Card-based Secure Ranking Computations[⋆]

Ken Takashima[1], Yuta Abe[1], Tatsuya Sasaki[1], Daiki Miyahara[1,4], Kazumasa Shinagawa[2,4], Takaaki Mizuki[3], and Hideaki Sone[3]

[1] Graduate School of Information Sciences, Tohoku University, 6–3–09 Aramaki-Aza-Aoba, Aoba-ku, Sendai, 980–8578, Japan

[2] Graduate School of Information Sciences and Engineerings, Tokyo Institute of Technology, 2–12–1 Ookayama, Megro, Tokyo 152–8552, Japan

[3] Cyberscience Center, Tohoku University, 6–3 Aramaki-Aza-Aoba, Aoba-ku, Sendai, 980–8578, Japan

[4] National Institute of Advanced Industrial Science and Technology, 2–4–7, Aomi, Koto-ku, Tokyo, 135–0064, Japan

Abstract. Consider a group of people who want to know the "rich list" among them, namely the ranking in terms of their total assets, without revealing any information about the actual value of their assets. This can be achieved by a "secure ranking computation," which was first considered by Jiang and Gong (CT-RSA 2006); they constructed a secure ranking computation protocol based on a public-key cryptosystem. In this paper, instead of using a public-key cryptosystem, we use a deck of physical cards to provide secure ranking computation protocols. Therefore, our card-based protocols do not rely on computers, and they are simple and easy for humans to implement. Specifically, we design four protocols considering tradeoffs between the number of cards and the number of shuffles required to execute protocols. We also present a guide to choose an appropriate protocol according to the number of people participating in the protocol and the size of the input range.

Key words: Card-based protocols, Secure ranking computation, Secure multiparty computations, Millionaire problem, Deck of cards

1 Introduction

Assume that there are n players $P_1, \ldots, P_n$ such that each player P_i, $1 \leq i \leq n$, holds x_i dollars with $1 \leq x_i \leq m$. They want to know the "rich list" $(r_1, \ldots, r_n)$ among them, namely the ranking in terms of their total assets, without revealing any information about the actual values of $x_1, \ldots, x_n$. More formally, they want to securely compute the *ranking function* $\mathsf{rk} : \{1, 2, \ldots, m\}^n \to \{1, 2, \ldots, n\}^n$ defined as

$$\mathsf{rk} : (x_1, \ldots, x_n) \mapsto (r_1, \ldots, r_n)$$

$$\text{such that } r_i = 1 + \Big|\{x_j \mid x_j > x_i, 1 \leq j \leq n\}\Big| \text{ for every } i,\ 1 \leq i \leq n,$$

⋆ This paper appears in Proceedings of COCOA 2019. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-36412-0_37.

where $|X|$ denotes the cardinality of a set X. For example,

$$\mathsf{rk}(3,1,3,5) = (2,4,2,1),$$

which means that P_4 is the richest, P_2 is the poorest, and both P_1 and P_3 are tied; all the players want to know only the ranking list $(2,4,2,1)$ without revealing the amounts of their individual money $(3,1,3,5)$.

This is a *secure ranking computation* problem, which was first considered by Jiang and Gong [5]. They constructed a secure ranking computation protocol based on a public-key cryptosystem. That is, their protocol allows $P_1 \cdots, P_n$ whose secret properties are $(x_1,\ldots,x_n)$ to know only the ranking $\mathsf{rk}(x_1,\ldots,x_n) = (r_1,\ldots,r_n)$ without revealing any information about the inputs $(x_1,\ldots,x_n)$. Note that this falls into the category of secure multiparty computations (MPCs) [12].

1.1 Card-based Protocols

It is well known that MPCs can be conducted by using a deck of physical cards (such as $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\cdots$), and such a research area is called *card-based cryptography* (e.g. [1,7]). The goal of this paper is to construct efficient card-based protocols for solving the secure ranking computation problem. Compared to the use of a public-key cryptosystem, there are certain advantages in using a deck of cards; their computation, correctness, and secrecy are easier to understand.

As elementary card-based computations, efficient NOT, AND, OR, XOR, and COPY protocols have been proposed [2,8]. Therefore, for any n-variable function f, we can construct a card-based protocol that securely computes f by combining these elementary protocols. Thus, if one created a logical circuit representing the ranking function rk, then one could construct a card-based protocol securely computing rk. However, such a circuit would be huge and complicated, and hence, the resulting protocol would no longer be practical.

While generic constructions as above tend to be impractical, a protocol specialized for a specific problem is usually more efficient. For example, consider the millionaire problem [12]: Two players, Alice and Bob, want to know who is richer. Indeed, efficient card-based millionaire protocols were proposed [3,6,9]. The goal of this paper is to design specialized card-based protocols that efficiently solve the ranking computation problem.

One may notice that the secure ranking computation can be realized by repeatedly executing a millionaire protocol. However, our proposed protocols perform the secure ranking computation more efficiently than by repeating the existing millionaire protocol, as seen later.

We use a deck of four types of cards: black cards $\boxed{\clubsuit}$, red cards $\boxed{\heartsuit}$, a joker $\boxed{\mathrm{Jo}}$, and number cards $\boxed{1}, \boxed{2}, \ldots, \boxed{m}$, which satisfy the following properties:

1. Cards with the same symbol on their faces cannot be distinguished from each other.
2. All cards have identical backs $\boxed{?}$, and the pattern of backs is vertically asymmetric so that we can distinguish between a face-down card that is right-side up $\boxed{?}$ and one that is upside down $\boxed{¿}$.

We call black cards, red cards, and the joker as *color cards*.

1.2 Contribution

In this study, we propose four secure ranking computation protocols using a deck of cards. The protocols are named as follows: two-phase, one-phase, shuffle-efficient, and card-efficient protocols. Table 1 summarizes their performance. The two-phase protocol and the one-phase protocol are not finite-runtime, but they are expected finite-runtime protocols that use an expected finite number of shuffles, i.e., they are Las Vegas algorithms. By contrast, both the shuffle-efficient protocol and the card-efficient protocol are finite-runtime; they terminate with a finite number of shuffles. More details along with the organization of this paper are as follows.

Table 1. The performance of our proposed protocols.

Protocol	finite-runtime?	# of color cards	# of num. cards	(Expected) # of shuffles
two-phase	no	$n(2m+1)$	0	$k+2+(m-k+1)\sum_{i=1}^{k}\frac{1}{i}(=S)$
one-phase	no	$n(m+1)$	0	$\leq S+1+(m+1)\sum_{i=1}^{m-k}\frac{1}{i}$
shuffle-efficient	yes	$(n+k)(m+1)$	$m+1$	$k+3$
card-efficient	yes	$n(m+1)$	$m+1$	$3k+1$

In Section 2, we review fundamental shuffling operations in card-based cryptography.

In Section 3.1, we propose the two-phase protocol. The first phase of this protocol computes the "equality" of input $(x_1,\ldots,x_n)$, i.e., it reveals the quotient set $\{1,\ldots,n\}/\{(i,j)\mid x_i=x_j\}$; each equivalent class corresponds to a group of players who have the same amount of money. We often use k to denote

$$k = m - |\{1,\ldots,n\}/\{(i,j)\mid x_i=x_j\}|$$

throughout this paper (including Table 1); in other words, k is the number of amounts of money that nobody has. Making the use of k, the second phase of this protocol outputs $\mathsf{rk}(x_1,\ldots,x_n)$. Note that knowing the ranking $\mathsf{rk}(x_1,\ldots,x_n)$ implies learning the "equality," and hence, revealing such a quotient set will not leak any information beyond the ranking.

While the two-phase protocol mentioned above has two phases, we propose the one-phase protocol in Section 3.2. That is, we get rid of the "equality check" phase so that the number of required cards is $n(m+1)$, which is the smallest among all the proposed protocols.

Remember that both the two-phase and one-phase protocols are not finite-runtime. In Section 4, we focus our attention on constructing finite-runtime protocols that always terminate with a fixed number of shuffles. We first propose the finite-runtime shuffle-efficient protocol, which uses only $k + 3$ shuffles although it requires a relatively large number of cards. We can modify this protocol to reduce the number of required cards by increasing the number of shuffles. Carrying this technique further, we next propose the finite-runtime card-efficient protocol.

In Section 5, we compare the two finite-runtime protocols with the existing millionaire protocol and discuss which protocol is appropriate in practical situations.

In Section 6, we summarize this paper and mention future work.

2 Preliminaries

In this section, we review three types of shuffles that are often used in card-based protocols. As seen later, we can conceal input values by applying a shuffle operation to a sequence of face-down cards.

2.1 Random Cut

Consider a sequence of ℓ face-down cards $\boxed{?}\boxed{?}\cdots\boxed{?}$ denoted by $(c_1, c_2, \ldots, c_\ell)$. A *random cut* randomly and cyclically shifts a sequence of cards. Applying a random cut to the sequence $(c_1, c_2, \ldots, c_\ell)$ results in $(c_{1+r}, c_{2+r}, \ldots, c_{\ell+r})$, where r is uniformly randomly generated from $\mathbb{Z}/\ell\mathbb{Z}$ and is unknown to everyone. Note that a random cut can be implemented securely by human hands [11].

2.2 Pile-Scramble Shuffle

Consider a pile of face-down cards $\boxed{?}\!\!|||$. Assume that we have ℓ such piles $\boxed{?}\!\!|||\,\boxed{?}\!\!|||\cdots\boxed{?}\!\!|||$, all of whose sizes are the same; we denote it by $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_\ell)$. A *pile-scramble shuffle* [4] is a shuffle operation for such a sequence of piles. Applying a pile-scramble shuffle to the sequence $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_\ell)$ results in $(\boldsymbol{p}_{\pi^{-1}(1)}, \boldsymbol{p}_{\pi^{-1}(2)}, \ldots, \boldsymbol{p}_{\pi^{-1}(\ell)})$, where $\pi \in S_\ell$ is a uniformly distributed random permutation and S_ℓ is the symmetric group of degree ℓ. We can easily implement a pile-scramble shuffle by making piles with rubber bands or envelopes.

Alternatively, we can reduce the implementation of a pile-scramble shuffle to random cuts. We make a sequence of piles by using upside down cards as follows:

$$\underbrace{\boxed{\text{¿}}\boxed{?}\cdots\boxed{?}}_{\boldsymbol{p}_1}\underbrace{\boxed{\text{¿}}\boxed{?}\cdots\boxed{?}}_{\boldsymbol{p}_2}\;\cdots\;\underbrace{\boxed{\text{¿}}\boxed{?}\cdots\boxed{?}}_{\boldsymbol{p}_l}.$$

Then, after applying a random cut to it, we take out the first pile, which is randomly chosen from ℓ piles. Next, do the same operation for the remaining $\ell - 1$ piles. Repeating this $\ell - 1$ times in total, we have a randomly permuted sequence of ℓ piles.

2.3 Pile-Shifting Shuffle

A *pile-shifting shuffle* [10] cyclically and randomly shifts a sequence of piles. Applying a pile-shifting shuffle to $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_\ell)$ results in $(\boldsymbol{p}_{1+r}, \boldsymbol{p}_{2+r}, \ldots, \boldsymbol{p}_{\ell+r})$, where r is uniformly randomly generated from $\mathbb{Z}/\ell\mathbb{Z}$ and is unknown to everyone.

Similar to the implementation of a pile-scramble shuffle based on upside down cards, as discussed in Section 2.2, we can implement a pile-shifting shuffle by applying a random cut once.

3 Simple Protocols

In this section, we propose two simple protocols. The first is the fundamental one, which has two phases. The second is obtained by removing one phase from the two-phase protocol. The numbers of cards and shuffles required to execute the two protocols were shown in Table 1; note that there is a tradeoff.

3.1 Proposal of Two-phase Protocol

This simple ranking protocol consists of two phases, namely, the Check-equality phase and the Create-ranking phase, after the setup.

Setup. Each player P_i holds one red card $\boxed{\heartsuit}$ and $m-1$ black cards $\boxed{\clubsuit}$s, and places a sequence of m cards face down, denoted by α_i, that encodes their private value $x_i \in \{1, \ldots, m\}$, as follows:

$$\alpha_i \quad \underbrace{\underset{\clubsuit}{\boxed{?}}}_{\text{1-st}} \; \underset{\clubsuit}{\boxed{?}} \; \cdots \; \underset{\clubsuit}{\boxed{?}} \; \underbrace{\underset{\heartsuit}{\boxed{?}}}_{x_i\text{-th}} \; \underset{\clubsuit}{\boxed{?}} \; \cdots \; \underbrace{\underset{\clubsuit}{\boxed{?}}}_{m\text{-th}} \; .$$

That is, P_i puts $\boxed{\heartsuit}$ at the x_i-th position and $\boxed{\clubsuit}$s at the remaining positions with their faces down. Likewise, each player P_i makes the following sequence of cards β_i:

$$\beta_i \quad \underbrace{\underset{\heartsuit}{\boxed{?}}}_{\text{1-st}} \; \underset{\heartsuit}{\boxed{?}} \; \cdots \; \underset{\heartsuit}{\boxed{?}} \; \underbrace{\underset{\clubsuit}{\boxed{?}}}_{x_i\text{-th}} \; \underset{\heartsuit}{\boxed{?}} \; \cdots \; \underbrace{\underset{\heartsuit}{\boxed{?}}}_{m\text{-th}} \; .$$

Check-equality phase. In this phase, we reveal the "equality" of the inputs $(x_1, \ldots, x_n)$. Formally speaking, we want to know the quotient set $\{1, \ldots, n\}/\{(i,j) \mid x_i = x_j\}$. Note that, as mentioned before, knowing $\mathsf{rk}(x_1, \ldots, x_n)$ implies learning the quotient set, and hence, revealing the equality does not leak any information beyond the ranking.

1. Take the i-th ($i = 1, \ldots, m$) cards from all players' input sequences $\alpha_1, \ldots, \alpha_n$ (while keeping the order unchanged) to make a pile $\boldsymbol{p}_i$ of size n:

$$\begin{array}{ccccc} \alpha_1 & \boxed{?} & \boxed{?} & \cdots & \boxed{?} \\ \alpha_2 & \boxed{?} & \boxed{?} & \cdots & \boxed{?} \\ \vdots & \vdots & \vdots & & \vdots \\ \alpha_n & \boxed{?} & \boxed{?} & \cdots & \boxed{?} \\ & \downarrow & \downarrow & & \downarrow \\ & \underbrace{\boxed{?}\!\|\!\|}_{\boldsymbol{p}_1} & \underbrace{\boxed{?}\!\|\!\|}_{\boldsymbol{p}_2} & \cdots & \underbrace{\boxed{?}\!\|\!\|}_{\boldsymbol{p}_m}. \end{array} \tag{1}$$

Note that if the j-th and ℓ-th cards of the pile $\boldsymbol{p}_i$ (for some j, ℓ) are red:

$$\boldsymbol{p}_i \quad \underset{\clubsuit}{\overset{1}{\boxed{?}}}\underset{\clubsuit}{\overset{2}{\boxed{?}}} \cdots \underset{\heartsuit}{\overset{j}{\boxed{?}}} \cdots \underset{\heartsuit}{\overset{\ell}{\boxed{?}}} \cdots \underset{\clubsuit}{\overset{m}{\boxed{?}}},$$

then both P_j and P_ℓ hold i dollars, i.e., $x_j = x_\ell = i$, and hence, they are tied.
2. Apply a pile-scramble shuffle to $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m)$.
3. Reveal all the piles $(\boldsymbol{p}_{\pi^{-1}(1)}, \boldsymbol{p}_{\pi^{-1}(2)}, \ldots, \boldsymbol{p}_{\pi^{-1}(m)})$. Then, two types of piles should appear: a pile consisting of only $\clubsuit$s (which we call an *empty pile*) or a pile containing $\heartsuit$s (which we call a *non-empty pile*).
If a non-empty pile contains $\heartsuit$s at j-th and ℓ-th positions for some j, ℓ, we have $x_j = x_\ell$, but we cannot know the actual value of $x_j = x_\ell$. Therefore, each non-empty pile tells us an equivalent class, and hence, all the non-empty piles reveal the equality, namely $\{1, \ldots, n\}/\{(i,j) \mid x_i = x_j\}$.

Create-ranking phase. Let k be the number of empty piles, i.e., $k = m - |\{1, \ldots, n\}/\{(i,j) \mid x_i = x_j\}|$.

1. Every player P_i adds a face-down joker[i] to the last of β_i:

$$\beta_i \quad \underbrace{\underset{\heartsuit}{\boxed{?}}}_{\text{1-st}} \; \underset{\heartsuit}{\boxed{?}} \cdots \underset{\heartsuit}{\boxed{?}} \; \underbrace{\underset{\clubsuit}{\boxed{?}}}_{x_i\text{-th}} \; \underset{\heartsuit}{\boxed{?}} \cdots \underset{\heartsuit}{\boxed{?}} \; \underbrace{\underset{\text{Jo}}{\boxed{?}}}_{m+1\text{-st}} \; . \tag{2}$$

2. Create a sequence $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{m+1})$ in the same way as in (1) in Section 3.1.
3. Repeat the following until k empty piles[ii] are removed.
   (a) Apply a pile-shifting shuffle to the sequence.
   (b) Choose (any) one pile and reveal all the cards of the pile.
      – If it is an empty pile, remove it from the sequence.

[i] $\boxed{\text{Jo}}$ can be substituted by $\boxed{\clubsuit}$; we use $\boxed{\text{Jo}}$ for the purpose of easy understanding.
[ii] An empty pile consists of only $\heartsuit$ (and a joker) here.

   - If it is not an empty pile, turn over all the face-up cards and return the pile to the sequence.
4. Apply a pile-shifting shuffle to the sequence.
5. Reveal all the remaining piles. Then, we obtain the ranking by cyclically shifting the revealed piles so that the pile consisting of only jokers is at the end of the sequence.

3.2 Proposal of One-phase Protocol

In this subsection, we construct a protocol without relying on checking equality.

Recall that in the Create-ranking phase of the two-phase protocol, we remove a pile if it is an empty pile. If it is not an empty pile, it goes back to the sequence. Notice that each non-empty pile is unique and contains information on players having the same amount of money, namely the corresponding equivalent class. Therefore, even if we do not know the "equality" in advance, we can find all empty piles by searching them until all piles are revealed. Players can memorize revealed piles so that they can determine when they finish.

Thus, we can skip the Check-equality phase, and our one-phase protocol proceeds as follows.

Setup. P_i creates a sequence of cards β_i in the same way as in (2) in Section 3.1.

Create-ranking phase.

1. Create $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{m+1})$ in the same way as in (1) in Section 3.1.
2. Repeat the following until all $m+1$ piles are revealed.
   (a) Apply a pile-shifting shuffle to the sequence.
   (b) Reveal (any) one pile.
      - If it is an empty pile, remove it from the sequence.
      - If it is not an empty pile, memorize it, turn over all the face-up cards, and return the pile to the sequence.
3. Apply a pile-shifting shuffle to the sequence.
4. Reveal all the remaining piles. We obtain the ranking by shifting the sequence of piles based on the pile consisting of only jokers.

4 Finite-runtime Protocols

The two proposed protocols in the previous section are not finite-runtime (because we have to repeat the shuffle until all empty piles are found). In this section, we solicit finite-runtime protocols that perform the secure ranking computation with a fixed number of shuffles. Specifically, we design two protocols considering the tradeoff between the number of cards and the number of shuffles. First, we propose the finite-runtime shuffle-efficient protocol specialized for reducing the number of shuffles. Next, we introduce a technique for reducing the number of cards, and propose the finite-runtime card-efficient protocol.

4.1 Proposal of Shuffle-efficient Protocol

In this protocol, we first find all empty piles at once, add markers to them, and search the empty piles with the help of the markers.

1. P_i creates a sequence of cards β_i in the same way as in (2) in Section 3.1.
2. Create $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{m+1})$ in the same way as in (1) in Section 3.1.
3. For every j, $1 \le j \le m+1$, add a number card j to the top of the pile $\boldsymbol{p}_j$ so that we have a new pile $\boldsymbol{p'}_j$ of size $n+1$.
4. Apply a pile-scramble shuffle to the sequence of piles:

$$\underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}(1)}}\ \underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}(2)}}\ \cdots\ \underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}(m+1)}}\ .$$

5. Reveal all the cards except for the number cards. Now, we find all the empty piles (whose positions are randomly permuted). Let k be the number of empty piles.
6. For each found empty pile, add a marker which consists of m ♣s and one ♡ being placed on the piles so that the red card ♡ indicates the empty pile, as follows:

$$\begin{array}{lcccc} \text{marker} & \boxed{?} & \boxed{?} & \cdots & \boxed{?} \\ & ♣ & ♡ & & ♣ \\ & \boxed{?} & \underbrace{\boxed{?}}_{\text{empty pile}} & \cdots & \boxed{?}. \end{array}$$

   One marker indicates one empty pile, and hence, we stack k markers on the sequence of piles. We denote the resulting sequence by $(\boldsymbol{p''}_1, \boldsymbol{p''}_2, \ldots, \boldsymbol{p''}_{m+1})$.
7. Apply a pile-scramble shuffle to the sequence of piles:

$$\underbrace{\boxed{?}}_{\boldsymbol{p''}_{\pi_2^{-1}(1)}}\ \underbrace{\boxed{?}}_{\boldsymbol{p''}_{\pi_2^{-1}(2)}}\ \cdots\ \underbrace{\boxed{?}}_{\boldsymbol{p''}_{\pi_2^{-1}(m+1)}}\ .$$

8. Reveal only the number cards, each of which is the $(k+1)$-st card from the top of the corresponding pile, and rearrange the order of the piles so that the number cards are in ascending order; remove all the number cards:

$$\underbrace{\boxed{?}}_{\boldsymbol{p'''}_1}\underbrace{\boxed{?}}_{\boldsymbol{p'''}_2}\ \cdots\ \underbrace{\boxed{?}}_{\boldsymbol{p'''}_{m+1}}\ .$$

9. We remove all empty piles without anyone getting to know the “distance” between any pair of non-empty piles: Repeat the following until k empty piles are removed.
   (a) Apply a pile-shifting shuffle to the sequence.
   (b) Reveal one marker and remove the corresponding empty pile.
10. Apply a pile-shifting shuffle to the sequence.
11. Reveal all the remaining piles. We obtain the ranking by shifting the sequence of piles based on the pile consisting of only jokers.

4.2 Proposal of Card-efficient Protocol

In the finite-runtime shuffle-efficient protocol presented in Section 4.1, we use k markers, i.e., we use $k(m+1)$ cards as markers. We consider a method of reducing the number of cards used for these markers. If we arrange only one marker in Step 6 of the finite-runtime shuffle-efficient protocol, we can compute the ranking by repeating the process from Step 3 to Step 9 k times. Then, the number of cards used for markers can be reduced from $k(m+1)$ to $m+1$. However, the number of required shuffles increases from $k+3$ to $3k+1$.

By extending this idea further, no marker is needed. We construct a finite-runtime card-efficient protocol that does not rely on markers. The procedure is shown below.

1. P_i creates a sequence of cards β_i in the same way as in (2) in Section 3.1.
2. Create $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{m+1})$ in the same way as in (1) in Section 3.1.
3. Repeat the following until k empty piles are removed (while repeating, the number of piles is decreasing from $m+1$ to $m+1-k$):
   (a) Apply a random cut to a sequence consisting of number cards:
   $$\underset{1}{\boxed{?}}\,\underset{2}{\boxed{?}}\cdots\underset{m+1}{\boxed{?}} \rightarrow \underset{1+r}{\boxed{?}}\,\underset{2+r}{\boxed{?}}\cdots\underset{m+1+r}{\boxed{?}}\,.$$
   (b) For each pile $\boldsymbol{p}_j$, add the number card $j+r$ to the pile so that we have a new pile $\boldsymbol{p'}_j$.
   (c) Apply a pile-scramble shuffle to the sequence of piles:
   $$\underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}(1)}}\;\underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}(2)}}\;\cdots\;\underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}(m+1)}}\,.$$
   (d) Reveal all the cards except for the number cards.
   (e) Remove one empty pile along with the number card. Note that because the removed number card does not indicate the position of the pile (because of the random permutation π_1), information on the input is not leaked.
   (f) Turn all the cards face-down and apply a pile-scramble shuffle to the sequence of piles:
   $$\underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}\left(\pi_2^{-1}(1)\right)}}\;\underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}\left(\pi_2^{-1}(2)\right)}}\;\cdots\;\underbrace{\boxed{?}}_{\boldsymbol{p'}_{\pi_1^{-1}\left(\pi_2^{-1}(m+1)\right)}}\,.$$
   (g) Reveal only the number cards, each of which is the top of the corresponding pile, and rearrange the order of the piles so that the number cards are in ascending order. Remove all the number cards.
4. Apply a pile-shifting shuffle to the sequence of piles.
5. Reveal all the remaining piles. We obtain the ranking by shifting the sequence of piles based on the pile consisting of only jokers.

5 Discussion

In this section, we discuss comparison of protocols and real use cases.

Comparing Protocols. We compare the two finite-runtime protocols (presented in Section 4) with the repetition of the existing millionaire protocol [6] in terms of the number of cards and shuffles, and the input range m. We fix the number of players at $n = 5$, the number of color cards at 52, and the number of number cards at 10.

Figure 1 shows the number of shuffles required to execute the three protocols. One can confirm that our protocols use fewer shuffles compared to the case of repeating the millionaire protocol.

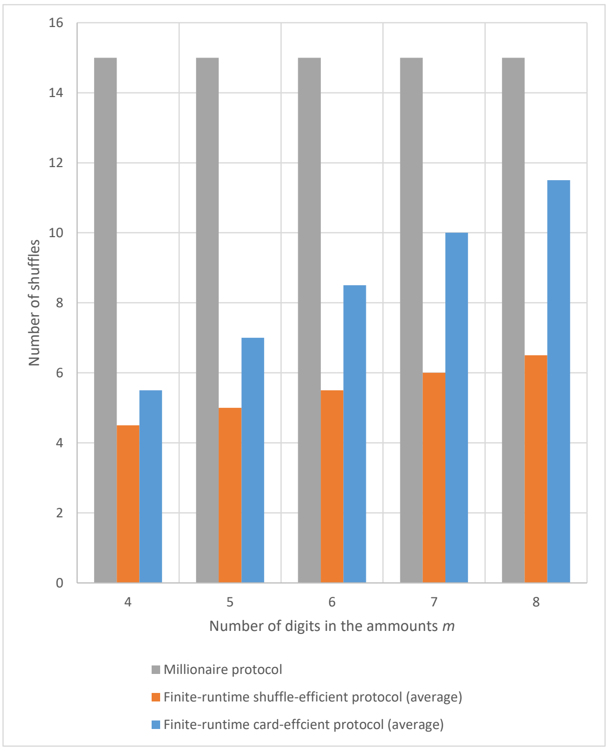

Fig. 1. Number of required shuffles ($n = 5$).

Use Case. Next, we discuss how our protocols can be used. To reduce the number of cards, it is reasonable to compare digits in the amounts instead of direct amounts of money. For example, if the amount of money that any player holds is between 10 and $10^{y+1} - 1$ for some $y \in \mathbb{N}$, we can set the input range to $\{1, \ldots, y\}$.

Assume that the number of players is $n = 5$ and the number of digits in the amounts is $m = 6$. For example, if a player P_i holds more than 10^3 dollars and less than 10^4 dollars, she sets to $x_i = 3$, if she holds 10^6 dollars, she sets to $x_i = 6$, and so on. In this case, 42 cards are required to execute our card-efficient protocol. The breakdown is as follows: 10 ♣s, 25 ♡s, and 7 number cards. Assume that, for instance, the number of empty piles is $k = 3$, and the number of required shuffles is 10. Both the numbers of cards and shuffles are within a feasible range in practice.

If there are two or more players at the same rank, it is possible to compare the amounts of money between the players in finer units. If there are three or more players to compare, using our finite-runtime shuffle-efficient protocol or finite-runtime card-efficient protocol is efficient; if there are only two players, using the existing millionaire protocol would be a good choice.

6 Conclusion

In this paper, we first proposed card-based secure ranking computation protocols. While our two simple protocols are Las Vegas algorithms, our shuffle-efficient protocol and card-efficient protocol always terminate with a finite number of shuffles. Future work will be to devise a protocol that works with a standard deck of playing cards. Finding lower bounds on the number of shuffles for a secure ranking computation protocol is also interesting.

Acknowledgement. This work was supported by JSPS KAKENHI Grant Numbers JP17K00001, JP17J01169, and JP19J21153. We would like to thank the anonymous reviewers for their fruitful comments.

References

1. den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology — EUROCRYPT '89. Lecture Notes in Computer Science, vol. 434, pp. 208–217. Springer, Berlin, Heidelberg (1990)
2. Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) Advances in Cryptology — CRYPTO' 93. Lecture Notes in Computer Science, vol. 773, pp. 319–330. Springer, Berlin, Heidelberg (1994)
3. Hibiki Ono, Y.M.: Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In: 2018 13th Asia Joint Conference on Information Security (AsiaJCIS). pp. 23–28. IEEE (2018)
4. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Calude, C.S., Dinneen, M.J. (eds.) Unconventional Computation and Natural Computation. Lecture Notes in Computer Science, vol. 9252, pp. 215–226. Springer, Cham (2015)

5. Jiang, S., Gong, G.: A round and communication efficient secure ranking protocol. In: Pointcheval, D. (ed.) Topics in Cryptology – CT-RSA 2006. Lecture Notes in Computer Science, vol. 3860, pp. 350–364. Springer, Berlin, Heidelberg (2006)
6. Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: Practical and easy-to-understand card-based implementation of Yao's millionaire protocol. In: Donghyun, K., Alexander, Z., Uma, R.N. (eds.) Combinatorial Optimization and Applications. Lecture Notes in Computer Science, vol. 11346, pp. 246–261. Springer, Cham (2018)
7. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E100.A**(1), 3–11 (2017). https://doi.org/10.1587/transfun.E100.A.3
8. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) Frontiers in Algorithmics. Lecture Notes in Computer Science, vol. 5598, pp. 358–369. Springer, Berlin, Heidelberg (2009)
9. Nakai, T., Tokushige, Y., Misawa, Y., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations. In: Foresti, S., Persiano, G. (eds.) Cryptology and Network Security. Lecture Notes in Computer Science, vol. 10052, pp. 500–517. Springer, Cham (2016)
10. Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E101.A**(9), 1494–1502 (2018). https://doi.org/10.1587/transfun.E101.A.1494
11. Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Martín-Vide, C., Mizuki, T., Vega-Rodríguez, M.A. (eds.) Theory and Practice of Natural Computing. Lecture Notes in Computer Science, vol. 10071, pp. 58–69. Springer, Cham (2016)
12. Yao, A.C.: Protocols for secure computations. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science. pp. 160–164. FOCS '82, IEEE Computer Society, Washington, DC, USA (1982). https://doi.org/10.1109/SFCS.1982.88, `http://dx.doi.org/10.1109/SFCS.1982.88`