Practical Card-Based Implementations of Yao's Millionaire Protocol*

Daiki Miyahara[a], Yu-ichi Hayashi[b], Takaaki Mizuki[c], Hideaki Sone[c]

[a] *Graduate School of Information Sciences, Tohoku University*
[b] *Graduate School of Information Sciences, Nara Institute of Science and Technology*
[c] *Cyberscience Center, Tohoku University*

Abstract

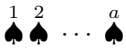
Yao's millionaire protocol enables Alice and Bob to know whether or not Bob is richer than Alice by using a public-key cryptosystem without revealing the actual amounts of their properties. In this paper, we present simple and practical implementations of Yao's millionaire protocol using a physical deck of playing cards; we straightforwardly implement the idea behind Yao's millionaire protocol so that even non-experts can easily understand their correctness and secrecy. Our implementations are based partially on the previous card-based scheme proposed by Nakai, Tokushige, Misawa, Iwamoto, and Ohta; their scheme admits players' private actions on a sequence of cards called Private Permutation (PP), implying that a malicious player could make an active attack (for example, he/she could exchange some of the cards stealthily when doing such a private action). By contrast, our implementations rely on a familiar shuffling operation called a random cut, and hence, they can be conducted completely publicly so as to avoid any active attack. More specifically, we present two card-based implementations of Yao's millionaire protocol; one uses a two-colored deck of cards (which consists of black and red cards), and the other uses a standard deck of playing cards. Furthermore, we also provide card-based protocols that rely on a logical circuit representing the comparison.

Keywords: Card-based protocols, Real-life hands-on cryptography, Secure multi-party computations, Yao's millionaire protocol, Deck of cards

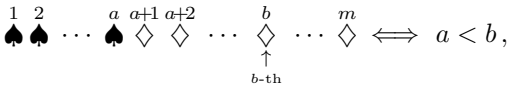
*An earlier version of this paper was presented at the 12th Annual International Conference on Combinatorial Optimization and Applications, COCOA 2018, Atlanta, USA, December 15–17, 2018, and appeared in Proc. COCOA 2018, Vol.11346 of LNCS, pp.246–261, 2018 [1]. This is an accepted manuscript for publication in Theoretical Computer Science. The formal publication is available via DOI: 10.1016/j.tcs.2019.11.005.

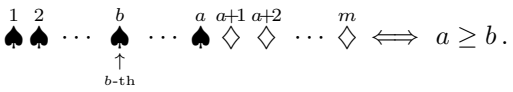
1. Introduction

Assume that Alice and Bob have a and b dollars, respectively, such that $a, b \in \{1, 2, \ldots, m\}$ for some natural number m. They want to know who is richer without revealing any information about their values (more than that is necessary), i.e., they want to determine only whether $a < b$ or not. This is the famous *millionaires' problem* proposed by Yao [2] in 1982, and he designed a protocol, which we call *Yao's millionaire protocol*, to solve the problem based on a public-key cryptosystem. The fundamental principle behind Yao's millionaire protocol could be interpreted as follows. If Alice arranges m symbols consisting of a number a of ♠s and a number $(m-a)$ of ♢s as



and Bob points at the b-th symbol, then the b-th symbol being ♢ implies $a < b$, and the b-th symbol being ♠ implies $a \geq b$:





While Yao's millionaire protocol relies on the public-key cryptosystem to implement the above principle without leaking actual values a and b, Nakai, Tokushige, Misawa, Iwamoto, and Ohta [3] considered the use of a deck of physical cards in 2016. That is, following the fundamental principle above, they constructed a card-based scheme using cards of two types such as

whose backs are all identical ?. Roughly speaking, in their scheme, Alice first encodes her secret value a with a sequence of face-down cards, and then Bob "privately" changes the positions of cards according to his secret value b. We will describe the details in Section 2. Since many people on earth are familiar with playing cards, their card-based scheme is human-friendly and useful. Its only drawback is that it requires a player's "private" action, called *Private Permutation* (PP) [3], which permits Bob to rearrange the sequence of cards privately (for example, he is allowed to manipulate the cards behind his back). Private Permutation is considered to be such a strong assumption that a malicious player may do an active attack. Hereinafter, we refer to their scheme as the *NTMIO protocol with PP*; the acronym NTMIO is made of the initial letters of the names of the authors [3].

Thus, it is preferable to construct a card-based scheme which does not rely on Private Permutation, in order to avoid possible malicious actions. To this

Table 1: The PP-free millionaire protocols.

	Deck	#Cards	#Shuffles	Section
Our implementation with RC	Two-colored	$3m+1$	1	§ 3
Using a standard deck	Standard	$4m$	4	§ 4
The previous circuit-based [3]	Two-colored	$4\lceil \log m \rceil + 4$	$7\lceil \log m \rceil - 6$	§ 5
Our improved circuit-based	Two-colored	$4\lceil \log m \rceil + 2$	$2\lceil \log m \rceil - 1$	§ 6

end, in this paper, we present a "PP-free" scheme, which implements the fundamental principle behind Yao's millionaire protocol; instead of using Private Permutation, we use a familiar shuffling operation called the *random cut* (RC). A random cut is a cyclic shuffle, which can be easily implemented by humans as in the case of usual card games (e.g. [4, 5, 6]). Therefore, our scheme, named the *PP-free protocol with RC*, can be conducted completely publicly, and hence, any malicious action can be detected. As will be seen in Section 3, we straightforwardly implement the above principle. Therefore, we believe that even non-experts can easily understand the correctness and secrecy of our scheme, and can practically use it in everyday life.

Similarly to the NTMIO protocol with PP, our PP-free protocol with RC uses a two-colored deck of cards $\boxed{\clubsuit}\boxed{\clubsuit}\cdots\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\cdots\boxed{\heartsuit}$. Compared with such a two-colored deck of cards, a standard deck of playing cards is more familiar with us. Therefore, we extend our protocol so that we can solve the millionaire problem using a standard deck of playing cards (which is sold in many toy stores all over the world). We note that simply replacing a two-colored deck with a standard deck in the PP-free protocol with RC does not work. Note also that our protocol uses another simple shuffle aside from RC. These results will be presented in Section 4.

It should be noted that Nakai et al. [3] proposed a PP-free protocol as well; they presented a card-based scheme, which follows not the above-mentioned fundamental principle but a logical circuit representing the comparison $a < b$. This PP-free circuit-based protocol relies on a shuffling operation called the random bisection cut [7] (instead of Private Permutation). In this paper, we improve upon this existing protocol; we will reduce the number of required random bisection cuts to around $2/7$. We will explain the details in Sections 5 and 6.

Table 1 summarizes the performance of the PP-free protocols.

The remainder of this paper is organized as follows. In Section 2, we introduce the NTMIO protocol with PP [3]. In Section 3, we present our implementation, the PP-free protocol with RC. We confirm that simply replacing a two-colored deck with a standard deck in the PP-free protocol with RC cannot solve the millionaire problem, and then present how to resolve the issue in Section 4. As for circuit-based protocols, we introduce the previous protocol in Section 5, and give an improved protocol in Section 6. Moreover, for the circuit-

based protocols, we consider the use of a standard deck of cards in Section 7. We conclude this paper in Section 8.

An earlier version of this paper was presented and appeared as a conference paper [1]. The main difference is that we have added Sections 4 and 7 in this extended version of the paper: While the conference paper deals only with a two-colored deck of cards, this paper first proposes implementations using a standard deck of cards.

2. The Previous Scheme: the NTMIO Protocol with PP

In this section, we introduce the NTMIO protocol with PP [3].

Recall the fundamental principle behind Yao's millionaire protocol; Alice arranges m symbols:

$$\overset{1}{\spadesuit}\ \overset{2}{\spadesuit}\ \cdots\ \overset{a}{\spadesuit}\ \overset{a+1}{\diamondsuit}\ \overset{a+2}{\diamondsuit}\ \cdots\ \overset{m}{\diamondsuit}\,.$$

Using a pair of physical cards $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, let us encode each symbol as follows:

$$\boxed{\clubsuit}\boxed{\heartsuit} = \spadesuit,\quad \boxed{\heartsuit}\boxed{\clubsuit} = \diamondsuit\,.$$

Thus, Alice can encode her private value a using m pairs of $\boxed{\clubsuit}\boxed{\heartsuit}$, and put the cards with their faces down such that Bob does not see the order of the cards. For such a sequence of m pairs encoding Alice's secret value a, Bob needs to point at the b-th pair without leaking any information about his secret value b; to this end, Bob is permitted to use Private Permutation. Specifically, the NTMIO protocol with PP proceeds as follows.

1. Alice holding m $\boxed{\clubsuit}$s and m $\boxed{\heartsuit}$s places a number a of $\boxed{\clubsuit}\boxed{\heartsuit}$s on a table with their faces down, and then puts $(m-a)$ $\boxed{\heartsuit}\boxed{\clubsuit}$s next to them:

$$\overset{1}{\boxed{?}\boxed{?}}_{\clubsuit\ \heartsuit}\ \overset{2}{\boxed{?}\boxed{?}}_{\clubsuit\ \heartsuit}\cdots\overset{a}{\boxed{?}\boxed{?}}_{\clubsuit\ \heartsuit}\ \overset{a+1}{\boxed{?}\boxed{?}}_{\heartsuit\ \clubsuit}\ \overset{a+2}{\boxed{?}\boxed{?}}_{\heartsuit\ \clubsuit}\cdots\overset{m}{\boxed{?}\boxed{?}}_{\heartsuit\ \clubsuit},$$

   while Bob does not see the order of each pair.
2. Bob uses Private Permutation; he takes the sequence of cards and move them behind his back. Then, he moves the b-th pair to the first without Alice seeing which pair comes first:

$$\begin{array}{cc} & \overset{1}{\boxed{?}\boxed{?}}\cdots\overset{b-1}{\boxed{?}\boxed{?}}\ \overset{b}{\boxed{?}\boxed{?}}\ \overset{b+1}{\boxed{?}\boxed{?}}\cdots\overset{m}{\boxed{?}\boxed{?}} \\ \rightarrow & \overset{b}{\boxed{?}\boxed{?}}\ \overset{1}{\boxed{?}\boxed{?}}\cdots\overset{b-1}{\boxed{?}\boxed{?}}\ \overset{b+1}{\boxed{?}\boxed{?}}\cdots\overset{m}{\boxed{?}\boxed{?}}\,. \end{array}$$

3. The first pair of cards is revealed.
   - If the revealed cards are $\boxed{\heartsuit}\boxed{\clubsuit}$, $a < b$.
   - If the revealed cards are $\boxed{\clubsuit}\boxed{\heartsuit}$, $a \geq b$.

This is the existing card-based solution to the millionaires' problem using Private Permutation[1]. Let us stress that Bob needs to use Private Permutation in Step 2.

The use of Private Permutation is so powerful as to contribute to improving the efficiency of card-based protocols [3, 10, 11, 12, 13], and also it is used in other physical secure protocols [14, 15]; however, it might lead to some issues. To implement Step 2 of this protocol, the following issues are considered. (1) If Bob were malicious, he could make an active attack; for instance, he could replace the sequence of cards with another set of cards (prepared by himself beforehand) behind his back so that he would be able to peep the exact value of a later; because this attack is done behind his back, Alice does not notice it. (2) Alice and/or audience watching the execution of the protocol could learn Bob's secret value b by observing his tiny shoulder movement. (3) Permuting some cards behind one's back might be challenging because one only has to rely on the sense of hands; the case of $b = 1$ or $b = m$ might be no problem, but if $b = m/2$, Bob might have difficulty in searching the desired pair of cards.

In the next section, we design a simple PP-free protocol.

3. Our Implementation Using a Random Cut

In this section, we present our card-based implementation of Yao's millionaire protocol; instead of relying on Private Permutation, we use

- a random cut (RC), which is a well-known and easy-to-perform shuffle, and
- cards whose backs are $\boxed{\#}$, which is a different pattern from $\boxed{?}$.

3.1. How to Proceed

Our PP-free protocol with RC proceeds as follows.

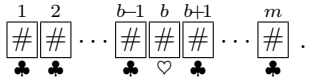
1. Alice holds m $\boxed{\clubsuit}$s and $(m-1)$ $\boxed{\heartsuit}$s. Depending on her secret value a, she places a number a of $\boxed{\clubsuit}$s on a table with their faces down, and then puts a number $(m-a)$ of $\boxed{\heartsuit}$s next to them. The resulting sequence is Alice's input:

$$\overset{1}{\boxed{?}}\underset{\clubsuit}{} \overset{2}{\boxed{?}}\underset{\clubsuit}{} \cdots \overset{a}{\boxed{?}}\underset{\clubsuit}{} \overset{a+1}{\boxed{?}}\underset{\heartsuit}{} \overset{a+2}{\boxed{?}}\underset{\heartsuit}{} \cdots \overset{m}{\boxed{?}}\underset{\heartsuit}{} .$$

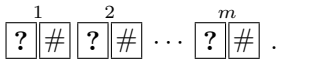
   On the other hand, Bob holds $(m-1)$ cards of $\clubsuit$ whose backs are $\boxed{\#}$ and a card of $\heartsuit$ whose back is also $\boxed{\#}$. Then, he places these m cards with their

[1] It should be noted that Fagin, Naor, and Winkler proposed a similar idea to solve the socialist millionaires' problem [8] where Alice and Bob want to know whether they think the same person in mind or not (see Solution 11 in [9]). In addition, Nakai et al. [3] presented another card-based scheme with Private Permutation, which compares a and b bit by bit with the help of "storage" cards.

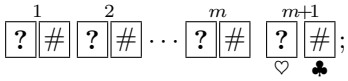
faces down on the table such that only the b-th card is $\heartsuit$. The resulting sequence is Bob's input:



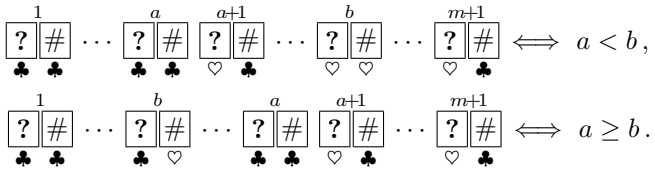
2. Take every card from Alice's input sequence and Bob's input sequence from the left alternately one by one, and put it to the right of the previous card:



We further add two cards to the sequence:

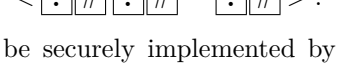


these two cards are put for handling the case of $a = b = m$. Note that recalling the fundamental principle behind Yao's millionaire protocol, the left card of Bob's $\heartsuit$-card determines whether $a < b$ or not:



Note, furthermore, that when $a \geq b$, the $(b{+}1)$-st pair determines whether $a{=}b$ or $a > b$: if the $(b{+}1)$-st pair is $\boxed{\heartsuit}\boxed{\clubsuit}$ then $a{=}b$; if it is $\boxed{\clubsuit}\boxed{\clubsuit}$ then $a > b$. Of course, we cannot open Bob's cards $\boxed{\#}$ now; hence, we add a randomization in the next step.

3. Apply a random cut to the sequence of $(2m{+}2)$ cards, which means shuffling the card sequence cyclically (we denote this operation by $<\cdot>$):



The random cut can be securely implemented by the shuffle operation called the "Hindu cut" [6]; the shuffle may be repeated by Alice and Bob, or even other people until they are all satisfied with the result. Note that the random cut can be done completely publicly [6], and hence, each player can notice any illegal action if any.

4. Reveal all the cards whose backs are $\boxed{\#}$ (namely, the m cards placed by Bob and the additional card); then, one card of $\heartsuit$ appears. Reveal the card on its left.
   - If the revealed card is $\boxed{\heartsuit}$, $a < b$.
   - If the revealed card is $\boxed{\clubsuit}$, we have $a \geq b$. To see whether equality holds or not, open the card to the right of Bob's $\heartsuit$-card (apart from cyclic rotation). If the opened card is $\boxed{\heartsuit}$, $a{=}b$. If it is $\boxed{\clubsuit}$, $a > b$.

This is our PP-free protocol with RC. It uses $(3m+1)$ cards in total and uses one shuffle. In Step 1, Alice places a $\boxed{\clubsuit}$s; if Alice has only a $\boxed{\clubsuit}$s at first, the value a might be leaked from the number of cards that Alice holds. Therefore, Alice needs to have m $\boxed{\clubsuit}$s at first (the number of $\boxed{\heartsuit}$ is similar). Since we apply a random cut in Step 3, revealing Bob's cards in Step 4 does not expose where Bob placed the $\heartsuit$-card. If $a=b$, Alice and Bob will learn the exact value; note that their values are not leaked to any other people watching the execution of the protocol.

As for the use of a different back $\boxed{\#}$, we were inspired by the technique called the "Chosen Cut" that Koch and Walzer proposed [5][2]. If the back-side symbol of the cards is vertically asymmetric, we do not need cards of different backs like $\boxed{\#}$: It suffices that Bob puts his cards upside down as follows:

$$\boxed{?}\boxed{\text{ʖ}}\boxed{?}\boxed{\text{ʖ}}\cdots\boxed{?}\boxed{\text{ʖ}}.$$

Our protocol can be executed completely publicly. Any malicious action will be noticed. Moreover, we can automatically confirm that Bob put his input in a correct format when we reveal all Bob's cards in Step 4. We can even be convinced that Alice put her input in a correct format by applying the idea in [16] with some additional cards.

3.2. A Pseudocode

In this subsection, we present a more formal description of our protocol, that is, we show a pseudocode that follows the computational model of card-based protocols, which was formalized in [17, 18, 19].

First, let us describe an input card sequence. Remember that, for example, if $a = b = 1$, then Alice and Bob will arrange their inputs with two additional cards as:

$$\Gamma^{(1,1)} = (\overbrace{\frac{?}{\clubsuit}, \frac{?}{\heartsuit}, \ldots, \frac{?}{\heartsuit}}^{m \text{ cards}}, \overbrace{\frac{\#}{\heartsuit}, \frac{\#}{\clubsuit}, \ldots, \frac{\#}{\clubsuit}}^{m \text{ cards}}, \frac{?}{\heartsuit}, \frac{\#}{\clubsuit}).$$

Generally, for $a, b \in \{1, 2, \ldots, m\}$, we define

$$\Gamma^{(a,b)} = (\overset{1}{\frac{?}{\clubsuit}}, \ldots, \overset{a-1}{\frac{?}{\clubsuit}}, \overset{a}{\frac{?}{\clubsuit}}, \overset{a+1}{\frac{?}{\heartsuit}}, \ldots, \overset{m}{\frac{?}{\heartsuit}}, \overset{m+1}{\frac{\#}{\clubsuit}}, \ldots, \overset{m+b-1}{\frac{\#}{\clubsuit}}, \overset{m+b}{\frac{\#}{\heartsuit}}, \overset{m+b+1}{\frac{\#}{\clubsuit}}, \ldots, \overset{2m}{\frac{\#}{\clubsuit}}, \overset{2m+1}{\frac{?}{\heartsuit}}, \overset{2m+2}{\frac{\#}{\clubsuit}}).$$

Next, we need to define the following operations applied to a card sequence $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_d)$:

- (turn, T) for $T \subseteq \{1, 2, \ldots, d\}$, i.e., turning over cards is denoted by a set T such that every card whose position in T is turned over;

[2] Koch and Walzer [5] showed that one can securely "choose" a permutation from a specific set using helping cards with a different color.

- (perm, π) for $\pi \in S_d$, where S_i denotes the symmetric group of degree i, i.e., a rearranging operation is denoted by permutation π;
- $(\mathsf{shuf}, \Pi, \mathcal{F})$ for $\Pi \subseteq S_d$ and a probability distribution $\mathcal{F}$ on Π, i.e., a shuffling operation is denoted by a permutation set Π and a probability distribution $\mathcal{F}$ on Π. If $\mathcal{F}$ is uniform, we simply write it as (shuf, Π);
- (result, e) for some expression e. This indicates that the protocol terminates with the output e.

Based on the above formalization, a pseudocode of our PP-free protocol with RC is shown as follows, where "visible seq." denotes what we can look at for a card sequence on the table, and we define

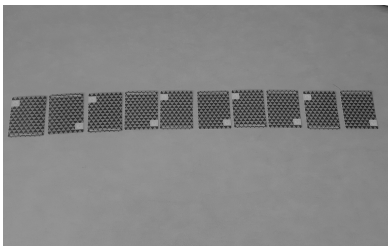
$$\sigma \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 2 & 3 & \cdots & m & m{+}1 & m{+}2 & \cdots & 2m & 2m{+}1 & 2m{+}2 \\ 1 & 3 & 5 & \cdots & 2m{-}1 & 2 & 4 & \cdots & 2m & 2m{+}1 & 2m{+}2 \end{pmatrix},$$

which corresponds to the action for taking cards alternately in Step 2 of the protocol, and

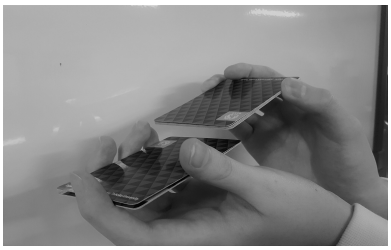
$$\mathsf{RC}_{2m+2} \stackrel{\text{def}}{=} \{(1\ 2\ 3\ \cdots\ 2m{+}2)^j \mid 1 \le j \le 2m{+}2\},$$

where $(1\ 2\ 3\ \cdots\ 2m{+}2)$ is a cyclic permutation, meaning that $1 \mapsto 2$, $2 \mapsto 3$, and so on.

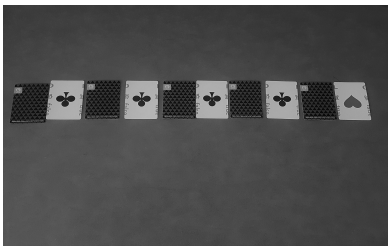
The PP-free protocol with RC
input set: $\left\{\Gamma^{(a,b)} \mid 1 \le a, b \le m\right\}$
(perm, σ)
$(\mathsf{shuf}, \mathsf{RC}_{2m+2})$
if visible seq. $= (\#, ?, \#, ?, \ldots, \#, ?)$ **then**
  $(\mathsf{perm}, (2m{+}2\ \ 2m{+}1 \cdots 1))$
$(\mathsf{turn}, \{2, 4, \ldots, 2m{+}2\})$
let r **s.t.** visible seq. $= (\overbrace{?, \clubsuit}^{1\text{st}}, \ldots, \overbrace{?, \clubsuit}^{(r-1)\text{-st}}, \overbrace{?, \heartsuit}^{r\text{-th}}, \overbrace{?, \clubsuit}^{(r+1)\text{-st}}, \ldots, \overbrace{?, \clubsuit}^{(m+1)\text{-st}})$
$(\mathsf{turn}, \{2r{-}1\})$
if visible seq. $= (?, \clubsuit, \ldots, \overbrace{\heartsuit, \heartsuit}^{r\text{-th}}, \ldots, ?, \clubsuit)$ **then** $(\mathsf{result}, \text{“}a < b\text{”})$
else if visible seq. $= (?, \clubsuit, \ldots, \overbrace{\clubsuit, \heartsuit}^{r\text{-th}}, \ldots, ?, \clubsuit)$ **then**
  $(\mathsf{turn}, \{2r{+}1 \ (\mathsf{mod}\ 2m{+}2)\})$
  if visible seq. $= (?, \clubsuit, \ldots, \overbrace{\clubsuit, \heartsuit}^{r\text{-th}}, \overbrace{\heartsuit, \clubsuit}^{(r+1)\text{-st}}, \ldots, ?, \clubsuit)$ **then** $(\mathsf{result}, \text{“}a = b\text{”})$
  else if visible seq. $= (?, \clubsuit, \ldots, \overbrace{\clubsuit, \heartsuit}^{r\text{-th}}, \overbrace{\clubsuit, \clubsuit}^{(r+1)\text{-st}}, \ldots, ?, \clubsuit)$ **then** $(\mathsf{result}, \text{“}a > b\text{”})$



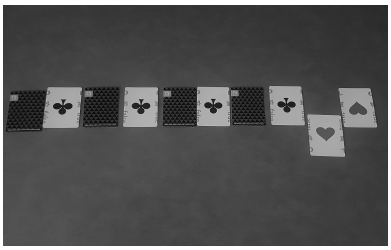
(a) Arrange Alice's input and Bob's input; the upside-down cards are put by Bob.



(b) Apply a random cut to the sequence.



(c) Bob's sequence (of upside-down cards) is revealed and then one red card appears.



(d) The left card of the red card is revealed; we have $a < b$ because the revealed card is red.

Figure 1: An implementation of our PP-free protocol with RC when $m = 4$.

3.3. Example of Real Execution

Our protocol is quite simple and easy-to-implement. For example, two colleagues, Alice and Bob, in a company are easily able to compare their bonuses by using our protocol, where Alice's bonus is 10^a dollars and Bob's bonus is 10^b dollars. The protocol falls into real world cryptography; Fig. 1 shows a real execution of our protocol for $m = 4$, i.e., $10^a, 10^b \in \{\$10, \$100, \$1000, \$10000\}$, requiring only 13 cards.[3] Card-based protocols are far more practical than might be imagined.

4. A Millionaire Protocol Using a Standard Deck

Remember that our implementation with RC explained in Section 3 uses a deck of black and red cards. As mentioned in Section 1, it would be great if we can perform the same task using a standard deck of playing cards instead of using a two-colored deck.

[3]Remember that the protocol requires $3m + 1$ (=13) cards to allow Alice to input a such that $1 \leq a \leq m$ (although there are only 10 cards on the table in Fig. 1).

In this section, we first show that simply replacing the two-colored deck with a standard deck of cards does not work. That is, such a straight-forward protocol leaks information about Bob's input, as shown in Section 4.1. In Section 4.2, we construct a subprotocol as a new technique that prevents Alice from knowing Bob's input b. Based on this, we present the full description of our protocol using a standard deck of playing cards in Section 4.3.

4.1. Toward Using a Standard Deck of Cards

Let us first define a total order that captures a standard deck of playing cards. A *standard deck of playing cards* is a 52-card deck consisting of numbered cards $\boxed{1}\boxed{2}\cdots\boxed{52}$ such that no two cards have the same number. The back sides are all identical $\boxed{?}$. For convenience, in the sequel, we often regard an odd-number card as a black card $\boxed{\clubsuit}$ and an even-number card as a red card $\boxed{\heartsuit}$.

Let us execute our implementation with RC using the above standard deck instead of a two-colored deck of cards $\boxed{\clubsuit}\boxed{\clubsuit}\cdots\boxed{\clubsuit}\ \boxed{\heartsuit}\boxed{\heartsuit}\cdots\boxed{\heartsuit}$. Assume that $m = 4$. Because Alice requires four black cards and three red cards to represent her input, she has $\boxed{1}\boxed{3}\boxed{5}\boxed{7}$ and $\boxed{2}\boxed{4}\boxed{6}$. Bob has $\boxed{9}\boxed{11}\boxed{13}$ and $\boxed{8}$ as three black cards and one red card. Remembering Step 2 in which the $(m+1)$-st additional pair of cards is put, we let Alice hold $\boxed{10}$ and Bob hold $\boxed{15}$. Consider, for instance, the case where $a = 4$ and $b = 2$. Alice and Bob place the following sequences of cards:

$$\underset{5\ \ 3\ \ 7\ \ 1\ \ 10}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}},\qquad \underset{\uparrow\ a\text{-th}}{}$$

$$\underset{15\ \ 8\ \ 9\ \ 11\ \ 13}{\boxed{\text{ʖ}}\boxed{\text{ʖ}}\boxed{\text{ʖ}}\boxed{\text{ʖ}}\boxed{\text{ʖ}}},\qquad \underset{\uparrow\ b\text{-th}}{}$$

where Alice's black cards $\boxed{5}\boxed{3}\boxed{7}\boxed{1}$ and red card $\boxed{10}$ are randomly chosen and placed, and Bob's sequence is arranged by shifting $\boxed{8}\boxed{9}\boxed{11}\boxed{13}\boxed{15}$. Note that Alice can memorize the order of numbers in her input. Thus, after Steps 1 and 2, we have

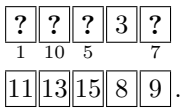
$$\underset{5\ \ 15\ \ 3\ \ 8\ \ 7\ \ 9\ \ 1\ \ 11\ \ 10\ \ 13}{\boxed{?}\boxed{\text{ʖ}}\boxed{?}\boxed{\text{ʖ}}\boxed{?}\boxed{\text{ʖ}}\boxed{?}\boxed{\text{ʖ}}\boxed{?}\boxed{\text{ʖ}}}.$$

They apply a random cut in Step 3 and then reveal Bob's cards in Step 4. For instance, assume that the resulting sequence in Step 4 becomes the following sequence. (We here fix Bob's upside down cards for convenience.)

$$\underset{1\ \ 10\ \ 5\ \ 3\ \ 7}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}}$$
$$\boxed{11}\boxed{13}\boxed{15}\boxed{8}\boxed{9}.$$

Remember that $\boxed{8}$ corresponds to the red card. In this case, the position of $\boxed{8}$ placed by Bob in Step 1 is hidden due to the random cut. Therefore, replacing

Bob's sequence of cards with standard cards does not cause a problem. However, a security issue occurs when Alice's card is revealed. In this example, they reveal the fourth card of Alice's input:



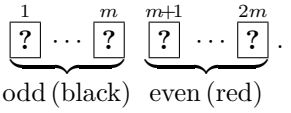
At this time, Alice gets to know that $b = 2$. That is, because Alice remembers that she placed $\boxed{3}$ at the second position in Step 1, she learns that Bob put the red card $\boxed{8}$ at the second position, which implies that $b = 2$. Therefore, if we execute our protocol presented in Section 3 using a standard deck of cards, the value of b would be leaked when Alice's card is revealed in Step 4. That is, this straight-forward implementation is not secure, and we need a new technique.

4.2. Subprotocol

As seen in Section 4.1, simply replacing the deck with a standard deck of playing cards does not work (namely, the value of b would be leaked to Alice). Let us confirm again the reason why the simple replacement described in Section 4.1 makes Alice know Bob's input b. That occurs when Alice's b-th card is opened in Step 4 (because Alice memorizes the order of her cards). Therefore, we construct a subprotocol that hides the order of Alice's cards from her while guaranteeing the "format" of Alice's input, i.e., placing odd-number cards at every position from the first to the a-th and even-number cards from the $(a+1)$-st to the last. If we execute such a subprotocol in advance, b is hidden from Alice even if they reveal the b-th card in Alice's input sequence because Alice does not know where the revealed card was in the input.

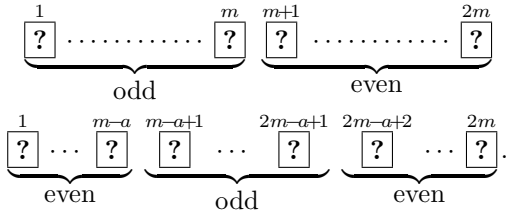
The subprotocol proceeds as follows:

1. Alice and Bob shuffle m odd-number cards (corresponding to black) and then place them on the table with their faces down. Similarly, they shuffle m even-number cards (corresponding to red) and then place them:



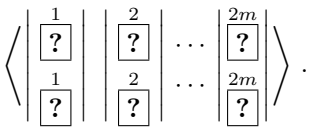
2. Alice takes $m + 1$ odd-number cards and $m - 1$ even-number cards from the deck. Then, she places a sequence of cards consisting of odd-number cards at positions from the $(m - a + 1)$-st to the $(2m - a + 1)$-st (in any order) and even-number cards at the remaining positions (in any order)

below the sequence of cards placed in Step 1 with their faces down:



Note that the subsequence just above the $(m+1)$ odd-number cards placed now becomes an encode of the value of a. We call the sequence placed by Alice the sequence in the second row.

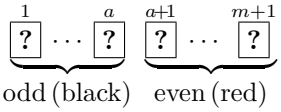
3. Considering the two cards in the same row as a pile, apply a *Pile-shifting Shuffle* to the sequence of piles:



This cyclically shuffles a sequence of piles. To implement this shuffle, we use a physical case that can store a pile of cards, such as boxes and envelopes [20]; one cyclically shuffles them by hand until nobody can trace the offset.[4]

4. Reveal the sequence in the second row. The subsequence of $m+1$ cards above the revealed odd-number cards represents a. The revealed $2m$ cards can be reused.

Thus, we can obtain a sequence of $m+1$ cards representing $a \in \{1, 2, \ldots, m\}$ by executing the above subprotocol:



Note that, by virtue of the shuffle in Step 1, neither Alice nor Bob can know the order of the obtained sequence of the number cards encoding Alice's value a in Step 4. Because they apply the Pile-Shifting Shuffle in Step 3, revealing the sequence in the second row does not leak any information about the value of a. More specifically, we show that no information about Alice's input a is leaked during the execution of the subprotocol, as follows. In Steps 1, 2, and 3, a is not leaked because players just place cards with their faces down and

[4] This operation is the same as Bob puts his cards upside down and then they apply a random cut (as described in Section 3.1). If the back-side symbol of the standard deck of cards is vertically asymmetric, using a random cut is more efficient because an additional tool is not required.

publicly apply shuffle operations. In Step 4, the sequence of cards in the second row placed in Step 2 is revealed. Then, odd-number cards appear from the $(m-a+1+r)$-th to the $(2m-a+1+r)$-th (apart from cyclic rotation) where $r \in \{0, 1, \ldots, 2m-1\}$ is a random value generated by the Pile-Shifting Shuffle in Step 3. Therefore, a is not leaked by revealing the sequence of cards in the second row.

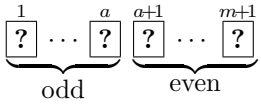
The total number of shuffles required for the subprotocol is three because two shuffles in Step 1 and one shuffle in Step 3 are applied. The number of required cards is $4m$.

4.3. Description

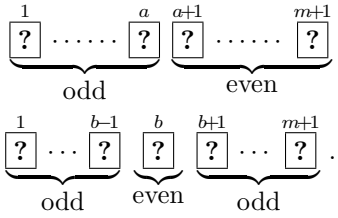
We are now ready to present our protocol using a standard deck of playing cards; our implementation is obtained by combining the subprotocol explained in Section 4.2 with the PP-free protocol with RC explained in Section 3.

The protocol proceeds as follows:

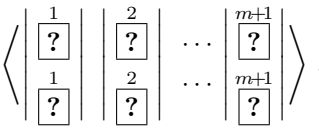
1. Execute the subprotocol explained in Section 4.2 to obtain Alice's input sequence of cards:



   The revealed $2m$ cards in the subprotocol can be reused; Bob takes them and then places $m+1$ cards (namely, Bob's input sequence) below Alice's, such that only the b-th card has an even number and the remaining m odd-number cards starting at the $(b+1)$-st position (apart from cyclic rotation) are sorted in ascending order (or in any order):



2. Apply the Pile-Shifting Shuffle:



3. Reveal Bob's cards. Then, one even-number card appears. Reveal the card just above the even-number card.
   - We have $a < b$ if the revealed card is even.
   - We have $a \geq b$ if the revealed card is odd. To see whether equality holds or not, open the card to the right of the revealed card. If the opened card is even, we have $a = b$. If it is odd, we have $a < b$.

Thus, we can solve the millionaire problem by executing the above protocol. In Steps 1 and 2, information about Alice's input a and Bob's input b is not leaked because they just execute the subprotocol, which is shown to be secure in Section 4.2, and publicly apply a shuffle operation. In Step 3, Bob's input sequence is revealed at first. Then, one even-number card appears in the $(b+r)$-th position (apart from cyclic rotation) where $r \in \{0, 1, \ldots, m\}$ is a random value generated by the Pile-Shifting Shuffle in Step 2. Therefore, the value of b is not leaked by revealing Bob's input sequence. Then, the card just above the revealed even-number card, i.e., the b-th card in Alice's input sequence is revealed. This revealed card does not leak b because nobody knows where the revealed card was placed in the input due to the subprotocol. Therefore, the value of b is not leaked by revealing the b-th card in Alice's input sequence. Opening the $(b+1)$-st card of Alice's input sequence is similar.

4.4. The Efficiency

The total number of shuffles required for our implementation using a standard deck is four because three shuffles in the subprotocol and one shuffle in Step 2 are applied. The number of required cards is $4m$ because they reuse cards to execute the protocol after executing the subprotocol with $4m$ cards.[5] See Table 1 again.

5. The Existing Circuit-Based Protocol

Hereinafter, we deal with another approach for solving the millionaires' problem: We introduce the existing circuit-based protocol [3] in this section, and then we will improve upon it in the next section.

Consider the following encoding:

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

Then, Alice and Bob can place sequences of cards corresponding to the binary representations of $a = (a_n, \ldots, a_1)_2$ and $b = (b_n, \ldots, b_1)_2$, respectively, where $n = \lceil \log_2 m \rceil$:

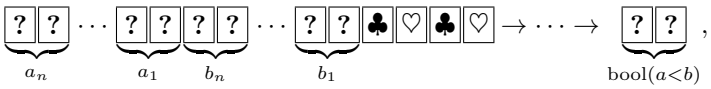
$$\underbrace{\boxed{?}\boxed{?}}_{a_n} \cdots \underbrace{\boxed{?}\boxed{?}}_{a_1} \quad \underbrace{\boxed{?}\boxed{?}}_{b_n} \cdots \underbrace{\boxed{?}\boxed{?}}_{b_1}.$$

Such a pair of face-down cards

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

[5] Note that the computational model of card-based protocols [18] assumes that all inputs are given at the beginning. If we follow this assumption, i.e., Bob should put input cards at the beginning of the protocol, the number of required cards is $5m$.

corresponding to a bit $x \in \{0,1\}$ is called a *commitment* to x. Given the above card sequence along with some additional cards, the existing circuit-based protocol given by Nakai et al. [3] determines whether $a < b$ or not:



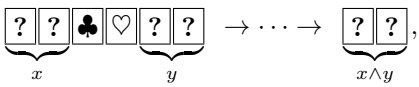
where $\mathrm{bool}\,(a < b)$ represents

$$\mathrm{bool}\,(a < b) \stackrel{\mathrm{def}}{=} \begin{cases} 0 & \text{if } a \geq b, \\ 1 & \text{if } a < b. \end{cases}$$

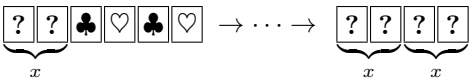
Their protocol proceeds based on the following logical circuit.

The circuit-based protocol [3]
input : $a = (a_n, \ldots, a_1)_2, b = (b_n, \ldots, b_1)_2$;
$f_1 = \bar{a}_1 \wedge b_1$;
for $(i : 2$ to $n)$ {
  $f_i = (\bar{a}_i \wedge b_i) \vee ((\bar{a}_i \vee b_i) \wedge f_{i-1})$;
}
output : $f_n\,(= \mathrm{bool}\,(a < b))$.

To implement this circuit, one requires AND (OR) and COPY protocols; Nakai et al. [3] used the six-card AND protocol [7] (shown in Appendix A), producing a commitment to $x \wedge y$ from the input commitments to x and y:



and the six-card COPY protocol [7] (shown in Appendix B), producing two commitments to x from an input commitment to x:



Let us count the number of required cards for implementing the circuit. First, two additional cards ♣♡ are required to compute $f_1 = \bar{a_1} \wedge b_1$ using the AND protocol [7]. Note that we have four reusable cards ♣♣♡♡ directly after computing f_1 (see Appendix A for details). Then, because six additional cards ♣♣♣♡♡♡ are required to duplicate the commitments to a_2 and b_2 in order to compute $f_2 = (\bar{a}_2 \wedge b_2) \vee ((\bar{a}_2 \vee b_2) \wedge f_1)$, another two cards ♣♡ are required. Consequently, four additional cards ♣♣♡♡ are necessary before computing f_1, and hence, the total number of required cards is $4\lceil \log m \rceil + 4$. (Note that directly after computing f_j, $2 \leq j \leq n-1$, we have enough reusable cards to compute f_{j+1}, i.e., four additional cards are sufficient to implement the circuit.)

Next, let us count the number of required shuffles. Note that each of the AND [7] and COPY [7] protocols uses one shuffle and we have two reusable cards after the protocol terminates; furthermore, we can obtain two more reusable cards after using the AND protocol [7] if we apply one more shuffle (as seen in Appendices A and B). First, one shuffle is required to compute f_1, and one more shuffle is required to produce four reusable cards. Then, the circuit-based protocol uses the COPY protocol [7] twice and the AND (OR) protocol [7] four times to compute f_i, $2 \leq i \leq n$. Moreover, one more shuffle is required to produce reusable cards enough to compute f_{i+1}. That is, seven shuffles are required to compute f_j, $2 \leq j \leq n-1$, and six shuffles are required to compute f_n. Consequently, the total number of required shuffles is $7\lceil \log m \rceil - 6$.

It should be noted that, as seen above, this existing circuit-based protocol produces a commitment to $\mathrm{bool}\,(a < b)$ (while our implementations with RC presented in Sections 3 and 4 reveal the value of it to the players at the end of the protocols). Therefore, one can reuse the commitment in a larger protocol.

6. Our Improved Circuit-Based Protocol

In this section, we improve upon the circuit-based protocol introduced in Section 5, i.e., we present an improved circuit-based protocol that uses a less number of shuffles and cards. We first show the idea behind our improved circuit-based protocol in Section 6.1, and then show the procedure of our improved circuit-based protocol in Section 6.2.

6.1. The Idea

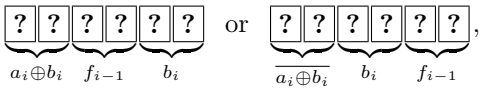
We borrow the idea behind the storage protocol [3]; it uses Private Permutation and regards f_i shown in Section 5 as:

$$f_i = \begin{cases} f_{i-1} & \text{if } \ a_i = b_i, \\ b_i & \text{if } \ a_i \neq b_i. \end{cases} \tag{2}$$

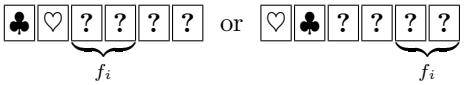
That is, the storage protocol is supposed to choose f_{i-1} or b_i depending on whether $a_i = b_i$ or not. More specifically,

- f_{i-1} is equal to $\mathrm{bool}\,((a_{i-1}, \ldots, a_1) < (b_{i-1}, \ldots, b_1))$;
- $a_i = b_i$ implies $f_i = f_{i-1}$;
- $a_i = 0$ and $b_i = 1$ imply $(a_i, \ldots, a_1) < (b_i, \ldots, b_1)$, and hence $f_i = 1 = b_i$ while $a_i = 1$ and $b_i = 0$ imply $(a_i, \ldots, a_1) > (b_i, \ldots, b_1)$, and hence $f_i = 0 = b_i$.

Such a choice can be made without Private Permutation; if we can let a six-card sequence be either

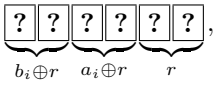


then, we can obtain a commitment to f_i by revealing the first two cards as follows:

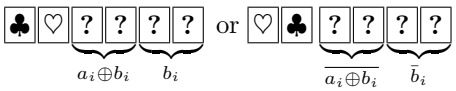


The above flow can be accomplished by using the procedure of the six-card AND protocol [7] shown in Appendix A.

Moreover, we can easily obtain commitments to $a_i \oplus b_i$ and b_i by using the six-card COPY protocol [7] shown in Appendix B. That is, from the following sequence where r is a uniform random bit:



it is determined whether $r = b_i$ or $r = \bar{b}_i$ by revealing the first two cards, and then we obtain commitments to $a_i \oplus b_i$ and b_i as:

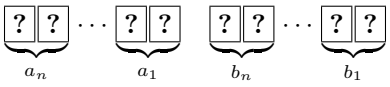


Note that revealing the first two cards leaks no information about b_i because r is a random bit.

As described above, by using the procedure of the COPY and AND protocols [7], we can obtain a commitment to f_i according to Eq. (2) without revealing the values of a_i, b_i, and f_{i-1}. It should be noted that f_i in Eq. (2) is the three-input majority function of a_i, b_i, and f_{i-1}. An efficient card-based protocol for the three-input majority function was proposed by Nishida et al. [21] in 2013, which was based on the same idea mentioned above.

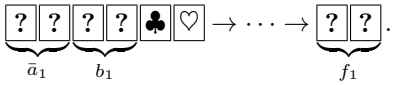
6.2. The Description of Our Protocol

Based on the idea presented in Section 6.1, we construct an improved circuit-based protocol. Given the input card sequence



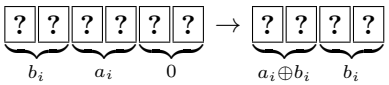
and two additional cards, our protocol proceeds as follows.

1. Compute $f_1 = \bar{a}_1 \wedge b_1$ by using the six-card AND protocol [7]:

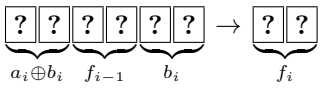


   Now, two reusable cards remain.
2. Repeat the following computation from $i = 2$ to $i = n$.

(a) Obtain commitments to $a_i \oplus b_i$ and b_i from the commitments to a_i and b_i by using the six-card COPY protocol [7] (and the NOT computation):



(b) Obtain a commitment to f_i from the commitments to $a_i \oplus b_i$, b_i, and f_{i-1} by using the six-card AND protocol:



3. Then, a commitment to $f_n = \mathrm{bool}\,(a < b)$ can be obtained:

Due to the use of two additional cards, this protocol uses $(4\lceil \log m \rceil + 2)$ cards in total. The number of required shuffles is $2\lceil \log m \rceil - 1$ in total, because this protocol repeats each procedure of the AND protocol and the COPY protocol from $i = 2$ to $i = n$ after AND computation for f_1, as shown in Table 1.

This improved circuit-based protocol is a combination of the existing information theoretically secure card-based protocols, and hence, it is guaranteed to be secure.

7. Circuit-based Protocols with a Standard Deck

In this section, let us run the circuit-based protocols presented in Sections 5 and 6 with a standard deck of cards. To achieve this, we consider the use of the existing elementary protocols with a standard deck of cards, namely the AND and COPY protocols listed in Table 2.

First, combine these elementary protocols with the (NTMIO) circuit-based protocol [3] presented in Section 6. Table 3 shows the numbers of required cards and shuffles; they can be counted in a similar way to Section 5.

Next, let us run our improved circuit-based protocol explained in Section 6 with a standard deck of cards. Because Step 2(b) of our protocol uses not a normal AND computation but a special choice of cards, we cannot use the elementary AND protocols straightforwardly. Fortunately, we found that the choice can be made by the (extended) existing AND protocol [22] easily. The extended protocol requires four additional cards and five shuffles.[6] Therefore, the resulting circuit-based protocol uses $(4\lceil \log m \rceil + 4)$ cards and $(6\lceil \log m \rceil - 2)$ shuffles in total.

[6] More specifically, the extended protocol proceeds in a similar way to the existing AND protocol [22] except for producing "opaque" commitments to f_{i-1} and b_i; refer to the paper [22] for details.

Table 2: The existing elementary protocols with a standard deck of cards. Note that the number of required shuffles for each of the AND protocol proposed by Niemi and Renvall [23] and the one proposed by Koch, Schrempp, and Kirsten [24] is an expected value.

	Function	**#Cards**	**#Shuffles**
Niemi–Renvall [23]	AND	5	9.5
Mizuki [22]	AND	8	4
Koch–Schrempp–Kirsten [24]	AND	4	6
Mizuki [22]	COPY	6	1

Table 3: The NTMIO circuit-based protocol with a standard deck of cards using the existing elementary protocols [22, 23, 24].

	#Cards	**#Shuffles**
NTMIO with Niemi–Renvall [23]	$4\lceil \log m \rceil + 4$	$40\lceil \log m \rceil - 30.5$
NTMIO with Mizuki [22]	$4\lceil \log m \rceil + 6$	$18\lceil \log m \rceil - 14$
NTMIO with Koch–Schrempp–Kirsten [24]	$4\lceil \log m \rceil + 4$	$26\lceil \log m \rceil - 20$

8. Conclusion

In this paper, we proposed three card-based protocols to solve the millionaires' problem without using Private Permutation. See Table 1 again for the performance of the PP-free millionaire protocols. In particular, the PP-free protocol with RC proposed in Section 3 uses only one random cut, and its correctness and secrecy are clear. Therefore, we believe that even non-experts such as high school students can easily understand and use it practically. Note that a random cut can be easily and securely implemented by using the Hindu cut [6]. When preparing a two-colored deck of cards is hard, our protocol presented in Section 4 will be benefical because it can be executed with a standard deck of cards, which is sold almost all over the world.

Moreover, we can use our protocols in didactic contexts in order to invite young people and students to cryptography; they would be an ideal tool to exhibit the concept of secure multiparty computations, as often pointed out, e.g., [25, 26].

Regarding the number of required cards, one might think of the use of the existing four-card AND [17], the five-card AND [27], and the five-card COPY protocols [28] in the previous circuit-based protocol because the number of required cards should be reduced. However, the numbers of required shuffles for the four-card and five-card AND protocols are eight and seven on average,

respectively. Moreover, the five-card COPY protocol requires ideal cases for execution. Therefore, for practicality, we considered only the six-card AND protocol [7] and the six-card COPY protocol [7].

Acknowledgment

We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. This work was supported by JSPS KAKENHI Grant Number JP17K00001.

References

[1] D. Miyahara, Y. Hayashi, T. Mizuki, H. Sone, Practical and easy-to-understand card-based implementation of Yao's millionaire protocol, in: Combinatorial Optimization and Applications, Vol. 11346 of Lecture Notes in Computer Science, Springer, Cham, 2018, pp. 246–261.

[2] A. C. Yao, Protocols for secure computations, in: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, FOCS '82, IEEE Computer Society, Washington, DC, USA, 1982, pp. 160–164. doi:10.1109/SFCS.1982.88.
URL http://dx.doi.org/10.1109/SFCS.1982.88

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

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

[5] A. Koch, S. Walzer, Foundations for actively secure card-based cryptography, Cryptology ePrint Archive, Report 2017/423, https://eprint.iacr.org/2017/423 (2017).

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

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

[8] M. Jakobsson, M. Yung, Proving without knowing: On oblivious, agnostic and blindfolded provers, in: N. Koblitz (Ed.), Advances in Cryptology — CRYPTO '96, Vol. 1109 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1996, pp. 186–200.

[9] R. Fagin, M. Naor, P. Winkler, Comparing information without leaking it, Commun. ACM 39 (5) (1996) 77–85. doi:10.1145/229459.229469.
URL http://doi.acm.org/10.1145/229459.229469

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

[11] H. Ono, Y. Manabe, Efficient card-based cryptographic protocols for the millionaires' problem using private input operations, in: 2018 13th Asia Joint Conference on Information Security (AsiaJCIS), 2018, pp. 23–28. doi:10.1109/AsiaJCIS.2018.00013.

[12] H. Ono, Y. Manabe, Card-based cryptographic protocols with the minimum number of cards using private operations, in: N. Zincir-Heywood, G. Bonfante, M. Debbabi, J. Garcia-Alfaro (Eds.), Foundations and Practice of Security, Vol. 11358 of Lecture Notes in Computer Science, Springer, Cham, 2019, pp. 193–207.

[13] Y. Watanabe, Y. Kuroki, S. Suzuki, Y. Koga, M. Iwamoto, K. Ohta, Card-based majority voting protocols with three inputs using three cards, in: 2018 International Symposium on Information Theory and Its Applications (ISITA), 2018, pp. 218–222. doi:10.23919/ISITA.2018.8664324.

[14] J. Balogh, J. A. Csirik, Y. Ishai, E. Kushilevitz, Private computation using a PEZ dispenser, Theoretical Computer Science 306 (1) (2003) 69–84. doi:https://doi.org/10.1016/S0304-3975(03)00210-X.
URL http://www.sciencedirect.com/science/article/pii/S030439750300210X

[15] T. Mizuki, Y. Kugimoto, H. Sone, Secure multiparty computations using the 15 puzzle, in: A. Dress, Y. Xu, B. Zhu (Eds.), Combinatorial Optimization and Applications, Vol. 4616 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007, pp. 255–266.

[16] T. Mizuki, H. Shizuya, Practical card-based cryptography, in: A. Ferro, F. Luccio, P. Widmayer (Eds.), Fun with Algorithms, Vol. 8496 of Lecture Notes in Computer Science, Springer, Cham, 2014, pp. 313–324.

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

[18] T. Mizuki, H. Shizuya, A formalization of card-based cryptographic protocols via abstract machine, International Journal of Information Security 13 (1) (2014) 15–23. doi:10.1007{\slash}s10207-013-0219-4.

[19] T. Mizuki, H. Shizuya, Computational model of card-based cryptographic protocols and its applications, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E100.A (1) (2017) 3–11. doi:10.1587/transfun.E100.A.3.

[20] A. Nishimura, Y. Hayashi, T. Mizuki, H. Sone, Pile-shifting scramble for card-based protocols, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences 101 (9) (2018) 1494–1502.

[21] T. Nishida, T. Mizuki, H. Sone, Securely computing the three-input majority function with eight cards, in: A.-H. Dediu, C. Martín-Vide, B. Truthe, M. A. Vega-Rodríguez (Eds.), Theory and Practice of Natural Computing, Vol. 8273 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2013, pp. 193–204.

[22] T. Mizuki, Efficient and secure multiparty computations using a standard deck of playing cards, in: S. Foresti, G. Persiano (Eds.), Cryptology and Network Security, Vol. 10052 of Lecture Notes in Computer Science, Springer, Cham, 2016, pp. 484–499.

[23] V. Niemi, A. Renvall, Solitaire zero-knowledge, Fundam. Inf. 38 (1,2) (1999) 181–188.
URL http://dl.acm.org/citation.cfm?id=2377856.2377870

[24] A. Koch, M. Schrempp, M. Kirsten, Card-based cryptography meets formal verification, in: Advances in Cryptology – ASIACRYPT 2019, Lecture Notes in Computer Science, Springer, to appear, https://eprint.iacr.org/2019/1037.

[25] G. Hanaoka, Towards user-friendly cryptography, in: R. C.-W. Phan, M. Yung (Eds.), Paradigms in Cryptology – Mycrypt 2016. Malicious and Exploratory Cryptology, Vol. 10311 of Lecture Notes in Computer Science, Springer, Cham, 2017, pp. 481–484.

[26] A. Marcedone, Z. Wen, E. Shi, Secure dating with four or fewer cards, Cryptology ePrint Archive, Report 2015/1031, https://eprint.iacr.org/2015/1031 (2015).

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

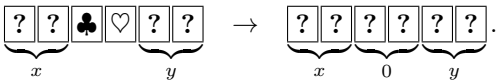
[28] A. Nishimura, T. Nishida, Y. Hayashi, T. Mizuki, H. Sone, Five-card secure computations using unequal division shuffle, in: A.-H. Dediu, L. Magdalena, C. Martín-Vide (Eds.), Theory and Practice of Natural Computing,

Vol. 9477 of Lecture Notes in Computer Science, Springer, Cham, 2015, pp. 109–120.

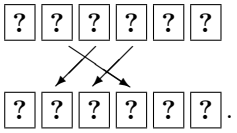
Appendix A. The Six-Card AND Protocol

In 2009, Mizuki and Sone [7] invented the following six-card AND protocol, which securely outputs a commitment to $x \wedge y$ from the commitments to x and y (and two additional cards).

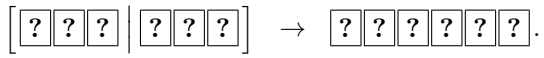
1. Place input commitments and additional cards of black and red, and then turn over the two cards in the center:



2. Rearrange the sequence:

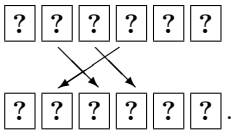


3. Apply a *random bisection cut*, which means bisecting the sequence and switching the two halves randomly:

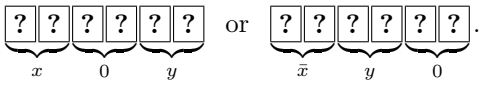


After applying this shuffling operation, the six-card sequence results in either the same sequence as the original one or a sequence whose left and right halves are switched; each case occurs with a probability of 1/2.

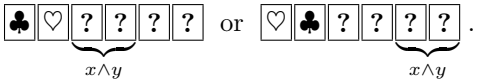
4. Rearrange the sequence:



After this rearranging operation, the six-card sequence will be as follows:



5. Reveal the first two cards. Then, a commitment to $x \wedge y$ can be obtained as:



Note that we can reuse the two revealed two cards, and moreover, the other two cards not being a commitment to $x \wedge y$ can be reused by revealing them after shuffling.

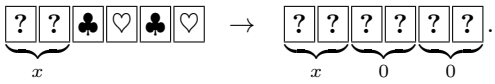
As mentioned above, we can obtain a commitment to $x \wedge y$ (keeping its value secret). It is well known that in the literature [6], a random bisection cut can be implemented by humans securely so that nobody knows the resulting card sequence.

An OR protocol can be obtained in a similar way.

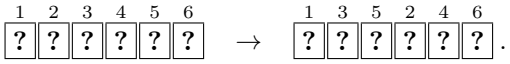
Appendix B. The Six-Card COPY Protocol

The following six-card COPY protocol proposed by Mizuki and Sone [7] produces two commitments to x from a commitment to x and four additional cards.

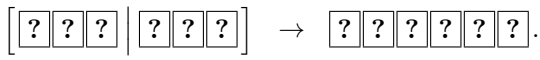
1. Place an input commitment and black and red additional cards, and then turn over the additional cards:



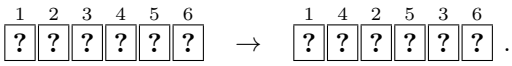
2. Rearrange the sequence:



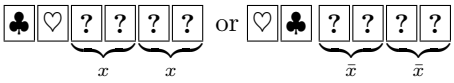
3. Apply a random bisection cut:



4. Rearrange the sequence:



5. Reveal the first two cards. Then, two commitments to x can be obtained as follows (two revealed cards can be reused in the next protocol):



   In the latter case, we can easily get two commitments to x (from commitments to $\bar{x}$) by using the NOT protocol (swapping the two cards of each commitment).