Efficient Card-Based Majority Voting Protocols

Yoshiki Abe · Takeshi Nakai · Yoshihisa Kuroki · Shinnosuke Suzuki · Yuta Koga · Yohei Watanabe · Mitsugu Iwamoto · Kazuo Ohta

Received: 3 September 2021 / Accepted: 8 February 2022 / Published online: 23 March 2022
© The Author(s) 2022, corrected publication 2022

Abstract
Card-based cryptography is a variety of secure multiparty computation (MPC). Recently, a new technique called private operations was introduced because the protocol can be implemented with fewer cards than that by using the conventional technique called the shuffle. For example, Nakai et al. showed that if the private operations are available, secure computations of AND and OR operations for two inputs can be realized simultaneously by using four cards, and the protocol is applied to a four-card majority voting protocol with three inputs. This paper shows that only three cards are sufficient to construct a majority voting protocol with three inputs. Specifically, we propose two constructions of three-input majority voting protocols. One is a protocol assuming that players can announce their output, and the other is not allowed. Compared to Nakai et al.'s protocol, the protocol with the announcement is realized without any additional private operations and communications. On the other hand, the second construction requires two more private operations and communications because it removes the assumption on the announcement from the first construction. More importantly, the idea of the second protocol can be extended to an $n$-input majority voting protocol with $n$ cards, which is the main result of this paper.

Keywords Multiparty computation · Card-based cryptography · Private operation · Majority voting protocol

Mathematics Subject Classification 94A60

A part of this work was done when Yohei Watanabe was at the National Institute of Information and Communications Technology, Japan. This work was supported by JSPS KAKENHI Grant Numbers JP17H01752, JP18H03238, JP18H05289, JP18K11293, JP18K19780, JP20J21248, and JP21H03395.

Yoshiki Abe
yoshiki@uec.ac.jp

1 Graduate School of Informatics and Engineering, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan

2 Cyber Physical Security Research Center (CPSEC), National Institute of Advanced Industrial Science and Technology (AIST), 2-3-26 Aomi, Koto-Ku, Tokyo 135-0064, Japan
Introduction

Background

Secure multiparty computation (MPC), introduced by Yao [12], allows players, each with input, to compute a function without revealing any information about their inputs other than what leaked from the output. MPC is usually realized using a computer, but den Boer showed that MPC of AND operation could be realized using five cards without using a computer [1]. Such MPC using physical cards like playing cards is called card-based cryptography.

Most of the existing works of card-based cryptography, e.g., [2,6], rely on a technique called a shuffle. In using the shuffle, we assume that cards are permuted in public repeatedly until all players cannot follow their order, for providing security. Note that the randomness in shuffles is assumed to be unknown to all players, including those who shuffle the deck of cards. Furthermore, since all shuffles are executed in public, a player who performs the shuffle cannot cheat. The shuffle is a technique involving such physical assumptions.

Instead of shuffles, Nakai et al. [8] and Marcedone et al. [4] independently introduced private permutations (or private operations), which are operations that a player permutes cards behind the player’s back to conceal the operation from the other players. Private operations allow players to use private randomness and capture local computation in (ordinary) secure multiparty computation. However, instead of removing the physical assumptions on shuffles, private operations are vulnerable to malicious behavior in the private area. Hence, in this paper, we assume that card-based cryptography with private operations is executed under the semi-honest model, i.e., a model in which every player always follows a protocol.¹

Although we have to assume the semi-honest model, private operations are useful because they make existing protocols simpler and more efficient. Hence, several card-based cryptographic protocols have been developed recently. For instance, the AND operation can be realized with only two or three cards (depending on the physical assumptions, e.g., players are allowed to announce their outputs) by using private operations [4], whereas the most efficient AND protocol using shuffles needs four cards (and relatively complicated procedures) [2]. The XOR and OR operations using private operations can be realized with two and three cards, respectively [7]. Furthermore, Nakai et al. [7] proposed a protocol that simultaneously computes AND and OR with four cards (called the AND/OR protocol) and showed that it could be used to construct a secure three-input majority voting protocol, which is a secure multiparty computation protocol that takes binary inputs from three players and outputs the result of majority voting.²

In the above protocols using private operations, formats of inputs and outputs are not always consistent. The protocols that have the same format of inputs and outputs are called commitment. Usually, a binary input is represented by a commitment using ¹ Several works for detecting malicious behavior have been proposed [3,10], but this research direction is outside the scope of this paper.
² We can classify majority voting protocols into two types: One outputs the total number of votes for 1 [5], and the other outputs whether votes for 1 have a majority or not [9]. We adopt the latter in this paper.
a pair of cards such as 0 $\mapsto$ ♣ and 1 $\mapsto$ ♥ unless an upside-down card is used. In any protocol with the commitment format, two cards are necessary for each input, so at least four cards are necessary for computing a function with two inputs.\(^3\) We note that when a binary input is represented by a commitment using a card such as 0 $\mapsto$ ♣ and 1 $\mapsto$ ♥, four cards are still needed to represent all possible inputs $x \in \{0, 1\}^2$. This is a disadvantage of the protocols with committed formats, but it has an advantage that such protocols are easy to be combined securely.

**Our Contributions**

*Proposal of majority voting protocols* In this paper, we show that three cards are sufficient to construct a three-input majority voting protocol by using private operations. Specifically, we first construct a protocol that simultaneously computes logical AND, XOR, and NOR operations (for two binary inputs), called an AND/XOR/NOR protocol, using only three cards.

We then show a three-input majority voting protocol (Protocol 3: MAJ\(_3\)wA) using our three-card AND/XOR/NOR protocol, whereas a three-input majority voting protocol is built on the AND/OR protocol by Nakai et al. [7]. The core idea of this construction is to allow players to announce their outputs to realize the OR operation based on the NOR operation. Namely, a player who gets the result of the NOR operation can announce the result of OR (not NOR), e.g., a player negates the value of NOR in the player’s mind and tells the negated result (the value of OR) to other players by voice. However, such an announcement is not desirable since additional cards or assumptions are required compared to the case where announcements are not used. Fortunately, it is possible to remove the announcement without increasing the number of cards at the cost of two additional private operations. We propose such a protocol (Protocol 4: MAJ\(_3\)). Furthermore, we can extend the three-input majority voting protocol (MAJ\(_3\)) to n-input majority voting protocols using n cards (gMAJ\(_n\)) shown in Protocols 6 and 7. Based on the observations of MAJ\(_3\), we assume some of the procedures of an n-input majority voting protocol. As a result, we can construct gMAJ\(_n\) by combining cyclic shifts of cards’ sequences.

*Efficiency comparisons* The proposed protocols in this paper is more efficient than the previous ones. We compare our protocols with those of Nakai et al. [7] in the case of $n = 3$, and with those of Ono and Manabe [10] for general $n$.

Table 1 compares efficiencies among Nakai et al.’s protocol and the proposed protocols MAJ\(_3\)wA and MAJ\(_3\). Compared with Nakai et al.’s protocol, the proposed protocols can reduce one card for computing three-input majority functions. In particular, MAJ\(_3\)wA (Protocol 3) increases neither the number of private operations nor the number of communications, though we allow Carol to announce the result. On the other hand, MAJ\(_3\) (Protocol 4) realizes three-input majority voting by only three cards without announcement. Instead, we need two more private operations and communications.

\(^3\) In [10], it is shown that any two-input logical functions can be computed with four cards, which means that four cards are optimal for computing two-input logical functions using commitment-format protocols.
Table 1  Comparison among three-input majority voting protocols

| Protocol             | #Priv. ops. | #Comm. | #Cards |
|----------------------|-------------|--------|--------|
| Nakai et al. [7]     | 3           | 2      | 4      |
| MAJ3wA (Protocol 3)  | 3           | 2      | 3      |
| MAJ3 (Protocol 4)    | 5           | 4      | 3      |

#Priv. ops., #Comm., and #Cards mean the number of private operations, communications, and cards, respectively

Table 2  Comparison among $n$-input majority voting protocols

| Protocol                        | #Priv. ops. | #Comm. | #Cards | Input           |
|---------------------------------|-------------|--------|--------|-----------------|
| Ono and Manabe [10]             | $O(2^n)$    | $O(2^n)$ | $2n + 4$ | Commitment     |
| Ono and Manabe [10]             | $O(2^n)$    | 2      | $2^{n+1}$ | Commitment     |
| gMAJ$_n$ (Protocols 6 and 7)    | $O(n)$      | $O(n)$ | $n$    | Non-commitment |

#Priv. ops., #Comm., and #Cards mean the number of private operations, communications, and cards, respectively

Table 2 shows a comparison among Ono and Manabe’s protocols [10] and proposed protocols gMAJ$_n$ (Protocols 6 and 7). Note that two Ono and Manabe’s protocols can compute any logical function with $n$ inputs, including $n$-input majority voting. Compared with the two Ono and Manabe’s protocols, the number of cards and private operations are lower in gMAJ$_n$. Regarding the number of communications, the second protocol of Ono and Manabe’s protocol is much more efficient than gMAJ$_n$. However, it requires an exponential number of cards.

Differences from the conference version  This paper is the full version of [11]. In this paper, [11] is extended by newly proposing gMAJ$_n$, $n$-input majority voting protocol, in Sect. 4 and Appendix A. In Sect. 4, the case where $n$ is odd is mainly described. Since the case where $n$ is even is almost the same as the case where $n$ is odd, the details of the case where $n$ is even are described in Appendix A.

Organization

The rest of this paper is organized as follows: in Sect. 2, we define notations, majority voting, and a left cyclic shift. Two three-input majority voting protocols are proposed in Sect. 3. The first one is the protocol with the announcement, and the second one is modified so that all operations are executed by cards only. Section 4 is devoted to extending the modified three-input majority voting protocol to an $n$-input majority voting protocol. We conclude this paper in Sect. 5. The details on the $n$-input majority voting protocol for even $n$, omitted in Sect. 4 due to similarities with the case of odd $n$, are given in Appendix A.
Preliminaries

Notations

We use two kinds of cards such as ♣ and ♥, where we assume that the cards with the same suit are indistinguishable. We also assume that all cards are indistinguishable if they are turned over, and they are denoted by ♦. For simplicity, we omit the frame of cards hereafter, i.e., ♠, ♡, and ♦ are used instead of using ♠, ♡, and ♦, respectively.

There are two models of card operations: a shuffle model and a private operation model. We adopt the latter model. According to [7,8], we design the proposed protocols based on the following operations of cards:

– **Permutation** Permutation of face-down cards in public.
– **Turn over** Face-down or open cards.
– **Private operation** Operations privately executed by a player, and their result is kept secret to the other players.
– **Announcement** Operations which make some information public. The information to be announced can be computed privately by the announcer, but that cannot be represented by the cards running the protocol.

Private operations are assumed to be executed easily, for instance, behind the player’s back. See [4,8] for details.

Majority Voting with Three Inputs

For three binary inputs \(a, b, c \in \{0, 1\}\), we define the majority voting for three inputs by:

\[
\text{maj}(a, b, c) = \begin{cases} 
0, & \text{if } a + b + c \leq 1 \\
1, & \text{if } a + b + c \geq 2,
\end{cases}
\] (1)

which is also represented as [7]

\[
\text{maj}(a, b, c) = \begin{cases} 
\text{\(a \land b\)}, & \text{if } c = 0 \\
\text{\(a \lor b\)}, & \text{if } c = 1.
\end{cases}
\] (2)

---

4 In [7,8], this operation is called private permutation. Private operation is named with non-permutation operations in mind (e.g., adding/removing cards), but operations that can be realized by private operations can also be realized by private permutations, so they can be considered the same. In the example of adding/removing cards, we can think of the addition/removal as increasing/decreasing the number of cards to be permuted in pre-putted cards.
Majority Voting with $n$-Inputs

For $n$ binary inputs $x_0, x_1, \ldots, x_{n-1} \in \{0, 1\}$, we define the threshold function for $n$ inputs by:

$$\text{maj}_n^t(x_0, x_1, \ldots, x_{n-1}) = \begin{cases} 0, & \text{if } \sum_{i=0}^{n-1} x_i < t \\ 1, & \text{if } \sum_{i=0}^{n-1} x_i \geq t \end{cases}$$

(3)

The ordinary majority voting is the case where $t = \lceil n/2 \rceil$.

Left Cyclic Shift

Suppose that there exists a sequence of $n$ cards, which is represented by $12 \ldots n$.

Note that numbers are written in order to specify the order of the cards.

Then, a left cyclic shift over the $n$ cards is defined as an operation such that the leftmost card of the sequence moves to the rightmost position of the sequence. The following shows the sequence of cards before and after a left cyclic shift of 5 cards.

$$12345 \rightarrow 23451.$$

A left cyclic shift can also be performed over a part of the card sequence. For example, if we perform a left cyclic shift over the left three cards out of five, we get the following.

$$12345 \rightarrow 23145.$$

The left cyclic shift has a feature that the order of the cards is restored after $m$ left cyclic shifts over $m$ cards.

Three-Input Majority Voting Protocol

In this section, we first propose a three-input majority voting protocol with announcement. This protocol is obtained from reducing a card from Nakai et al.’s four-card AND/OR protocol. Next, the three-input majority protocol is modified to be executable without announcement.
Protocol 1: Four-Card AND/OR Protocol [7]

Inputs: Alice has \(a\) \(\in\{0, 1\}\), and Bob has \(b\) \(\in\{0, 1\}\).
Setup: Both Alice and Bob have ♠ and ♥.

1) If \(a = 0\), then Alice sends face-down ♠ to Bob. Otherwise, she sends face-down ♥ to Bob.
2) If \(b = 0\), Bob places ♠ behind his back to the left of the card he received. Otherwise, he places ♥ behind his back to the right.

Outputs: The left card represents \(a \land b\), and the right card represents \(a \lor b\). If it is ♠, the output is 0; otherwise, 1.

Table 3: Combinations of cards after Step 2) in four-card AND/OR protocol [7]

| \(a\) | \(b\) | Combinations of cards after Step 2) |
|------|------|----------------------------------|
| 0    | 0    | ♠Bob ♠Alice                      |
| 0    | 1    | ♠Alice ♥Bob                      |
| 1    | 0    | ♠Bob ♥Alice                      |
| 1    | 1    | ♥Alice ♥Bob                      |

MAJ_{3wA}: Three-Input Majority Voting Protocol with Announcement

Previous Work: Four-card AND/OR Protocol

Assume that Alice, Bob, and Carol possess binary inputs \(a\), \(b\), and \(c\), respectively. Equation (2) implies that \(\text{maj}(a, b, c)\) can be securely computed by Carol if the results of \(a \land b\) and \(a \lor b\) are simultaneously computed securely, and she can choose one of them in her back privately depending on her bit \(c\). Specifically, for computing \(\text{maj}(a, b, c)\) securely, Carol chooses \(a \land b\) if \(c = 0\); otherwise, i.e., \(c = 1\), she chooses \(a \lor b\).

Based on this idea, Nakai et al. [7] proposed a four-card AND/OR protocol, which is described in Protocol 1. We note that this protocol requires four cards for representing all combinations of inputs, although only two cards are used to execute this protocol. Table 3 shows the combinations of cards after step 2) of this protocol for readers’ convenience, where the subscript indicates who has the card. This protocol is obtained by merging a three-card AND protocol [4] and a three-card OR protocol [7].

In Protocol 1, binary inputs are encoded as

\[
1 \leftrightarrow ♥, \quad 0 \leftrightarrow ♠, \quad \text{(4)}
\]

which is used throughout this paper.

Three-Card AND/XOR/NOR Protocol Based on Protocol 1, we propose a new card-based protocol to compute AND/NOR simultaneously instead of computing AND/OR. The proposed protocol is implemented by three cards, which is a result of reducing one card from Protocol 1. Furthermore, the proposed protocol can compute XOR of two binary inputs simultaneously with AND/NOR. The application of this protocol to majority voting will be discussed in Sect. 3.1.
Fig. 1 Difficulty of AND/OR protocol by three cards

Idea behind the protocol To see our idea, we try to construct a new AND/OR protocol by reducing a card from Protocol 1 at first. In Protocol 1, Alice has ♣ and ♥ in the setup. It seems hard to reduce the number of her cards because she has to securely input her bit by a face-down card. Hence, if we can reduce the number of cards in Protocol 1, the only way is to reduce one card from Bob. Which one of ♣ or ♥ should Bob have? To answer this question, observe the following properties (see also Fig. 1):

Case of $b = 0$: (a) It holds that $a \land b = b (= 0)$ regardless of the value of $a$.
(b) It holds that $a \lor b = a$.
Case of $b = 1$: (c) It holds that $a \land b = a$.
(d) It holds that $a \lor b = b (= 1)$ regardless of the value of $a$.

The properties (a)–(c) suggest Bob having ♣, and placing it to the leftmost of the cards he received from Alice if $b = 0$; otherwise, to the right. However, this approach obviously fails to realize (d) because $a \lor b = 1$ holds, whereas ♣ is placed on the right.

Considering the above discussion of the difficulty in realizing the AND/OR protocol with three cards, we alternatively show that a secure AND/NOR protocol is realizable with three cards as follows (see also Fig. 2): rewrite (b) and (d) as the equivalent statements such that:

(b’) It holds that $\neg(a \lor b) = \neg a$, and
(d’) It holds that $\neg(a \lor b) = \neg b (= 0)$ regardless of the value of $a$.

respectively. Recalling that ♣ indicates 0, properties (b’) and (d’) imply that it is possible to output $\neg(a \lor b)$ at the left of the cards if Alice prepares two cards $\spadesuit a$ in step 1) instead of one card $\spadesuit a$ in Fig. 1 and Bob places ♣ to the left if $b = 0$; otherwise, to the right.

Three-card AND/XOR/NOR protocol Observing Fig. 2, we find that the center of three cards is $a$ in the case of $b = 0$, whereas it is $\neg a$ in the case of $b = 1$. Noticing that $a = a \oplus 0$ and $\neg a = a \oplus 1$, the center of three cards indicates $a \oplus b$, where $\oplus$ denotes the XOR operation. Therefore, surprisingly, the AND/NOR protocol discussed in Fig. 2 also succeeds in computing XOR simultaneously. Hence, we call this protocol as a card-based AND/XOR/NOR protocol. The card-based AND/XOR/NOR protocol can be described in Protocol 2. We show the combinations of cards after the Step 2) in Table 4 for readers’ convenience.
Fig. 2  Three-card AND/XOR/NOR protocol
($\bar{x} := \neg x$ for $x \in \{0, 1\}$)

Protocol 2 Three-Card AND/XOR/NOR Protocol

**Inputs:** Alice has $a \in \{0, 1\}$, Bob has $b \in \{0, 1\}.$

**Setup:** Alice has ♣ and ♥. Bob has ♣.

1) Alice sends face-down ♣♥ to Bob if $a = 0$, otherwise, she sends face-down ♥♣ to Bob.
2) Behind his back, Bob places face-down ♣ to the left of the two cards he received. Otherwise, he places face-down ♣ to the right.

**Outputs:** From left to right, three cards are the results of AND, XOR, and NOR, where ♣ and ♥ indicate 0 and 1, respectively.

| $a$ | $b$ | Combinations of cards after the Step 2 |
|-----|-----|----------------------------------------|
| 0   | 0   | ♣Bob ♥Alice ♣Alice                      |
| 0   | 1   | ♣Bob ♥Alice ♥Alice ♣Bob                 |
| 1   | 0   | ♣Bob ♥Alice ♥Alice ♣Bob                 |
| 1   | 1   | ♥Alice ♥Alice ♣Bob                      |

MAJ$_3$wA: Three-Card Majority Voting with Announcement

Although we failed to construct a three-card AND/OR protocol, we succeeded to construct the three-card AND/XOR/NOR protocol (Protocol 2). Even if the AND/XOR/NOR protocol is used instead of a AND/OR protocol, three-input majority voting can be easily constructed by allowing Carol to announce.

If Carol inputs $c = 1$, she shall choose the card indicating $\neg(a \lor b)$ behind her back, and makes $a \lor b$ public by negating $\neg(a \lor b)$ in her mind. Making $a \lor b$ public is possible for Carol if she is allowed, for instance, to announce her output.\(^\text{5}\) It is obvious that she can obtain no information other than $a \lor b$ in this process.

One might wonder if this announcement leaks important information since Carol can see $a \land b$ (resp., $a \lor b$) if $c = 0$ (resp., $c = 1$). We would like to note that it does not violate security since Carol eventually obtains the information from the output of the majority voting.

\(^\text{5}\) Instead of announcing $a \lor b$, for instance, Carol can show the card indicating $a \lor b$ from extra cards.
Protocol 3 MAJ3wA: Three-Input Majority Voting Protocol with Announcement

Inputs: Alice inputs $a \in \{0, 1\}$, Bob inputs $b \in \{0, 1\}$, and Carol inputs $c \in \{0, 1\}$.

Setup: Alice has ♣ and ♥, Bob has ♣, and Carol has nothing at hand.

1) If $a = 0$, then Alice sends Bob face-down ♣♥ in this order. Otherwise, she sends face-down ♥♣ to Bob.

2) If $b = 0$, Bob places face-down ♣ at the left of the card pair he received. If $b = 1$, he places face-down ♥ at the right of the card pair. Then, he sends all face-down cards to Carol.

3) (Output) Carol opens the leftmost card behind her back if $c = 0$, and announces 1 (0) if the card she opens is ♥ (♣). If $c = 1$, she opens the rightmost card behind her back, and announces 1 (0) if the card she opens is ♣ (♥). Carol discard all cards in the deck.

Table 5 Three-card majority voting protocol with three inputs with announcement

| $a$ | $b$ | $c$ | Combinations of cards after the Step 2) | Output |
|-----|-----|-----|----------------------------------------|--------|
| 0   | 1   | 0   | ♣Alice ♥Alice ♣Bob                      | 0      |
| 1   | 0   | 0   | ♣Alice ♥Alice ♣Bob                      | 0      |
| 1   | 1   | 0   | ♥Alice ♣Alice ♣Bob                      | 1      |
| 0   | 1   | 1   | ♥Alice ♣Alice ♣Bob                      | 1      |
| 1   | 1   | 1   | ♥Alice ♣Alice ♣Bob                      | 1      |

In summary, the proposed protocol MAJ3wA is described in Protocol 3. For readers’ convenience, we show in Table 5 all combinations of cards after the step 2) of Protocol 3, where the underline indicates the card that Carol opens behind her back. In the case of $c = 1$, we can see that the bit represented by the card she opens and the output she announces are flipped.

MAJ3: Three-Input Majority Voting Protocol Without Announcement

In this subsection, we show how we modify MAJ3wA in the previous subsection so that all the operations (including the output) are represented by cards without increasing the number of cards.

Construction Idea

To construct a card-based majority voting protocol with three inputs without announcement, we take another look at Eq. (1) from the viewpoint of $a = b$ or not.

If $a = b$, it holds that either $(a, b) = (0, 0)$ or $(a, b) = (1, 1)$. In the case of $(a, b) = (0, 0)$, $\text{maj}(a, b, c) = 0 = a \land b$ regardless of the value of $c \in \{0, 1\}$. Similarly, we have $\text{maj}(a, b, c) = 1 = a \land b$, in the case of $(a, b) = (1, 1)$ regardless of the value of $c \in \{0, 1\}$. 
Protocol 4 MAJ3: Three-Input Majority Voting Protocol Without Announcement

Inputs: Alice inputs $a \in \{0, 1\}$, Bob inputs $b \in \{0, 1\}$, and Carol inputs $c \in \{0, 1\}$.

Setup: Alice has ♠ and ♥, Bob has ♠, and Carol has nothing at hand.

1) The same as Protocol 3-1).
2) The same as Protocol 3-2).
3) If $c = 0$, Carol swaps the center card for the right one behind her back; otherwise, she does nothing. Then, she sends them to Bob.
4) If $b = 0$, Bob swaps the center card for the left one behind his back; otherwise, he does nothing. Then, he sends them to Alice.
5) If $a = 0$, Alice swaps the center card for the left one behind her back; otherwise, she does nothing. Then, Alice opens the leftmost card, and discards the others.

Output: If the card Alice opens at Step 5 is ♠, the output is 0; otherwise (if it is ♥), the output is 1.

On the other hand, if $a \neq b$, it holds that $(a, b) = (1, 0)$ or $(a, b) = (0, 1)$, i.e., $a + b = 1$, which leads to $\text{maj}(a, b, c) = c$ from the definition of $\text{maj}(\cdot, \cdot, \cdot)$ given by (1).

In summary, (1) is equivalent to

$$\text{maj}(a, b, c) = \begin{cases} a \land b, & \text{if } a = b \text{(for any } c) \\ c, & \text{if } a \neq b. \end{cases}$$  \hspace{1cm} (6)

Suppose that Step 2) of MAJ3wA has been completed. Then, the leftmost card represents $a \land b$ (see also Protocol 2 and Fig. 2), which is equivalent to the output of $\text{maj}(a, b, c)$ in the case where $a = b$. Therefore, we additionally require the procedure that, the leftmost card is replaced by the card representing $c$ if $a \neq b$, or otherwise it is untouched. The following two steps are sufficient so as to realize this requirement:

Step A) Keeping the leftmost card representing $a \land b$, let the center card represent $c$ if $a \neq b$.
Step B) Swap the leftmost and the center cards if $a \neq b$.

For realizing step A), observe that $a \oplus b = 1$ and $\neg (a \lor b) = 0$ hold in the case of $a \neq b$, and, after the Step 2) of Protocol 3, the center and the rightmost cards represent $a \oplus b$ and $\neg (a \lor b)$, respectively, (see Protocol 2 and Fig. 2). Hence, Carol can realize Step A) without knowing the actual values of $a$ and $b$ by swapping the center card for the right one (behind her back) if $c = 0$, or does nothing otherwise.

Realization of Step B) is easy: it is sufficient that Alice and Bob successively swap the left card for the center card if and only if his/her input is 0, since the swap is done only once if and only if $a \neq b$ under this rule.

MAJ3: Three-Card Majority Voting for Three Inputs Without Announcement

Based on the construction idea, we construct MAJ3, a three-input majority voting protocol with three cards without any announce operations, which is described in Protocol 4. See also Fig. 3, where the question marks denote the face-down cards a player received from the previous player.
The outline of our construction approach is as follows. First, based on the procedure of MAJ₃, we assume gMAJₙ has three phases (called placement, permutation, and output). In addition, we also assume the operations in the permutation phase and the output phase. Then, we deduce the card sequence in the placement phase based on the assumed permutation and output phase.

**Our Construction Approach**

First, we divide the procedures of our three-input protocol MAJ₃ into three phases as follows.

- **Placement phase**: Alice and Bob place all of three cards face down and perform private operations to generate a card sequence depending on the value of a and b (cf., Steps 1–2)).
- **Permutation phase**: Each player performs private operations to permute a part of the card sequence (cf., Steps 3–5)). More concretely, these procedures can be organized into the following two sub-phases.
  - **Permutation phase 1**: At Step 3), Carol swaps the center card for the rightmost one if c = 0.
– **Permutation phase 2:** At Steps 4) and 5), Bob and Alice swap the center card for the leftmost one if \( b = 0 \) and \( a = 0 \), respectively.

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

Toward realizing \( n \)-input protocol \( g\text{MAJ}_n \), we abstract the above procedures as follows. Suppose there are two groups (or two sets) of players, \( \mathcal{P}_L \) and \( \mathcal{P}_S \), such that \( |\mathcal{P}_L| = |\mathcal{P}_S| + 1 \). 6 Let \( \text{op}_L \) and \( \text{op}_S \) be two distinct private operations. Then, the procedures of \( \text{MAJ}_3 \) can be abstracted as follows.

– **Placement phase:** All players in \( \mathcal{P}_L \) place all of three cards face down and perform private operations to generate a card sequence depending on their inputs.

– **Permutation phase:**

  – **Permutation phase 1:** Each player in \( \mathcal{P}_S \) performs the private operation \( \text{op}_S \) if the player’s input is zero.
  
  – **Permutation phase 2:** Each player in \( \mathcal{P}_L \) performs the private operation \( \text{op}_L \) if the player’s input is zero.

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

In the case of \( \text{MAJ}_3 \), Alice and Bob belong to the large group \( \mathcal{P}_L \) and Carol belongs to the small group \( \mathcal{P}_S \). \( \text{op}_L \) is the replacement of the center card with the leftmost one, and \( \text{op}_S \) is the replacement of the center card with the rightmost one.

We take the above constructing approach based on \( \text{MAJ}_3 \): during the placement phase, all players in \( \mathcal{P}_L \) place all cards face down in the some order; during the permutation phase, each player performs private operations (i.e., \( \text{op}_L \) or \( \text{op}_S \)) if the player’s input is zero; and during the output phase, the protocol outputs the leftmost card of the card sequence. Note that players belonging to the same group perform the same operation in the permutation phases 1 and 2. Therefore, what we need to determine to realize \( g\text{MAJ}_n \) is threefold:

– The concrete card sequence at the end of the placement phase.

– How the players in \( \mathcal{P}_L \) generate the card sequence during the placement phase.

– The concrete procedures of \( \text{op}_L \) and \( \text{op}_S \) in the permutation phase.

To determine the above three for \( g\text{MAJ}_n \), particularly for the first one (i.e., to figure out the concrete card sequence), we take a closer look at \( \text{MAJ}_3 \) in the next subsection. We here describe the overview. First, we analyze how the private operations in \( \text{MAJ}_3 \) affect the output of the protocol. Namely, we analyze where the output card (i.e., the leftmost card at the output phase) is placed at each step in \( \text{MAJ}_3 \). The analysis clarifies the minimum requirement of the card sequence at the end of the placement phase; it determines the suits and positions of some cards in the card sequence. In the subsection after that, we then extend the minimum requirement to the arbitrary-number input setting, and determines the suits of the rest of cards as well as the concrete procedures during the placement and permutation phases, respectively.

6 Although this formalization implicitly means that we will consider the case that \( n = |\mathcal{P}_L| + |\mathcal{P}_S| \) is always odd, we finally show a construction for even \( n \) by extending the construction for odd \( n \) in Sect. 4.6.
Table 6  Card sequences after the permutation phase 1 in MAJ₃

| a + b | Card sequence |
|-------|---------------|
| 0     | ■             |
| 1     | ■             |
| 2     | ■             |

Table 7  Card sequences before the permutation phase 1 in MAJ₃

| a + b | Card sequence |
|-------|---------------|
| 0     | ■             |
| 0     | ■             |
| 0     | ■             |

Observation About MAJ₃

For simplicity, we describe card sequences in MAJ₃ by ■ and □, where ■ denotes the output card of MAJ₃, i.e., the leftmost card at the output phase. Namely, the card sequence at the output phase (or the output sequence) is represented as “■ □ □.” Note that the output sequence (hence the output card) is always the above one for any a, b, c ∈ {0, 1}.

We begin with the analysis of the card sequences for any fixed a, b, c ∈ {0, 1} during the permutation phase. It is obvious that the card sequence after the permutation phase 2 (i.e., before the output phase) is “■ □ □” for any a, b, c ∈ {0, 1}.

How about the card sequence after Step 3), i.e., after the permutation phase 1? At Steps 4) and 5) (i.e., during the permutation phase 2), the leftmost card is swapped with the center one depending on the value of b and a, respectively. Therefore, we can summarize the card sequence after the permutation phase 1 in Table 6. If a + b = 1, the center card at the end of the permutation phase 1 will be used for the output of the protocol since the left and center cards are swapped once during the permutation phase 2; otherwise, the cards are swapped twice or not swapped at all during the permutation phase 2, so the leftmost card at the end of the permutation phase 1 will be used as the output card.

Next, to determine the card sequence at the beginning of the permutation phase, we look at the procedure at the permutation phase 1. If c = 0, the center card is swapped with the rightmost one. As can be seen in Table 6, the swap affects the position of the output card only when a + b = 1, and the position of the output card varies depending on the value of c. Therefore, we can summarize the card sequences at the beginning of the permutation phase in Table 7.

Due to the correctness of MAJ₃, the suits of ■ in Table 7 are uniquely determined; Table 8 shows the minimum requirement for the suits and positions of some cards in the
Table 8  The minimum requirement of card sequences (at the beginning of the permutation phase) that MAJ₃ correctly works

| a + b | Card sequence |
|-------|---------------|
| 0     | ♣             |
| 1     | ♥             |
| 2     | ♣             |

Table 9  Concrete card sequences at the beginning of the permutation phase

| a + b | Card sequence |
|-------|---------------|
| 0     | ♣             |
| 1     | ♥             |
| 2     | ♣             |

card sequences at the beginning of the permutation phase, i.e., before Step 3). Table 9 shows the concrete card sequences in MAJ₃, and one can see that the sequences meet the minimum requirement in Table 8. Moreover, one can also check that the placement phase of MAJ₃, i.e., Steps 1) and 2), correctly generates the card sequences in Table 9 for any a, b ∈ {0, 1} by manually inputting all combinations of a, b ∈ {0, 1}.

Toy Example: the Case of n = 5

Based on the above observation, let us consider how to construct a five-input majority voting protocol gMAJ₅ as an example. Suppose that each player Pᵢ has a binary input xᵢ ∈ {0, 1} for i ∈ {0, 1, . . . , 4}, and let Pₖ := {P₀, P₁, P₂} and Pₕ := {P₃, P₄}. We assume gMAJ₅’s procedure as follows based on that of MAJ₃. Note that, just as three cards are used in the case where n = 3, we also assume that five cards are used in the case where n = 5. The suit of each of the five cards is specified later.

- **Placement phase:** P₀, P₁, P₂ ∈ Pₖ place all of five cards face down and perform private operations to generate a card sequence depending on their inputs.

- **Permutation phase:**
  - **Permutation phase 1:** Each of P₃, P₄ ∈ Pₕ performs the private operation opₕ if the player’s input is zero.
  - **Permutation phase 2:** Each of P₀, P₁, P₂ ∈ Pₖ performs the private operation opₖ if the player’s input is zero.

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

As in the observation on MAJ₃, we construct gMAJ₅ based on the above assumptions. Concretely, we first consider the card sequence during the permutation phase. Then, the requirement of card sequences at the beginning of the permutation phase is specified. Finally, we determine the suits of cards and the operations in the placement phase.
Table 10 Card sequences after the permutation phase 1 in gMAJ5

| \( \sum_{i=0}^{2} x_i \) | Card sequence |
|--------------------------|--------------|
| 0                        | [ ]           |
| 1                        | [ ] [ ]      |
| 2                        | [ ] [ ] [ ]  |
| 3                        | [ ] [ ] [ ]  |

Table 11 Card sequences at the beginning of the permutation phase in gMAJ5, where \( X = x_3 + x_4 \)

| \( \sum_{i=0}^{2} x_i \) | Card sequence |
|--------------------------|--------------|
| 0                        | [ ]           |
| 1                        | [ ] [ ] [ ]  |
| 2                        | [ ] [ ] [ ]  |
| 3                        | [ ] [ ] [ ]  |

Table 10, which is extended from Table 6 in the five-input setting, shows the expected card sequences after the permutation phase 1. Note that swapping two cards and a left cyclic shift over two cards are equivalent. Therefore, we consider the swapping in the permutation phase 2 in MAJ3 as a left cyclic shift and extend it to the case of five inputs. That is, we assume that \( \text{op}_l \) is the left cyclic shift over three cards on the left. As a result, the output card is placed at the leftmost position after the permutation phase 1 if \( \sum_{P_i \in P_l} x_i = 0 \) or \( \sum_{P_i \in P_l} x_i = 3 \) (\( \mid P_l \mid \)). Note that this is the same as we can see in MAJ3 if \( a + b = 0 \) or \( a + b = 2 \). Then, at the end of the permutation phase 1 in gMAJ5, the output card is placed at the center of the sequence if the inputs of two players in \( P_l \) will be zero (i.e., \( \sum_{P_i \in P_l} x_i = 1 \)). Similarly, the output card is placed the second one from the left if only one player in \( P_l \) will be zero, i.e., \( \sum_{P_i \in P_l} x_i = 2 \).

Now, recall the private operation \( \text{op}_s \) in MAJ3, i.e., the replacement of the center and rightmost cards, during the permutation phase 1; it can be also regarded as the left cyclic shift over two cards on the right. By extending it, we consider the following private operation as \( \text{op}_s \) during the permutation phase 1 in gMAJ5: the left cyclic shift over four cards on the right.

By fixing \( \text{op}_l \) and \( \text{op}_s \) as above, we can find out the card sequences and the minimum requirement of card sequences at the beginning of the permutation phase (see Tables 11, 12, respectively).

The last issue we have to resolve is how the players place cards and perform private operations so that the resulting card sequence is consistent with the minimum requirement, \( \text{op}_l \), and \( \text{op}_s \). Through the second and third card sequences in Table 12, let us
Protocol 5 gMAJ$_5$: Five-Input Majority Voting Protocol

1. Inputs: $P_0, P_1, \ldots, P_4$ have $x_0, x_1, \ldots, x_4 \in \{0, 1\}$, respectively.

2. Setup: Publicly put a sequence of five face-down cards $♣♥♥♣♣$ in this order. Then, $P_0$ receives the card sequence.

1) From $i = 0$ to $i = 2$, if $x_i = 0$, then $P_i$ does nothing behind the player’s back; otherwise, $P_i$ does a left cyclic shift behind the player’s back over the two cards from the left of the received card sequence. Then, $P_i$ sends the cards to $P_{i+1}$.

2) From $i = 3$ to $i = 4$, if $x_i = 0$, then $P_i$ does a left cyclic shift behind the player’s back over the four cards from the right of the received card sequence; otherwise, $P_i$ does nothing behind the player’s back. Then, $P_i$ sends the resulting card sequence to $P_{i+1 \mod 5}$.

3) From $i = 0$ to $i = 2$, if $x_i = 0$, then $P_i$ does a left cyclic shift behind the player’s back over the three cards from the left of the received card sequence; otherwise, $P_i$ does nothing behind the player’s back. Then, if $i \neq 2$, $P_i$ sends the resulting card sequence to $P_{i+1}$. If $i = 2$, $P_2$ opens the leftmost card of the five cards, and discards the others.

3. Output: If the card $P_2$ opens at step 3) is $♣$, the output is 0; otherwise (i.e., if the card is $♥$), the output is 1.

| $\sum_{i=0}^{2} x_i$ | Card sequence |
|----------------------|---------------|
| 0                    | $♣$           |
| 1                    | $♣$           |
| 2                    | $♣$           |
| 3                    | $♣$           |

Table 12 The minimum requirement of card sequences (at the beginning of the permutation phase) that gMAJ$_5$ correctly works

assume that the three cards on the right are the same regardless of any input of $P_L$, i.e., $x_0$, $x_1$, and $x_2$. From Table 12, the leftmost card should be $♣$ if $\sum_{i \in P_L} x_i = 0$, whereas it should be $♥$ if $\sum_{i \in P_L} x_i = 3$. Moreover, the second card from the left should be $♥$ if $\sum_{i \in P_L} x_i = 2$. It means we need a certain private operation to make the protocol consistent. Hence, we set the private operation during the placement phase is the left cyclic shift over the two cards $♣♥$ on the left, and each player in $P_L$ performs it if their input is one. Then, the cards $♣♥$ are shifted left (swapped) $\sum_{i \in P_L} x_i$ times, which satisfies the requirements shown in Table 12 because if $\sum_{i \in P_L} x_i$ is even, the two cards result in $♣♥$, and if $\sum_{i \in P_L} x_i$ is odd, the two cards result in $♥♣$. Consequently, an initial card sequence (i.e., a card sequence at the beginning of the placement protocol) should be $♣♥♥♣♣$. Table 13 summarizes the concrete card sequences at the beginning of the permutation phase, and our five-input majority voting protocol gMAJ$_5$ is formally given in Protocol 5.

We note that we may construct different protocols if we make different assumptions about the number of cards, the procedure of the protocol (i.e., placement, permutation, and output phases), the operations $\text{op}_L$ and $\text{op}_S$ in the permutation phase, and the suits of three cards on the right (assumed for determining the placement phase).
Construction Idea for Arbitrary \( n \)

We generalize the above construction for \( n = 5 \) and construct \( \text{gMAJ}_n \), a majority voting protocol with \( n \) inputs. We assume \( \text{gMAJ}_n \)'s generalized procedure for odd \( n = 2m + 1 \) and extend it to even \( n \) later. Suppose that each player \( P_i \) has a binary input \( x_i \in \{0, 1\} \) for \( i \in \{0, 1, \ldots, n\} \), and let \( \mathcal{P}_L := \{P_0, \ldots, P_m\} \) and \( \mathcal{P}_S := \{P_{m+1}, \ldots, P_{2m}\} \). In the generalized procedure, the operations \( \text{op}_S \) and \( \text{op}_L \) in the permutation phases 1 and 2 are assumed to be a left cyclic shift refer to the case where \( n = 5 \). In addition, we assume to use \( n \) (\( = 2m + 1 \)) cards, although the suit of each card is determined later.

- **Placement phase:** All of \( n \) cards are (publicly) placed face down in some order (will be determined as in \( \text{gMAJ}_5 \)), and players \( P_i \in \mathcal{P}_L \) perform private operations to generate a sequence of cards depending on their inputs.

- **Permutation phase:**
  - **Permutation phase 1:** Each player \( P_i \in \mathcal{P}_S \) \( (i \in \{m + 1, \ldots, 2m\}) \) performs the left cyclic shift over \( n - 1 \) (\( = 2m \)) cards on the right as the private operation \( \text{op}_S \) if \( x_i = 0 \).
  - **Permutation phase 2:** Each player \( P_i \in \mathcal{P}_L \) \( (i \in \{0, \ldots, m\}) \) performs the left cyclic shift over \( m + 1 \) cards on the left as the private operation \( \text{op}_L \) if \( x_i = 0 \).

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

By assuming the above procedure, all we have to do to construct an \( n \)-input majority voting protocol becomes to figure out the operations in the placement phase. As we constructed \( \text{gMAJ}_5 \), we specify the placement phase by following steps:

1. Specify the position of the output card depending on players’ inputs.
2. Specify the requirement of card sequences which has to be satisfied at the beginning of the permutation phase.
3. Specify the suits of cards and operations in the placement phase to generate the card sequence satisfying the requirement.

Indeed, Tables 14, 15, and 16 support these steps, respectively, as in the case of \( \text{gMAJ}_5 \) in the previous subsection.

| \( \sum_{i=0}^{2} x_i \) | Card sequence |
|-------------------------|---------------|
| 0                      | ♠             |
| 1                      | ♠             |
| 2                      | ♠             |
| 3                      | ♠             |
Table 14  Card sequences at the beginning of permutation phase in $gMAJ_{2m+1}$, where $X = \sum_{i=m+1}^{2m} x_i$

| $\sum_{i=0}^{m} x_i$ | Card sequence |
|-----------------------|---------------|
| 0                     | 0 1 2 ... $m-1$ $m$ $m+1$ ... $2m$ |
| 1                     | if $X = m$ if $X = m-1$ if $X = 0$ |
| 2                     | if $X = m$ if $X = m-2$ if $X = 0$ |
| $m-1$                 | if $X = m$ if $X = m$ |
| $m$                   | if $X = m$ if $X = m$ |
| $m+1$                 | if $X = m$ if $X = m$ |

Table 15  The minimum requirement (at the beginning of the permutation phase) that $gMAJ_{2m+1}$ correctly works

| $\sum_{i=0}^{m} x_i$ | Card sequence |
|-----------------------|---------------|
| 0                     | ♠ 2 ♣ ♠ ♠ ♠ ♠ ..... ♠ ♠ ♠ ♠ ♠ |
| 1                     | ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ |
| 2                     | if $X = m$ if $X = m$ |
| $m-1$                 | if $X = m$ if $X = m$ |
| $m$                   | if $X = m$ if $X = m$ |
| $m+1$                 | if $X = m$ if $X = m$ |

Odd Case: Majority Voting Protocol with $2m + 1$ Inputs Where $m \geq 2$

Based on the construction idea in Sect. 4.4, we construct an $n$-input majority voting protocol for $n = 2m + 1$, $m \geq 2$ called $gMAJ_{2m+1}$. By tracing the output card during the assumed permutation phase, the card sequence at the end of the placement phase can be represented as shown in Table 14. Note that the numbers 0, 1, ..., $2m$ in the table’s heading row indicate the index of the order in the sequence of cards.
Table 16 Concrete card sequences after the placement phase in gMAJ_{2m+1}

| \( \sum_{i=0}^{m} x_i \) | Card sequence |
|-----------------|---------------|
| 0               | ♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣♣ Craig_2022.png
Number of Cards, Private Operations, and Communication Required in \( g\text{MAJ}_{2m+1} \) (Protocol 6).

\( g\text{MAJ}_{2m+1} \) is executable with \( m + 1 \) \spadesuit cards and \( m \heartsuit \) cards, i.e., \((m + 1) + m = n\) cards. Then, \( m + 1, m, \) and \( m + 1 \) players perform a private operation in Step 1), 2), and 3), respectively. Therefore, the number of private operations is \((m + 1) + m + (m + 1) = n + \lfloor n/2 \rfloor + 1\).

The sequence of cards is sent to

\[
\begin{align*}
\text{Step 1) } & P_0, P_1, \ldots, P_m, P_{m+1}, P_{m+2}, \ldots, P_{2m}, P_0, P_1, \ldots, P_m \\
\text{Step 2) } & P_0, P_1, \ldots, P_m, P_{m+1}, P_{m+2}, \ldots, P_{2m}, P_0, P_1, \ldots, P_m \\
\text{Step 3) } & P_0, P_1, \ldots, P_m, P_{m+1}, P_{m+2}, \ldots, P_{2m}, P_0, P_1, \ldots, P_m
\end{align*}
\]

in this order. Thus, a total of \((m + 1) + m + m = n + \lfloor n/2 \rfloor \) communications are performed.

Even Case: Majority Voting Protocol with 2m Inputs Where \( m \geq 2 \)

Next, for \( n = 2m, m \geq 2 \), we consider to construct an \( n \)-input majority voting protocol, called \( g\text{MAJ}_{2m} \), in the same way as for \( n = 2m + 1 \). In this case, we consider two types of majority voting protocols, one computes \( \text{maj}^m_{2m} \), and the other computes \( \text{maj}^{m+1}_{2m} \). Suppose that each player \( P_i \) has a binary input \( x_i \in \{0, 1\} \) for \( i \in \{0, 1, \ldots, 2m - 1\} \). Basically, we can construct \( g\text{MAJ}_{2m} \) in the same way as for \( g\text{MAJ}_{2m+1} \) in Sect. 4.5, but we need to be careful about how to divide the players into two groups \( P_L \) and \( P_S \). Our construction framework works well if \( 2m \) players are divided into \( m + 1 \) players and \( m - 1 \) players, i.e., \(|P_L| = m + 1\) and \(|P_S| = m - 1\). (See Appendix A for details.)

However, suppose that \( 2m \) players are divided into two groups so that \(|P_L| = |P_S| = m\) holds, i.e., \( P_0, P_1, \ldots, P_{m-1} \in P_L \) and \( P_m, P_{m+1}, \ldots, P_{2m-1} \in P_S \). Then, our approach to construct a majority voting protocol does not work well because the minimum requirement for the correctness of \( g\text{MAJ}_{2m} \) cannot be satisfied.

Let us consider the computation of \( \text{maj}^m_{2m} \) at first. In this case, the leftmost card of the card sequence cannot be determined uniquely when \( \sum_{i=0}^{m-1} x_i = 0 \). Concretely, if there exists at least one of players in \( P_S \) inputs 0, i.e., \( \sum_{i=m}^{2m-1} x_i < m \), then the leftmost card has to be \( \spadesuit \) (because \( \sum_{i=0}^{m-1} x_i < 0 + m = m \)); otherwise, it has to be \( \heartsuit \) (because \( \sum_{i=0}^{2m-1} x_i = 0 + m = m \)). Since choosing one of \( \spadesuit \) or \( \heartsuit \) does not allow the other condition to be satisfied at the same time, the approach of dividing the players into two groups of equal numbers does not work. Table 17 shows the situation.

Analogously, in the computation of \( \text{maj}^{m+1}_{2m} \), the leftmost card cannot be uniquely determined when \( \sum_{i=0}^{m-1} x_i = m \). If all players in \( P_S \) input 0, i.e., \( \sum_{i=m}^{2m-1} x_i = 0 \), then the leftmost card when \( \sum_{i=0}^{m-1} x_i = m \) has to be \( \spadesuit \), otherwise, it has to be \( \heartsuit \). Thus, we cannot achieve both requirements, which results in our construction approach not working well for the case where \(|P_L| = |P_S| = m| \).

\footnote{\( \text{maj}^m_{2m} \) and \( \text{maj}^{m+1}_{2m} \) output different values only when the number of inputs of 0 and that of 1 are equal. The former outputs 1, and the latter outputs 0.}
Table 17 Unattainable requirements for correctness for the computation of $\text{maj}_m^m$ where $X = \sum_{i=m}^{2m-1} x_i$

| $\sum_{i=0}^{m-1} x_i$ | Card sequence |
|-------------------------|---------------|
| 0                       | ♣ and ♥       |
| 1                       | ♥             |
| ...                     | ...           |
| $m-1$                   | ♥             |
| $m$                     | ♥             |

Conclusion

We showed efficient card-based majority voting protocols under the assumption of private operations. Our three-input majority voting protocols (called MAJ$_3$wA and MAJ$_3$) are implemented by only three cards, whereas previous work requires four cards [7]. MAJ$_3$wA (Protocol 3) uses an AND/XOR/NOR hybrid protocol (Protocol 2) obtained by modifying the AND/OR protocol proposed by Nakai et al. [7]. Our idea of transforming the AND/XOR/NOR protocol to MAJ$_3$wA is to allow Carol to announce her output. To get rid of this extra condition, we proposed MAJ$_3$ (Protocol 4) that enables us to output the result of majority voting by a card without announcement. In addition, $n$-input majority voting protocols using $n$ cards (Protocols 6 and 7: called gMAJ$_n$) were constructed with combinations of cyclic shifts by deciding a part of the protocol procedures based on the observations of MAJ$_3$.

Recalling that eight cards are sufficient for realizing three-input majority voting protocols using shuffle operations [9], the proposed protocols (and Nakai et al.’s protocols [7]) are very efficient thanks to the power of private operations [8]. Proving the optimality of MAJ$_3$wA and MAJ$_3$ for three-input majority voting or that of gMAJ$_n$ for $n$-input majority voting is important future work. Note that since a (natural) lower bound of the number of cards required to a protocol seems to be the number of bits required for encoding the result (e.g., for the majority voting, the sum of all inputs, i.e., $\lceil \log_2(n+1) \rceil$), there also seems to be room for improvements. On the other hand, it is easy to see that the proposed AND/XOR/NOR protocol (Protocol 2) is optimal in terms of the number of cards since three cards are obviously necessary for representing three respective outputs (AND/XOR/NOR) simultaneously.

In this paper, we assume that players are semi-honest, and we did not use the announcement, which was used in MAJ$_3$wA, in the extended protocol gMAJ$_n$. However, it could be future work to consider protocols that allow other operations such as the announcement or those in other security settings, e.g., there are malicious players who do not follow the protocol. Furthermore, it is also challenging to propose...
**Protocol 7 gMAJ\(_{2m}\): (2m)-Input Majority Voting Protocol**

Inputs: \(P_0, P_1, \ldots, P_{2m-1}\) have \(x_0, x_1, \ldots, x_{2m-1} \in \{0, 1\}\), respectively.

\[
\begin{array}{c}
\text{Setup: Publicly put } 2m \text{ face-down cards } ♣\copyright\cdot\cdot\cdot♣\cdot\cdot\cdot♣\cdot\cdot\cdot♣\text{ in this order. Note that } \copyright \text{ (resp., } ♣\text{) is placed in the position of } \copyright\text{ when we compute } \text{maj}^m_{2m} \text{ (resp., } \text{maj}^{m+1}_{2m}\text{). Then, } P_0 \text{ receives these } 2m \text{ cards.} \\
1) \text{From } i = 0 \text{ to } i = m, \text{ if } x_i = 0, \text{ then } P_i \text{ does nothing behind the player’s back; otherwise, } P_i \text{ does a right cyclic shift behind the player’s back over the } m \text{ cards from the left of the received cards. Then, } P_i \text{ sends the cards to } P_{i+1}. \\
2) \text{From } i = m + 1 \text{ to } i = 2m - 1, \text{ if } x_i = 0, \text{ then } P_i \text{ does a left cyclic shift behind the player’s back over the } 2m - 1 \text{ cards from the right of the received cards; otherwise, } P_i \text{ does nothing behind the player’s back. Then, } P_i \text{ sends the cards to } P_{(i+1) \mod 2m}. \\
3) \text{From } i = 0 \text{ to } i = m, \text{ if } x_i = 0, \text{ then } P_i \text{ does a left cyclic shift behind the player’s back over the } m \text{ cards from the left of the received cards; otherwise, } P_i \text{ does nothing behind the player’s back. Then, if } i \neq m, P_i \text{ sends the cards to } P_{i+1}. \text{ If } i = m, P_m \text{ opens the leftmost card of the } 2m \text{ cards, and discards the others.} \\
\end{array}
\]

Output: If the card \(P_m\) opens at step 3) is ♣, the output is 0; otherwise (i.e., if the card is ♣), the output is 1.

**An Even Case: Majority Voting Protocol with 2m Inputs Where \(m \geq 2\)**

For \(n = 2m, \ m \geq 2\), we construct a \(n\)-input majority voting protocol in the same way as for \(n = 2m + 1\). In this case, we propose two types of majority voting protocol, one computes \(\text{maj}^m_{2m}\), and the other computes \(\text{maj}^{m+1}_{2m}\). Suppose that each player \(P_i\) has a binary input \(x_i \in \{0, 1\}\) for \(i \in \{0, 1, \ldots, 2m - 1\}\) and let \(\mathcal{P}_L := \{P_0, P_1, \ldots, P_m\}\) and \(\mathcal{P}_S := \{P_{m+1}, P_{m+2}, \ldots, P_{2m-1}\}\).

We assume \(g\text{MAJ}_{2m}\)’s procedure as follows. Moreover, we assume that \(n \ (= 2m)\) cards are used.

- **Placement phase:** All of \(n\) cards are (publicly) placed face down in some order, and players \(P_i \in \mathcal{P}_L\) perform private operations to generate a sequence of cards depending on their inputs.
- **Permutation phase:**
  - **Permutation phase 1:** Each player \(P_i \in \mathcal{P}_S \ (i \in \{m + 1, \ldots, 2m - 1\})\) performs the left cyclic shift over \(n - 1 \ (= 2m - 1)\) cards on the right as the private operation \(\text{op}_S\) if \(x_i = 0\).
Table 18 Card sequences after the placement phase in $g\text{MAJ}_{2m}$ where $X = \sum_{i=m+1}^{2m-1} x_i$

| $\sum_{i=0}^{m} x_i$ | Card sequence |
|------------------------|---------------|
| 0                     | 0 1 2 $\cdots$ $m-1$ $m$ $m+1$ $\cdots$ $2m-1$ |
| 1                     | 0 0 0 $\cdots$ 0 if $x_{m-1}$ 0 if $x_{m-2}$ $\cdots$ 0 if $x_0$ |
| 2                     | 0 0 0 $\cdots$ if $x_{m-1}$ if $x_{m-2}$ if $x_{m-3}$ $\cdots$ 0 |
| $\vdots$              | $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ $\vdots$ |
| $m-1$                 | 0 0 0 if $x_{m-1}$ if $x_{m-2}$ if $x_{m-3}$ if $x_{m-4}$ if $x_{m-5}$ $\cdots$ 0 |
| $m$                   | 0 0 0 if $x_{m-1}$ if $x_{m-2}$ if $x_{m-3}$ if $x_{m-4}$ if $x_{m-5}$ $\cdots$ 0 |
| $m+1$                 | 0 0 0 if $x_{m-1}$ if $x_{m-2}$ if $x_{m-3}$ if $x_{m-4}$ if $x_{m-5}$ $\cdots$ 0 |

- **Permutation phase 2:** Each player $P_i \in P_L$ ($i \in \{0, \ldots, m+1\}$) performs the left cyclic shift over $m+1$ cards on the left as the private operation $op_L$ if $x_i = 0$.

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

Then, based on the construction idea in Sect. 4.4, the card sequences after the placement phase and the minimum requirement for computing $\text{maj}^m_{2m}$ (or $\text{maj}^{m+1}_{2m}$) correctly can be represented as shown in Tables 18 and 19, respectively.

To specify the cards at the non-output positions in the sequences of cards after the placement phase, let us assume that $m$ cards from the right are the same regardless of $P_L$. Then, $m$ cards from the left can be considered to sequences of cards obtained by $\sum_{i=0}^{m} x_i$ times left cyclic shift over the $m$ cards $\spadesuit \heartsuit \cdots \heartsuit$. Actually, by performing the operations, $\spadesuit$ appears at the $((m - \sum_{i=0}^{m} x_i) \mod m)$-th leftmost place in the sequence and other places are filled by $\heartsuit$. We can see that such a sequence satisfies the requirement shown in Table 19. Therefore, the specified sequences of cards and constructed protocol become as shown in Table 20 and Protocol 7, respectively.

**A.1 Number of Cards, Private Operations, and Communication Required in $g\text{MAJ}_{2m}$ (Protocol 7).**

$g\text{MAJ}_{2m}$ is executable with $m$ $\spadesuit$ cards and $m$ $\heartsuit$ cards for computing $\text{maj}^m_{2m}$, or $m+1$ cards $\spadesuit$ and $m-1$ cards $\heartsuit$ for computing $\text{maj}^{m+1}_{2m}$. In either case, $m + m = (m + 1) + (m - 1) = n$ cards are required.
Table 19 The minimum requirement (at the beginning of the permutation phase) that gMAJ$_{2m}$ correctly works where ♦ means ♥ (resp., ♣) in the computation of maj$_{2m}$ (resp., maj$_{2m+1}$)

| $\sum_{i=0}^{m} x_i$ | Card sequence |
|----------------------|---------------|
| 0                    | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| 1                    | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| 2                    | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| ...                  | ...           |
| $m - 1$              | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| $m$                  | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| $m + 1$              | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |

Table 20 Concrete card sequences after the placement phase in gMAJ$_{2m}$, where ♦ means ♥ (resp., ♣) in the computation of maj$_{2m}$ (resp., maj$_{2m+1}$)

| $\sum_{i=0}^{m} x_i$ | Card sequence |
|----------------------|---------------|
| 0                    | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| 1                    | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| 2                    | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| ...                  | ...           |
| $m - 1$              | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| $m$                  | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |
| $m + 1$              | ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ ♣ |

Then, $m + 1$, $m - 1$, and $m + 1$ players perform a private operation in Step 1), 2), and 3), respectively. Therefore, the number of private operations is $(m + 1) + (m - 1) + (m + 1) = n + \lfloor n/2 \rfloor + 1$.

The sequence of cards is sent to

$$ P_0, P_1, \ldots, P_m, P_{m+1}, P_{m+2}, \ldots, P_{2m-1}, P_0, P_1, \ldots, P_m $$

in this order. Thus, a total of $(m + 1) + (m - 1) + m = n + \lfloor n/2 \rfloor$ communications are performed.
References

1. den Boer, B.: More efficient match-making and satisfiability: the five card trick. In: J. Quisquater, J. Vandewalle (eds.) Proceedings, Lecture Notes in Computer Science, Advances in Cryptology-EUROCRYPT ’89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10–13, vol. 434, pp. 208–217. Springer (1989). https://doi.org/10.1007/3-540-46885-4_23

2. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: T. Iwata, J.H. Cheon (eds.) Proceedings, Part I. Lecture Notes in Computer Science, Advances in Cryptology-ASIACRYPT 2015–21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, vol. 9452, pp. 783–807. Springer (2015). https://doi.org/10.1007/978-3-662-48797-6_32

3. Manabe, Y., Ono, H.: Secure card-based cryptographic protocols using private operations against malicious players. In: D. Maimut, A. Oprina, D. Sauveron (eds.) Revised Selected Papers, Lecture Notes in Computer Science, Innovative Security Solutions for Information Technology and Communications-13th International Conference, SecITC 2020, Bucharest, Romania, November 19–20, 2020, vol. 12596, pp. 55–70. Springer (2020). https://doi.org/10.1007/978-3-030-69255-1_5

4. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. IACR Cryptol. ePrint Arch. 2015, 1031 (2015). http://eprint.iacr.org/2015/1031

5. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: G. Mauri, A. Dennunzio, L. Manzoni, A.E. Porreca (eds.) Proceedings, Lecture Notes in Computer Science, Unconventional Computation and Natural Computation-12th International Conference, UCNC 2013, Milan, Italy, July 1–5, 2013, vol. 7956, pp. 162–173. Springer (2013). https://doi.org/10.1007/978-3-642-39074-6_16

6. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: X. Deng, J.E. Hopcroft, J. Xue (eds.) Proceedings, Lecture Notes in Computer Science, Frontiers in Algorithmic, Third International Workshop, FAW 2009, Hefei, China, June 20–23, 2009, vol. 5598, pp. 358–369. Springer (2009). https://doi.org/10.1007/978-3-642-02270-8_36

7. Nakai, T., Shirouchi, S., Iwamoto, M., Ohta, K.: Four cards are sufficient for a card-based three-input voting protocol utilizing private permutations. In: J. Shikata (ed.) Proceedings, Lecture Notes in Computer Science, Information Theoretic Security-10th International Conference, ICITS 2017, Hong Kong, China, November 29–December 2, 2017, vol. 10681, pp. 153–165. Springer (2017). https://doi.org/10.1007/978-3-319-72089-0_9

8. Nakai, T., Tokushige, Y., Misawa, Y., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for millionaires’ problem utilizing private permutations. In: S. Foresti, G. Persiano (eds.) Proceedings, Lecture Notes in Computer Science, Cryptology and Network Security-15th International Conference, CANS 2016, Milan, Italy, November 14–16, 2016, vol. 10052, pp. 500–517 (2016). https://doi.org/10.1007/978-3-319-48965-0_30

9. Nishida, T., Mizuki, T., Sone, H.: Securely computing the three-input majority function with eight cards. In: A. Dediu, C. Martín-Vide, B. Truthe, M.A. Vega-Rodriguez (eds.) Proceedings, Lecture Notes in Computer Science, Theory and Practice of Natural Computing-Second International Conference, TPNC 2013, Cáceres, Spain, December 3–5, 2013, vol. 8273, pp. 193–204. Springer (2013). https://doi.org/10.1007/978-3-642-45008-2_16

10. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Gener. Comput. 39(1), 19–40 (2021). https://doi.org/10.1007/s00354-020-00113-z

11. Watanabe, Y., Kuroki, Y., Suzuki, S., Koga, Y., Iwamoto, M., Ohta, K.: Card-based majority voting protocols with three inputs using three cards. In: International Symposium on Information Theory and Its Applications, ISITA 2018, Singapore, October 28–31, 2018, pp. 218–222. IEEE (2018). https://doi.org/10.23919/ISITA.2018.8664324

12. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3–5 November 1982, pp. 160–164. IEEE Computer Society (1982). https://doi.org/10.1109/SFCS.1982.38

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.