Card-Based Secure Sorting Protocols Based on the Sorting Networks

Kota Kato, Takeshi Nakai and Koutarou Suzuki

September 30, 2024

Card-based Secure Sorting Protocols based on the Sorting Networks

Kota Kato
Toyohashi University of Technology
Aichi, Japan
kato.kota.gc@tut.jp

Takeshi Nakai
Toyohashi University of Technology
Aichi, Japan
nakai@cs.tut.ac.jp

Koutarou Suzuki
Toyohashi University of Technology
Aichi, Japan
suzuki@cs.tut.ac.jp

***Abstract*—Card-based cryptography enables us to realize secure computation using physical cards with simple manual operations. The efficiency of a card-based protocol is evaluated by the number of shuffles and cards. The Haga et al. (IWSEC2022) is the only work to address card-based secure sorting. They proposed two protocols: a Las Vegas protocol and a finite-runtime protocol. The finite-runtime means that the number of shuffles is deterministic. In this paper, we propose a general converter that transforms an arbitrary sorting network into a finite-runtime card-based sorting protocol. The converter uses Haga et al.'s finite-runtime protocol for $n = 2$ as its building block. Furthermore, we show card-based sorting protocol obtained by applying the converter to the AKS sorts. As a result, we improve the number of additional cards from $O(n^2 + m)$ to $O(n + m)$ compared to Haga et al. On the other hand, the number of shuffles increases from $O(nm)$ to $O(nm \log n)$.**

***Index Terms*—Card-based Cryptography, Secure Sorting Protocol, Sorting Network**

I. Introduction

A. Background

Secure computation involves cryptographic protocols that enable mutually distrustful parties to jointly compute a function on their private inputs [25]. While secure computation protocols are supposed to be implemented on computers typically, there is a line of work to realize them using physical objects with manual operations. Since such physical protocols allow hands-on and visual understanding, they are suitable as educational and recreational tools.

Card-based cryptography is one of such physical protocols, and uses a deck of physical cards in the implementation [6], [7]. A card-based protocol consists of simple manual operations: permuting a card order, flipping a card, and shuffling a deck of cards. We evaluate the efficiency of a card-based protocol by the number of cards and the number of shuffles.

Many researches in card-based cryptography have been devoted to constructing efficient protocols for basic logical functions, such as AND, XOR, and COPY [6], [7], [10], [11], [15], [16], [19], [24]. This is because a composition of them can realize a secure protocol for any function [20]. Besides, there is another approach of constructing specific protocols for more advanced functions, e.g., the millionaires' problem [12], [17], [22], the majority voting [1], [2], [13], [18], the private set intersection [5], and the secure sorting [8].

This work was supported by JSPS KAKENHI Grant Number JP23K16880.

Haga et al. [8] were the first to address the secure sorting in card-based cryptography. A secure sorting protocol enables parties to sort a sequence of encrypted (or secret-shared) values without revealing the underlying values. A card-based sorting protocol provides a similar functionality for a face-down card sequence. They proposed two card-based sorting protocols, a Las Vegas protocol and a finite-runtime (which means that the number of shuffles is deterministic) protocol. The Las Vegas protocol returns the correct output in a probabilistic manner. Thus, players may require an infinitely large number of steps to obtain the correct output. The expected value of the number of shuffles is $O(nm)$, where n is the number of elements to be sorted and m is the bit length of each element. This protocol requires $O(n + m)$ additional cards. (The additional cards refer to the cards that need to be used other than cards representing the input sequence.) Haga et al. showed a way of converting the protocol into a finite-runtime protocol, which returns the correct output in a fixed number of steps, their protocol is not efficient since the number of additional cards is $O(n^2 + m)$. Also, their finite-runtime protocol requires $O(nm)$ shuffles.

B. Our Contribution

In this paper, we construct efficient finite-runtime card-based sorting protocols in terms of the number of cards. We first present a general converter that transforms an arbitrary sorting network to a finite-runtime card-based sorting protocol. The converter uses as its building block a card-based sorting protocol of which the number of input elements is two. This building block corresponds to comparators in sorting networks. By realizing it with Haga et al.'s protocol for $n = 2$, we show that the protocol obtained by our converter can be implemented with $m + 20$ cards for an arbitrary sorting network. However, this evaluation is based on the assumption that the input satisfies the conditions required by the underlying sorting network. Note that some sorting networks have a limitation to their input length.

We show card-based sorting protocols obtained by applying this coverter to concrete sorting networks: the AKS [3], [4] sorting networks. It require that the number of input elements is a power of two. In the case where the input is not a power of two, we must adjust the length by appending cards as a padding to the input. Hence, in this case, the protocols require more additional cards than the $m+20$ ones described above.

Tables I and II show the efficiency comparisons in the case where the number of input elements n is and is not a power of two, respectively. In the case where n is a power of two, our protocol can be constructed with $O(m)$ cards, which is more efficient than Haga et al.'s finite-runtime protocol that requires $O(n^2+m)$ cards. On the other hand, regarding the number of shuffles, Haga et al.'s protocol requires $O(nm)$ shuffles, which is more efficient than our protocol that $O(nm\log n)$ shuffles.

In the case where n is not a power of two, our protocol can be constructed with $O(n+m)$ cards due to the padding cards for the input, while Haga et al.'s finite-runtime protocol requires $O(n^2+m)$ cards. Regarding the number of shuffles, the evaluations are the same as the case where n is a power of two.

II. Preliminaries

This section gives the notations and basic definitions of card-based protocols. A protocol to compute a function f is correct if players obtain $f(x)$ for an arbitrary input x. (Note that the function f is a sort function in this paper.) We say a protocol fulfills privacy if an arbitrary player cannot learn any additional information than the output from his/her view of the protocol.

A. Notations

We use two types of colored cards, red $\boxed{\heartsuit}$ and black $\boxed{\clubsuit}$. Our protocol makes secondarily use of several other types of cards; numbered cards $\boxed{1}\boxed{2}$, marker cards $\boxed{*}$, and arrow cards $\boxed{\leftarrow}\boxed{\rightarrow}$. We assume that the cards with the same suit are indistinguishable. We also assume that back sides of all cards have the same pattern, denoted by $\boxed{?}$, and are indistinguishable.

A Boolean value is encoded as $0 \mapsto \boxed{\clubsuit}\boxed{\heartsuit}$ and $1 \mapsto \boxed{\heartsuit}\boxed{\clubsuit}$. For a non-negative integer x, let $x[i]$ show the i-th bit value of x. For an m-bit integer, the most and least significant bits are referred to $(m-1)$-th and 0th bits, respectively. An m-bit non-negative integer x is represented with $2m$ face-down cards. We call a sequence of face-down cards representing a non-negative integer x a commitment of x.

$$\begin{array}{ccc}
\boxed{?}\boxed{?} & \leftarrow & x[m-1] \\
\vdots & & \vdots \\
\boxed{?}\boxed{?} & \leftarrow & x[1] \\
\boxed{?}\boxed{?} & \leftarrow & x[0].
\end{array}$$

In the depiction of a commitment, the top (resp. bottom) row corresponds to the most (resp. least) significant bit.

In this paper, a protocol consists of the following operations.

- Permutation: permuting a card order deterministically.
- Turn: changing the face of a card.
- Shuffle: permuting a card order with some probability distribution.

Formally, for the d-th symmetric group S_d corresponding to a card sequence, a shuffle for S_d is defined with two parameters, a permutation set $\Pi \subseteq S_d$ and a probability distribution $\mathcal{F}$ on Π. That is, a shuffle $(\mathsf{shuffle}, \Pi, \mathcal{F})$ is a permutation for S_d according to $\pi \in \Pi$ sampled from $\mathcal{F}$. We assume that any player cannot identify the sampled permutation π, i.e., no player learn the result of the shuffle.

B. Shuffles Used in Our Protocols

We call a pile a sequence of cards regarded as one bundle.

Pile-Scramble Shuffle: A pile-scramble shuffle [9] is a shuffle that completely randomizes the order of multiple piles consisting of the same number of cards. Suppose that for a positive integer s, we have s piles $(\vec{p}_0, \vec{p}_1, \ldots, \vec{p}_{s-1})$, each having the same number of cards. Let r be a permutation chosen uniformly at random from the s-th symmetric group. Let $[\,\cdot\,|\,\cdot\,]$ represent a pile-scramble shuffle. The pile-scramble shuffle result is as follows.

$$\underbrace{\boxed{?}\cdots\boxed{?}}_{\vec{p}_{r^{-1}(0)}}\underbrace{\boxed{?}\cdots\boxed{?}}_{\vec{p}_{r^{-1}(1)}}\cdots\underbrace{\boxed{?}\cdots\boxed{?}}_{\vec{p}_{r^{-1}(s-1)}}$$

Pile-Shifting Shuffle: A pile-shifting shuffle [21] is a shuffle that cyclically and randomly shifts the order of piles consisting of the same number of cards. Suppose that for a positive integer s, we have s piles of cards $(\vec{p}_0, \vec{p}_1, \ldots, \vec{p}_{s-1})$ each having the same number of cards. Let r be chosen uniformly at random from $\{0, 1, \cdots, s-1\}$. The pile shifting shuffle result is as follows.

$$\underbrace{\boxed{?}\cdots\boxed{?}}_{\vec{p}_{r}}\underbrace{\boxed{?}\cdots\boxed{?}}_{\vec{p}_{r+1}}\cdots\underbrace{\boxed{?}\cdots\boxed{?}}_{\vec{p}_{r+s-1}}$$

III. Existing Card-Based Sorting Protocol

A. Definition of Card-based Sorting Protocol

A card-based sorting protocol aims to sort a sequence of commitments while keeping their values secret. Precisely, it provides the following functionality with privacy.

Input: $X = (\tilde{x}_0, \ldots, \tilde{x}_{n-1})$, where $\tilde{x}_i$ is a commitment of an m-bit non-negative integer x_i.

Output: $X' = (\tilde{x}_{\phi(0)}, \ldots, \tilde{x}_{\phi(n-1)})$, where ϕ is the permutation on $\{0, \ldots, n-1\}$ that satisfies $x_{\phi(i)} \leq x_{\phi(i+1)}$ for all $i \in \{0, 1 \cdots, n-2\}$.

B. Procedure of Haga et al.'s Protocol

Haga et al.'s work [8] is the first and only work so far to address card-based sorting protocol. It shows two card-based sorting protocols: Las Vegas and finite-runtime protocols. We here describe the finite-runtime one. Although the protocol is general for any number of inputs n, we focus on the case of $n = 2$, which is the only case we use to construct our protocol.

The two-input sorting protocol is as follows.

0. Players hold $\tilde{x}_0$ and $\tilde{x}_1$, which are commitments of m-bit non-negative integers x_0 and x_1, respectively. They also have $2 \times \boxed{\heartsuit}$, $8 \times \boxed{\clubsuit}$, $2 \times \boxed{1}$, $2 \times \boxed{2}$, $2 \times \boxed{\leftarrow}$, $2 \times \boxed{\rightarrow}$, and $(m+2) \times \boxed{*}$.

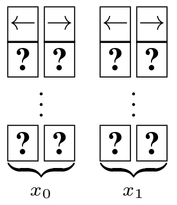
TABLE I
Comparison of card-based sorting protocols in the case where the number of input elements is a power of two.
(n: the number of input elements, m: the bit length of each element)

Protocol	# additional cards	# shuffles	Underlying sort	Runtime
Haga et al. [8]	$O(n+m)$	$O(nm)$	Radix sort	Las Vegas
Haga et al. [8]	$O(n^2+m)$	$O(nm)$	Radix sort	Finite
Ours (Section V-A)	$O(m)$	$O(nm\log n)$	AKS sort	Finite

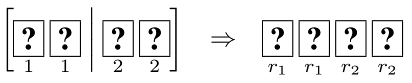
TABLE II
Comparison of sorting protocols in the case where the number of input elements is not a power of two.

Protocol	# additional cards	# shuffles	Underlying sort	Runtime
Haga et al. [8]	$O(n+m)$	$O(nm)$	Radix sort	Las Vegas
Haga et al. [8]	$O(n^2+m)$	$O(nm)$	Radix sort	Finite
Ours (Section V-A)	$O(n+m)$	$O(nm\log n)$	AKS sort	Finite

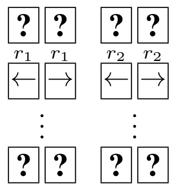
1. Arrange the arrow cards above the commitments as follows.



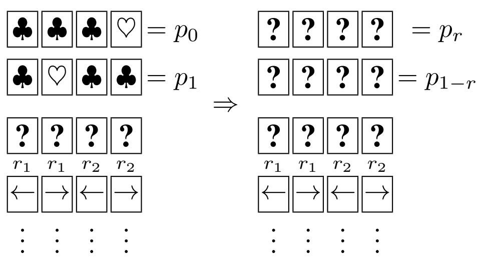
2. For $i = 0, 1, \cdots, m-1$, perform the following operations:
   2-1. Prepare the numbered cards $\boxed{1}\boxed{1}\boxed{2}\boxed{2}$, and make two piles, $\boxed{1}\boxed{1}$ and $\boxed{2}\boxed{2}$. Afterwards, apply a pile-scramble shuffle to them.



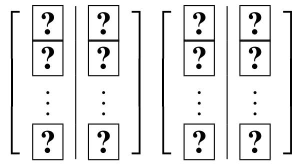
   2-2. Place these four cards above the arrow cards as follows:



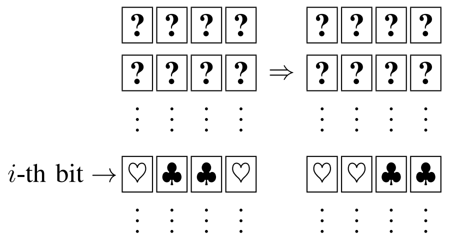
   2-3. Prepare $6\times\boxed{\clubsuit}$ and $2\times\boxed{\heartsuit}$, and make two piles $p_0 = \boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}$ and $p_1 = \boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}$. Then, apply a pile-scramble shuffle to the two piles, denoted by (p_r, p_{1-r}), $r \in \{0,1\}$ the result, and place them the above of the numbered cards as follows.



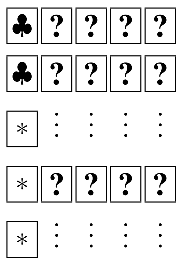
   2-4. Make the arrow cards face down. Afterwards, apply a pile-scramble shuffle to each of the 0th and 1st columns, and the 2nd and 3rd columns as follows:



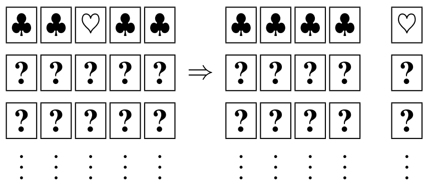
   2-5. Reveal the cards corresponding to the i-th bits and perform a stable sort according to the relationship $\boxed{\heartsuit} > \boxed{\clubsuit}$ while keeping the card order of each column unchanged. Then, make all of the revealed cards face-down.



   2-6. Prepare $2\times\boxed{\clubsuit}$ and $(m+2)\times\boxed{*}$, and arrange them at the leftmost of the card sequences as follows.

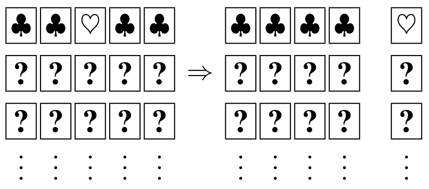


   2-7. Regard each column as a pile, i.e., five piles, and apply a pile-shifting shuffle to them.
   2-8. Reveal all cards on the topmost row, and move the pile of the $\boxed{\heartsuit}$ columns. Then, remove all of the revealed cards. (Note that the revealed cards correspond to the pile p_r)

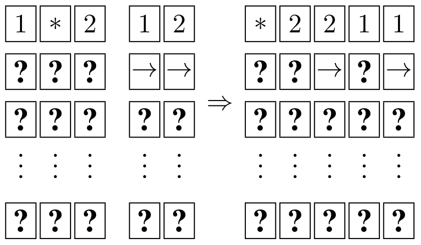


2-9. Regard each column as a pile (i.e., there is five piles), and apply a pile-shifting shuffle to them.

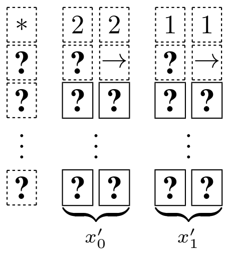
2-10. Reveal all cards on the topmost row, and move the pile of the $\boxed{\heartsuit}$ columns. Then, remove all of the revealed cards. (Note that the revealed cards correspond to the pile p_{1-r})



2-11. Reveal all cards on the topmost row, and permute them cyclically so that the pile of the $\boxed{*}$ column is in the leftmost. Then restore the commitments by moving the pile of the right-arrow columns to the appropriate positions as follows:



2-12. Remove all cards except the commitments. (In the following figure, the cards whose frames are dotted correspond to the cards to be removed.)



The protocol outputs the sequence of two commitments $(\tilde{x}'_0, \tilde{x}'_1)$. Then, the underlying values (x'_0, x'_1) is always the sorted result of (x_0, x_1). Hereafter, we denote by $\Pi^{n=2}_{\mathrm{HTS+}}$ the above protocol.

Note that both numbered and arrow cards can be substituted by the two-colored cards in $\Pi^{n=2}_{\mathrm{HTS+}}$, such as $\boxed{1} \to \boxed{\clubsuit}$, $\boxed{2} \to \boxed{\heartsuit}$, $\boxed{\leftarrow} \to \boxed{\clubsuit}$, and $\boxed{\rightarrow} \to \boxed{\heartsuit}$. In other words, the protocol can be constructed with only the two-colored cards and the marker cards.

C. Efficiency

$\Pi^{n=2}_{\mathrm{HTS+}}$ uses $m + 20$ additional cards as described in the previous subsection. It is important to note that if the protocol is carried out multiple times sequentially, we can reuse the additional cards. That is to say, we can realize multiple runs with the same $m + 20$ additional cards.

Regarding the number of shuffles, $\Pi^{n=2}_{\mathrm{HTS+}}$ requires three pile scramble shuffles and two pile shifting shuffles per bit. Since each input element consists of m-bit, it requires $5m$ shuffles.

IV. Proposed "Sorting Network to Card-based Sorting Protocol" Converter

A. Comparator Network and Sorting Network

Let X be a set of non-negative integers. A comparator is defined as a mapping $X^n \to X^n$ as follows.

Definition 1. *Let $[l, r]$ be a mapping $X^n \to X^n$, where $l, r \in \{0, \cdots, n-1\}$. Denote by $[l, r](x)_k$ the k-th element of $[l, r](x)$. We call $[l, r]$ comparator if it fulfills the following properties for any $x = (x_1, \cdots, x_n) \in X^n$:*

$$[l, r](x)_l = \mathsf{min}(x_l, x_r)$$
$$[l, r](x)_r = \mathsf{max}(x_l, x_r)$$

Moreover, $[l, r](x)_k = x_k$ holds for all $k \in \{0, \cdots, n-1\} \setminus \{l, r\}$.

That is, the comparator $[l, r]$ is the sort between the l-th and r-th elements. The compartor and sorting networks are defined based on the comparator as follows.

Definition 2. *A comparator stage $S = \{[l_1, r_1], ..., [l_k, r_k]\}$ is a composition of comparators such that $l_1, \cdots, l_k, r_1, \cdots, r_k$ are distinct.*[1]

Definition 3. *A comparator network is a composition of comparator stages.*

Definition 4. *A sorting network is a comparator network such that, for an arbitrary input, its output sequence, denoted by $(x'_0, \cdots, x'_{n-1})$, fulfills $x'_i \le x'_{i+1}$ for all $i \in \{0, \ldots, n-2\}$.*

B. Our Proposed Converter

Before presenting our converter, we define card-based comparator protocol, denoted by $\mathsf{Compare}(X; l, r)$, as follows, where X is a private input and $l, r \in \{0, \ldots, n-1\}$ are public inputs.

Input: $X = (\tilde{x}_0, \ldots, \tilde{x}_{n-1})$, where $\tilde{x}_i$ is a commitment of an m-bit non-negative integer x_i, and indices $l, r \in \{0, \ldots, n-1\}$.

Output: The sequence of commitments that consist of $[l, r](X)$.

Obviously, we can realize the comparator protocol by a single execution of $\Pi^{n=2}_{\mathrm{HTS+}}$ for $(\tilde{x}_l, \tilde{x}_r)$.

We present a general converter that transforms any sorting network into a card-based sorting protocol. See Protocol 1 that shows our proposed converter. Given a sequence of commitments, the compiled protocol returns the sorting result of the commitments. This protocol consists of multiple card-based comparator protocols, so that each of them corresponds to a comparator in the given sorting network. More generally,

[1]Precisely, a comparator stage is a mapping $X^n \to X^n$. However, we denote a stage as a set of comparators for ease of describing our protocol.

our converter can also be used to transform any comparator network into a card-based protocol that provides the same functionality of the comparator network.

Correctness: The comparator protocol guarantees to sort two commitments correctly in each comparison. Since each comparator protocol corresponds to a comparator in the underlying sorting network, the protocol provides the result of the sorting network. Thus, the proposed protocol ensures parties to always obtain the correct output.

Privacy: Since the comparator protocol fulfills the privacy, it guarantees to keep the values of commitments secret in each comparison. Moreover, the process of the sorting network is deterministic and depends only on the input length n. Thus, the proposed protocol hides information other than the output.

C. *Efficiency*

We evaluate the protocol obtained by our converter in the case where a comparator protocol is realized with $\Pi^{n=2}_{\text{HTS+}}$. Since our sorting protocol uses $\Pi^{n=2}_{\text{HTS+}}$ sequentially, we can reuse the additional cards of $\Pi^{n=2}_{\text{HTS+}}$. Thus, the number of additional cards in our sorting protocol is $m + 20$.

$\Pi^{n=2}_{\text{HTS+}}$ requires $5m$ shuffles, and the compiled protocol requires executes $\Pi^{n=2}_{\text{HTS+}}$ for c times, where c is the number of comparators of the underlying sorting networks. Thus, the number of shuffles is $5mc$.

Note that there are sorting networks that have a limitation to the number of input elements n. For instance, the AKS sort requires that n is a power of two. The above evaluation is based on the assumption that the input meets a condition required by the underlying sorting network.

The efficiency with parallel executions: Let $k_0, \cdots, k_{t-1}$ be the number of comparators in each stage. Also, we denote by k_{max} the maximum number in $\{k_0, \cdots, k_{t-1}\}$. For any sorting network, comparators in a common stage can run in parallel. This is also true for our card-based sorting protocol. That is, we can carry out comparator protocols for a common stage in parallel. It is known that multiple shuffles performed simultaneously can be composed in one shuffle [14]. Thus, if we apply the parallel manner, we can realize a secure sorting protocol with $5mt$ shuffles. Then, it requires $k_{\text{max}}(m + 20)$ additional cards.

V. Our Card-based Sorting Protocols

In this section, we show card-based sorting protocol obtained by applying our converter, shown in the previous section, to the AKS sorting network [3], [4].

A. *Card-based Sorting Protocol Based on AKS Sort*

The AKS sorting network is known as the optimal sorting network.

The AKS sorting network uses as its component a comparator network, called an ϵ-halver. Given $X = (x_0, \ldots, x_{n-1})$ as the input, the ϵ-halver returns two blocks $L = \{x'_0, \cdots, x'_{(n-1)/2-1}\}$ and $R = \{x'_{(n-1)/2}, \cdots, x'_{n-1}\}$ for a parameter $\epsilon < 1$. Let $\hat{X} = (\hat{x}_0, \ldots, \hat{x}_{n-1})$ be the ordered sequence of X, and let $\hat{X}_{L,i} := \{\hat{x}_0, \cdots, \hat{x}_i\}$ and $\hat{X}_{R,i} := \{\hat{x}_{n-(i+1)}, \cdots, \hat{x}_{n-1}\}$ for $i \in \{0, \ldots, (n-1)/2 - 1\}$. Then, for any $i \in \{0, \ldots, (n-1)/2 - 1\}$, the ϵ-halver ensures that $|\hat{X}_{L,i} \cap R| < \epsilon i$ and $|\hat{X}_{R,i} \cap L| < \epsilon i$.

The ϵ-halver can be constructed with a constant number of comparator stages by using expander graphs. (The constant number depends on ϵ.) Ajtai et al. [3], [4] showed that there is a sorting network with $O(\log n)$ stages and $O(n \log n)$ comparators based on this fact.[2] (See [3], [4], [23] for the detail.)

From the above, we can claim that there is a card-based sorting protocol that consists of $O(n \log n)$ comparator protocols. The AKS sort has the limitation that the number of input elements n must be a power of two. Hence, we use the padding if the input does not hold the condition, as shown in the previous subsection. If the number of input elements is not a power of two, it is necessary to adjust the input as follows.

How to adjust the input length.: We present a way to apply our protocol where the number of input elements n is not a power of two. For an input X, which consists of n commitments (n is not a power of two), we consider d dummies, where d is the minimum positive integer such that $n + d$ is a power of two. Also, we prepare one card per dummy, and the dummies are treated as pseudo-zero in the sorting. We use these dummies to store the intermediate sorting result of the underlying sorting network. Players append the dummies to any positions in the input sequence and carry out the sorting protocol. In a comparison between a dummy and an input element, the players can determine that the dummy is smaller than or equals to the input element without running a comparator protocol since we set the dummy as zero. Similarly, in a comparison between two dummies, the players can skip the comparison. At the end of the sorting protocol, players remove all of the dummy inputs. Note that since we set the dummies to minimum number, they always locate at the leftmost positions in the output sequence. Hence, players can obtain the sorting result of X by removing the leftmost d dummies.

Privacy: We discuss the privacy of our card-based sorting protocol based on the AKS sort. Then, we omit the cases where n is a power of two since the proof is the same as in Section IV-B. Hence, we here focus on the case where n is not a power of two. In this case, we append the dummies to the input sequence. The comparison results for the dummies are obvious because the value of the dummies is the minimum number. In addition, the sorting network process is deterministic and depends only on the input length n. As a result, we know the movement transitions of the dummies and they are always in the leftmost positions in the output sequence. However, we do not know the movement transitions of the non-dummies. Thus, this case hides information other than the output.

[2]The constant factor is very large as stated in [3], and thus the AKS sort is not practical. Due to this fact, the card-based sorting protocol obtained from this sort is also impractical.

Protocol 1 Card-based sorting protocol based on a sorting network

converter: Given a sorting network $N = (S_0, \cdots, S_{t-1})$, where t be the number of comparator stages, and $S_i = \{[l_{i,0}, r_{i,0}], ..., [l_{i,k_i-1}, r_{i,k_i-1}]\}$, the converter generates the following card-based sorting protocol.

Input: $X_0 = (\tilde{x}_0, \ldots, \tilde{x}_{n-1})$, where $\tilde{x}_i$ is a commitment of m-bit non-negative integer x_i.
Output: $(\tilde{x}_{\phi(0)}, \ldots, \tilde{x}_{\phi(n-1)})$, where ϕ is the permutation that satisfies $x_{\phi(i)} \leq x_{\phi(i+1)}$ for all $i \in \{0, 1, \cdots, n-2\}$.

```
1: Initialize $c = 0$.
2: for $i = 0$ to $t-1$ do
3:     for $[l_{i,j}, r_{i,j}] \in S_i$ do
4:         Run $X_{c+1} \leftarrow \mathsf{Compare}(X_c; l_{i,j}, r_{i,j})$.
5:         $c = c + 1$.
```

Efficiency: We evaluate the efficiency separately for the cases where the input length n is a power of two or not. As in Section IV, we suppose that comparator protocols are realized with $\Pi^{n=2}_{\mathrm{HTS}+}$ and they run sequentially.

In the case where n is a power of two, the number of additional cards is $m + 20$. The number of shuffles is $5mc$, where c is the number of comparators of the underlying sorting networks. Note that c is evaluated as $O(n \log n)$ and the exact value of c is not provided in [3], [4], [23]. Thus, the protocol requires $O(nm \log n)$ shuffles.

Next, let us consider the case where n is not a power of two. To consider the worst case, suppose $n = 2^v + 1$ for a positive integer v. In this case, we must prepare $n - 2$ dummy cards. In addition, $\Pi^{n=2}_{\mathrm{HTS}+}$ requires $m + 20$ additional cards. Thus, the sorting protocol requires $n + m + 18$ cards in total. It also requires $O(nm \log n)$ shuffles.

References

[1] Yoshiki Abe, Takeshi Nakai, Yoshihisa Kuroki, Shinnosuke Suzuki, Yuta Koga, Yohei Watanabe, Mitsugu Iwamoto, and Kazuo Ohta. Efficient card-based majority voting protocols. *New Generation Computing*, 40(1):173–198, apr 2022.

[2] Yoshiki Abe, Takeshi Nakai, Yohei Wawanabe, Mitsugu Iwamoto, and Kazuo Ohta. A computationally efficient card-based majority voting protocol with fewer cards in the private model. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, advpub:2022CIP0021, 2022.

[3] M. Ajtai, J. Komlós, and E. Szemerédi. An 0(n log n) sorting network. *Proceedings of the fifteenth annual ACM symposium on Theory of computing - STOC '83*, 1983.

[4] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in clog (n) parallel steps. *Combinatorica*, 3:1–19, 1983.

[5] Doi Anastasiia, Ono Tomoki, Nakai Takeshi, Shinagawa Kazumasa, Watanabe Yohei, Nuida Koji, and Iwamoto Mitsugu. Card-based cryptographic protocols for private set intersection. In *ISITA 2022*. IEEE, 2022.

[6] Crépeau Claude and Kilian Joe. Discreet solitary games. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Proceedings*, pages 319–330, 1993.

[7] Bert den Boer. More efficient match-making and satisfiability: *The Five Card Trick.* In *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, pages 208–217, 1989.

[8] Rikuo Haga, Kodai Toyoda, Yuto Shinoda, Daiki Miyahara, Kazumasa Shinagawa, Yuichi Hayashi, and Takaaki Mizuki. Card-based secure sorting protocol. In *Advances in Information and Computer Security*, pages 224–240, Cham, 2022. Springer International Publishing.

[9] Rie Ishikawa, Eikoh Chida, and Takaaki Mizuki. Efficient card-based protocols for generating a hidden random permutation without fixed points. In *Unconventional Computation and Natural Computation - 14th International Conference, UCNC, Proceedings*, pages 215–226, 2015.

[10] Julia Kastner, Alexander Koch, Stefan Walzer, Daiki Miyahara, Yuichi Hayashi, Takaaki Mizuki, and Hideaki Sone. The minimum number of cards in practical card-based protocols. In *Advances in Cryptology – ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Proceedings, Part III*, page 126–155. Springer-Verlag, 2017.

[11] Alexander Koch, Stefan Walzer, and Kevin Härtel. Card-based cryptographic protocols using a minimal number of cards. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part I*, pages 783–807, 2015.

[12] Daiki Miyahara, Yuichi Hayashi, Takaaki Mizuki, and Hideaki Sone. Practical card-based implementations of yao's millionaire protocol. *Theoretical Computer Science*, 803:207–221, 2020.

[13] Takaaki Mizuki, Isaac Kobina Asiedu, and Hideaki Sone. Voting with a logarithmic number of cards. In *Unconventional Computation and Natural Computation - 12th International Conference, UCNC 2013, Milan, Italy, July 1-5, 2013. Proceedings*, pages 162–173, 2013.

[14] Takaaki Mizuki and Hiroki Shizuya. A formalization of card-based cryptographic protocols via abstract machine. *Int. J. Inf. Sec.*, 13(1):15–23, 2014.

[15] Takaaki Mizuki and Hideaki Sone. Six-card secure AND and four-card secure XOR. In *Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 20-23, 2009. Proceedings*, pages 358–369, 2009.

[16] Takaaki Mizuki, Fumishige Uchiike, and Hideaki Sone. Securely computing XOR with 10 cards. *The Australasian Journal of Combinatorics*, 36:279–293, 2006.

[17] Takeshi Nakai, Yuto Misawa, Yuuki Tokushige, Mitsugu Iwamoto, and Kazuo Ohta. How to solve millionaires' problem with two kinds of cards. *New Generation Computing*, 39:73–96, 01 2021.

[18] Takeshi Nakai, Satoshi Shirouchi, Yuuki Tokushige, Mitsugu Iwamoto, and Kazuo Ohta. Secure computation for threshold functions with physical cards: Power of private permutations. *New Generation Computing*, 40(1):95–113, 2022.

[19] Valtteri Niemi and Ari Renvall. Secure multiparty computations without computers. *Theor. Comput. Sci.*, 191(1-2):173–183, 1998.

[20] Takuya Nishida, Yuichi Hayashi, Takaaki Mizuki, and Hideaki Sone. Card-based protocols for any boolean function. In *Theory and Applications of Models of Computation - 12th Annual Conference, TAMC, Proceedings*, pages 110–121, 2015.

[21] Akihiro Nishimura, Yuichi Hayashi, Takaaki Mizuki, and Hideaki Sone. Pile-shifting scramble for card-based protocols. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E101.A(9):1494–1502, 2018.

[22] H. Ono and 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)*, pages 23–28, 2018.

[23] M. S. Patterson. Improved sorting networks with o(log n)depth. *Algorithmica*, 5:75–92, 1990.

[24] Anton Stiglic. Computations with a deck of cards. *Theor. Comput. Sci.*, 259(1-2):671–678, 2001.

[25] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.