapping two cards if the input is one and doing nothing if it is zero. Thus, private-permutation-based protocols can be executed with fewer than $2n$ cards. Note that the security model for private-permutation-based protocols is restricted to the semi-honest model [8]–[11] because players can not detect malicious behaviors in a private space[***]. However, despite the advantage of the number of cards, there are only a few n-input protocols with fewer than n cards, such as protocols for the millionaires' problem [12], [13] and a general protocol for any Boolean function [9] utilizing a branching program and Barrington's theorem [14]. Majority voting protocols are actively studied in card-based cryptography, e.g. [8], [9], [11], [15]. With the exception of Shinagawa's protocol [9], existing n-input majority voting protocols need n or more cards. Further, the exception is inefficient in terms of the number of private permutations [9]. Concretely, the exception, the general protocol [9], can execute with only five cards regardless of the number of inputs; however, the number of private permutations required for the protocol is exponential in d, where d is the circuit depth of a function to be computed.

1.2 Our Contributions

In this paper, we propose an n-input majority voting protocol (denoted by MAJ_n) with $\lceil n/2 \rceil + 1$ cards and $2n - 1$ private permutations for $n \geq 3$. Our protocol is constructed

Manuscript received March 15, 2022.
Manuscript revised August 26, 2022.
Manuscript publicized October 20, 2022.

[†]The authors are with The University of Electro-Communications, Chofu-shi, 182-8585 Japan.
[††]The authors are with National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, 135-0064 Japan.
[†††]The author is with Toyohashi University of Technology, Toyohashi-shi, 441-8580 Japan.
[*]This work was done while the author was with The University of Electro-Communications.
a) E-mail: yoshiki@uec.ac.jp

DOI: 10.1587/transfun.2022CIP0021

[**]There is an unconventional encoding of one bit with a rotationally asymmetric card such as $0 \mapsto$ ♣ and $1 \mapsto$ ♥ [7].

[***]There are studies to prevent and detect malicious behaviors using additional tools or introducing a player who monitors the operations (See, e.g., [16]–[18]).

Table 1 Comparison among n-input majority voting protocols. #Cards, #P.P., and #Comm. means the number of cards, private permutations, and communication, respectively. d is the depth of the circuit for n-input majority voting.

Protocol	#Cards	#P.P.	#Comm.
Proposed protocol (MAJ_n)	$\lceil n/2 \rceil + 1$	$2n-1$	$2n-2$
Ono and Manabe [10, Protocol 4]	$2n+4$	$O(2^n)$	$O(2^n)$
Ono and Manabe [10, Protocol 5]	2^{n+1}	$O(2^n)$	2
Nakai et al. [11]	$n+1$	n	$n-1$
Abe et al. ($\mathsf{MAJ}_n^{\mathsf{A}}$) [15]	n	$n + \lfloor n/2 \rfloor + 1$	$n + \lfloor n/2 \rfloor$
Shinagawa [9]	5	$O(4^d)$	$O(4^d)$
Watanabe et al. ($\mathsf{MAJ}_3^{\mathsf{W}}$) [8]	3	5	4

by extending Watanabe et al.'s three-input majority voting protocol (denoted by $\mathsf{MAJ}_3^{\mathsf{W}}$) [8].

Table 1 shows the comparison among the proposed protocol and existing protocols on efficiency measures of private-permutation-based protocol; the number of cards, private permutations, and communications. Note that the bottom row of Table 1 is a three-input majority voting protocol. Compared with Ono and Manabe's protocols [10, Protocols 4, 5], the number of cards and the number of private permutations are significantly reduced. Since both of Ono and Manabe's protocols in Table 1 can compute any logical function f with n inputs, our protocol specialized for majority voting is more efficient than those protocols. Note that their protocol [10, Protocol 4] computes f based on the Shannon's expansion and their protocol [10, Protocol 5] computes f based on the truth table of f. Compared with Nakai et al.'s protocol [11] (resp., Abe et al.'s protocol [15], denoted by $\mathsf{MAJ}_n^{\mathsf{A}}$), although the number of private permutations and the number of communications are increased about two times (resp. 4/3 times), the number of cards is reduced by about half. Abe et al.'s protocol $\mathsf{MAJ}_n^{\mathsf{A}}$ is constructed by extending $\mathsf{MAJ}_3^{\mathsf{W}}$ similar to our protocol MAJ_n. However, $\mathsf{MAJ}_n^{\mathsf{A}}$ is not a generalized protocol of $\mathsf{MAJ}_3^{\mathsf{W}}$ in the sense that $\mathsf{MAJ}_3^{\mathsf{A}}$ is not equivalent to $\mathsf{MAJ}_3^{\mathsf{W}}$ and actually does not work well (See Sect. 3.2.2 for details). On the other hand, our protocol MAJ_n includes $\mathsf{MAJ}_3^{\mathsf{W}}$ as a special case for $n = 3$.

1.3 Organization

The remaining part of this paper is organized as follows. In Sect. 2, we introduce the notation of cards, the definition of the majority voting, and cyclic shifts used in this paper. We show existing protocols, Watanabe et al.'s three-input majority voting protocol $\mathsf{MAJ}_3^{\mathsf{W}}$ [8] and Abe et al.'s n-input majority voting protocol $\mathsf{MAJ}_n^{\mathsf{A}}$ [15], in Sect. 3. Section 4 is devoted to describe our new n-input majority voting protocol. We conclude this paper in Sect. 5.

2. Preliminaries

In this paper, we consider protocols in the semi-honest model, i.e., players follow a protocol procedure that may include input operations multiple times.

2.1 Notation

We use two types of cards, denoted by $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. The backs of these cards are denoted by $\boxed{?}$. We assume the same type of cards are indistinguishable and the backs of all cards are also indistinguishable. In addition, a bit is encoded as $0 \mapsto \boxed{\clubsuit}$, $1 \mapsto \boxed{\heartsuit}$. For simplicity, $\clubsuit$ and $\heartsuit$ are used instead of $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, respectively, in the following text.

2.2 Majority Voting

For n binary inputs $x_1, x_2, \ldots, x_n \in \{0,1\}$, n-input majority voting, denoted by maj_n is defined as follows.

$$\mathsf{maj}_n(x_1, x_2, \ldots, x_n) := \begin{cases} 0 & \text{if } \sum_{i=1}^n x_i \leq \lfloor n/2 \rfloor \\ 1 & \text{if } \sum_{i=1}^n x_i > \lfloor n/2 \rfloor \end{cases} \quad (1)$$

If the number of inputs of one is equal to $n/2$ (i.e., the voting result is a tie), maj_n outputs zero. We can consider another majority voting, denoted by $\mathsf{maj}_n^{\mathrm{half}}$ and defined as follows.

$$\mathsf{maj}_n^{\mathrm{half}}(x_1, x_2, \ldots, x_n) := \begin{cases} 0 & \text{if } \sum_{i=1}^n x_i < \lfloor n/2 \rfloor \\ 1 & \text{if } \sum_{i=1}^n x_i \geq \lfloor n/2 \rfloor \end{cases} \quad (2)$$

Only when the voting result is a tie, $\mathsf{maj}_n^{\mathrm{half}}$ outputs a value different from maj_n. Since we can compute $\mathsf{maj}_n^{\mathrm{half}}$ by flipping the bits of inputs and output in a protocol for computing maj_n, we construct a protocol for computing Eq. (1) in this paper.

2.3 Cyclic Shift

For explanation, we use numeric cards instead of $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ to show the order of a card sequence only in this subsection.

For a sequence of n cards $\boxed{1}\boxed{2}\boxed{3}\boxed{4}\cdots\boxed{n}$, a *left cyclic shift* is defined as a permutation that the leftmost card of the sequence is moved to the rightmost position. For instance, a left cyclic shift over a sequence of five cards can be represented as follows.

$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5} \mapsto \boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{1}$$

In addition, a left cyclic shift can be applied to a part of the sequence of cards. For example, the following shows the before and after of a left cyclic shift over the left four cards

of a sequence of five cards.

$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5} \mapsto \boxed{2}\boxed{3}\boxed{4}\boxed{1}\boxed{5}$$

Analogously, a *right cyclic shift* is defined as a permutation that the rightmost card of the sequence of cards is moved to the leftmost position. A right cyclic shift over five cards and a right cyclic shift over the left four cards of a sequence of five cards are represented as follows, respectively.

$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5} \mapsto \boxed{5}\boxed{1}\boxed{2}\boxed{3}\boxed{4}$$
$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5} \mapsto \boxed{4}\boxed{1}\boxed{2}\boxed{3}\boxed{5}$$

3. Previous Works

Before constructing our proposed protocol, we show two existing protocols which become the base of our protocol. One is Watanabe et al.'s three-input majority voting protocol ($\mathrm{MAJ}_3^{\mathsf{W}}$) [8]. The other is Abe et al.'s n-input majority voting protocol ($\mathrm{MAJ}_n^{\mathsf{A}}$) [15].

3.1 $\mathrm{MAJ}_3^{\mathsf{W}}$: Three-Input Majority Voting Protocol [8]

Suppose that there are three players P_1, P_2, P_3 and P_i ($i = 1,2,3$) has an binary input $x_i \in \{0,1\}$. Watanabe et al.'s three-input majority voting protocol ($\mathrm{MAJ}_3^{\mathsf{W}}$) [8] is shown in Protocol 1. Note that the operations of putting cards, swapping cards, and doing nothing in steps 1) – 5) of $\mathrm{MAJ}_3^{\mathsf{W}}$ are performed in the operator's private space.

$\mathrm{MAJ}_3^{\mathsf{W}}$ is constructed based on the following another representation of maj_3,

$$\mathsf{maj}_3(x_1,x_2,x_3) = \begin{cases} \mathsf{and}_2(x_1,x_2) & \text{if } x_1 + x_2 \neq 1, \\ x_3 & \text{if } x_1 + x_2 = 1, \end{cases} \quad (3)$$

where and_2 is the two-input logical AND function. Namely, in $\mathrm{MAJ}_3^{\mathsf{W}}$, after executing and_2 protocol (corresponding to steps 1) and 2), the operations are performed which swap the cards representing $\mathsf{and}_2(x_1,x_2)$ and x_3 only when $x_1 + x_2 = 1$ (corresponding to steps 3)–5)).

Hereafter, we consider $\mathrm{MAJ}_3^{\mathsf{W}}$ consists of the following three phases, which become the base idea to extend $\mathrm{MAJ}_3^{\mathsf{W}}$ in Sects. 3.2 and 4.

Placement phase: P_1, P_2 place all of three cards face-down and perform private permutations to generate a card sequence depending on their inputs. (cf. Steps 1) and 2) of $\mathrm{MAJ}_3^{\mathsf{W}}$.)

Permutation phase: Each player P_i ($i = 1,2,3$) performs private permutations depending on the player's input. (cf. Steps 3)–5) of $\mathrm{MAJ}_3^{\mathsf{W}}$.) Specifically, this phase consists of the following two sub-phases.

Permutation phase 1: P_3 swaps the center and rightmost cards of the card sequence if $x_3 = 0$; otherwise, P_3 does nothing. (cf. Step 3) of $\mathrm{MAJ}_3^{\mathsf{W}}$.)

Protocol 1 $\mathrm{MAJ}_3^{\mathsf{W}}$: Three-input Majority Voting Protocol [8]

Inputs: P_1, P_2, P_3 input $x_1, x_2, x_3 \in \{0,1\}$, respectively.
Setup: P_1 has two cards, ♣ and ♡. P_2 has a card, ♣. P_3 has nothing.

1) If $x_1 = 0$, P_1 sends P_2 face-down cards ♣♡ in this order; otherwise, P_1 sends P_2 face-down cards ♡♣ in this order.
2) If $x_2 = 0$, P_2 places a face-down ♣ at the left of the card pair P_2 received; otherwise, P_2 places a face-down ♣ at the right of the card pair. Then, P_2 sends all face-down cards to P_3.
3) If $x_3 = 0$, P_3 swaps the center and rightmost cards of the cards P_3 received; otherwise, P_3 does nothing. Then, P_3 sends all cards to P_2.
4) If $x_2 = 0$, P_2 swaps the center and leftmost cards of the cards P_2 received; otherwise, P_2 does nothing. Then, P_2 sends all cards to P_1.
5) If $x_1 = 0$, P_1 swaps the center and leftmost cards of the cards P_1 received; otherwise, P_1 does nothing. Then, P_1 opens the leftmost card, and discards the others without opening.

Output: If the card P_1 opens at step 5) is ♣, the output is 0; otherwise (if it is ♡), the output is 1.

Permutation phase 2: Each player P_i ($i = 1,2$) swaps the center and leftmost cards of the card sequence if $x_i = 0$; otherwise, P_i does nothing. (cf. Steps 4) and 5) of $\mathrm{MAJ}_3^{\mathsf{W}}$.)

Output phase: The protocol outputs the leftmost card of the card sequence. (cf. Output of $\mathrm{MAJ}_3^{\mathsf{W}}$.)

3.2 n-Input Majority Voting Protocol [15]

We show Abe et al.'s n-input majority voting protocol ($\mathrm{MAJ}_n^{\mathsf{A}}$) [15] for odd n ($= 2m+1$) in Protocol 2. As in $\mathrm{MAJ}_3^{\mathsf{W}}$, the operations of a left cyclic shift and doing nothing in steps 1)–3) of $\mathrm{MAJ}_n^{\mathsf{A}}$ are performed in the operator's private space. Note that steps 1), 2), and 3) of $\mathrm{MAJ}_n^{\mathsf{A}}$ correspond to the placement phase, the permutation phases 1, and the permutation phase 2, shown in Sect. 3.1, respectively.

3.2.1 How to Extend $\mathrm{MAJ}_3^{\mathsf{W}}$ to $\mathrm{MAJ}_5^{\mathsf{A}}$

We show how to construct $\mathrm{MAJ}_n^{\mathsf{A}}$ because our construction described in Sect. 4 is based on the same construction method for $\mathrm{MAJ}_n^{\mathsf{A}}$. In the method, $\mathrm{MAJ}_n^{\mathsf{A}}$ is constructed as follows: (a) fix operations corresponding to the permutation phase of $\mathrm{MAJ}_3^{\mathsf{W}}$; and (b) based on (a), deduce concrete operations corresponding to the placement phase of $\mathrm{MAJ}_3^{\mathsf{W}}$. Since there is no essential difference, we show the case where $n = 5$ instead of the general n.

Suppose that each player P_i ($i = 1,2,\ldots,5$) has a binary input $x_i \in \{0,1\}$. At first, we fix $\mathrm{MAJ}_n^{\mathsf{A}}$ as consisting of the following three phases, similar to $\mathrm{MAJ}_3^{\mathsf{W}}$.

Placement phase: P_1, P_2, P_3 place all of five cards face-down and perform private permutations to generate a card sequence depending on their inputs. *We note that concrete operations in this phase are determined later.*

Permutation phase: Each player P_i ($i = 1,2,\ldots,5$) performs private permutations depending on the player's

Protocol 2 $\mathrm{MAJ}^{\mathsf{A}}_{2m+1}$: $(2m+1)$-input Majority Voting Protocol [15]

Inputs: $P_1, P_2, \ldots, P_{2m+1}$ input $x_1, x_2, \ldots, x_{2m+1} \in \{0,1\}$, respectively.

Setup: Publicly put $2m+1$ face-down cards $\overbrace{\clubsuit\heartsuit\cdots\heartsuit}^{m-1}\heartsuit\overbrace{\clubsuit\cdots\clubsuit}^{m}$ in this order. Then, P_1 receives these $2m+1$ cards.

1) From $i = 1$ to $i = m+1$, if $x_i = 0$, P_i does nothing; otherwise, P_i does a left cyclic shift over the m cards from the left of the received cards. Then, P_i sends the cards to P_{i+1}.
2) From $i = m+2$ to $i = 2m+1$, if $x_i = 0$, P_i does a left cyclic shift over the $2m$ cards from the right of the received cards; otherwise, P_i does nothing. Then, P_i sends the cards to $P_{(i \bmod (2m+1))+1}$.
3) From $i = 1$ to $i = m+1$, if $x_i = 0$, P_i does a left cyclic shift over the $m+1$ cards from the left of the received cards; otherwise, P_i does nothing. Then, if $i \neq m+1$, P_i sends the cards to P_{i+1}. If $i = m+1$, P_{m+1} opens the leftmost card of the $2m+1$ cards, and discards the others without opening.

Output: If the card P_{m+1} opens at step 3) is ♣, the output is 0; otherwise (if it is ♡), the output is 1.

Table 2 Card sequences after the permutation phase 1 in $\mathrm{MAJ}^{\mathsf{A}}_5$.

$\sum_{i=1}^{3} x_i$	Card Sequence
0	■ □ □ □ □
1	□ □ ■ □ □
2	□ ■ □ □ □
3	■ □ □ □ □

Table 3 Card sequences after the placement phase in $\mathrm{MAJ}^{\mathsf{A}}_5$, where $X := x_4 + x_5$.

$\sum_{i=1}^{3} x_i$	Card Sequence
0	■ □ □ □ □
1	□ □ ■ ■ ■ (if $X=2$ if $X=1$ if $X=0$)
2	□ ■ ■ ■ □ (if $X=2$ if $X=1$ if $X=0$)
3	■ □ □ □ □

Table 4 Requirements to compute maj_5 correctly.

$\sum_{i=1}^{3} x_i$	Card Sequence
0	♣ □ □ □ □
1	□ □ ♡ ♣ ♣
2	□ ♡ ♡ ♣ □
3	♡ □ □ □ □

Table 5 Card sequences resulting of the placement phase in $\mathrm{MAJ}^{\mathsf{A}}_5$.

$\sum_{i=1}^{3} x_i$	Card Sequence
0	♣ ♡ ♡ ♣ ♣
1	♡ ♣ ♡ ♣ ♣
2	♣ ♡ ♡ ♣ ♣
3	♡ ♣ ♡ ♣ ♣

input. Specifically, this phase consists of the following two sub-phases.

Permutation phase 1: Each player P_i $(i = 4, 5)$ performs a left cyclic shift over the four cards on the right of the card sequence if $x_i = 0$; otherwise, P_i does nothing.

Permutation phase 2: Each player P_i $(i = 1, 2, 3)$ performs a left cyclic shift over the three cards on the left of the card sequence if $x_i = 0$; otherwise, P_i does nothing.

Output phase: The protocol outputs the leftmost card of the card sequence.

Note that the operations in the above three phases of $\mathrm{MAJ}^{\mathsf{A}}_5$ are extended from those in $\mathrm{MAJ}^{\mathsf{W}}_3$ as follows. The number of cards placed in the placement phase, whose suits are specified later, is extended from three to five based on the number of inputs. Moreover, the swapping of center and rightmost (resp., leftmost) cards performed by P_3 (resp., P_2, P_1) in step 3) (resp., steps 4) and 5)) of $\mathrm{MAJ}^{\mathsf{W}}_3$ is extended to the left cyclic shift over the four cards on the right (resp., the three cards on the left) of the card sequence performed by P_4, P_5 (resp., P_1, P_2, P_3) in step 2) (resp., step 3)) of $\mathrm{MAJ}^{\mathsf{A}}_5$.

Next, we consider the position of the (finally) output card at the end of the placement phase for each possible input to determine the operations in the placement phase. Concretely, we analyze which card is output in the card sequence at the end of each phase by retracing the position back from the output phase to after the placement phase. Then, from the analyzed position of the output card and the output value for corresponding inputs, the requirements, i.e., some cards' suits and positions, for the card sequence created in the placement phase is determined. Finally, $\mathrm{MAJ}^{\mathsf{A}}_5$ is constructed by deducing the operations in the placement phase that satisfy the requirements.

Based on the above three phases, the specific placement phase of $\mathrm{MAJ}^{\mathsf{A}}_5$ is determined as follows. Here, we use ■ and □ to describe card sequences of $\mathrm{MAJ}^{\mathsf{A}}_5$, where ■ indicates the output card of $\mathrm{MAJ}^{\mathsf{A}}_5$, and □ indicates non-output cards in $\mathrm{MAJ}^{\mathsf{A}}_5$. First, since the leftmost card is output at the output phase, the card sequence of five cards at the output phase is represented by ■□□□□ for any $x_1, x_2, \ldots, x_5 \in \{0,1\}$. Analyzing the output card's position during the permutation phases 2 and 1, the card sequence changes as shown in Tables 2 and 3, respectively. Note that ■ with text below indicates the position of the output card for each case of $X := x_4 + x_5$ in Table 3. Then, we can specify the suits of the cards at the position of ■ from the values of $\sum_{i=1}^{3} x_i$ and X to compute maj_5 correctly. Table 4 shows the specified suits.

Finally, we consider the operations in the placement phase, which can create the card sequence satisfying the requirements shown in Table 4. The requirements for the right three cards of the sequences in Table 4 can be satisfied by firstly putting ♡♣♣ in this order and by not moving them during the placement phase. In addition, the requirement for the left two cards of the card sequences can be satisfied by putting ♣♡ in this order and by swapping these two cards (or doing a left cyclic shift over these cards) $\sum_{i=1}^{3} x_i$ times. Therefore, the operations of the placement phase of $\mathrm{MAJ}^{\mathsf{A}}_5$ are determined as operations such that, for pre-put face-down cards ♣♡♡♣♣, each player P_i $(i = 1, 2, 3)$ swaps the left two cards of the card sequence if $x_i = 1$; otherwise, P_i does nothing. The specific card sequences created in the placement phase are shown in Table 5.

3.2.2 Not-Working Case: $n = 3$

$\mathrm{MAJ}^{\mathsf{A}}_3$ cannot compute maj_3 and is different from $\mathrm{MAJ}^{\mathsf{W}}_3$.

Especially, step 1) in $\mathrm{MAJ}_3^{\mathsf{A}}$ does not work well because the cards to be manipulated become the leftmost one of the three cards. (See step 1) in Protocol 2 with $m = 1$.) As a result, the card sequence put in the setup is not changed in step 1) and the card sequence after step 1) does not satisfy the requirements to compute maj_3 correctly. In this sense, $\mathrm{MAJ}_n^{\mathsf{A}}$ is not a generalized protocol of $\mathrm{MAJ}_3^{\mathsf{W}}$, although $\mathrm{MAJ}_n^{\mathsf{A}}$ is constructed by extending the number of cards and swapping operations in $\mathrm{MAJ}_3^{\mathsf{W}}$.

4. Extension to n-Input Majority Voting Protocol

In this section, we describe another construction of an n-input majority voting protocol, denoted by MAJ_n.

4.1 Construction Idea

We construct MAJ_n similarly to $\mathrm{MAJ}_n^{\mathsf{A}}$. Namely, we first fix MAJ_n as consisting of three phases: placement, permutation, and output. The operations in the permutation and output phases are also fixed. Then, we analyze the position of the output card at the end of the placement phase for each input. Finally, we determine the operations in the placement phase. The difference between the constructions of $\mathrm{MAJ}_n^{\mathsf{A}}$ and MAJ_n is how to fix the operations in the permutation phase. The permutation phase of $\mathrm{MAJ}_n^{\mathsf{A}}$ is fixed extending the operation of $\mathrm{MAJ}_3^{\mathsf{W}}$. On the other hand, the permutation phase of MAJ_n is fixed based on the following representation of maj_n:

$$\mathsf{maj}_n(x_1, x_2, \ldots, x_n) = \begin{cases} \mathsf{maj}_{n-1}(x_1, x_2, \ldots, x_{n-1}) & \text{if } \sum_{i=1}^{n-1} x_i \neq \lfloor n/2 \rfloor, \\ x_n & \text{if } \sum_{i=1}^{n-1} x_i = \lfloor n/2 \rfloor. \end{cases} \tag{4}$$

Note that Eq. (4) is a general expression of Eq. (3) since $\mathsf{and}_2(x_1, x_2) = \mathsf{maj}_2(x_1, x_2)$ holds for the cases where $x_1 + x_2 \neq 1$ holds.

$\mathrm{MAJ}_3^{\mathsf{W}}$ is constructed based on an and_2 protocol. After computing $\mathsf{and}_2(x_1, x_2)$, the operations are performed that output x_3 only when $x_1 + x_2 = 1$. In contrast, we construct MAJ_n based on the operations in the permutation phase for outputting x_n only when $\sum_{i=1}^{n-1} x_i = \lfloor n/2 \rfloor$. Concretely, we first fix such operations to output x_n as the permutation phase. Then, we specify operations in the placement phase to output $\mathsf{maj}_{n-1}(x_1, x_2, \ldots, x_{n-1})$, taking care of the subsequent operations in the permutation phase.

4.2 MAJ_5: Five-Input Majority Voting Protocol

Let us construct MAJ_5 as a simple but essential example.

4.2.1 Three Phases of MAJ_5

We first fix the procedure of MAJ_5 as consisting of the following three phases.

Placement phase: P_1, P_2, P_3, P_4 place all of four cards face-down and perform private permutations to generate a card sequence depending on their inputs. *Concrete operations are determined later.*

Permutation phase: Each player P_i ($i = 1, 2, \ldots, 5$) performs private permutations depending on the player's input. Specifically, this phase consists of the following two sub-phases.

- **Permutation phase 1:** P_5 performs a left cyclic shift over the two cards on the right (or swaps right two cards) of the card sequence if $x_i = 0$; otherwise, P_5 does nothing.
- **Permutation phase 2:** Each player P_i ($i = 1, 2, 3, 4$) performs a left cyclic shift over the three cards on the left of the card sequence if $x_i = 0$; otherwise, P_i does nothing.

Output phase: The protocol outputs the leftmost card of the card sequence.

Note that the above procedure is fixed in order to output x_5 only when $\sum_{i=1}^{4} x_i = 2 (= \lfloor 5/2 \rfloor)$, as Eq. (4). Concretely, the number of cards and the operations in the permutation phases are determined based on the following considerations.

Suppose that we use k cards. After we determine the operations in the permutation phases 1 and 2, we specify the value of k. To output x_5 only when $\sum_{i=1}^{4} x_i = 2$, we suppose the operations in the permutation phases 1 and 2 as follows: in the permutation phase 1, the card representing x_5 is positioned at the second from the right (i.e., the $(k-1)$-th from the left) of the card sequence when $\sum_{i=1}^{4} x_i = 2$, and in the permutation phase 2, the $(k-1)$-th card is moved to the leftmost position only when $\sum_{i=1}^{4} x_i = 2$.

Specifically, in the permutation phase 1, the right two cards of the card sequence are swapped if $x_5 = 0$; otherwise, nothing is done. (cf. Step 3) in $\mathrm{MAJ}_3^{\mathsf{W}}$.) In the permutation phase 2, each player P_i ($i = 1, 2, 3, 4$) performs a left cyclic shift over the left $k-1$ cards of the sequence if $x_i = 0$; otherwise, nothing is done. (cf. Steps 4) and 5) in $\mathrm{MAJ}_3^{\mathsf{W}}$. Swapping the two cards is extended to a left cyclic shift over the $k-1$ cards.) Through the permutation phase 2, the $(k-1)$-th card is shifted left $\sum_{i=1}^{4} x_i$ times. Therefore, if $\sum_{i=1}^{4} x_i = k-2$ holds, the $(k-1)$-th card moves to the leftmost position. Since we want to move the $(k-1)$-th card to the leftmost position only when $\sum_{i=1}^{4} x_i = 2$, we should use four cards so that $2 = \sum_{i=1}^{4} x_i = k-2$, i.e., $k = 4$ holds.

4.2.2 Specification of MAJ_5

Next, we deduce the operations in the placement phase based on the procedure fixed in Sect. 4.2.1. Similarly to what we see in Sect. 3.2.1, we analyze the position of the output card. At the output phase, the sequence of four cards is represented by for any $x_1, x_2, \ldots, x_5 \in \{0, 1\}$. Then, the card sequence changes during the permutation phases 2 and 1 as shown in Tables 6 and 7, respectively. From the values of

Table 6 Card sequences after the permutation phase 1 in MAJ_5.

$\sum_{i=1}^{4} x_i$	Card Sequence
0	□ ■ □ □
1	■ □ □ □
2	□ □ ■ □
3	□ ■ □ □
4	■ □ □ □

Table 7 Card sequences after the placement phase in MAJ_5.

$\sum_{i=1}^{4} x_i$	Card Sequence
0	□ ■ □ □
1	■ □ □ □
2	□ □ ■ (if $x_5 = 1$) ■ (if $x_5 = 0$)
3	□ ■ □ □
4	■ □ □ □

Table 8 Requirements to compute maj_5 correctly.

$\sum_{i=1}^{4} x_i$	Card Sequence
0	□ ♣ □ □
1	♣ □ □ □
2	□ □ ♡ ♣
3	□ ♡ □ □
4	♡ □ □ □

Table 9 Card sequences resulting of the placement phase in MAJ_5.

$\sum_{i=1}^{4} x_i$	Card Sequence
0	♡ ♣ ♣ ♣
1	♣ ♣ ♣ ♡
2	♣ ♣ ♡ ♣
3	♣ ♡ ♣ ♣
4	♡ ♣ ♣ ♣

Protocol 3 MAJ_5: Five-input Majority Voting Protocol

Inputs: $P_1, P_2, \ldots, P_5$ input $x_1, x_2, \ldots, x_5 \in \{0, 1\}$, respectively.
Setup: Publicly put four face-down cards ♡♣♣♣ in this order. Then, P_1 receives these $2m + 1$ cards.

1) From $i = 1$ to $i = 4$, if $x_i = 0$, then P_i does a right cyclic shift over the received cards; otherwise, P_i does nothing. Then, P_i sends the cards to P_{i+1}.
2) If $x_5 = 0$, P_5 does a left cyclic shift over the two cards from the right of the received cards; otherwise, P_5 does nothing. Then, P_5 sends the cards to P_1.
3) From $i = 1$ to $i = 4$, if $x_i = 0$, P_i does a left cyclic shift over the four cards from the left of the received cards; otherwise, P_i does nothing. Then, if $i \neq 4$, P_i sends the cards to P_{i+1}. If $i = 4$, P_4 opens the leftmost card of the four cards, and discards the others without opening.

Output: If the card P_4 opens at step 3) is ♣, the output is 0; otherwise (if it is ♡), the output is 1.

$\sum_{i=1}^{4} x_i$ and x_5 in Table 7, we can specify the suits of the cards at the position of ■, which indicate the requirement to compute maj_5 correctly, and are summarized as Table 8.

Finally, we determine the operations in the placement phase satisfying the requirement. Observing the bottom three rows of Table 8, we can see that the position of ♡ shifts one place to the right for every one decrease in the weight $\sum_{i=1}^{4} x_i$. Thus, we come up with the following operations as the placement phase. First, suppose that four cards ♡♣♣♣ are put in this order. Then, each player P_i $(i = 1, 2, 3, 4)$ does a right cyclic shift over the card sequence of the four cards if $x_i = 0$; otherwise, P_i does nothing. The card sequences generated by these operations for each input are shown in Table 9. Since these sequences satisfy the requirements shown in Table 8, we determine the operations in the placement phase as the above operations over a pre-put card sequence ♡♣♣♣. We show the constructed protocol MAJ_5 in Protocol 3.

4.3 MAJ_{2m+1}: $(2m + 1)$-Input Majority Voting Protocol

Generalizing the construction of MAJ_5, we construct an n-input majority voting protocol for odd $n = 2m + 1$ $(m \geq 1)$, denoted by MAJ_{2m+1}. First, we fix MAJ_{2m+1} as consisting of the following three phases.

Placement phase: $P_1, P_2, \ldots, P_{2m}$ place all of $m + 2$ cards face-down and perform private permutations to generate a card sequence depending on their inputs.

Permutation phase: Each player P_i $(i = 1, 2, \ldots, 2m + 1)$ performs private permutations depending on the player's input. Specifically, this phase consists of the following two sub-phases.

- **Permutation phase 1:** P_{2m+1} performs a left cyclic shift over the two cards on the right (or swaps the right two cards) of the card sequence if $x_{2m+1} = 0$; otherwise, P_{2m+1} does nothing.
- **Permutation phase 2:** Each player P_i $(i = 1, 2, \ldots, 2m)$ performs a left cyclic shift over the $m + 1$ cards on the left of the card sequence if $x_i = 0$; otherwise, P_i does nothing.

Output phase: The protocol outputs the leftmost card of the card sequence.

The number of cards and the operations in the permutation phase are fixed in order to output x_n only when $\sum_{i=1}^{n-1} x_i = \lfloor n/2 \rfloor$, similar to Sect. 4.2.1.

Suppose that we use k cards. In the permutation phase 1, the right two cards of the card sequence are swapped if $x_n = 0$; otherwise, nothing is done. Then, the card representing x_n is positioned at the $(k - 1)$-th from the left of the card sequence when $\sum_{i=1}^{n-1} x_i = \lfloor n/2 \rfloor$. In the permutation phase 2, each player P_i $(i = 1, 2, \ldots, 2m)$ performs a left cyclic shift over the left $k - 1$ cards of the sequence if $x_i = 0$; otherwise, nothing is done. Through the permutation phase 2, the $(k - 1)$-th card from the left is shifted left $\sum_{i=1}^{n-1} x_i$ times. Thus, if $\sum_{i=1}^{n-1} x_i = k - 2$ holds, the $(k - 1)$-th card moves to the leftmost position. Since we want to move the $(k - 1)$-th card to the leftmost position only when $\sum_{i=1}^{n-1} x_i = \lfloor n/2 \rfloor$, we should use $m + 2$ cards so that $m = \lfloor n/2 \rfloor = \sum_{i=1}^{n-1} x_i = k - 2$, i.e., $k = m + 2$ holds.

Then, we determine the operations in the placement phase. Table 10 shows the positions of the output card retraced from the output phase to after the placement phase. From the value of $\sum_{i=1}^{2m} x_i$ and x_{2m+1}, the requirements to compute maj_{2m+1} correctly are specified, shown in Table 11. Concerning the operations in the placement phase, creating a card sequence satisfying the requirement in Table 11 is possible using equivalent operations as in MAJ_5. As a result, we obtain Table 12 and Protocol 4 shows the constructed protocol MAJ_{2m+1} from above discussion. Note that the numbers in the header rows of the tables after Table 10

Table 10 Card sequences after the placement phase in MAJ_{2m+1}.

$\sum_{i=1}^{2m} x_i$	Card Sequence						
	1	2	$\cdots$	$m-1$	m	$m+1$	$m+2$
0	□	□	$\cdots$	□	■	□	□
1	□	□	$\cdots$	■	□	□	□
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m-1$	■	□	$\cdots$	□	□	□	□
m	□	□	$\cdots$	□	□	■ if $x_{2m+1}=1$	■ if $x_{2m+1}=0$
$m+1$	□	□	$\cdots$	□	■	□	□
$m+2$	□	□	$\cdots$	■	□	□	□
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2m$	■	□	$\cdots$	□	□	□	□

Table 11 Requirements to compute maj_{2m+1} correctly.

$\sum_{i=1}^{2m} x_i$	Card Sequence						
	1	2	$\cdots$	$m-1$	m	$m+1$	$m+2$
0	□	□	$\cdots$	□	♣	□	□
1	□	□	$\cdots$	♣	□	□	□
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m-1$	♣	□	$\cdots$	□	□	□	□
m	□	□	$\cdots$	□	□	♡	♣
$m+1$	□	□	$\cdots$	□	♡	□	□
$m+2$	□	□	$\cdots$	♡	□	□	□
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2m$	♡	□	$\cdots$	□	□	□	□

Table 12 Card sequences resulting of the placement phase in MAJ_{2m+1}.

$\sum_{i=1}^{2m} x_i$	Card Sequence						
	1	2	$\cdots$	$m-1$	m	$m+1$	$m+2$
0	♣	♣	$\cdots$	♡	♣	♣	♣
1	♣	♣	$\cdots$	♣	♣	♣	♣
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$m-1$	♣	♣	$\cdots$	♣	♣	♣	♡
m	♣	♣	$\cdots$	♣	♣	♡	♣
$m+1$	♣	♣	$\cdots$	♣	♡	♣	♣
$m+2$	♣	♣	$\cdots$	♡	♣	♣	♣
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2m$	♡	♣	$\cdots$	♣	♣	♣	♣

indicate the cards' positions in the card sequence.

4.4 MAJ_{2m}: $2m$-Input Majority Voting Protocol

We can construct an n-input majority voting protocol for even $n = 2m$ ($m \geq 2$), denoted by MAJ_{2m}, in the same way as MAJ_{2m+1}. The requirements in the construction of MAJ_{2m}, corresponding to Table 11, differ only slightly from those of MAJ_{2m+1}. However, The rest of the construction process is the same. Therefore, the detail of the construction of MAJ_{2m} are provided in Appendix.

Protocol 4 MAJ_{2m+1}: $(2m+1)$-input Majority Voting Protocol

Inputs: $P_1, P_2, \ldots, P_{2m+1}$ input $x_1, x_2, \ldots, x_{2m+1} \in \{0, 1\}$, respectively.

Setup: Publicly put $m+2$ face-down cards $\heartsuit \overbrace{\clubsuit \cdots \clubsuit}^{m+1}$ in this order. Then, P_1 receives these $m+2$ cards.

1) From $i = 1$ to $i = 2m$, if $x_i = 0$, then P_i does a right cyclic shift over the received cards; otherwise, P_i does nothing. Then, P_i sends the cards to P_{i+1}.
2) If $x_{2m+1} = 0$, P_{2m+1} does a left cyclic shift over the two cards from the right of the received cards; otherwise, P_{2m+1} does nothing. Then, P_{2m+1} sends the cards to P_1.
3) From $i = 1$ to $i = 2m$, if $x_i = 0$, P_i does a left cyclic shift over the $m+1$ cards from the left of the received cards; otherwise, P_i does nothing. Then, if $i \neq 2m$, P_i sends the cards to P_{i+1}. If $i = 2m$, P_{2m} opens the leftmost card of the $m+2$ cards, and discards the others without opening.

Output: If the card P_{2m} opens at step 3) is ♣, the output is 0; otherwise (if it is ♡), the output is 1.

4.5 MAJ_n: n-Input Majority Voting Protocol

For positive integers n and m, the following holds.

$$\lceil n/2 \rceil = \begin{cases} m+1 & \text{if } n = 2m+1 \\ m & \text{if } n = 2m \end{cases} \tag{5}$$

Therefore, MAJ_{2m+1} and MAJ_{2m} can be unified as Protocol 5.

Let us check the efficiency measures of MAJ_n. The number of cards is $\lceil n/2 \rceil + 1$. The number of private permutations is $2n-1$ ($n-1, 1$, and $n-1$ for steps 1), 2), and 3), respectively). The number of communications is $2n-2$ since the card sequence is send in the order of $P_1 \to P_2 \to \ldots \to P_n \to P_1 \to \cdots \to P_{n-1}$.

4.6 Correctness and Privacy of MAJ_n

The correctness of MAJ_n is obvious since the card sequences created in the placement phase shown in Tables 12 and A·3 satisfy the requirements shown in Tables 11 and A·2, respectively. In addition, we can see that MAJ_n does not reveal any information about input except what leaks from the output because of the following two reasons. First, the card sequence is always face-down after the setup until the output phase. Since the sequence is created not depending on any input and the backs of all cards are indistinguishable, no information leaks during executing MAJ_n. Second, the non-output cards are discarded without opening. Therefore, no player can know the information except the output.

5. Conclusion

We showed an n-input majority voting protocol (denoted by

Protocol 5 MAJ_n: n-input Majority Voting Protocol

Inputs: $P_1, P_2, \ldots, P_n$ input $x_1, x_2, \ldots, x_n \in \{0, 1\}$, respectively.

Setup: Publicly put $\lceil n/2 \rceil + 1$ face-down cards $\heartsuit\overbrace{\clubsuit\cdots\clubsuit}^{\lceil n/2 \rceil}$ in this order. Then, P_1 receives these $\lceil n/2 \rceil + 1$ cards.

1) From $i = 1$ to $i = n - 1$, if $x_i = 0$, then P_i does a right cyclic shift over the received cards; otherwise, P_i does nothing. Then, P_i sends the cards to P_{i+1}.

2) If $x_n = 0$, P_n does a left cyclic shift over the two cards from the right of the received cards; otherwise, P_n does nothing. Then, P_n sends the cards to P_1.

3) From $i = 1$ to $i = n - 1$, if $x_i = 0$, P_i does a left cyclic shift over the $\lceil n/2 \rceil$ cards from the left of the received cards; otherwise, P_i does nothing. Then, if $i \neq n - 1$, P_i sends the cards to P_{i+1}. If $i = n - 1$, P_{n-1} opens the leftmost card of the $\lceil n/2 \rceil + 1$ cards, and discards the others without opening.

Output: If the card P_{n-1} opens at step 3) is $\clubsuit$, the output is 0; otherwise (if it is $\heartsuit$), the output is 1.

MAJ_n), which uses only $\lceil n/2 \rceil + 1$ cards and $2n - 1$ private permutations. Our protocol MAJ_n requires the smallest number of cards among the n-input majority voting protocols using linear number of private permutations. Similar to Abe et al.'s work [15] (denoted $\mathrm{MAJ}_n^{\mathsf{A}}$), our protocol MAJ_n is constructed by extending Watanabe et al.'s work [8] (denoted by $\mathrm{MAJ}_3^{\mathsf{W}}$). However, MAJ_n includes $\mathrm{MAJ}_3^{\mathsf{W}}$ as a special case, different from $\mathrm{MAJ}_n^{\mathsf{A}}$.

MAJ_n is constructed as follows: (a) fix a part of its procedure based on $\mathrm{MAJ}_3^{\mathsf{W}}$; and (b) deduce the rest of the procedure based on (a). This construction method is identical to Abe et al.'s work. Unlike Abe et al.'s work, we fixed the procedures based on the following equation, which can be derived naturally from $\mathrm{MAJ}_3^{\mathsf{W}}$:

$$\mathsf{maj}_n(x_1, x_2, \ldots, x_n) = \begin{cases} \mathsf{maj}_{n-1}(x_1, x_2, \ldots, x_{n-1}) & \text{if } \sum_{i=1}^{n-1} x_i \neq \lfloor n/2 \rfloor, \\ x_n & \text{if } \sum_{i=1}^{n-1} x_i = \lfloor n/2 \rfloor. \end{cases}$$

As a result, we succeed to reduce the number of cards by half compared with $\mathrm{MAJ}_n^{\mathsf{A}}$ due to the benefit of achieving the generalization. To the best of our knowledge, MAJ_n is the first efficient protocol to realize the n-input majority voting with less than n cards.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers JP22J10137, JP22H03590, JP21H03395, JP20J21248, JP18H05289, and JP18K11293 and by MEXT Leading Initiative for Excellent Young Researchers.

References

[1] A.C. Yao, "Protocols for secure computations (extended abstract)," 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, Nov. 3–5, 1982, pp.160–164, IEEE Computer Society, 1982.

[2] B. den Boer, "More efficient match-making and satisfiability: *The Five Card Trick*," Advances in Cryptology – EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10–13, 1989, Proceedings, J. Quisquater and J. Vandewalle, eds., Lecture Notes in Computer Science, vol.434, pp.208–217, Springer, 1989.

[3] C. Crépeau and J. Kilian, "Discreet solitary games," Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, Aug. 22–26, 1993, Proceedings, D.R. Stinson, ed., Lecture Notes in Computer Science, vol.773, pp.319–330, Springer, 1993.

[4] T. Mizuki and H. Sone, "Six-card secure AND and four-card secure XOR," Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 20–23, 2009. Proceedings, X. Deng, J.E. Hopcroft, and J. Xue, eds., Lecture Notes in Computer Science, vol.5598, pp.358–369, Springer, 2009.

[5] A. Marcedone, Z. Wen, and E. Shi, "Secure dating with four or fewer cards," IACR Cryptol. ePrint Arch., vol.2015, p.1031, 2015.

[6] T. Nakai, Y. Tokushige, Y. Misawa, M. Iwamoto, and K. Ohta, "Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations," Cryptology and Network Security – 15th International Conference, CANS 2016, Milan, Italy, Nov. 14–16, 2016, Proceedings, S. Foresti and G. Persiano, eds., Lecture Notes in Computer Science, vol.10052, pp.500–517, 2016.

[7] T. Mizuki and H. Shizuya, "Practical card-based cryptography," Fun with Algorithms - 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1–3, 2014. Proceedings, A. Ferro, F. Luccio, and P. Widmayer, eds., Lecture Notes in Computer Science, vol.8496, pp.313–324, Springer, 2014.

[8] Y. Watanabe, Y. Kuroki, S. Suzuki, Y. Koga, M. Iwamoto, and K. Ohta, "Card-based majority voting protocols with three inputs using three cards," International Symposium on Information Theory and Its Applications, ISITA 2018, Singapore, Oct. 28–31, 2018, pp.218–222, IEEE, 2018.

[9] K. Shinagawa, "Deterministic cryptographic protocols with active security using a deck of cards, envelops and chains," 2018 Symposium on Cryptography and Information Security, pp.3B1–3, 2018.

[10] H. Ono and Y. Manabe, "Card-based cryptographic logical computations using private operations," New Generat. Comput., vol.39, no.1, pp.19–40, 2021.

[11] T. Nakai, S. Shirouchi, Y. Tokushige, M. Iwamoto, and K. Ohta, "Secure computation for threshold functions with physical cards: Power of private permutations," New Generat. Comput., vol.40, no.1, pp.95–113, 2022.

[12] H. Ono and Y. Manabe, "Efficient card-based cryptographic protocols for the millionaires' problem using private input operations," 2018 13th Asia Joint Conference on Information Security (AsiaJCIS), pp.23–28, 2018.

[13] T. Nakai, Y. Misawa, Y. Tokushige, M. Iwamoto, and K. Ohta, "How to solve millionaires' problem with two kinds of cards," New Generat. Comput., vol.39, no.1, pp.73–96, 2021.

[14] D.A.M. Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1," J. Comput. Syst. Sci., vol.38, no.1, pp.150–164, 1989.

[15] Y. Abe, T. Nakai, Y. Kuroki, S. Suzuki, Y. Koga, Y. Watanabe, M. Iwamoto, and K. Ohta, "Efficient card-based majority voting protocols," New Generat. Comput., vol.40, no.1, pp.173–198, 2022.

[16] Y. Manabe and H. Ono, "Card-based cryptographic protocols with malicious players using private operations," New Generat Comput., vol.40, no.1, pp.67–93, 2022.

[17] A. Koch and S. Walzer, "Foundations for actively secure card-based cryptography," 10th International Conference on Fun with Algorithms, FUN 2021, May 30–June 1, 2021, Favignana Island, Sicily, Italy, M. Farach-Colton G. Prencipe, and R. Uehara, ed., LIPIcs, vol.157, pp.17:1–17:23, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

[18] Y. Abe, M. Iwamoto, and K. Ohta, "How to detect malicious be-

haviors in a card-based majority voting protocol with three inputs," International Symposium on Information Theory and Its Applications, ISITA 2020, Kapolei, HI, USA, Oct. 24–27, 2020, pp.377–381, IEEE, 2020.

Appendix: MAJ_{2m}: $2m$-Input Majority Voting Protocol

First, we consider that MAJ_{2m} consists of the following three phases.

Placement phase: $P_1, P_2, \ldots, P_{2m-1}$ place all of $m+1$ cards face-down and perform private permutations to generate a card sequence depending on their inputs.

Permutation phase: Each player P_i $(i = 1, 2, \ldots, 2m)$ performs private permutations depending on the player's input. Specifically, this phase consists of the following two sub-phases.

- **Permutation phase 1:** P_{2m} performs a left cyclic shift over the two cards on the right (or swaps the right two cards) of the card sequence if $x_{2m} = 0$; otherwise, P_{2m} does nothing.
- **Permutation phase 2:** Each player P_i $(i = 1, 2, \ldots, 2m-1)$ performs a left cyclic shift over the m cards on the left of the card sequence if $x_i = 0$; otherwise, P_i does nothing.

Output phase: The protocol outputs the leftmost card of the card sequence.

Then, we determine the operations in the placement phase. Table A· 1 shows the positions of the output card retraced from the output phase to after the placement phase. Note that there exists the different part from the case of MAJ_{2m+1} when $\sum_{i=1}^{2m-1} x_i = 0$. Namely, the position of the output card changes depending on the value of x_{2m}. Therefore, the requirements to compute MAJ_{2m} correctly, shown in Table A· 2, also differ from those of MAJ_{2m+1}. However, we can generate the card sequences shown in Table A· 3 that satisfy the requirements by the equivalent operations in the placement phase of MAJ_{2m+1}. The constructed protocol is shown in Protocol A.

Table A· 1 Card sequences after the placement phase in MAJ_{2m}.

$\sum_{i=1}^{2m-1} x_i$	Card Sequence 1	2	⋯	$m-2$	$m-1$	m	$m+1$
0	□	□	⋯	□	□	■ if $x_{2m}=1$	■ if $x_{2m}=0$
1	□	□	⋯	□	■	□	□
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮
$m-1$	■	□	⋯	□	□	□	□
m	□	□	⋯	□	□	■ if $x_{2m}=1$	■ if $x_{2m}=0$
$m+1$	□	□	⋯	□	■	□	□
$m+2$	□	□	⋯	■	□	□	□
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮
$2m-1$	■	□	⋯	□	□	□	□

Table A· 2 Requirements to compute maj_{2m} correctly.

$\sum_{i=1}^{2m-1} x_i$	Card Sequence 1	2	⋯	$m-2$	$m-1$	m	$m+1$
0	□	□	⋯	□	□	♣	♣
1	□	□	⋯	□	♣	□	□
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮
$m-1$	♣	□	⋯	□	□	□	□
m	□	□	⋯	□	□	♡	♣
$m+1$	□	□	⋯	□	♡	□	□
$m+2$	□	□	⋯	♡	□	□	□
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮
$2m-1$	♡	□	⋯	□	□	□	□

Table A· 3 Card sequences resulting of the placement phase in MAJ_{2m}.

$\sum_{i=1}^{2m-1} x_i$	Card Sequence 1	2	⋯	$m-2$	$m-1$	m	$m+1$
0	♣	♣	⋯	♣	♡	♣	♣
1	♣	♣	⋯	♡	♣	♣	♣
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮
$m-1$	♣	♣	⋯	♣	♣	♣	♡
m	♣	♣	⋯	♣	♣	♡	♣
$m+1$	♣	♣	⋯	♣	♡	♣	♣
$m+2$	♣	♣	⋯	♡	♣	♣	♣
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮
$2m-1$	♡	♣	⋯	♣	♣	♣	♣

Protocol A MAJ_{2m}: $(2m)$-input Majority Voting Protocol

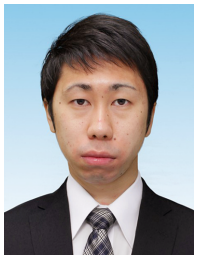
Inputs: $P_1, P_2, \ldots, P_{2m}$ input $x_1, x_2, \ldots, x_{2m} \in \{0, 1\}$, respectively.

Setup: Publicly put $m+1$ face-down cards $\heartsuit \overbrace{\clubsuit \cdots \clubsuit}^{m}$ in this order. Then, P_1 receives these $m+1$ cards.

1) From $i = 1$ to $i = 2m-1$, if $x_i = 0$, then P_i does a right cyclic shift over the received cards; otherwise, P_i does nothing. Then, P_i sends the cards to P_{i+1}.
2) If $x_{2m} = 0$, P_{2m} does a left cyclic shift over the two cards from the right of the received cards; otherwise, P_{2m} does nothing. Then, P_{2m} sends the cards to P_1.
3) From $i = 1$ to $i = 2m-1$, if $x_i = 0$, P_i does a left cyclic shift over the m cards from the left of the received cards; otherwise, P_i does nothing. Then, if $i \neq 2m-1$, P_i sends the cards to P_{i+1}. If $i = 2m-1$, P_{2m-1} opens the leftmost card of the $m+1$ cards, and discards the others without opening.

Output: If the card P_{2m-1} opens at step 3) is ♣, the output is 0; otherwise (if it is ♡), the output is 1.

Yoshiki Abe received the B.E. and M.E. degrees from the University of Electro-Communications in 2019 and 2021, respectively. He is currently a Ph.D. student and a JSPS research fellow in the University of Electro-Communications and also serves as Research Assistant at AIST.



Takeshi Nakai received the B.E., M.E., and Ph.D. degrees from the University of Electro-Communications in 2015, 2017, and 2021, respectively. He is currently Assistant Professor at Toyohashi University of Technology. His research interests include cryptography and information security. He is a member of IEICE, IPSJ, and IACR.

Yohei Watanabe received the B.E., M.E., and Ph.D. degrees in information science from Yokohama National University in 2011, 2013, and 2016, respectively. He is currently Assistant Professor at the University of Electro-Communications, and also serves as Invited Adviser at NICT and Collaborative Researcher at AIST. His research interests include cryptography and information security. He is a member of IEICE, IPSJ, IEEE, and IACR.

Mitsugu Iwamoto received the B.E., M.E., and Ph.D. degrees from the University of Tokyo, Tokyo, Japan, in 1999, 2001, and 2004, respectively. In 2004, he joined the University of Electro-Communications, where he is currently a Professor of Department of Informatics. His research interests include information theory, information security, and cryptography. He is a member of IEICE, IEEE, and IACR.

Kazuo Ohta received the B.S., M.S., and Dr.S. degree from Waseda University, Tokyo, Japan, in 1977, 1979, and 1990 respectively. He is an emeritus professor. He had been a Professor at The University of Electro-Communications between 2001 and 2020. He had been a researcher at NTT laboratories between 1979 and 2001. He is presently engaged in research on information security. He is a fellow of IEICE, and a member of IACR.