Upper Bounds on the Number of Shuffles for Two-Helping-Card Multi-Input AND Protocols*

Takuto Yoshida[1], Kodai Tanaka[2], Keisuke Nakabayashi[2], Eikoh Chida[1], and Takaaki Mizuki[2]

[1] National Institute of Technology, Ichinoseki College, Ichinoseki, Iwate, Japan
chida+lncs[atmark]g.ichinoseki.ac.jp
[2] Tohoku University, Sendai, Japan
mizuki+lncs[atmark]tohoku.ac.jp

Abstract. Card-based cryptography uses a physical deck of cards to achieve secure computations. To evaluate the performance of card-based protocols, the numbers of helping cards and shuffles required to execute are often used as evaluation metrics. In this paper, we focus on n-input AND protocols that use at most two helping cards, and investigate how many shuffles suffice to construct such a two-helping-card AND protocol. Since the Mizuki–Sone two-input AND protocol uses two helping cards and it can be repeatedly applied $n-1$ times to perform a secure n-input AND computation, an obvious upper bound on the number of required shuffles is $n-1$. In this paper, to obtain better bounds (than $n-1$), we consider making use of the "batching" technique, which was developed by Shinagawa and Nuida in 2020 to reduce the number of shuffles. Specifically, we first formulate the class of two-helping-card n-input AND protocols obtained by applying the batching technique to the Mizuki–Sone AND protocol, and then show n-input AND protocols requiring the minimum number of shuffles (among the class) for the case of $2 \le n \le 500$.

Keywords: Card-based cryptography · Secure computation · Real-life hands-on cryptography · AND protocols

1 Introduction

Secure computations [36] enable players holding individual private inputs to evaluate a predetermined function without revealing the input values more than necessary. The method of secure computation using a physical deck of cards is called *card-based cryptography* [7, 19]. Typically, two types of cards are used, where the reverse side is indistinguishable as $\boxed{?}$ and the front side is either $\boxed{\clubsuit}$ or $\boxed{\heartsuit}$. These cards are arranged in the following way to represent Boolean values:

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

* This paper appears in Proceedings of CANS 2023. This version of the contribution has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-981-99-7563-1_10. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use: https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms.

When two cards placed face down according to this encoding rule represent a bit $x \in \{0, 1\}$, these two cards are called a *commitment* to x and are represented as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{x}\,.$$

A two-input AND protocol takes as input two commitments to bits $x, y \in \{0, 1\}$ along with some helping cards like $\boxed{\clubsuit}\boxed{\heartsuit}$, and performs a secure computation of the AND value $x \wedge y$ via a series of actions such as shuffling, rearranging, and turning over cards. When the output is obtained as a commitment to $x \wedge y$, such a protocol is called a *committed-format* protocol:

$$\underbrace{\boxed{?}\boxed{?}}_{x}\underbrace{\boxed{?}\boxed{?}}_{y}\boxed{\clubsuit}\boxed{\heartsuit}\cdots\rightarrow\cdots\rightarrow\underbrace{\boxed{?}\boxed{?}}_{x\wedge y}.$$

This paper deals with committed-format AND protocols, especially multi-input AND protocols, as seen later.

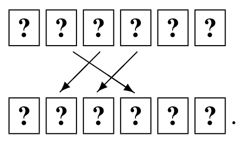
1.1 The Mizuki–Sone AND Protocol

The most practical committed-format two-input AND protocol currently known would be the protocol proposed by Mizuki and Sone in 2009 [20]. Hereinafter, we refer to this as the *MS-AND protocol*, and the procedure is described below.

1. Place two input commitments to $x, y \in \{0, 1\}$ along with two helping cards, turning the middle two cards face down, as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{x}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{y}\rightarrow\underbrace{\boxed{?}\boxed{?}}_{x}\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{y}.$$

2. Rearrange the sequence as follows:

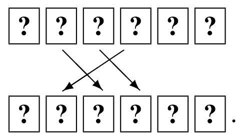


3. A *random bisection cut* (hereafter sometimes called an *RBC* for short), denoted by $[\cdots|\cdots]$, is applied to the sequence of six cards, meaning that we split the card sequence in half and randomly swap the left and right sides (until anyone loses track of the move):

$$\left[\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\right]\rightarrow\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

It is known that a random bisection cut can be securely implemented using familiar tools such as envelopes [35].

4. Rearrange the sequence as follows:



5. The two cards are turned over from the left. Depending on the order of the two revealed cards, we obtain a commitment to $x \wedge y$ as follows:

$$\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{x\wedge y}\boxed{?}\boxed{?} \quad \text{or} \quad \boxed{\heartsuit}\boxed{\clubsuit}\boxed{?}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{x\wedge y}.$$

The above is the MS-AND protocol, which, given commitments to x and y, uses two helping cards and one random bisection cut to output a commitment to $x \wedge y$.

After the protocol terminates, the two cards that were turned over in Step 5 can be used as helping cards in another protocol run; we call such face-up cards *free cards*.

We will also call the two face-down cards that are not a commitment to $x \wedge y$ a *garbage commitment*[iii]. A garbage commitment can be transformed into two free cards by applying a (normal) shuffle to the two cards (composing the garbage commitment) and turning them over.

1.2 Committed-Format Multi-Input AND Protocol

As mentioned above, the subject of this paper is to construct committed-format multi-input AND protocols. That is, given n input commitments

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n},$$

we want to produce a commitment to $x_1 \wedge x_2 \wedge \cdots \wedge x_n$.

Applying the MS-AND protocol described in Sect. 1.1 to commitments to x_1 and x_2 yields a commitment to $x_1 \wedge x_2$ together with two free cards. Thus, we can continue to apply the MS-AND protocol to commitments to $x_1 \wedge x_2$ and x_3. By repeating this a total of $n-1$ times, a committed-format n-input AND protocol can be constructed [15]:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_n}\boxed{\clubsuit}\boxed{\heartsuit} \rightarrow \cdots \rightarrow \underbrace{\boxed{?}\boxed{?}}_{x_1\wedge x_2\wedge\cdots\wedge x_n} \quad .$$

In this case, the number of required helping cards is two, and the number of required shuffles (namely, the number of random bisection cuts) is $n-1$.

iii These two cards are actually a commitment to $\overline{x} \wedge y$.

1.3 Contribution of This Paper

As described in Sect. 1.2, an obvious upper bound on the number of required shuffles for a committed-format n-input AND protocol is $n - 1$ under the condition that two helping cards are available. On the other hand, there is a technique called the "batching" proposed in 2020 by Shinagawa and Nuida [31] that can reduce the number of shuffles.

Therefore, in this paper, we apply the batching technique to the MS-AND protocol so that we can construct two-helping-card n-input AND protocols having a smaller number of shuffles. Such an application naturally formulates the class of committed-format two-helping-card n-input AND protocols. Within the class, we present a simple generic protocol having a smaller number of shuffles. Furthermore, we reduce the problem of constructing a protocol to the "MSbatching move-sequence" problem (which is a kind of a computational problem), and by analyzing the latter problem, we show the minimum number of shuffles among the class of protocols in the range of $2 \leq n \leq 500$. It turns out that the generic proposed protocol is optimal in terms of the number of shuffles for many cases of n. For every n, $2 \leq n \leq 500$, such that the proposed protocol is not optimal, we find optimal protocols, as well.

1.4 Related Works

The history of committed-format two-input AND protocols dates back to 1993 [2], and since then, a couple protocols have been invented [22, 33], followed by the MS-AND protocol in 2009 [20] (which uses six cards and one random bisection cut as seen above). Subsequently, four- and five-card protocols have been developed using complex shuffles [8, 9, 25].

As for committed-format multi-input AND protocols, those using only one or two shuffles have recently been proposed [11] (although many helping cards are required). In addition, several specialized protocols have been known [6, 15].

The research area of card-based cryptography has been growing rapidly in recent years [16, 17]. Examples of active topics are: physical zero-knowledge proof protocols [4, 10, 23, 24], private-model secure computations [1, 12, 21], symmetric function evaluation [26–28], information leakage due to operative or physical errors [18, 29], graph automorphism shuffles [14, 30], multi-valued protocols with a direction encoding [34], the half-open action [13], card-minimal protocols [3, 8], and applications to private simultaneous messages protocols [32].

1.5 Organization of This Paper

The remainder of this paper is organized as follows. In Sect. 2, we introduce the batching technique along with the "pile-scramble shuffle" required for it. In Sect. 3, we show how the batching technique can be applied to multiple MS-AND protocols. In Sect. 4, we formulate the class of protocols obtained by applying the batching technique, and reduce the problem of finding protocols having fewer shuffles to the "MSbatching move-sequence" problem and propose a simple generic protocol in Sect. 5. In Sect. 6, the MSbatching move-sequence problem is analyzed by a dynamic programming algorithm, and we show optimal n-input AND protocols for $2 \leq n \leq 500$ in the sense that the number of shuffles is minimum among all the protocols in the class. Finally, the conclusion is given in Sect. 7.

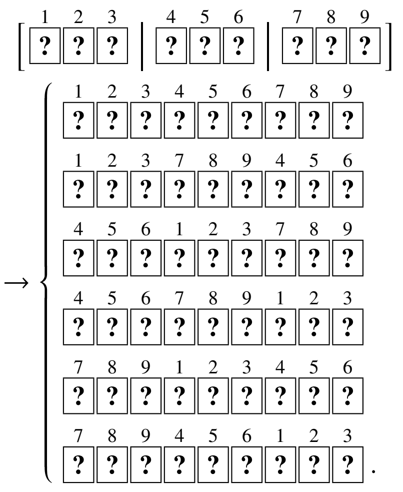
2 Preliminaries

In this section, we first introduce the pile-scramble shuffle [5] and then explain the batching technique [31].

2.1 Pile-Scramble Shuffle

A *pile-scramble shuffle* [5] is a shuffling operation that divides a sequence of cards into multiple piles of the same size and then rearranges the order of those piles uniformly at random (while the order of cards inside each pile is kept unchanged).

As an example, applying a pile-scramble shuffle, denoted by $[\,\cdot \,|\, \cdot \,|\, \cdots \,|\, \cdot\,]$, to a sequence of nine cards consisting of three piles yields one of the following six sequences with a probability of exactly $1/6$:



A pile-scramble shuffle can be easily implemented by placing each pile in an envelope and randomly stirring the envelopes.

A random bisection cut that appears in the MS-AND protocol can be said to be a pile-scramble shuffle for two piles (each consisting of three cards).

2.2 Batching Technique

The batching technique [31] combines multiple pile-scramble shuffles that can be executed in parallel into a single pile-scramble shuffle, thereby reducing the number of shuffles. Simply put, after adding "identifiers" with some helping cards to the piles of each pile-scramble shuffle, we perform a single pile-scramble shuffle together, and then open the identifier to return each pile to the position of its original pile-scramble shuffle.

As an example, suppose that we want to apply two random bisection cuts (RBCs) in parallel, which appear in the MS-AND protocol, and that we want to use the batching technique. In other words, we want to perform two RBCs

$$\left[\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\right],\left[\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\right]$$

simultaneously using a single shuffle by the batching technique.

1. To identify two RBCs, we use $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. That is, at the head of each pile, a helping card for identification is placed as follows:

   $$\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}.$$

   In the sequel, we call such helping cards *identifier cards*.
2. Turn over the identifier cards and apply a pile-scramble shuffle to the four piles:

   $$\begin{aligned}&\left[\boxed{?}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\boxed{?}\middle|\boxed{?}\boxed{?}\boxed{?}\boxed{?}\right]\\ \rightarrow&\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.\end{aligned}$$

3. Turn the identifier cards face up. For instance, suppose that the following sequence of cards is obtained:

   $$\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}.$$

4. Sort the piles so that the pile with $\boxed{\clubsuit}$ at the head is on the left and the pile with $\boxed{\heartsuit}$ at the head is on the right, as when the identifier cards were inserted in Step 1. In the example above, the fourth pile is moved in front of the second pile:

   $$\begin{aligned}&\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\\ \rightarrow&\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\;\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}.\end{aligned}$$

   Note that sorting the piles here is done publicly as seen just above (and hence, we need no additional shuffle).
5. Remove the face-up identifier cards; then, we have performed two RBCs by one pile-scramble shuffle.

In this example, two sets of $\boxed{\clubsuit}\boxed{\heartsuit}$ were used to identify the two RBCs. If we want to identify four RBCs, two sets of

$$\boxed{\clubsuit}\boxed{\clubsuit}\;\boxed{\clubsuit}\boxed{\heartsuit}\;\boxed{\heartsuit}\boxed{\clubsuit}\;\boxed{\heartsuit}\boxed{\heartsuit}$$

suffice. That is, we can distinguish the piles by two binary digits according to the one-card-per-bit encoding: $\boxed{\clubsuit} = 0$, $\boxed{\heartsuit} = 1$. In general, when we want to apply the batching technique to k RBCs, we need $2k\lceil\log_2 k\rceil$ identifier cards (i.e., $k\lceil\log_2 k\rceil$ free cards for each of $\clubsuit$ and $\heartsuit$).

The batching technique can be applied not only to RBCs, but more generally to multiple pile-scramble shuffles. However, this paper only utilizes it for RBCs (of six cards each).

3 Application of Batching to MS-AND Protocol

As seen in Sect. 1.2, executing repeatedly the MS-AND protocol provides a two-helping-card n-input AND protocol using $n-1$ shuffles, namely $n-1$ RBCs. In this section, we utilize the batching technique to reduce the number of shuffles.

In Sect. 3.1, we mention the idea behind our approach. In Sect. 3.2, we present how to batch MS-AND protocols. In Sect. 3.3, we show an example of a protocol based on our approach.

3.1 Idea

As described in Sect. 2.2, the batching technique can be used to convert multiple RBCs into a single pile-scramble shuffle. This can be applied to the execution of multiple MS-AND protocols to reduce the number of shuffles. Remember that, in order to perform the batching technique, some free cards as identifier cards must be provided to identify each pile pair. Remember furthermore that in the problem setup, only two helping cards are available:

$$\underbrace{\boxed{?}\,\boxed{?}}_{x_1}\,\underbrace{\boxed{?}\,\boxed{?}}_{x_2}\cdots\underbrace{\boxed{?}\,\boxed{?}}_{x_n}\,\boxed{\clubsuit}\,\boxed{\heartsuit}.$$

Therefore, the batching technique cannot be applied immediately (because of shortage of free cards as identifier cards).

Let us recall the procedure of the MS-AND protocol described in Sect. 1.1; then, a garbage commitment arises in Step 5. That is, one execution of the MS-AND protocol yields one garbage commitment. Several garbage commitments can be turned into free cards by shuffling all the cards (of the garbage commitments) and revealing them. This leads to increasing the number of free cards to be used as identifier cards even if there are only two helping cards at the beginning.

More specifically, for the first m input commitments, if the MS-AND protocol is repeated $m-1$ times, we obtain $2m-2$ free cards:

$$\underbrace{\boxed{?}\,\boxed{?}}_{x_1}\cdots\underbrace{\boxed{?}\,\boxed{?}}_{x_m}\,\boxed{\clubsuit}\,\boxed{\heartsuit}\;\rightarrow\;\underbrace{\boxed{?}\,\boxed{?}}_{x_1\wedge x_2\wedge\cdots\wedge x_m}\;\boxed{\clubsuit}\,\boxed{\heartsuit}\;\underbrace{\boxed{\clubsuit}\,\boxed{\heartsuit}\cdots\boxed{\clubsuit}\,\boxed{\heartsuit}}_{2m-2\text{ cards}}$$

Thus, some of the $2m$ free cards can be used as identifier cards for the batching technique. In this way, we first apply the MS-AND protocol for the first several input commitments to produce free cards enough for the batching technique to execute.

3.2 MSbatching: How to Batch MS-AND Protocols

This subsection describes in detail how the batching technique is applied to the execution of multiple MS-AND protocols.

Before we begin, let us define a couple of terms. As described in Sect. 3.1, free cards can be created by collecting garbage commitments, shuffling all the cards (constituting the garbage commitments), and then turning them over. This procedure is called the *garbage collection*. For convenience, we will refer to a pair of free cards (of different colors) $\boxed{\clubsuit}\boxed{\heartsuit}$ placed face up as a *free pair*.

First, as an example, assume that there are four commitments

$$\underbrace{\boxed{?}\boxed{?}}_{y_1}\underbrace{\boxed{?}\boxed{?}}_{y_2}\quad\underbrace{\boxed{?}\boxed{?}}_{y_3}\underbrace{\boxed{?}\boxed{?}}_{y_4}$$

and we want to produce commitments to $y_1 \wedge y_2$ and $y_3 \wedge y_4$ by executing the MS-AND protocol twice. Recalling the MS-AND protocol procedure, we need two helping cards, i.e., one free pair $\boxed{\clubsuit}\boxed{\heartsuit}$, per run; therefore, we require two free pairs:

$$\underbrace{\boxed{?}\boxed{?}}_{y_1}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{y_2}\quad\underbrace{\boxed{?}\boxed{?}}_{y_3}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{y_4}.$$

For each of these, an RBC is applied (after reordering). Using the batching technique, this can be achieved with a single shuffle. In this case, however, two more free pairs

$$\boxed{\clubsuit}\boxed{\heartsuit}\ \boxed{\clubsuit}\boxed{\heartsuit}$$

are required (for identifier cards). After applying the batching technique and turning over the four identifier cards, we return to the two MS-AND protocols and terminate each protocol. Then, the following commitments are obtained:

$$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\quad\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{y_1\wedge y_2}\underbrace{\boxed{?}\boxed{?}}_{\text{garbage}}\quad\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{y_3\wedge y_4}\underbrace{\boxed{?}\boxed{?}}_{\text{garbage}}.$$

In summary, given four commitments and four free pairs, one shuffle suffices to output two commitments (to the AND values), four free pairs, and two garbage commitments.

Next, more generally, consider running k MS-AND protocols in parallel (i.e., the number of input commitments is $2k$). As mentioned earlier, one MS-AND protocol requires one free pair, and hence, k free pairs are needed for this amount. In addition, when applying the batching technique, free cards are also needed for serving identifier cards; as mentioned in Sect. 2.2, we require $2k\lceil\log_2 k\rceil$ free cards, which are $k\lceil\log_2 k\rceil$ free pairs. Thus, a total of $k + k\lceil\log_2 k\rceil$ free pairs are required:

$$\underbrace{\boxed{?}\boxed{?}}_{y_1}\underbrace{\boxed{?}\boxed{?}}_{y_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{y_{2k-1}}.\underbrace{\boxed{?}\boxed{?}}_{y_{2k}}\quad\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\times(k + k\lceil\log_2 k\rceil).$$

Applying the batching technique to this sequence of cards, the $2k$ commitments become k commitments (to the AND values) after one shuffle, resulting in k garbage commitments and no change in the number of free pairs:

$$\underbrace{\boxed{?}\boxed{?}}_{y_1\wedge y_2}\cdots\underbrace{\boxed{?}\boxed{?}}_{y_{2k-1}\wedge y_{2k}}\quad\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\times(k + k\lceil\log_2 k\rceil)\quad\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{garbage}}\times k.$$

This procedure will henceforth be referred to as *k-MSbatching*.

The number of free pairs required for k-MSbatching is given by $k+k\lceil\log_2 k\rceil$, where the specific numbers are shown in Table 1. Note that 1-MSbatching is the MS-AND protocol itself (i.e., one run of the protocol).

Table 1: The number of free pairs required for k-MSbatching

k	1	2	3	4	5	6	7	8	9
# of free pairs	1	4	9	12	20	24	28	32	45

3.3 Example of Two-Helping-Card AND Protocol by MSbatching

In this subsection, we illustrate a two-helping-card protocol by using MSbatching.

Suppose that the number of inputs is 48, i.e., $n = 48$. Then, 48 commitments and one free pair are the input sequence:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\underbrace{\boxed{?}\boxed{?}}_{x_3}\underbrace{\boxed{?}\boxed{?}}_{x_4}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_{48}}\;\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}.$$

Recall that the obvious upper bound on the number of shuffles is to repeat the MS-AND protocol 47 times, which is 47 shuffles. The following protocol requires a smaller number of shuffles to produce a commitment to the AND value of 48 inputs.

1. Perform 1-MSbatching 15 times for the commitments to $x_1, \ldots, x_{16}$ to obtain a commitment to $x_1 \wedge \cdots \wedge x_{16}$ (by 15 shuffles):
$$\underbrace{\boxed{?}\boxed{?}}_{x_1\wedge\cdots\wedge x_{16}}\;\underbrace{\boxed{?}\boxed{?}}_{x_{17}}\underbrace{\boxed{?}\boxed{?}}_{x_{18}}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_{48}}\;\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\;\underbrace{\boxed{?}\boxed{?}}_{\text{garbage}}\times 15.$$

2. The garbage collection is performed to generate 15 free pairs (by one shuffle):
$$\underbrace{\boxed{?}\boxed{?}}_{x_1\wedge\cdots\wedge x_{16}}\;\underbrace{\boxed{?}\boxed{?}}_{x_{17}}\underbrace{\boxed{?}\boxed{?}}_{x_{18}}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_{48}}\;\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\times 16.$$

3. Perform 4-MSbatching for $x_{17}, \ldots, x_{24}$ and for $x_{25}, \ldots, x_{32}$, followed by 4-MSbatching, 2-MSbatching, and 1-MSbatching, in this order, to obtain a commitment to $x_{17} \wedge \cdots \wedge x_{32}$ (by five shuffles):
$$\underbrace{\boxed{?}\boxed{?}}_{x_1\wedge\cdots\wedge x_{16}}\;\underbrace{\boxed{?}\boxed{?}}_{x_{17}\wedge\cdots\wedge x_{32}}\;\underbrace{\boxed{?}\boxed{?}}_{x_{33}}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_{48}}\;\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\times 16\;\underbrace{\boxed{?}\boxed{?}}_{\text{garbage}}\times 15.$$

4. Perform 1-MSbatching for $x_1 \wedge \cdots \wedge x_{16}$ and $x_{17} \wedge \cdots \wedge x_{32}$ to obtain a commitment to $x_1 \wedge \cdots \wedge x_{32}$ (by one shuffle):
$$\underbrace{\boxed{?}\boxed{?}}_{x_1\wedge\cdots\wedge x_{32}}\;\underbrace{\boxed{?}\boxed{?}}_{x_{33}}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_{48}}\;\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\times 16\;\underbrace{\boxed{?}\boxed{?}}_{\text{garbage}}\times 16.$$

5. The garbage collection is performed (by one shuffle):
$$\underbrace{\boxed{?}\boxed{?}}_{x_1\wedge\cdots\wedge x_{16}}\;\underbrace{\boxed{?}\boxed{?}}_{x_{17}\wedge\cdots\wedge x_{32}}\;\underbrace{\boxed{?}\boxed{?}}_{x_{33}}\cdots\underbrace{\boxed{?}\boxed{?}}_{x_{48}}\;\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\times 32.$$

6. Execute 8-MSbatching, 4-MSbatching, 2-MSbatching, and 1-MSbatching for $x_{33}, \ldots, x_{48}$ in this order to obtain a commitment to $x_{33} \wedge \cdots \wedge x_{48}$ (by four shuffles):
$$\underbrace{\boxed{?}\boxed{?}}_{x_1\wedge\cdots\wedge x_{32}}\ \underbrace{\boxed{?}\boxed{?}}_{x_{33}\wedge\cdots\wedge x_{48}}\ \underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\times 32\ \underbrace{\boxed{?}\boxed{?}}_{\text{garbage}}\times 15.$$
7. Perform 1-MSbatching for $x_1 \wedge \cdots \wedge x_{32}$ and $x_{33} \wedge \cdots \wedge x_{48}$ to obtain a commitment to $x_1 \wedge \cdots \wedge x_{48}$ (by one shuffle):
$$\underbrace{\boxed{?}\boxed{?}}_{x_1\wedge\cdots\wedge x_{48}}\ \underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{\text{free}}\times 32\ \underbrace{\boxed{?}\boxed{?}}_{\text{garbage}}\times 16.$$

As shown above, the total number of shuffles is 28, which is a significant reduction from the obvious upper bound of 47 shuffles.

4 Class of MSbatching Protocols and Corresponding Problem

As seen in Sect. 3, given n input commitments and one free pair, we can produce a commitment to the AND value via a series of MSbatching (including 1-MSbatching) and the garbage collection with fewer shuffles than the obvious upper bound. In this section, we first formulate the class of two-helping-card AND protocols, called the "MSbatching protocols," naturally created by the combination of MSbatching and the garbage collection. We then introduce the "MSbatching move-sequence" problem; as seen soon, finding a protocol in the class corresponds to solving the problem.

4.1 MSbatching Protocols

This subsection clarifies the class of protocols obtained by MSbatching.

First, let $k \geq 1$ and consider the conditions under which k-MSbatching can be performed. As mentioned in Sect. 3.2, at least $k + k\lceil \log_2 k \rceil$ free pairs are required. Also, to be able to run the MS-AND protocol k times in parallel, there must be at least $2k$ input commitments.

Next, if we want to perform the garbage collection on g garbage commitments for $g \geq 1$, there must be at least g garbage commitments. Hereafter, the garbage collection for g garbage commitments is sometimes referred to as g-GC.

Bearing these in mind, we naturally obtain a class of one-free-pair n-input AND protocols, which we call the *MSbatching protocols*, as follows.

1. $a := n$, $b := 1$.
2. Now there are a commitments and b free pairs (and hence, there are $n - a - b + 1$ garbage commitments). Perform one of the followings.
   (a) Apply k-MSbatching such that $a \geq 2k$ and $b \geq k + k\lceil \log_2 k \rceil$. In this case, the number of commitments decreases by k, the number of garbage commitments increases by k, and the number of free pairs remains the same. Thus, we set
$$a := a - k,\ b := b.$$

(b) Apply g-GC such that $1 \leq g \leq n - a - b + 1$. Since g free pairs arise, we set

$$a := a, \; b := b + g.$$

3. If $a \geq 2$, then return to Step 2.

As described above, determining the strategy of selection in Step 2 stipulates one protocol. When $a \geq 2$, 1-MSbatching is always applicable, so it is never unselectable in Step 2. Note that the number of times Step 2 is executed is directly the number of shuffles the protocol uses.

4.2 MSbatching Move-Sequence Problem

Each protocol in the class of MSbatching protocols defined in Sect. 4.1 changes the current number of commitments a and the current number of free pairs b according to the selection in Step 2. Therefore, let us represent the current state at each iteration of Step 2 by a pair (a, b) and consider it as a point (a, b) on the ab-plane.

When $a = 1$, there is exactly one commitment and the protocol terminates; thus, we call any point $(1, b)$ a *terminal* point.

Assume a point (a, b) which is not terminal; from the point (a, b), we transition to another point by either of the following operations.

1. Transition to the point $(a - k, b)$ by k-MSbatching (provided that $a \geq 2k$ and $b \geq k + k\lceil \log_2 k \rceil$). Denote this by $B^k(a, b) = (a - k, b)$.
2. Transition to the point $(a, b + g)$ by g-GC (provided that $1 \leq g \leq n - a - b + 1$). Denote this by $GC^g(a, b) = (a, b + g)$.

In the ab-plane, starting from point $(n, 1)$, the current state (a, b) moves by Transition 1 or 2. Transition 1 moves horizontally (left) and Transition (2) moves vertically (up). The number of transitions required to reach a terminal point from the start $(n, 1)$ corresponds to the number of shuffles. That is, the length of the move-sequence connecting the start and terminal points corresponds to the number of shuffles used in the corresponding protocol. Figure 1 shows the move-sequence corresponding to the protocol illustrated in Sect. 3.3.

Figure 2 illustrates the area to which Transition 1 can be applied. From the lightest color to the darkest, they represent the regions to which k-MSbatching can be applied with $k = 2, 3, \ldots, 9$.

The *MSbatching move-sequence problem* is defined as easily imagined: given a start point $(n, 1)$, find a move-sequence to an terminal point on the ab-plane where only Transitions 1 and 2 are applicable. Such a move-sequence uniquely corresponds an MSbatching protocol, and the length of the move-sequence corresponds to the number of shuffles. Therefore, finding a shortest move-sequence is equivalent to constructing an optimal MSbatching protocol in terms of the number of shuffles.

5 Proposed Protocol

In this section, we propose a generic construction for an n-input MSbatching protocol by giving how to choose Transitions 1 and 2.

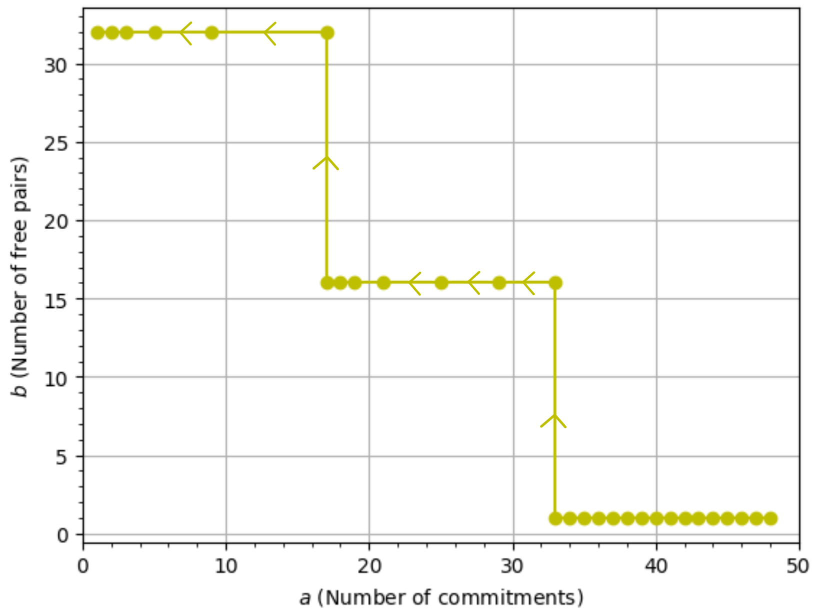


Fig. 1: The 48-input MSbatching protocol given in Sect. 3.3

5.1 Description of Proposed Protocol

First, we explain the idea of how to choose Transitions 1 and 2 in the proposed protocol. Basically, k-MSbatching to be applied is limited to those where k is a power of 2. If the garbage collection would allow for a larger MSbatching size compared to the currently applicable MSbatching, then perform the garbage collection.

To describe the above idea formally, the proposed protocol chooses a transition for a point (a, b) such that $a \geq 2$ as follows, where let k_b be the maximum value of k that satisfies $a \geq 2k$ and $b \geq k + k\lceil \log_2 k \rceil$, and let k_{b+g} be the maximum value of k that satisfies $a \geq 2k$ and $b + g \geq k + k\lceil \log_2 k \rceil$.

- If $k_{b+g} > k_b$ and $k_{b+g} = 2^i$ for some integer i, perform $(n - a - b + 1)$-GC and transition to $GC^{(n-a-b+1)}(a, b) = (a, n - a + 1)$.
- Otherwise, perform k_b-MSbatching and transition to $B^{k_b}(a, b) = (a - k_b, b)$.

5.2 Proposed Protocol for $n = 48$

Here, we illustrate the proposed protocol for the case of $n = 48$ as an example. That is, Figure 3 shows the move-sequence of the proposed protocol in the case of 48 inputs.

For further explanation, 1-MSbatching is performed until enough garbage commitments have been accumulated for 2-MSbatching, i.e., for $a = 48, 47, 46$. Next, after performing 3-GC when $a = 45$ is reached, 2-MSbatching is applied. Then, 2-MSbatching is performed until enough garbage commitments are accumulated for 4-MSbatching,

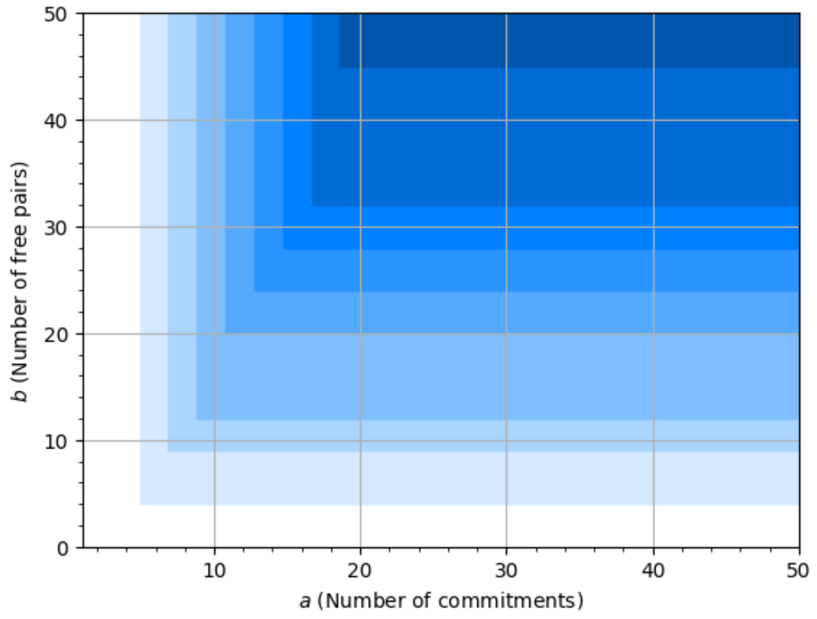


Fig. 2: The area to which k-MSbatching can be applied

i.e., until $a = 39$. When $a = 37$ is reached, 8-GC is performed and then 4-MSbatching is applied. In the same way, continue 4-MSbatching for a while and apply 8-MSbatching at $a = 17$. Then apply 4-MSbatching, 2-MSbatching, and 1-MSbatching for $a = 9, 5, 3, 2$.

Since the number of transitions above is 20, the number of shuffles in the proposed protocol is 20. Since the number of shuffles for the protocol introduced in Sect. 3.3 is 28, it is a successful improvement.

We show the number of shuffles in our protocol for $2 \le n \le 50$ in Figure 4.

6 Search for Optimal Protocols

In this section, we verify whether the protocol proposed in Sect. 5 is optimal in the sense that it minimizes the number of shuffles among all the MSbatching protocols. Specifically, we find shortest move-sequences in the MSbatching move-sequence problem by a dynamic programming algorithm in the range up to $n = 500$.

First, in Sect. 6.1, two lemmas are given to narrow the space where we have to search. Next, in Sect. 6.2, we present the strategy for finding shortest move-sequences. After that, in Sect. 6.3, we compare the number of shuffles in the proposed protocol with the minimum number of shuffles.

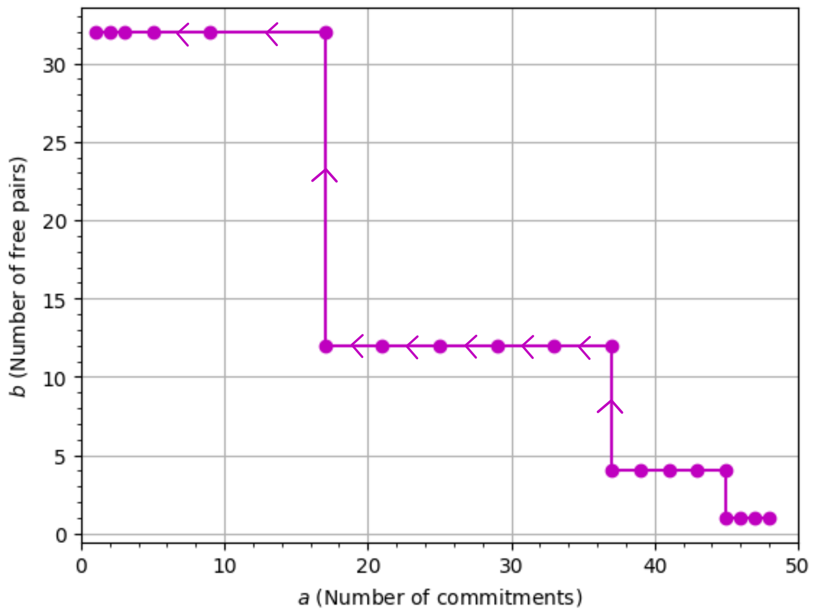


Fig. 3: Proposed protocol for $n = 48$

6.1 Lemmas to Narrow Search Space

When searching for shortest move-sequences in the MSbatching move-sequence problem presented in Sect. 4, the following two lemmas imply that there are transitions that do not need to be considered, narrowing the search space. Hereafter, n is fixed and $M(a, b)$ denotes the shortest move-sequence length from point (a, b) to a terminal point.

The following Lemma 1 indicates that whenever the garbage collection is performed, it should be done on all the remaining garbage commitments.

Lemma 1 *For any g and g' such that $1 \leq g < g' \leq n - a - b + 1$, $M(a, b + g) \geq M(a, b + g')$.*

Proof. We prove that $M(a, b + g) \geq M(a, b + g + 1)$ because it implies the lemma. Let P be a shortest move-sequence from point $(a, b + g)$ to a terminal point, and use induction on the length of P. When P has length 1, $(a, b + g)$ is terminated by some k-MSbatching. Since the same k-MSbatching can be applied to $(a, b + g + 1)$, $M(a, b + g) = M(a, b + g + 1) = 1$ and the claim holds. Assume inductively that when P has length 2 or more, the claim holds for those having a smaller length. When the first move of P transitions upward from the point $(a, b + g)$ to $GC^{g''}(a, b) = (a, b + g'')$, the point $(a, b + g + 1)$ can also transition to the same point $(a, b + g'')$ (or $b + g + 1 = b + g''$), and hence, $M(a, b + g) = M(a, b + g'') + 1$ and $M(a, b + g + 1) \leq M(a, b + g'') + 1$, from which we have $M(a, b + g) \geq M(a, b + g + 1)$ as desired. When the first move of P is leftward, some k-MSbatching transitions to $B^k(a, b+g) = (a-k, b+g)$ and $M(a, b+g) =$

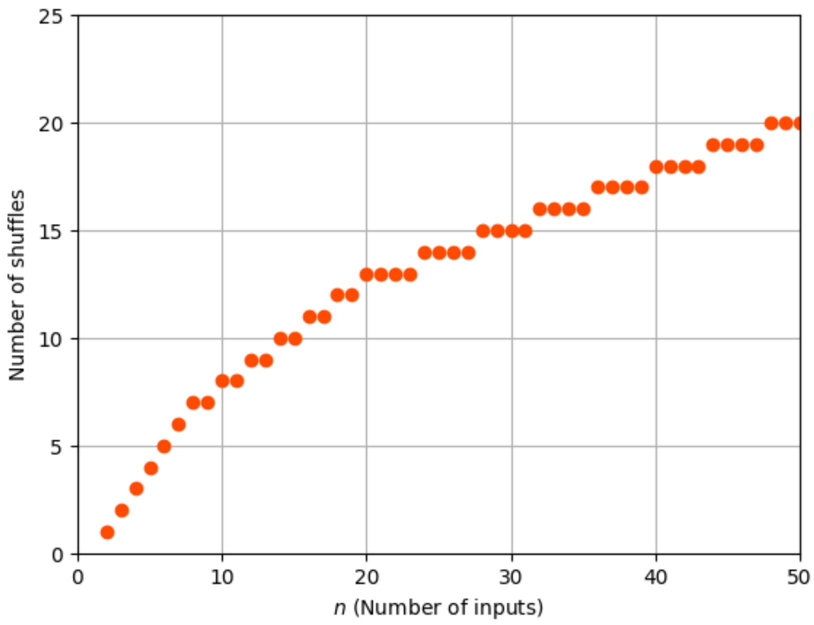


Fig. 4: Number of shuffles in our protocol for $2 \leq n \leq 50$

$M(a-k, b+g)+1$. Since the same k-MSbatching can be applied to the point $(a, b+g+1)$ and $B^k(a, b+g+1) = (a-k, b+g+1)$, we have $M(a, b+g+1) \leq M(a-k, b+g+1)+1$. Also, from the induction assumption, $M(a-k, b+g) \geq M(a-k, b+g+1)$. Therefore, the claim holds. □

The following Lemma 2 indicates that when MSbatching is perfomed, it should be the largest size.

Lemma 2 *If $a < a'$, then $M(a, b) \leq M(a', b)$.*

Proof. We prove that $M(a, b) \leq M(a+1, b)$ because it implies the lemma. Let P be a shortest move-sequence from point $(a+1, b)$ to a terminal point, and use induction on the length of P. When P has length 1, we have $a+1 = 2$, i.e., $a = 1$. Therefore, since $M(a, b) = 0$, the claim holds. Assume inductively that when P has length 2 or more, the claim holds for those having a smaller length. When the first move of P is upward, there exists some g such that $M(a+1, b) = M(a+1, b+g)+1$. Also, from the induction assumption, $M(a, b+g) \leq M(a+1, b+g)$. Since $M(a, b) \leq M(a, b+g)+1$, the claim holds. When the first move of P is leftward, by some k-MSbatching, $M(a+1, b) = M(a-k+1, b)+1$ and $a+1 \geq 2k$. Therefore, $M(a, b) \leq M(a-(k-1), b)+1 = M(a-k+1, b)+1$ since $(k-1)$-MSbatching can be applied to the point (a, b). Therefore, the claim holds. □

Algorithm 1 Shortest move-sequence search

```
1:  function MIN_ COST($n$)
2:      $n \leftarrow$ number of inputs
3:      SMS$[n][n+1]$
4:      for $1 \leq a \leq n$ do
5:          for $n-a+1 \geq b \geq 1$ do
6:              if $b == n-a+1$ then
7:                  SMS$[a][b]$ = SMS$[a-k_b][b]+1$
8:              else if $b < n-a+1$ then
9:                  if SMS$[a][n-a+1] \leq$ SMS$[a-k_b][b]$ then
10:                     SMS$[a][b]$ = SMS$[a][n-a+1]+1$
11:                 else
12:                     SMS$[a][b]$ = SMS$[a-k_b][b]+1$
13:                 end if
14:             end if
15:         end for
16:     end for
17: end function
```

6.2 Shortest Move-Sequence Search

Here, the shortest move-sequence length for each $n \leq 500$ is obtained by running a dynamic programming algorithm for the search space of the MSbatching move-sequence problem, which is narrowed by Lemmas 1 and 2 in Sect. 6.1. The pseudo code for the search is shown in Algorithm 1. The algorithm is explained below.

To search the entire move-sequence space from point $(n, 1)$ to a terminal point, we define a 2-dimensional array named SMS (Shortest Move Sequence) in Line 3 of Algorithm 1.

First, the search range is explained. Since it is assumed that the move-sequence is explored in reverse, a is processed in ascending order over the range of $1 \leq a \leq n$ and b is processed in descending order over the range of $1 \leq b \leq n-a+1$. When $b = n-a+1$, it means that there is no garbage commitment, which is the boundary. Figure 5 shows the search area (boundary) when $n = 48$, for example.

Next, the transition selection method is explained. Line 6 is the condition on the boundary $b = n-a+1$ shown in Figure 5. Since no further GC can be performed on the boundary, add 1 to the value at the transition moved to the left by k_b-MSbatching, i.e., SMS$[a-k_b][b]$. The conditions from Line 8 are about the inside of the search area. Line 9 adds 1 to SMS$[a][n-a+1]$ since Transition 2 is better. Line 11 adds 1 to SMS$[a-k_b][b]$ because Transition 1 is better.

This algorithm was executed on a computer for up to $n = 500$.

6.3 Comparison

Here, we compare the number of shuffles for the protocol proposed in Sect. 5 with the shortest move-sequence length calculated in Sect. 6.2.

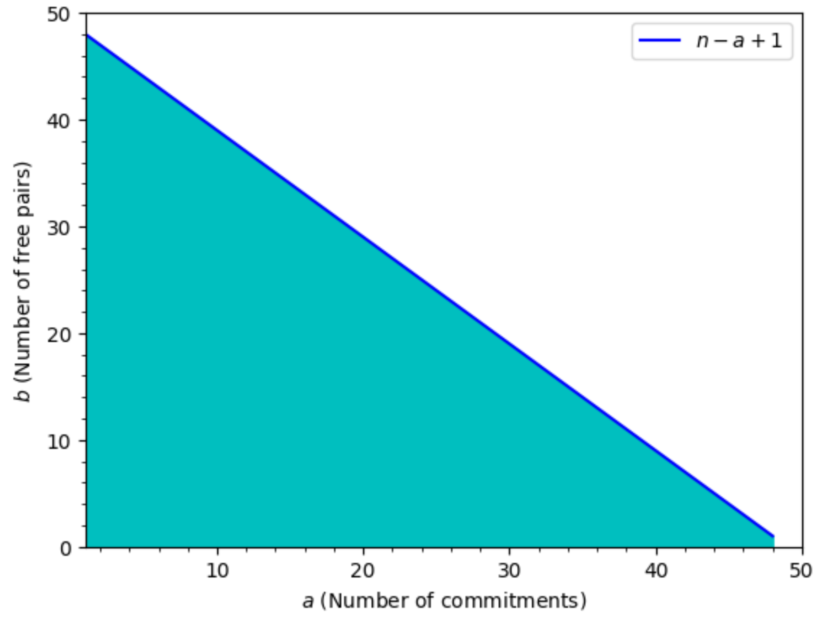


Fig. 5: Possible points to visit for $n = 48$

As mentioned above, the search was performed within $n \leq 500$ of inputs, and the shortest move-sequence lengths were obtained by a computer. Compared to the number of shuffles for the proposed protocol in Sect. 5, in many cases they are consistent and the proposed protocol is optimal. On the other hand, in some cases, the proposed protocol is not optimal, and specifically, there is a protocol that is better than the proposed protocol at n as shown in Table 2.

As shown in Table 2, the number of shuffles of the proposed protocol is found to be minimum or only one more than the minimum. Although only the shortest lengths are shown in Table 2 when the proposed protocol is not optimal, the shortest move-sequences themselves were of course obtained for all the range $n \leq 500$.

No rule of thumb has been found for cases where the proposed protocol is not optimal. Nor is the specific procedure for giving an optimal protocol known. While it is interesting to consider these issues, since card-based cryptography is expected to be performed by human hands, it may be sufficient to have optimal protocols up to $n = 500$ figured out.

7 Conclusion

In this paper, we gave a natural class of committed-format two-helping-card n-input AND protocols based on the batching technique and the MS-AND protocol, and showed optimal protocols among them in terms of the minimum number of shuffles up to $n = 500$.

Table 2: Numbers of inputs that do not minimize the number of shuffles in the proposed protocol

Number of inputs n	Proposed protocol's shuffles	Shortest length
48 - 49	20	19
56	21	20
104	27	26
112 - 115	28	27
128 - 129	29	28
240 - 247	36	35
256 - 271	37	36
288 - 295	38	37
320 - 323	39	38
352	40	39

Note that the "optimality" here was discussed within the class of MSbatching protocols, and hence, it is still open to determine whether the upper bounds on the number of shuffles obtained in this paper are also lower bounds on the number of shuffles for *any* committed-format two-helping-card AND protocols (that are not necessarily based on the MS-AND protocol or the batching technique).

Acknowledgements

We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. This work was supported in part by JSPS KAKENHI Grant Numbers JP21K11881 and JP23H00479.

References

1. Abe, Y., Nakai, T., Kuroki, Y., Suzuki, S., Koga, Y., Watanabe, Y., Iwamoto, M., Ohta, K.: Efficient card-based majority voting protocols. New Gener. Comput. **40**, 173–198 (2022), https://doi.org/10.1007/s00354-022-00161-7
2. Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) Advances in Cryptology—CRYPTO' 93. LNCS, vol. 773, pp. 319–330. Springer, Berlin, Heidelberg (1994), https://doi.org/10.1007/3-540-48329-2_27
3. Haga, R., Hayashi, Y., Miyahara, D., Mizuki, T.: Card-minimal protocols for three-input functions with standard playing cards. In: Batina, L., Daemen, J. (eds.) Progress in Cryptology—AFRICACRYPT 2022. LNCS, vol. 13503, pp. 448–468. Springer, Cham (2022)
4. Hand, S., Koch, A., Lafourcade, P., Miyahara, D., Robert, L.: Check alternating patterns: A physical zero-knowledge proof for Moon-or-Sun. In: Shikata, J., Kuzuno, H. (eds.) Advances in Information and Computer Security. LNCS, vol. 14128, pp. 255–272. Springer, Cham (2023), https://doi.org/10.1007/978-3-031-41326-1_14

5. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Calude, C.S., Dinneen, M.J. (eds.) Unconventional Computation and Natural Computation. LNCS, vol. 9252, pp. 215–226. Springer, Cham (2015), https://doi.org/10.1007/978-3-319-21819-9_16
6. Isuzugawa, R., Toyoda, K., Sasaki, Y., Miyahara, D., Mizuki, T.: A card-minimal three-input AND protocol using two shuffles. In: Chen, C.Y., Hon, W.K., Hung, L.J., Lee, C.W. (eds.) Computing and Combinatorics. LNCS, vol. 13025, pp. 668–679. Springer, Cham (2021), https://doi.org/10.1007/978-3-030-89543-3_55
7. Koch, A.: The landscape of security from physical assumptions. In: IEEE Information Theory Workshop. pp. 1–6. IEEE, NY (2021), https://doi.org/10.1109/ITW48936.2021.9611501
8. Koch, A.: The landscape of optimal card-based protocols. Mathematical Cryptology **1**(2), 115–131 (2022), https://journals.flvc.org/mathcryptology/article/view/130529
9. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology—ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer, Berlin, Heidelberg (2015), https://doi.org/10.1007/978-3-662-48797-6_32
10. Komano, Y., Mizuki, T.: Card-based zero-knowledge proof protocol for pancake sorting. In: Bella, G., Doinea, M., Janicke, H. (eds.) Innovative Security Solutions for Information Technology and Communications. LNCS, vol. 13809, pp. 222–239. Springer, Cham (2023), https://doi.org/10.1007/978-3-031-32636-3_13
11. Kuzuma, T., Toyoda, K., Miyahara, D., Mizuki, T.: Card-based single-shuffle protocols for secure multiple-input AND and XOR computations. In: ASIA Public-Key Cryptography. pp. 51–58. ACM, NY (2022), https://doi.org/10.1145/3494105.3526236
12. Manabe, Y., Ono, H.: Card-based cryptographic protocols with malicious players using private operations. New Gener. Comput. **40**, 67–93 (2022), https://doi.org/10.1007/s00354-021-00148-w
13. Miyahara, D., Mizuki, T.: Secure computations through checking suits of playing cards. In: Frontiers in Algorithmics. Lecture Notes in Computer Science, Springer, Cham (2022), to appear
14. Miyamoto, K., Shinagawa, K.: Graph automorphism shuffles from pile-scramble shuffles. New Gener. Comput. **40**, 199–223 (2022), https://doi.org/10.1007/s00354-022-00164-4
15. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theor. Comput. Sci. **622**(C), 34–44 (2016), https://doi.org/10.1016/j.tcs.2016.01.039
16. Mizuki, T.: Preface: Special issue on card-based cryptography. New Gener. Comput. **39**, 1–2 (2021), https://doi.org/10.1007/s00354-021-00127-1
17. Mizuki, T.: Preface: Special issue on card-based cryptography 2. New Gener. Comput. **40**, 47–48 (2022), https://doi.org/10.1007/s00354-022-00170-6
18. Mizuki, T., Komano, Y.: Information leakage due to operative errors in card-based protocols. Inf. Comput. **285**, 104910 (2022), https://doi.org/10.1016/j.ic.2022.104910
19. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Trans. Fundam. **E100.A**(1), 3–11 (2017), https://doi.org/10.1587/transfun.E100.A.3
20. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) Frontiers in Algorithmics. LNCS, vol. 5598, pp. 358–369. Springer, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-642-02270-8_36

21. Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: Secure computation for threshold functions with physical cards: Power of private permutations. New Gener. Comput. **40**, 95–113 (2022), https://doi.org/10.1007/s00354-022-00153-7
22. Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theor. Comput. Sci. **191**(1–2), 173–183 (1998), https://doi.org/10.1016/S0304-3975(97)00107-2
23. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Physical ZKP protocols for Nurimisaki and Kurodoko. Theor. Comput. Sci. **972**, 114071 (2023), https://doi.org/10.1016/j.tcs.2023.114071
24. Ruangwises, S.: Physical zero-knowledge proof for ball sort puzzle. In: Della Vedova, G., Dundua, B., Lempp, S., Manea, F. (eds.) Unity of Logic and Computation. LNCS, vol. 13967, pp. 246–257. Springer, Cham (2023), https://doi.org/10.1007/978-3-031-36978-0_20
25. Ruangwises, S., Itoh, T.: AND protocols using only uniform shuffles. In: van Bevern, R., Kucherov, G. (eds.) Computer Science–Theory and Applications. LNCS, vol. 11532, pp. 349–358. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-19955-5_30
26. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. Theor. Comput. Sci. **887**, 99–110 (2021), https://doi.org/10.1016/j.tcs.2021.07.007
27. Shikata, H., Miyahara, D., Mizuki, T.: Few-helping-card protocols for some wider class of symmetric Boolean functions with arbitrary ranges. In: 10th ACM Asia Public-Key Cryptography Workshop. pp. 33–41. APKC ’23, ACM, New York (2023), https://doi.org/10.1145/3591866.3593073
28. Shikata, H., Toyoda, K., Miyahara, D., Mizuki, T.: Card-minimal protocols for symmetric boolean functions of more than seven inputs. In: Seidl, H., Liu, Z., Pasareanu, C.S. (eds.) Theoretical Aspects of Computing – ICTAC 2022. pp. 388–406. Springer International Publishing, Cham (2022)
29. Shimano, M., Sakiyama, K., Miyahara, D.: Towards verifying physical assumption in card-based cryptography. In: Bella, G., Doinea, M., Janicke, H. (eds.) Innovative Security Solutions for Information Technology and Communications. LNCS, vol. 13809, pp. 289–305. Springer, Cham (2023), https://doi.org/10.1007/978-3-031-32636-3_17
30. Shinagawa, K., Miyamoto, K.: Automorphism shuffles for graphs and hypergraphs and its applications. IEICE Trans. Fundam. **E106.A**(3), 306–314 (2023), https://doi.org/10.1587/transfun.2022CIP0020
31. Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any Boolean circuit. Discrete Applied Mathematics **289**, 248–261 (2021), https://doi.org/10.1016/j.dam.2020.10.013
32. Shinagawa, K., Nuida, K.: Single-shuffle full-open card-based protocols imply private simultaneous messages protocols. Cryptology ePrint Archive, Paper 2022/1306 (2022), https://eprint.iacr.org/2022/1306, https://eprint.iacr.org/2022/1306
33. Stiglic, A.: Computations with a deck of cards. Theor. Comput. Sci. **259**(1–2), 671–678 (2001), https://doi.org/10.1016/S0304-3975(00)00409-6
34. Suga, Y.: A classification proof for commutative three-element semigroups with local AND structure and its application to card-based protocols. In: 2022 IEEE International Conference on Consumer Electronics - Taiwan. pp. 171–172. IEEE, NY (2022), https://doi.org/10.1109/ICCE-Taiwan55306.2022.9869063
35. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. Int. J. Inf. Secur. **19**(4), 445–452 (2020), https://doi.org/10.1007/s10207-019-00463-w
36. Yao, A.C.: Protocols for secure computations. In: Foundations of Computer Science. pp. 160–164. IEEE Computer Society, Washington, DC, USA (1982), https://doi.org/10.1109/SFCS.1982.88