

# Efficient Card-Based Millionaires’ Protocols via Non-Binary Input Encoding

Koji Nuida<sup>1,2</sup>

<sup>1</sup> Institute of Mathematics for Industry (IMI), Kyushu University, Japan

nuida@imi.kyushu-u.ac.jp

<sup>2</sup> National Institute of Advanced Industrial Science and Technology (AIST), Japan

## Abstract

Comparison of integers, a traditional topic in secure multiparty computation since Yao’s pioneering work on “Millionaires’ Problem” (FOCS 1982), is also well studied in card-based cryptography. For the problem, Miyahara et al. (Theoretical Computer Science, 2020) proposed a protocol using binary cards (i.e., cards with two kinds of symbols) that is highly efficient in terms of numbers of cards and shuffles, and its extension to number cards (i.e., cards with distinct symbols). In this paper, with a different design strategy which we name “Tug-of-War Technique”, we propose new protocols based on binary cards and on number cards. For binary cards, our protocol improves the previous protocol asymptotically (in bit lengths of input integers) in terms of numbers of cards and shuffles when adopting ternary encoding of input integers. For number cards, at the cost of increasing the number of cards, our protocol improves the number of shuffles of the previous protocol even with binary encoding, and more with  $q$ -ary encoding where  $q > 2$ .

*Keywords:* Card-based protocols, integer comparison, Millionaires’ Problem

## 1 Introduction

Secure multiparty computation (MPC) is a cryptographic technology that enables multiple parties to collaboratively derive a correct output while keeping each party’s input secret to the other parties. The notion of MPC was proposed by Yao [12] in 1982, where he discussed the problem of securely comparing two integers, nowadays called “Yao’s Millionaires’ Problem”. Besides such historical importance, comparison of integers has also been an active research topic in the area of MPC from the viewpoints of theoretical interests and ubiquitous practical applications. Moreover, it is also a major topic in card-based cryptography, which is a research area where MPC (as well as zero-knowledge proofs, etc.) is executed by using physical (non-electronic) objects such as a deck of cards [1].

Most of the card-based protocols in the literature are designed assuming a deck of “binary cards” with two kinds of front-side symbols, frequently denoted by  $\clubsuit$  and  $\heartsuit$ , and identical back-side symbols. There is also a line of studies for card-based protocols using a deck of “number cards” [8] where all cards have distinct front-side symbols, such as the standard playing cards with 52 different symbols (or 53, including a joker). An essential ingredient of card-based protocols is shuffle operations, which permute the sequence of cards according to some probability distribution, untraceably by any party. (We note that there are also card-based protocols using so-called private permutations [7] instead of shuffles, which are out of the scope of the present work.) The efficiency of card-based protocols is mainly evaluated by the numbers of cards and shuffles.

Among the existing card-based protocols based on binary cards in the literature for comparison of two input integers in a range, say  $\{0, 1, \dots, m - 1\}$ , a protocol proposed by Miyahara et al. [3] has the smallest number of cards to the authors’ best knowledge and also uses a fairly small number of shuffles. Namely, their

protocol uses  $4\ell + 2$  cards and  $2\ell - 1$  shuffles where  $\ell := \lceil \log_2 m \rceil$  denotes the bit length of each input integer. Their basic strategy itself is elementary: they compare the two bits in the binary expressions of two input integers from the most significant to the least significant bits, and adopt the earliest non-tie result. They implemented this strategy by combining known efficient protocols for fundamental gates and also applying some optimizations. In particular, they used the standard encoding of each bit of an input integer into the order of two cards  and . As their protocol requires just two extra cards in addition to the  $4\ell$  cards for encoding two  $\ell$ -bit integers, the number of cards cannot be largely reduced when assuming this encoding rule. Consequently, any significant improvement in the number of cards requires a change of encoding rules.

In fact, there is a known encoding technique for a non-binary, say  $q$ -ary value in  $\{0, 1, \dots, q - 1\}$  (with  $q > 2$ ) using the  $q$  possible positions of a single card  in a sequence of  $q$  cards where the other  $q - 1$  cards are . By using  $q = 3$ , i.e., ternary expressions of integers, the number of cards needed in encoding two input integers can be reduced in comparison to the binary encoding above (due to the effect that the number of digits is decreased). However, the previous protocol in [3] relies on known efficient protocols specific to binary AND and XOR gates, therefore it is not directly extendible to non-binary input encoding. A possible strategy is to apply some efficient protocol tailored to comparison of non-binary (and fairly small) integers, recursively from the most significant to the least significant digits. For example, Nakai et al. [7] proposed a card-based implementation of Yao’s original protocol in [12] using private permutations, and its conversion to the shuffle model (i.e., without private permutations) is also described in [3]. However, this protocol has a crucial drawback from the present viewpoint, inherited from Yao’s protocol, that it is in nature for only distinguishing two cases  $\alpha < \beta$  and  $\alpha \geq \beta$ , while such a recursive usage as above requires distinguishing three cases  $\alpha < \beta$ ,  $\alpha > \beta$ , and  $\alpha = \beta$ . To the authors’ best knowledge, an efficient card-based protocol for non-binary inputs that can distinguish these three cases is not known in the literature.

## 1.1 Our Contributions

In this paper, we introduce a new design strategy for integer comparison protocols, which we name “Tug-of-War Technique”, and propose a new card-based protocol based on the technique. Our Tug-of-War subprotocol distinguishes three cases  $\alpha_\nu < \beta_\nu$ ,  $\alpha_\nu = \beta_\nu$ , and  $\alpha_\nu > \beta_\nu$  for  $q$ -ary digits  $\alpha_\nu$  and  $\beta_\nu$  of input integers  $\alpha$  and  $\beta$ , respectively, therefore it can realize the strategy for recursive comparison mentioned in the previous paragraph. Table 1 shows a comparison of the efficiency of our proposed protocol with the previous protocol in [3], where some cells also include asymptotic values (when  $\log_2 m \rightarrow \infty$ ) of the numbers of cards or shuffles. See Section 2.4 for definitions of the shuffles appearing in the table. When using binary encoding, our proposed protocol is slightly less efficient compared to [3] in both numbers of cards and shuffles. On the other hand, our proposed protocol with ternary encoding achieves better efficiency than [3]; asymptotically, the number of cards is reduced to approximately 94.6% and the number of shuffles is reduced to 63.1%. (We note that the shuffles used in [3] are random bisection cuts, which are the simplest case of pile-shifting shuffles; while our protocol requires not only random bisection cuts but also more complicated pile-shifting shuffles.) In addition, Table 1 also includes the state-of-the-art protocol by Ono et al. [9] among card-based garbled circuits [11], which uses just a single shuffle but a significantly larger (though still linear in  $\log_2 m$ ) number of cards. See Section 3.4 for the details of the comparison of efficiency.

**Technical Overview.** We explain the central idea of our proposed Tug-of-War Technique for comparison of two input digits  $\alpha_\nu$  and  $\beta_\nu$ , by using the following simple but *insecure* protocol as an example:

1. Arrange cards sequentially (from left to right), where the card  (in default) at the middle point  $O$  is given in advance and the others are ’s.
2. Alice shifts the sequence by  $\alpha_\nu$  to the left.
3. Bob shifts the sequence by  $\beta_\nu$  to the right.

Table 1: Comparison of (committed-format) protocols for comparing integers  $\alpha, \beta \in \{0, 1, \dots, m-1\}$  (with binary output distinguishing  $\alpha < \beta$  and  $\alpha \geq \beta$ ): here PSS, CS, and US denote a pile-shifting shuffle, a complete shuffle, and a (complicated) uniform shuffle, respectively; for (\*), the bottom row is under an assumption that the cards can be shuffled before generating input commitments and each party is allowed to generate an input commitment by using private permutations (not counted in the table)

Cards	Protocols	# of Cards	# of Shuffles
Binary	Miyahara et al. [3], §6	$4\lceil \log_2 m \rceil + 2$ $\sim 4 \log_2 m$	$(2\lceil \log_2 m \rceil - 1)$ PSS $\sim 2 \log_2 m$
	Ono et al. [9]	$28\lceil \log_2 m \rceil - 16$ $\sim 28 \log_2 m$	1 US
	<b>Ours</b> , §3 $q$ -ary encoding	$2q\lceil \log_q m \rceil + 2q - 1$ $\sim (2q/\log_2 q) \log_2 m$	$(2\lceil \log_q m \rceil + 2)$ PSS $\sim (2/\log_2 q) \log_2 m$
	<b>Ours</b> , §3 binary encoding ( $q = 2$ )	$4\lceil \log_2 m \rceil + 3$ $\sim 4 \log_2 m$	$(2\lceil \log_2 m \rceil + 2)$ PSS $\sim 2 \log_2 m$
	<b>Ours</b> , §3 ternary encoding ( $q = 3$ )	$6\lceil \log_3 m \rceil + 5$ $\sim (3.7855 \dots) \log_2 m$	$(2\lceil \log_3 m \rceil + 2)$ PSS $\sim (1.2618 \dots) \log_2 m$
Number	Miyahara et al. [3], §7	$4\lceil \log_2 m \rceil + 4$	$(6\lceil \log_2 m \rceil - 2)$ PSS
	<b>Ours</b> , §4 $q$ -ary encoding (*)	$(6q - 3)\lceil \log_q m \rceil + 3$	$(3\lceil \log_q m \rceil + 2)$ PSS +1 CS
		$(5q - 3)\lceil \log_q m \rceil + 3$	$(2\lceil \log_q m \rceil + 2)$ PSS +1 CS
	<b>Ours</b> , §4 binary encoding ( $q = 2$ ) (*)	$9\lceil \log_2 m \rceil + 3$	$(3\lceil \log_2 m \rceil + 2)$ PSS +1 CS
$7\lceil \log_2 m \rceil + 3$		$(2\lceil \log_2 m \rceil + 2)$ PSS +1 CS	

- Alice (respectively, Bob) takes the cards at the left (respectively, right) side, exclusively relative to the point  $O$ ; a card at the point  $O$  remains not taken.

It is easily seen that, if  $\alpha_\nu > \beta_\nu$  (respectively,  $\alpha_\nu < \beta_\nu$ ), then Alice (respectively, Bob) takes the card  $\boxed{\heartsuit}$  at the end of the protocol; while if  $\alpha_\nu = \beta_\nu$ , then the card  $\boxed{\heartsuit}$  is not taken. (Intuitively, the party who pulls the rope of cards more strongly will get the card  $\boxed{\heartsuit}$ .) Moreover, when this protocol is repeated recursively from the most significant to the least significant digits, the card  $\boxed{\heartsuit}$  is not taken until  $\alpha_\nu \neq \beta_\nu$  holds, and once  $\alpha_\nu \neq \beta_\nu$  holds,  $\boxed{\heartsuit}$  is taken by the party with larger input at the round and  $\boxed{\heartsuit}$  does not appear in the subsequent rounds. This realizes correct comparison of input integers. Although the protocol above is insecure (as the position of  $\boxed{\heartsuit}$  at the end of a round tells the difference of the input digits), we can convert it into a secure protocol by applying a known technique of computing subtraction of integers (e.g., [10]) and using some post-processing phase to securely determine who has taken the  $\boxed{\heartsuit}$ .

**Extension to Number Cards.** The previous paper [3] also gives a variant of the comparison protocol that uses number cards instead of binary cards. To the authors' best knowledge, it is the protocol with minimal number of cards among such existing protocols using number cards in the literature. In fact, the number of cards is even kept almost unchanged from the case of binary cards. On the other hand, the number of shuffles is increased to almost three times larger than the case of binary cards. This is because the efficient protocols for fundamental gates used originally are tailored to binary cards, and their protocol with number cards has to rely on less efficient building-block protocols.

In contrast, our proposed protocol with binary cards has an advantage that the underlying mechanism,

especially that for Tug-of-War subprotocol, is not deeply dependent on the characteristic of binary cards. Our variant of the protocol using number cards is basically obtained by partitioning the number cards in a deck into “♣-cards” and “♥-cards” first and then replacing the originally used binary cards  $\boxed{\clubsuit}$  and  $\boxed{\heartsuit}$  with random ♣-cards and ♥-cards, respectively. Here, in fact, we need some additional care about the usage of number cards; e.g., some previously opened cards were reused in later steps for our protocol based on binary cards, but such reuse of cards should be avoided in the case of number cards to achieve the security. Even considering such points, our proposed protocol is still significantly more efficient than [3] in terms of the number of shuffles, while it uses an almost twice larger number of cards than [3]. See Table 1 for more information on the comparison with the previous work, and Section 4.3 for the details. We note also that the number of shuffles in our protocol can be decreased further (at the cost of increasing the number of cards) by using  $q$ -ary encoding with  $q > 2$ . It is a future research topic to improve further the trade-off between the number of cards and the number of shuffles.

## 1.2 Organization of the Paper

In Section 2, we summarize basic notations (Section 2.1) and our model of card-based protocols (Section 2.2), including the encoding rule of input integers (Section 2.3) and definitions of shuffles (Section 2.4). In Section 3, we introduce the Tug-of-War subprotocol (Section 3.1), and then describe our main protocol (Section 3.2). We also discuss a way of modifying the protocol to deal with ternary and/or non-committed outputs (Section 3.3), and explain a comparison with previous work (Section 3.4). In Section 4, we explain how an integer is encoded using number cards (Section 4.1), describe the extension of our proposed protocol based on binary cards to number cards (Section 4.2), and explain about the comparison of our protocols with the previous protocol (Section 4.3).

# 2 Preliminaries

## 2.1 Notations

In the paper, we let  $[a..b] := \{a, a + 1, \dots, b\} \subseteq \mathbb{Z}$  for two integers  $a, b$ . Let  $S_n$  denote the symmetric group on  $n$  letters. For a condition  $P$ , we define  $\chi[P] := 1$  if  $P$  is satisfied and  $\chi[P] := 0$  if  $P$  is not satisfied.

Let  $\vec{q} = (q_0, q_1, \dots, q_{\ell-1})$  be a sequence of  $\ell$  integers  $q_i \geq 2$ . Then we define

$$|\vec{q}| := q_0 + \dots + q_{\ell-1}, \pi(\vec{q}) := q_0 q_1 \dots q_{\ell-1}, \max(\vec{q}) := \max\{q_0, \dots, q_{\ell-1}\} .$$

Now in the same way as ordinary  $q$ -ary expressions of integers, for any  $a \in [0.. \pi(\vec{q}) - 1]$ , there is a unique expression of the form

$$a = \sum_{k=0}^{\ell-1} a_k q_0 q_1 \dots q_{k-1}, a_k \in [0.. q_k - 1]$$

(where  $q_0 q_1 \dots q_{k-1} := 1$  when  $k = 0$ ). We call it the  $\vec{q}$ -ary expression of  $a$  and write  $a = (a_{\ell-1} \dots a_1 a_0)_{\vec{q}}$ . When the  $q_k$ 's are a constant value  $q$  (as in the case of ordinary  $q$ -ary expressions), we also write the sequence  $\vec{q}$  as  $q^{\times \ell}$ .

## 2.2 Model of Card-Based Protocols

In the paper, we put the following assumptions on the cards. Each card has a front-side symbol and a back-side symbol. There are two possible states for cards, “face-up” and “face-down”, where only the front-side (respectively, back-side) symbol is visible for a face-up (respectively, face-down) card. All cards have identical back-side symbols, denoted by ‘?’, and hence face-down cards are indistinguishable from each other. Face-up cards with identical front-side symbols are also indistinguishable from each other. For the purpose of explanation, we often write a face-down card with invisible front-side symbol  $s$  as  $\boxed{[s]}$ .

In the paper, we deal with the following two kinds of cards.

**Binary cards** The front-side symbols are  $\clubsuit$  or  $\heartsuit$ .

**Number cards** There are a number, say  $N$ , of cards in the deck and each card has a mutually distinct number from 1 to  $N$  as the front-side symbol.

The card-based protocols in the paper can be modeled in the following manner (see e.g., [5] for a more formal treatment of card-based protocols). Here we only deal with protocols played by two parties, say Alice and Bob with inputs  $\alpha$  and  $\beta$ , respectively, and suppose that the number  $L$  of steps in a protocol is constant (possibly depending on a public parameter).

**Initial sequence** The initial card sequence is the concatenation of three sequences  $\text{In}_1(\alpha)$ ,  $\text{In}_2(\beta)$ , and  $\text{Aux}$ , where  $\text{In}_1(\alpha)$  and  $\text{In}_2(\beta)$  are sequences of face-down cards determined by a certain encoding rule from the input values  $\alpha$  and  $\beta$ , respectively, and  $\text{Aux}$  is a publicly known sequence of face-down cards. Moreover, a list  $\text{Vis}$  of “visible sequences” is initialized to be empty.

**Main loop** In  $\nu$ -th step ( $\nu = 1, 2, \dots, L$ ), the parties execute one of the following operations on the card sequence according to the current content of  $\text{Vis}$ , where  $M$  denotes the number of cards in the sequence:

**Permutation** Permute the cards in the sequence according to a publicly known permutation  $\sigma \in S_M$ .

**Turn** Switch the states of some (possibly multiple) cards, where we write “open” (respectively, “turn down”) to mean changing from “face-down” to “face-up” (respectively, from “face-up” to “face-down”).

**Shuffle** Choose a permutation  $\sigma \in S_M$  according to some probability distribution, and permute the card sequence according to  $\sigma$  in a way that  $\sigma$  is kept secret for both parties.

Let  $\text{vis}_\nu$  denote the sequence of visible symbols of the card sequence, called the *visible sequence*, after the operation above. The parties append  $\text{vis}_\nu$  to  $\text{Vis}$ . For example, if the card sequence after the operation is  $\boxed{\clubsuit} \boxed{\heartsuit} \boxed{?} \boxed{?} \boxed{\heartsuit}$ , then  $\text{vis}_\nu = (\clubsuit, \heartsuit, ?, ?, \heartsuit)$ . Let  $\text{Vis}_{\text{Out}}$  denote the list  $\text{Vis}$  after the  $L$ -th step.

**Output (for “non-committed-format” protocols)** According to a certain rule from the list  $\text{Vis}_{\text{Out}}$ , the parties output a value, denoted by  $\text{Out}$ .

**Output (for “committed-format” protocols)** According to a certain rule from the list  $\text{Vis}_{\text{Out}}$ , the parties determine a list of positions, pick up the (face-down) cards at those positions in the card sequence, and output the sequence of those picked cards. Now set  $\text{Out}$  to be the empty list.

We say that a protocol is *secure* (in the semi-honest model) if for any party with input  $\gamma = \alpha$  or  $\gamma = \beta$ , the conditional probability distribution of  $\text{Vis}_{\text{Out}}$  conditioned on a value of  $(\gamma, \text{Out})$  is independent of the other input  $\beta$  or  $\alpha$ .

In what follows, a description of a concrete protocol may use some intuitive explanations in order to improve the readability. For example, we allow a protocol to arrange (a part of) the cards in multiple rows rather than a single sequence, or to store cards into (or take cards from) some separate “regions”; such operations can in fact be simulated by some appropriate operations performed on a single sequence of cards as in the model above.

### 2.3 Encoding of Input Values

We use the following two kinds of encodings of an integer into binary cards.

- We use the term *k-vector commitment* of  $a \in [0..k-1]$  and write  $\text{vEnc}_k(a)$  to mean a sequence of  $k$  face-down binary cards in which the  $a$ -th card (counted from the 0-th) from the right has front-side symbol  $\heartsuit$  and the others have front-side symbols  $\clubsuit$ . For example, we have  $\text{vEnc}_5(3) = \boxed{\clubsuit} \boxed{\heartsuit} \boxed{\clubsuit} \boxed{\clubsuit} \boxed{\clubsuit}$ . When  $k = 2$ , it is the same as the standard definition of commitments of bits in card-based cryptography.

- Let  $\vec{q} = (q_0, q_1, \dots, q_{\ell-1})$  be a sequence of  $\ell$  integers  $q_i \geq 2$ . For any integer  $a \in [0.. \pi(\vec{q}) - 1]$  with  $a = (a_{\ell-1} \cdots a_1 a_0)_{\vec{q}}$  (see Section 2.1 for the notations), we use the term  $\vec{q}$ -ary commitment of  $a$  and write  $\mathbf{qEnc}_{\vec{q}}(a)$  to mean the sequence of  $q_k$ -vector commitments of each  $a_k$ :

$$\mathbf{qEnc}_{\vec{q}}(a) = (\mathbf{vEnc}_{q_{\ell-1}}(a_{\ell-1}), \dots, \mathbf{vEnc}_{q_1}(a_1), \mathbf{vEnc}_{q_0}(a_0)) .$$

When  $\vec{q} = 2^{\times \ell}$ , it is the same as the standard definition of commitments of integers, which is used in e.g., [3].

## 2.4 Shuffles Used in the Paper

**Complete Shuffle (CS)** This is a shuffle that permutes all the target cards uniformly at random.

**Pile-Shifting Shuffle (PSS) [2]** Suppose that an array of cards with  $r$  rows and  $c$  columns is given. A *column*-PSS is a shuffle that cyclically rotates the  $c$  columns uniformly at random (where the rows are synchronized). For example, a column-PSS applied to the following  $3 \times 3$  array of cards

[1]	[2]	[3]
[4]	[5]	[6]
[7]	[8]	[9]

yields one of the following three arrays with probability 1/3 each:

[1]	[2]	[3]	,	[2]	[3]	[1]	,	[3]	[1]	[2]	.
[4]	[5]	[6]		[5]	[6]	[4]		[6]	[4]	[5]	
[7]	[8]	[9]		[8]	[9]	[7]		[9]	[7]	[8]	

We also define a *row*-PSS by switching the roles of rows and columns in a column-PSS. In the example above, a row-PSS yields one of the following three arrays with probability 1/3 each:

[1]	[2]	[3]	,	[4]	[5]	[6]	,	[7]	[8]	[9]	.
[4]	[5]	[6]		[7]	[8]	[9]		[1]	[2]	[3]	
[7]	[8]	[9]		[1]	[2]	[3]		[4]	[5]	[6]	

We note that column-PSSs with  $c = 2$  columns and row-PSSs with  $r = 2$  rows are equivalent to *random bisection cuts* (RBCs) [6].

## 3 Our Proposed Protocol with Binary Cards

### 3.1 Tug-of-War Subprotocol

In this subsection, we introduce a subprotocol as in Protocol 1 used in our proposed protocol, which we name *Tug-of-War subprotocol*. In both of the subprotocol and our main protocol (Section 3.2), we use five separate regions named “Main Memory”, “♣-Garage”, “♥-Garage”, “Memory A”, and “Memory B”. Main Memory is endowed with two-dimensional coordinates and each card is placed at an integer point. For integers  $i \geq 1$  and  $j$ , we write “Point  $(i, j)$ ” to mean the  $j$ -th column (counted from left to right) of the  $i$ -th row (counted from top to bottom) in Main Memory. ♣-Garage and ♥-Garage are for storing face-down cards  $\boxed{\clubsuit}$  and  $\boxed{\heartsuit}$ , respectively. Memory A and Memory B are for storing face-down cards taken during a protocol by Alice and Bob, respectively.

---

**Protocol 1** Our Tug-of-War subprotocol using binary cards

---

**Input**  $q$ -vector commitments  $\mathbf{vEnc}_q(\alpha), \mathbf{vEnc}_q(\beta)$  of  $\alpha, \beta \in [0..q-1]$

(Main Memory involves only a single face-down card, say  $C$ , and it is at Point  $(1, 0)$ ; and  $\clubsuit$ -Garage must involve at least  $2q - 2$  cards  $\boxed{\clubsuit}$ )

**Output** A single face-down card, say  $C'$ , at Point  $(1, 0)$  in Main Memory, and  $q - 1$  new cards each in Memory A and Memory B, among which at most one card is  $\boxed{\heartsuit}$  and the others are  $\boxed{\clubsuit}$ , where

- if  $C = \boxed{\heartsuit}$  and  $\alpha > \beta$  (respectively,  $\alpha < \beta$ ), then  $\boxed{\heartsuit}$  is appended to Memory A (respectively, Memory B);
- if  $C = \boxed{\heartsuit}$  and  $\alpha = \beta$ , then  $C' = \boxed{\heartsuit}$ ;
- if  $C = \boxed{\clubsuit}$ , then all those cards are  $\boxed{\clubsuit}$ ;

and two new cards  $\boxed{\heartsuit}$  in  $\heartsuit$ -Garage

- 1: For each  $j \in [1..q-1]$ , move a face-down card from  $\clubsuit$ -Garage to Point  $(1, -j)$ .
  - 2: Put the cards in  $\mathbf{vEnc}_q(\alpha)$  at Points  $(2, -(q-1))$  to  $(2, 0)$  in reverse order; i.e., the rightmost (the 0-th) card of  $\mathbf{vEnc}_q(\alpha)$  comes to Point  $(2, -(q-1))$ .
  - 3: Apply column-PSS to Main Memory, and open all cards in the second row.
  - 4: For Main Memory, rotate the columns in the two rows synchronously in a way that  $\boxed{\heartsuit}$  in the second row comes to the leftmost place (i.e., Point  $(2, -(q-1))$ ).
  - 5: Move  $\boxed{\heartsuit}$  and  $q - 1$   $\boxed{\clubsuit}$ 's in the second row to  $\heartsuit$ -Garage and  $\clubsuit$ -Garage, respectively, and turn them face-down.
  - 6: For each  $j \in [1..q-1]$ , move two face-down cards from  $\clubsuit$ -Garage to Points  $(1, j)$  and  $(2, -j)$ , respectively.
  - 7: Put the cards in  $\mathbf{vEnc}_q(\beta)$  at Points  $(2, 0)$  to  $(2, q-1)$ .
  - 8: Apply column-PSS to Main Memory, and open all cards in the second row.
  - 9: For Main Memory, rotate the columns in the two rows synchronously in a way that  $\boxed{\heartsuit}$  in the second row comes to the rightmost place (i.e., Point  $(2, q-1)$ ).
  - 10: Move  $\boxed{\heartsuit}$  and  $2q - 2$   $\boxed{\clubsuit}$ 's in the second row to  $\heartsuit$ -Garage and  $\clubsuit$ -Garage, respectively, and turn them face-down.
  - 11: Move the cards at Points  $(1, j)$ ,  $j < 0$  by Alice to Memory A, and move the cards at Points  $(1, j)$ ,  $j > 0$  by Bob to Memory B.
-

Figure 1 shows an example of Protocol 1 for  $q = 4$ ,  $\alpha = 3$ , and  $\beta = 1$ , i.e.,  $\mathbf{vEnc}_q(\alpha) = \boxed{\heartsuit} \boxed{\clubsuit} \boxed{\clubsuit} \boxed{\clubsuit}$  and  $\mathbf{vEnc}_q(\beta) = \boxed{\clubsuit} \boxed{\clubsuit} \boxed{\heartsuit} \boxed{\clubsuit}$ . Here we suppose that the card originally put at Point  $(1, 0)$  is  $\boxed{\heartsuit}$ . In the figure, ‘\*’ denotes an empty point in Main Memory at which a card will be put. The four numbers in the right side show how the numbers of cards in  $\clubsuit$ -Garage,  $\heartsuit$ -Garage, Memory A, and Memory B are changed. In the example, at Step 11, one  $\boxed{\heartsuit}$  and  $q - 2 = 2$   $\boxed{\clubsuit}$ ’s are taken by Alice and  $q - 1 = 3$   $\boxed{\clubsuit}$ ’s are taken by Bob.

Let  $C$  be the card originally put at Point  $(1, 0)$ . In a general case, if the inputs satisfy that  $\alpha > \beta$  (respectively,  $\alpha < \beta$ ), then the card  $C$  moves  $\alpha - \beta$  cells to the left (respectively,  $\beta - \alpha$  cells to the right) through Steps 1–10, therefore  $C$  is taken by Alice (respectively, Bob) at Step 11. On the other hand, if  $\alpha = \beta$ , then the card  $C$  is still at the middle after Step 10, therefore  $C$  remains not taken at Step 11. As for the security, regardless of the input values, the visible sequences (focusing only on the second row) after Steps 3 and 8 are uniformly random sequences of  $\heartsuit$  and  $q - 1$  (respectively,  $2q - 2$ )  $\clubsuit$ ’s, and those after Steps 4 and 9 are public constants specified by the protocol. Hence no information leaks during Protocol 1.

### 3.2 The Main Protocol

Based on Tug-of-War subprotocol in Section 3.1, we describe our proposed protocol using binary cards in Protocol 2. For  $\vec{q}$ -ary commitments  $\mathbf{qEnc}_{\vec{q}}(\alpha)$  and  $\mathbf{qEnc}_{\vec{q}}(\beta)$  of input values  $\alpha = (\alpha_{\ell-1} \cdots \alpha_1 \alpha_0)_{\vec{q}}$  and  $\beta = (\beta_{\ell-1} \cdots \beta_1 \beta_0)_{\vec{q}}$ , respectively, first suppose that  $\alpha \neq \beta$  and  $\nu_0$  is the largest index with  $\alpha_{\nu_0} \neq \beta_{\nu_0}$ . Then the card  $\boxed{\heartsuit}$  is not taken by Alice nor by Bob in Tug-of-War subprotocols during the loops with  $\nu > \nu_0$ , and at the loop with  $\nu = \nu_0$ ,  $\boxed{\heartsuit}$  is taken by Alice (respectively, Bob) if  $\alpha_{\nu_0} > \beta_{\nu_0}$  (respectively,  $\alpha_{\nu_0} < \beta_{\nu_0}$ ). On the other hand, if  $\alpha = \beta$ , then the card  $\boxed{\heartsuit}$  is not taken by Alice nor by Bob during all the loops. As a result, for the 0-th to  $\lambda$ -th columns (where  $\lambda := |\vec{q}| - \ell$ ) after Step 5,  $\boxed{\heartsuit}$  exists in the first row if  $\alpha \geq \beta$  and in the second row if  $\alpha < \beta$ , while the other cards are  $\boxed{\clubsuit}$ . Now when  $\alpha \geq \beta$ , the two  $\boxed{\heartsuit}$ ’s in Main Memory are at the same (first) row after Step 5, which implies that  $\boxed{\heartsuit}$  comes to Point  $(2, -1)$  after Step 8 and therefore the protocol outputs the 2-vector commitment of the bit  $0 = \chi[\alpha < \beta]$ . On the other hand, when  $\alpha < \beta$ , the two  $\boxed{\heartsuit}$ ’s in Main Memory are at different rows after Step 5, which implies that  $\boxed{\heartsuit}$  comes to Point  $(1, -1)$  after Step 8 and therefore the protocol outputs the 2-vector commitment of the bit  $1 = \chi[\alpha < \beta]$ . Hence Protocol 2 outputs  $\mathbf{vEnc}_2(\chi[\alpha < \beta])$  correctly.

As for the security, it was shown in Section 3.1 that Tug-of-War subprotocol leaks no information. For the remaining steps, as the 0-th to  $\lambda$ -th columns after Step 5 involves precisely one  $\boxed{\heartsuit}$ , the column-PSS in Step 6 has an effect equivalent to the complete shuffle for these columns. Consequently, the visible sequence (focusing only on the 0-th to  $\lambda$ -th columns) after Step 7 is a uniformly random  $2 \times (\lambda + 1)$  array of one  $\heartsuit$  and  $2\lambda + 1$   $\clubsuit$ ’s regardless of the input values, and the visible sequence after Step 8 is uniquely determined by that after Step 7. Hence Protocol 2 is secure.

As for the efficiency, Protocol 2 uses  $2\ell$   $\boxed{\heartsuit}$ ’s and  $2|\vec{q}| - 2\ell$   $\boxed{\clubsuit}$ ’s for the two  $\vec{q}$ -ary commitments; and one  $\boxed{\heartsuit}$  and  $2 \cdot \max(\vec{q}) - 2$   $\boxed{\clubsuit}$ ’s prepared in Step 1 (the cards moved to  $\clubsuit$ -Garage and  $\heartsuit$ -Garage during Tug-of-War subprotocols can be reused as the extra  $\boxed{\heartsuit}$  and  $\boxed{\clubsuit}$ ’s in Step 5). Therefore the protocol uses  $2\ell + 1$   $\boxed{\heartsuit}$ ’s and  $2|\vec{q}| - 2\ell + 2 \cdot \max(\vec{q}) - 2$   $\boxed{\clubsuit}$ ’s, hence  $2|\vec{q}| + 2 \cdot \max(\vec{q}) - 1$  cards in total. When  $\vec{q} = q^{\times \ell}$ , these values are  $2\ell + 1$ ,  $2(q - 1)\ell + 2q - 2$ , and  $2q\ell + 2q - 1$ . On the other hand, Protocol 2 uses two PSSs per one Tug-of-War subprotocol and two PSSs after the loops, hence  $2\ell + 2$  PSSs in total.

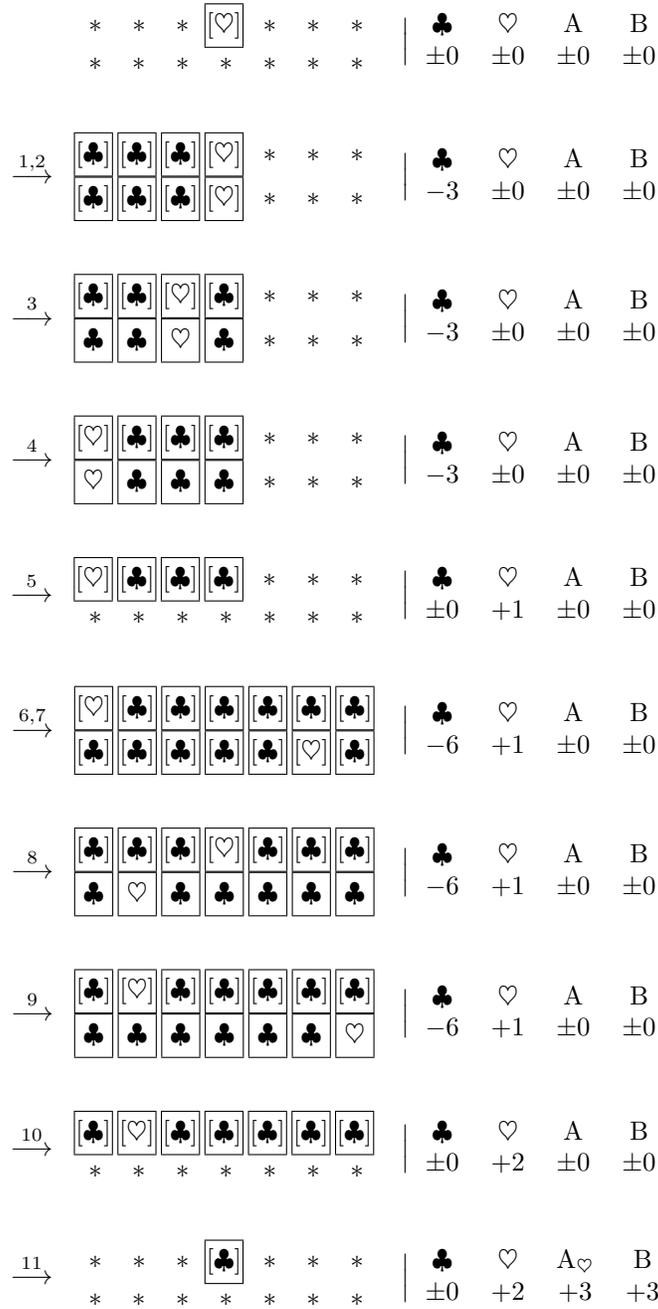


Figure 1: Example of Protocol 1 for  $q = 4$ ,  $\alpha = 3$ ,  $\beta = 1$ : the pictures after Step 3 and Step 8 show some possible results of PSS

---

**Protocol 2** Our proposed protocol using binary cards (committed-format)

---

**Input**  $\vec{q}$ -ary commitment  $\mathbf{qEnc}_{\vec{q}}(\alpha)$  of  $\alpha = (\alpha_{\ell-1} \cdots \alpha_1 \alpha_0)_{\vec{q}}$ , and  $\vec{q}$ -ary commitment  $\mathbf{qEnc}_{\vec{q}}(\beta)$  of  $\beta = (\beta_{\ell-1} \cdots \beta_1 \beta_0)_{\vec{q}}$

**Output** 2-vector commitment  $\mathbf{vEnc}_2(b)$  of the bit  $b = \chi[\alpha < \beta]$

- 1: Put a face-down card  $\boxed{\heartsuit}$  at Point  $(1, 0)$ ; put  $2 \cdot \max(\vec{q}) - 2$  face-down cards  $\boxed{\clubsuit}$  in  $\clubsuit$ -Garage; and initialize  $\heartsuit$ -Garage, Memory A, and Memory B to be empty.
  - 2: **for**  $\nu = \ell - 1, \dots, 1, 0$  **do**
  - 3:   Execute Tug-of-War subprotocol using  $q_\nu$ -vector commitments  $\mathbf{vEnc}_{q_\nu}(\alpha_\nu)$  and  $\mathbf{vEnc}_{q_\nu}(\beta_\nu)$  contained in  $\mathbf{qEnc}_{\vec{q}}(\alpha)$  and  $\mathbf{qEnc}_{\vec{q}}(\beta)$ , respectively.
  - 4: **end for** // Each of Memory A and Memory B involves  $\lambda := |\vec{q}| - \ell$  cards
  - 5: Move a card from  $\heartsuit$ -Garage to Point  $(1, -1)$ ; put all cards in Memory A at Points  $(1, 1), (1, 2), \dots, (1, \lambda)$ ; move two cards from  $\clubsuit$ -Garage to Points  $(2, -1)$  and  $(2, 0)$ ; and put all cards in Memory B at Points  $(2, 1), (2, 2), \dots, (2, \lambda)$ .
  - 6: Apply column-PSS to the 0-th to  $\lambda$ -th columns in Main Memory.
  - 7: Apply row-PSS to all cards in Main Memory, and open the cards in the 0-th to  $\lambda$ -th columns in Main Memory.
  - 8: For Main Memory, rotate the rows synchronously in a way that the face-up card  $\boxed{\heartsuit}$  comes to the bottom (i.e., the second row).
  - 9: Output a pair of the face-down cards at Points  $(1, -1)$  and  $(2, -1)$  in this order.
- 

### 3.3 Options for the Protocol

**Ternary Output.** We can modify Protocol 2 in a way that it will output the 3-vector commitment  $\mathbf{vEnc}_3(b)$  of the value  $b$  given by  $b = 0$  if  $\alpha > \beta$ ,  $b = 1$  if  $\alpha = \beta$ , and  $b = 2$  if  $\alpha < \beta$ . Instead of the array of cards with two rows in Step 5, we let the protocol arrange the face-down cards in a way that the first row is a sequence of  $\boxed{\heartsuit}$  and the cards in Memory A; the second row is a sequence of  $\boxed{\clubsuit}$ , the card remained (at Point  $(1, 0)$ ) after the loops, and  $\lambda - 1$   $\boxed{\clubsuit}$ 's from  $\clubsuit$ -Garage; and the third row is a sequence of  $\boxed{\clubsuit}$  and the cards in Memory B. (The correspondence of the value of  $b$  and the three conditions can be made arbitrarily by changing the order of the rows appropriately.) Steps 6–8 are executed similarly, with the only difference that the number of rows is increased by one and the number of columns is decreased by one. Finally, in Step 9, the output is changed to the triplet of the cards at Points  $(1, -1)$ ,  $(2, -1)$  and  $(3, -1)$  in this order. As for the number of cards, the new Step 5 uses one extra  $\boxed{\heartsuit}$  and  $\lambda + 1 = |\vec{q}| - \ell + 1$  extra  $\boxed{\clubsuit}$ 's, therefore we have to put  $\max\{2 \cdot \max(\vec{q}) - 2, |\vec{q}| - \ell + 1\}$   $\boxed{\clubsuit}$ 's to  $\clubsuit$ -Garage at Step 1. Hence this option requires  $2\ell + 1$   $\boxed{\heartsuit}$ 's and  $2|\vec{q}| - 2\ell + \max\{2 \cdot \max(\vec{q}) - 2, |\vec{q}| - \ell + 1\}$   $\boxed{\clubsuit}$ 's, therefore  $2|\vec{q}| + \max\{2 \cdot \max(\vec{q}) - 1, |\vec{q}| - \ell + 2\}$  cards in total. When  $\vec{q} = q^{\times \ell}$  and  $\ell \geq 2$ , these values are  $2\ell + 1$ ,  $3q\ell - 3\ell + 1$ , and  $3q\ell - \ell + 2$ . On the other hand, when  $\vec{q} = q^{\times 1}$ , these values are 3,  $4q - 4$ , and  $4q - 1$  (note that  $q \geq 2$ ). As for the shuffles, the number of shuffles is not changed by adopting the option.

**Non-Committed Format.** Protocol 2 is easily converted into a non-committed-format protocol by just opening the cards after the column-PSS in Step 6 and tells at which row the  $\boxed{\heartsuit}$  appears (we have  $\alpha \geq \beta$  if it is the first row and  $\alpha < \beta$  if it is the second row). This option does not require the two cards put at the  $(-1)$ -th column (among which the  $\boxed{\clubsuit}$  is a reused card) nor the row-PSS in Step 7. As a result, the number of  $\boxed{\heartsuit}$ 's is decreased by one and the number of  $\boxed{\clubsuit}$ 's is not changed, while the number of PSSs is decreased by one.

This non-committed-format option can be also modified to output a ternary value. In contrast to the committed-format case, here we do not need an extra row. Instead, we just change the procedure in the

Table 2: Comparison of the numbers of binary cards

Protocols	# of Cards		
	Total	For Inputs	Additional
Miyahara et al. [3], §6	$4\lceil\log_2 m\rceil + 2$	$4\lceil\log_2 m\rceil$	2
<b>Ours, Protocol 2 (any <math>q</math>)</b>	$2q\lceil\log_q m\rceil + 2q - 1$	$2q\lceil\log_q m\rceil$	$2q - 1$
<b>Ours, Protocol 2 (<math>q = 2</math>)</b>	$4\lceil\log_2 m\rceil + 3$	$4\lceil\log_2 m\rceil$	3
<b>Ours, Protocol 2 (<math>q = 3</math>)</b>	$6\lceil\log_3 m\rceil + 5$	$6\lceil\log_3 m\rceil$	5

previous paragraph in a way that the last column-PSS is applied to the 1-th (rather than the 0-th) to  $\lambda$ -th columns; the three possibilities can be distinguished by observing the position of  $\boxed{\heartsuit}$  among the opened cards. This even does not require the card put at Position (2, 0), but as it was a reused card, the number of cards (as well as the number of shuffles) used throughout the protocol is not changed from the previous paragraph.

### 3.4 Comparison to the Previous Work

To the authors' best knowledge, the existing committed-format protocol in the literature for integer comparison (with binary output) using minimal number of binary cards is the protocol by Miyahara et al. [3, Section 6]. Table 2 shows a comparison of the numbers of cards for our proposed protocol (with binary output in the committed format, i.e., the original Protocol 2) and for the previous one mentioned above. Here  $m$  denotes the size of the input domain, i.e.,  $\alpha, \beta \in [0..m - 1]$ . For our protocol, we simply use the parameter  $\vec{q} = q^{\times\ell}$  (though the use of  $\vec{q}$  with differing components  $q_\nu$  may sometimes decrease the number of cards further), where  $\ell = \lceil\log_q m\rceil$  to satisfy the requirement  $\pi(\vec{q}) \geq m$ . From the table we observe that, our protocol with  $q = 2$  requires one more additional card compared to [3] and consequently, the total number of cards for our protocol is also increased by one. On the other hand, when we use  $q = 3$ , the number of additional cards in our protocol is increased further by two, but now the number of cards for input commitments is asymptotically decreased (note that  $6\lceil\log_3 m\rceil \sim (3.7855 \dots) \cdot (\log_2 m)$ ) due to the effect that the number of digits in the input integers is decreased. As the latter effect is dominant, the total number of cards is also asymptotically decreased.

As for the shuffles, the protocol in [3] has  $\lceil\log_2 m\rceil$  loops. Each loop uses the six-card AND protocol in [6] with one RBC, which is equivalent (in our terminology) to row-PSS for two rows, and, except for the first loop, uses the six-card XOR protocol in [6] with one RBC. In total, the protocol in [3] uses  $2\lceil\log_2 m\rceil - 1$  PSSs. On the other hand, our protocol (with  $\vec{q} = q^{\times\ell}$ ) uses  $2\lceil\log_q m\rceil + 2$  PSSs. This number is larger than [3] by three when  $q = 2$ , while it is again asymptotically smaller than [3] when  $q = 3$ .

We also take into account another line of researches on card-based garbled circuits using only a single shuffle invented by Shinagawa and Nuida [11]. Among its follow-up work, the most efficient (committed-format) protocol known at the present (in the shuffle-based model as in our work, without private permutations) is the one by Ono et al. [9], using  $6g + 2n_{\text{in}} + 2n_{\text{out}}$  cards and a single (uniform, but not necessarily closed) shuffle, where  $g$  is the number of gates for a circuit to be computed,  $n_{\text{in}}$  is the number of input bits, and  $n_{\text{out}}$  is the number of output bits. By setting  $n_{\text{in}} = 2\lceil\log_2 m\rceil$  and  $n_{\text{out}} = 1$  as in our problem and applying it to the integer comparison circuit used in [3] with  $g = 4\lceil\log_2 m\rceil - 3$  gates, it follows that the number of cards used by the protocol of [9] becomes  $28\lceil\log_2 m\rceil - 16$ , which is still linear in  $\lceil\log_2 m\rceil$  but is significantly larger.

## 4 Our Proposed Protocol with Number Cards

### 4.1 Encoding of Integers Using Number Cards

Here we extend the definitions of commitments of integers based on binary cards to number cards. We partition the number cards into “♣-cards” and “♥-cards”. In our protocol, each ♣-card plays the role of a card  $\boxed{\clubsuit}$ , and each ♥-card plays the role of a card  $\boxed{\heartsuit}$ . Now the definitions of  $k$ -vector commitments and  $\vec{q}$ -ary commitments are naturally generalized and written in the same notations as the original, by replacing any binary card  $\boxed{\clubsuit}$  with some ♣-card and replacing any binary card  $\boxed{\heartsuit}$  with some ♥-card. For example, when ♣-cards and ♥-cards have front-side symbols in  $[1..7]$  and in  $\{8, 9\}$ , respectively, the two sequences  $\boxed{[0]}\boxed{[8]}\boxed{[3]}\boxed{[6]}$  and  $\boxed{[4]}\boxed{[9]}\boxed{[0]}\boxed{[1]}$  are examples of 4-vector commitments  $\mathbf{vEnc}_4(2)$  of an integer 2.

### 4.2 Extension of Our Protocol to Number Cards

Our proposed protocol with number cards is obtained basically by replacing the cards  $\boxed{\clubsuit}$  and  $\boxed{\heartsuit}$  in Protocol 2 with ♣-cards and ♥-cards, respectively. Therefore the correctness is inherited from the original protocol.

Regarding the security, the problem in contrast to the case of binary cards is that now the ♣-cards have non-identical front-side symbols. Consequently, the distribution of the ♣-cards opened simultaneously at some step should be independent of the input values. In particular, unlike Protocol 2 where some cards  $\boxed{\clubsuit}$  opened previously may be reused later, such reuse of ♣-cards in the current case violates the independency requirement. Hence we have to remove from the protocol the ♣-cards opened during a Tug-of-War subprotocol without being reused. As a result, each, say  $\nu$ -th, Tug-of-War subprotocol now requires  $3(q_\nu - 1)$  new ♣-cards, and the steps after the loops requires two more new ♣-cards. On the other hand, we do not need to concern about the distributions of the ♥-cards in the protocol and therefore a ♥-card opened at some loop may be reused in Step 5, as at most one ♥-card is opened at each step and the order of turns for ♥-cards in the protocol is determined independently of the input values. These arguments change the number of cards as follows: we need one ♥-card and  $3|\vec{q}| - 3\ell + 2$  ♣-cards in addition to  $2|\vec{q}|$  cards for input commitments, hence  $5|\vec{q}| - 3\ell + 3$  number cards in total. When  $\vec{q} = q^{\times \ell}$ , the last value is  $(5q - 3)\ell + 3$ .

In case where each input commitment is made from scratch at the beginning of the protocol by the party knowing the input value, the condition above on the distribution of ♣-cards can be satisfied in the following manner:

1. Before making the input commitments, a CS is applied to all the face-down ♣-cards. Then sufficient numbers of face-down ♣-cards and ♥-cards are separately given to each of Alice and Bob.
2. Alice generates a commitment  $\mathbf{qEnc}_{\vec{q}}(\alpha)$  by permuting her face-down cards privately and appropriately. Bob generates a commitment  $\mathbf{qEnc}_{\vec{q}}(\beta)$  by permuting his face-down cards privately and appropriately.

Although the second step seems to require private permutations, such usage of private permutations might be allowable, as it is reasonable to assume that an input commitment of each party is anyway generated privately. If we adopt this option, the whole protocol uses  $2\ell + 2$  PSSs and one CS.

In what follows, we consider another possibility that  $\vec{q}$ -ary commitments  $\mathbf{qEnc}_{\vec{q}}(\alpha)$  and  $\mathbf{qEnc}_{\vec{q}}(\beta)$  of input values are given in advance (e.g., as outputs of some other committed-format protocols). In this case, we assume that

(\*) there are disjoint sets  $S_{i,\nu}$  ( $i \in \{1, 2\}$ ,  $\nu \in [0..\ell - 1]$ ) of ♣-cards satisfying that the choice of ♣-cards in the  $q_\nu$ -vector commitment  $\mathbf{vEnc}_{q_\nu}(\alpha_\nu)$  (respectively,  $\mathbf{vEnc}_{q_\nu}(\beta_\nu)$ ) is uniformly random over all sequences of  $q_\nu - 1$  distinct cards in the set  $S_{1,\nu}$  (respectively,  $S_{2,\nu}$ ).

Now for, say  $\nu$ -th Tug-of-War subprotocol, the second row at Step 3 consists only of the cards from  $\mathbf{vEnc}_{q_\nu}(\alpha_\nu)$ . Therefore, the condition (\*) implies that the sequence of the  $q_\nu$  cards opened at Step 3 is a uniformly random sequence of a ♥-card and uniformly random  $q_\nu - 1$  distinct ♣-cards from  $S_{1,\nu}$ , which

is independent of the input values. On the other hand, if we naively use  $\mathbf{vEnc}_{q_\nu}(\beta_\nu)$  as is, then among the  $2q_\nu - 1$  cards opened at Step 8, the cards from  $\mathbf{vEnc}_{q_\nu}(\beta_\nu)$  become distinguishable from the others and hence the value of  $\beta_\nu$  can be guessed. To avoid the issue, we introduce a pre-processing phase at the beginning of the whole protocol to replace each  $\clubsuit$ -card in  $\mathbf{qEnc}_{\vec{q}}(\beta)$  with a random one among the remaining  $\clubsuit$ -cards (while keeping the committed value), as follows:

1. A CS is applied to all face-down  $\clubsuit$ -cards (not in  $\mathbf{qEnc}_{\vec{q}}(\alpha)$  nor in  $\mathbf{qEnc}_{\vec{q}}(\beta)$ ).
2. For each  $\nu \in [0..\ell - 1]$ , we generate a new  $q_\nu$ -vector commitment  $\mathbf{vEnc}_{q_\nu}(\beta_\nu)$  in the following manner:
  - (a) Reverse the order of cards of the original  $\mathbf{vEnc}_{q_\nu}(\beta_\nu)$ .
  - (b) Put a face-down  $\heartsuit$ -card and  $q_\nu - 1$  face-down  $\clubsuit$ -cards from left to right at the next row, making a  $2 \times q_\nu$  array of cards.
  - (c) Apply a column-PSS to the array, open all cards in the first row (i.e., where the original commitment was placed), and rotate each of the two rows synchronously in a way that the face-up  $\heartsuit$ -card comes to the rightmost. Then use the second row as a new  $\mathbf{vEnc}_{q_\nu}(\beta_\nu)$ .

This pre-processing itself is secure due to the condition (\*). After the pre-processing, the distribution of the  $\clubsuit$ -cards opened during the protocol except for Step 3 of each Tug-of-War subprotocol becomes uniformly random. Hence our protocol is secure as well. The pre-processing phase requires  $|\vec{q}|$  more cards, one CS, and  $\ell$  PSSs. In total, our protocol with this option uses  $6|\vec{q}| - 3\ell + 3$  number cards (which is  $(6q - 3)\ell + 3$  when  $\vec{q} = q^{\times\ell}$ ),  $3\ell + 2$  PSSs, and one CS.

**Options for the Protocol.** By the same idea as Section 3.3, the protocol above can be also modified to have a ternary and/or non-committed output.

- For the case of a ternary and committed output, the steps after the loops now use  $|\vec{q}| - \ell + 1$  new  $\clubsuit$ -cards, while it used two new  $\clubsuit$ -cards in the original protocol. Hence the total number of cards is increased by  $|\vec{q}| - \ell - 1$ , which becomes  $(q - 1)\ell - 1$  when  $\vec{q} = q^{\times\ell}$ .
- For the case of a binary and non-committed output, one of the two new  $\clubsuit$ -cards used after the loops becomes not necessary, therefore the total number of cards is decreased by one compared to the original (the  $\heartsuit$ -card therein is also not necessary, but it was a reused card and does not affect the total number of cards). We also note that a PSS is removed from the original.
- For the case of a ternary and non-committed output, the total number of cards is decreased further by one compared to the binary and non-committed case.

### 4.3 Comparison to the Previous Work

We compare the efficiency of our proposed protocol (with binary and committed-format output) with the number-card version of Miyahara et al.'s protocol [3, Section 7], which is (to the authors' best knowledge) the existing protocol with minimal number of shuffles among those based on number cards in the literature. Similarly to the case of binary cards discussed in Section 3.4, their protocol consists of  $\lceil \log_2 m \rceil$  loops (where  $m$  is the size of the input domain), each using an extended version of AND protocol in [4] and, except for the first loop, an XOR protocol in [4]. The total number of cards is  $4\lceil \log_2 m \rceil + 4$  and it uses  $6\lceil \log_2 m \rceil - 2$  RBCs. See Section 7 of [3] for details. On the other hand, our protocol with given  $\vec{q}$ -ary commitments for inputs (i.e., not made from scratch), where  $\vec{q} = 2^{\times\ell}$ , uses  $9\lceil \log_2 m \rceil + 3$  cards,  $3\lceil \log_2 m \rceil + 2$  PSSs, and one CS. If we allow the parties to make the input commitments from scratch by using private permutations, the total number of cards is decreased to  $7\lceil \log_2 m \rceil + 3$  and the number of shuffles is decreased to  $2\lceil \log_2 m \rceil + 2$  PSSs and one CS. In any case, there is a trade-off between the protocol in [3] and ours, the former using less cards but more shuffles, while the latter using more cards but less shuffles. When using larger parameter  $q \geq 3$ , the number of shuffles in our protocol is decreased further, at the cost of increasing the number of cards further.

## Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP19H01109, Japan.

## References

- [1] C. Crépeau and J. Kilian, “Discreet Solitary Games”, in: Proceedings of CRYPTO 1993, LNCS vol.773, pp.319–330, 1993
- [2] R. Ishikawa, E. Chida, and T. Mizuki, “Efficient Card-Based Protocols for Generating a Hidden Random Permutation Without Fixed Points”, in: Proceedings of UCNC 2015, LNCS vol.9252, pp.215–226, 2015
- [3] D. Miyahara, Y. Hayashi, T. Mizuki, and H. Sone, “Practical Card-Based Implementations of Yao’s Millionaire Protocol”, Theoretical Computer Science, vol.803, pp.207–221, 2020
- [4] T. Mizuki, “Efficient and Secure Multiparty Computations Using a Standard Deck of Playing Cards”, in: Proceedings of CANS 2016, LNCS vol.10052, pp.484–499, 2016
- [5] T. Mizuki and H. Shizuya, “A Formalization of Card-Based Cryptographic Protocols via Abstract Machine”, International Journal of Information Security, vol.13, pp.15–23, 2014
- [6] T. Mizuki and H. Sone, “Six-Card Secure AND and Four-Card Secure XOR”, in: Proceedings of FAW 2009, LNCS vol.5598, pp.358–369, 2009
- [7] T. Nakai, Y. Tokushige, Y. Misawa, M. Iwamoto, and K. Ohta, “Efficient Card-Based Cryptographic Protocols for Millionaires’ Problem Utilizing Private Permutations”, in: Proceedings of CANS 2016, LNCS vol.10052, pp.500–517, 2016
- [8] V. Niemi and A. Renvall, “Solitaire Zero-Knowledge”, Fundamenta Informaticae, vol.38, no.1–2, pp.181–188, 1999
- [9] T. Ono, K. Shinagawa, T. Nakai, Y. Watanabe, and M. Iwamoto, “Card-Based Protocols for Any Boolean Circuit with Six Cards per Gate” (in Japanese), in: Proceedings of 2023 Symposium on Cryptography and Information Security (SCIS 2023), article no.3D2-2, 2023
- [10] K. Shinagawa, T. Mizuki, J. C. N. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto, “Card-Based Protocols Using Regular Polygon Cards”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E100-A, no.9, pp.1900–1909, 2017
- [11] K. Shinagawa and K. Nuida, “A Single Shuffle Is Enough for Secure Card-Based Computation of Any Boolean Circuit”, Discrete Applied Mathematics, vol.289, pp.248–261, 2021
- [12] A. C.-C. Yao, “Protocols for Secure Computations”, in: Proceedings of FOCS 1982, pp.160–164, 1982