Card-Based Protocols Imply PSM Protocols

Kazumasa Shinagawa
Ibaraki University, Ibaraki, Japan
National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

Koji Nuida
Institute of Mathematics for Industry (IMI), Kyushu University, Fukuoka, Japan
National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

Abstract

Card-based cryptography is the art of cryptography using a deck of physical cards. While this area is known as a research area of recreational cryptography and is recently paid attention in educational purposes, there is no systematic study of the relationship between card-based cryptography and the other "conventional" cryptography. This paper establishes the first generic conversion from card-based protocols to private simultaneous messages (PSM) protocols, a special kind of secure multiparty computation. Our compiler supports "simple" card-based protocols, which is a natural subclass of finite-runtime protocols. The communication complexity of the resulting PSM protocol depends on how many cards are opened in total in all possible branches of the original card-based protocol. This result shows theoretical importance of such "opening complexity" of card-based protocols, which had not been focused in this area. As a consequence, lower bounds for PSM protocols imply those for simple card-based protocols. In particular, if there exists no PSM protocol with subexponential communication complexity for a function f, then there exists no simple card-based protocol with subexponential opening complexity for the same f.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Card-based cryptography, private simultaneous messages

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.72

Funding This work was supported by Institute of Mathematics for Industry, Joint Usage/Research Center in Kyushu University: FY2022 Short-term Visiting Researcher "On Minimal Construction of Private Simultaneous Messages Protocols" (2022a006), FY2023 Short-term Visiting Researcher "On the Relationship between Physical and Non-physical Secure Computation Protocols" (2023a009), and FY2024 Short-term Visiting Researcher "Private Simultaneous Messages and Card-Based Cryptography" (2024a015).
Kazumasa Shinagawa: This work was supported in part by JSPS KAKENHI Grant Numbers 21K17702 and 23H00479, and JST CREST Grant Number MJCR22M1.
Koji Nuida: This work was supported in part by JSPS KAKENHI Grant Number JP22K11906.

1 Introduction

1.1 Background

Secure computation allows a set of parties, each with a secret input, to compute an output value of a function of their inputs without revealing any information on the inputs beyond the output value. Although secure computation is typically assumed to be implemented on electronic devices, there is a line of research to implement secure computation using a deck of physical cards, which is called *card-based cryptography* [8, 11, 21].

In card-based cryptography, an important research topic is to determine the minimum number of cards required for secure compution of a function. For the logical AND function, for example, upper and lower bounds on the number of cards have been studied by constructing a protocol and by proving impossibility for a certain number of cards [16–19], respectively.

Other than AND protocols, lower bounds are so far known only for COPY protocols [16, 21] and protocols for generating a random permutation without fixed points [14], and as far as we know, no lower bounds for general functions have been explored.

On the other hand, in "conventional" (or "electronic") cryptography, there have been derived some lower bounds for general functions. For example, lower bounds on the communication complexity have been studied for secure multiparty computation (MPC) [10], private simultaneous messages (PSM) [2, 4, 5, 12], conditional disclosure of secrets (CDS) [1, 13], and secret sharing (SS) [9].

If there is some technical relationship between card-based and conventional cryptography, lower bounds in card-based cryptography might be obtained from lower bounds in conventional cryptography. Unfortunately, however, there has been no systematic study of the relationship between card-based and conventional cryptography so far.

In this line of research, den Boer [11], in the historically first paper on card-based protocols, proposed a conventional cryptographic protocol based on an idea from a card-based protocol. However, there is no other research on constructing conventional cryptographic protocols from card-based protocols; this line of research has not progressed at all for more than 30 years, long enough for many cryptographers to believe that there is no more technical relationship between card-based and conventional cryptography.

1.2 Our Contribution

In this paper, we study a technical relationship between card-based and conventional cryptography. In particular, we develop a generic conversion from card-based to PSM protocols.

The model of PSM was initially introduced by Feige, Kilian, and Naor [12] and later generalized by Ishai and Kushilevitz [15]. A PSM protocol is a kind of secure computation protocols with one-round and unidirectional communication from n input players, having input-independent pre-shared common randomness, to an output player called a referee.

Our compiler supports a class of finite-runtime card-based protocols, which we name *simple* protocols (see Section 4.1): Roughly speaking, a finite-runtime protocol is said to be simple if, throughout an execution, all cards in each step are face-down except when opening cards. When the total number of possibly opened cards in a simple protocol is t, the resulting PSM protocol has communication complexity nt. This parameter t, which we name the *opening complexity* of the protocol, is a new complexity measure for card-based protocols not well investigated in the previous studies. It is an interesting research direction to design card-based protocols with small opening complexity.

For a special case, we consider *single-shuffle full-open (SSFO)* protocols (see Section 3). Since SSFO protocols are simple and the opening complexity of an SSFO protocol is exactly the same as the number of cards d used in the protocol, the communication complexity of the resulting PSM protocol is nd.

The main contribution of this study is deepening our understanding of card-based cryptography by connecting to the world of conventional cryptography. We believe that our work enhances a new direction for future research on card-based cryptography: An interesting research topic is to determine a class of functions for which our conversion yields efficient PSM protocols; Another interesting topic is to develop deeper connections between card-based protocols and PSM protocols or other kinds of conventional cryptographic protocols.

As a direct consequence of our conversion, lower bounds on the opening complexity t in card-based protocols will be derived from lower bounds on the communication complexity of PSM protocols (Corollary 5). If there exists no PSM protocol for any function with subexponential communication complexity, which is an unproven but plausible statement

and is stated as Conjecture 6, then there exists no simple protocol computing any function with subexponential opening complexity (Corollary 7). To prove an unconditional non-trivial lower bound for card-based protocols, we need a *super-quadratic* lower bound for PSM protocols, i.e., $\omega(kn^2)$ for a function $f\colon (\{0,1\}^k)^n \to \{0,1\}$. Unfortunately, as far as we know, the state-of-the-art lower bound is $\Omega(kn^2/\log(nk))$ by Ball and Randolph [5] and no super-quadratic lower bound has been proved yet. We believe that our result provides a new motivation for obtaining such a lower bound.

Another application of our conversion is to provide a new method to construct PSM protocols. As a concrete example, we construct a card-based protocol for an *indirect storage access* (ISA, for short) function [23] and then convert it to a PSM protocol with communication complexity $O(KL^2 + K^2 L \log L)$, where K and L are parameters for the input length. Our protocol is more efficient than the existing constructions based on branching programs (BP, for short) and gate evaluation secret sharing (GESS, for short), both of which, to the best of our knowledge, are the only existing constructions for efficient PSM protocols effective for the ISA function.

2 Preliminaries

For an integer $n \geq 2$, we denote $[n] := \{1, 2, \ldots, n\}$. For an integer $k \geq 1$, we denote the k-th symmetric group by S_k. For a k-bit string $s = (s_1, s_2, \ldots, s_k) \in \{0,1\}^k$ and a permutation $\pi \in S_k$, a permuted string of s by π, denoted by $\pi(s)$, is defined by $\pi(s) := (s_{\pi^{-1}(1)}, s_{\pi^{-1}(2)}, \ldots, s_{\pi^{-1}(k)})$. For example, for a string $s = 111000 \in \{0,1\}^6$ and a cyclic permutation $\pi = (1\ 2\ 3\ 4\ 5\ 6) \in S_6$, we have $\pi(s) = 011100$. We denote by X^* the set of finite sequences (of various lengths) from a set X.

2.1 PSM Protocols

Feige, Kilian, and Naor [12] introduced a minimal model of secure two-party computation protocols. Ishai and Kushilevitz [15] generalized the model to the multiparty case, which is called *private simultaneous messages (PSM)*.

Suppose that n players $P_1, P_2, \ldots, P_n$ hold $x_1, x_2, \ldots, x_n \in X$, respectively. They want to tell the output value $f(x_1, x_2, \ldots, x_n)$ of a given function $f\colon X^n \to Y$ only to the *referee*, who is not one of the n players, without revealing any information on the inputs beyond the output value. The n players are assumed to share a *common randomness* and each of them is allowed to send a single message to the referee. The model of PSM is summarized in Figure 1.

Now we define the syntax and requirements for PSM protocols formally. Let X, Y be finite sets and $f\colon X^n \to Y$ a function. Let $R, M_1, M_2, \ldots, M_n$ be finite sets and $M := M_1 \times M_2 \times \cdots \times M_n$. A *private simultaneous messages (PSM) protocol* for f is an $(n+2)$-tuple $(\mathcal{R}, \mathsf{Enc}_1, \mathsf{Enc}_2, \ldots, \mathsf{Enc}_n, \mathsf{Dec})$, where $\mathcal{R}$ is a probability distribution over R, $\mathsf{Enc}_i\colon X \times R \to M_i$ is the i-th *encoding function* $(1 \leq i \leq n)$, and $\mathsf{Dec} : M \to Y$ is the *decoding function*. The protocol proceeds as follows: First, a common randomness $r \leftarrow \mathcal{R}$ is chosen and distributed to the n players $P_1, P_2, \ldots, P_n$. Then each player P_i holding (x_i, r), where $x_i \in X$ is an input of P_i, computes a message $m_i = \mathsf{Enc}_i(x_i, r)$ and sends it to the referee. Finally, on receiving $m_1, m_2, \ldots, m_n$, the referee determines an output value $y = \mathsf{Dec}(m_1, m_2, \ldots, m_n)$.

The protocol is said to be *correct* if we have $\mathsf{Dec}(\mathsf{Enc}_1(x_1, r), \ldots, \mathsf{Enc}_n(x_n, r)) = f(\vec{x})$ for any $\vec{x} = (x_1, \ldots, x_n) \in X^n$ and any $r \in R$ such that $\Pr[r \leftarrow \mathcal{R}] > 0$. The protocol is said to be *secure* if there exists an algorithm Sim called a simulator such that for any $\vec{x} \in X^n$,

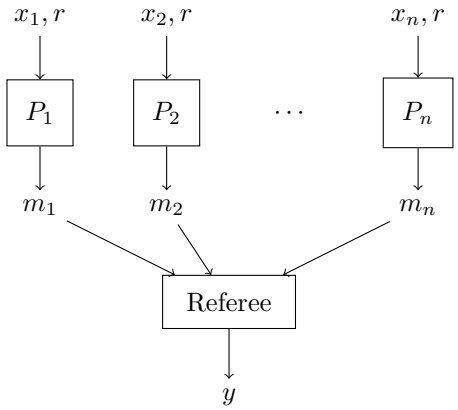


Figure 1 The model of PSM protocols.

the distribution of $\mathsf{Sim}(f(\vec{x}))$ equals the distribution of $(\mathsf{Enc}_1(x_1, \mathcal{R}), \ldots, \mathsf{Enc}_n(x_n, \mathcal{R}))$. The *communication complexity* of the protocol is defined by $\sum_{i=1}^{n} \log_2 |M_i|$. The *randomness complexity* of the protocol is defined by the Shannon entropy $H(\mathcal{R})$ of $\mathcal{R}$.

2.2 Card-Based Protocols

Mizuki and Shizuya [21] proposed a computational model for card-based protocols, referred to as the *Mizuki–Shizuya model*. We review the Mizuki–Shizuya model below. (We also follow the well-summarized description of [19].)

A *deck* $\mathcal{D}$ is a finite and non-empty multiset of symbols. For a symbol $c \in \mathcal{D}$, $\frac{c}{?}$ and $\frac{?}{c}$ denote a *face-up card* and *face-down card* with symbol c, respectively, where '?' is a special backside symbol that is not contained in $\mathcal{D}$. For a card (i.e., face-up card or face-down card) α, $\mathsf{top}(\alpha)$ is defined by $\mathsf{top}(\frac{c}{?}) := c$ and $\mathsf{top}(\frac{?}{c}) := ?$, and the *flipped card* of α is defined to be $\frac{?}{c}$ if $\alpha = \frac{c}{?}$ and to be $\frac{c}{?}$ if $\alpha = \frac{?}{c}$.

In this paper, we mainly deal with a *two-colored deck* $\mathcal{D}$, which consists of two types of symbols ♣ and ♡. The face-up cards $\frac{\clubsuit}{?}$ and $\frac{\heartsuit}{?}$ are depicted by $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, respectively, and a face-down card is depicted by $\boxed{?}$.

A *sequence of cards* from $\mathcal{D}$ is obtained by permuting $\mathcal{D}$ and, for each symbol $c \in \mathcal{D}$, choosing a face-up card $\frac{c}{?}$ or a face-down card $\frac{?}{c}$. For example, when $\mathcal{D} = [\clubsuit, \heartsuit, \heartsuit]$ (where the square brackets represent a multiset), $\Gamma = (\frac{\heartsuit}{?}, \frac{\clubsuit}{?}, \frac{?}{\heartsuit})$ is a sequence of cards from $\mathcal{D}$. For a sequence of cards $\Gamma = (\alpha_1, \ldots, \alpha_d)$, $\mathsf{vis}(\Gamma) := (\mathsf{top}(\alpha_1), \ldots, \mathsf{top}(\alpha_d))$ is called the *visible sequence* of Γ. We denote the set of all visible sequences of card sequences from $\mathcal{D}$ by $\mathsf{Vis}^{\mathcal{D}}$.

For a sequence of cards $\Gamma = (\alpha_1, \ldots, \alpha_d)$, we can apply the following three types of *actions*:

- (perm, π) for $\pi \in S_d$ is an action that converts Γ to $\pi(\Gamma) := (\alpha_{\pi^{-1}(1)}, \alpha_{\pi^{-1}(2)}, \ldots, \alpha_{\pi^{-1}(d)})$.
- $(\mathsf{shuf}, \Pi, \mathcal{F})$ for a subset $\Pi \subseteq S_d$ and a probability distribution $\mathcal{F}$ over Π is an action that converts Γ to $\pi(\Gamma)$, where $\pi \in \Pi$ is drawn from $\mathcal{F}$. Here, no player should know which permutation is chosen by the shuffle when it is applied to a sequence of face-down cards.
- (turn, T) for a subset $T \subseteq [d]$ is an operation that converts Γ to $(\beta_1, \ldots, \beta_d)$ where β_i is the flipped card of α_i if $i \in T$ and $\beta_i = \alpha_i$ otherwise.

The set of actions for card sequences from $\mathcal{D}$ is denoted by $\mathsf{Action}^{\mathcal{D}}$.

A *Mizuki–Shizuya (MS) protocol* is defined by a tuple $(\mathcal{D}, U, Q, A)$, where $\mathcal{D}$ is a deck, U is a set of input sequences (of cards), Q is a set of states including the *initial state* q_0 and the set of *final states* $Q_{\mathsf{fin}} \subseteq Q$, and $A\colon (Q \setminus Q_{\mathsf{fin}}) \times \mathsf{Vis}^{\mathcal{D}} \to Q \times \mathsf{Action}^{\mathcal{D}}$ is a

partial function called an *action function*. An MS protocol starts with an input sequence $\Gamma_0 \in U$ and the initial state q_0. For the current sequence Γ_i and the current state q, if $A(\mathsf{vis}(\Gamma_i), q) = (q', \mathsf{action})$, the MS protocol applies action to Γ_i and updates the state to q'. The MS protocol terminates when entering a final state $q \in Q_{\mathsf{fin}}$. The list of all sequences $(\Gamma_0, \Gamma_1, \ldots, \Gamma_k)$ for a protocol execution is called the *sequence trace* of the protocol execution, and $(\mathsf{vis}(\Gamma_0), \mathsf{vis}(\Gamma_1), \ldots, \mathsf{vis}(\Gamma_k))$ is called the *visible sequence trace* of the protocol execution.

We note that an MS protocol is defined as a mechanism for converting an input sequence into an output sequence, and does not explicitly address the function to be computed. To associate a function $f\colon X^n \to Y$ with an MS protocol $(\mathcal{D}, U, Q, A)$, we need to give a map $\mathsf{in}\colon X^n \to U$ called an *input function* and a map $\mathsf{out}\colon (\mathsf{Vis}^{\mathcal{D}})^* \to Y$ called an *output function*. For an input $\vec{x} \in X^n$, the input sequence of the protocol is set to $\mathsf{in}(\vec{x}) \in U$, and the output of the protocol is defined by $\mathsf{out}(v_0, v_1, \ldots, v_k) \in Y$ when the visible sequence trace is $(v_0, v_1, \ldots, v_k) \in (\mathsf{Vis}^{\mathcal{D}})^{k+1}$.

Let in_i be the function representing the i-th card of the input sequence, i.e., $\mathsf{in}(\vec{x}) := (\mathsf{in}_1(\vec{x}), \ldots, \mathsf{in}_d(\vec{x}))$. We naturally assume that the output of each function in_i depends only on at most one input, say $x_j \in X$ (including the case that the output of in_i is constant), referred to as the *locality* of in.

A *protocol* is defined by a tuple $(\mathcal{D}, U, Q, A, \mathsf{in}, \mathsf{out})$, where $(\mathcal{D}, U, Q, A)$ is an MS protocol, and in and out are input and output functions, respectively. A protocol for a function f is said to be *correct* if for all inputs $\vec{x} \in X^n$, the protocol outputs $f(\vec{x})$ whenever it terminates. A protocol for f is said to be *secure* if there exists an algorithm Sim called a simulator such that for any $\vec{x} \in X^n$, the distribution of $\mathsf{Sim}(f(\vec{x}))$ equals the distribution of the visible sequence trace of the protocol execution with input $\vec{x}$. A protocol is said to be *finite-runtime* if it terminates within a fixed number of steps. A protocol is said to be *Las Vegas* if its runtime is finite in expectation.

We give two remarks on the above definition. A pair of face-down cards with different symbols is called a *commitment* to a bit, via the encoding rule $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$. Our definition of in can deal with input commitments: For a sequence of n commitments to $x_1, x_2, \ldots, x_n \in \{0,1\}$, in our definition, the input function in can be represented by $\mathsf{in}(\vec{x}) = \left(\frac{?}{x_1}, \frac{?}{\overline{x_1}}, \frac{?}{x_2}, \frac{?}{\overline{x_2}}, \ldots, \frac{?}{x_n}, \frac{?}{\overline{x_n}}\right)$ using the encoding rule $\clubsuit = 0$ and $\heartsuit = 1$, where $\overline{x_i}$ denotes the negation of a bit x_i. Our definition also captures other encoding rules as long as they satisfy the locality.

There are two types of protocols in the area of card-based cryptography: a *committed-format protocol* and a *non-committed-format protocol*. A protocol of the former type produces a sequence of commitments as output, while a protocol of the latter type publicly outputs the value $f(x_1, x_2, \ldots, x_n)$. Our definition of the correctness and security supposed non-committed-format protocols, but committed-format protocols can be obviously converted to non-committed-format ones by just opening the output commitments and hence our proposed conversion method is also applicable to them.

For a protocol $\mathcal{P} = (\mathcal{D}, U, Q, A, \mathsf{in}, \mathsf{out})$ (or an MS protocol $\mathcal{P} = (\mathcal{D}, U, Q, A)$), the *protocol diagram* of $\mathcal{P}$ is defined as the directed edge-labeled graph as follows:

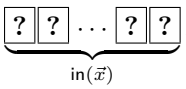
- The set of vertices is Q.
- A directed edge labeled by $\mathsf{action} \in \mathsf{Action}^{\mathcal{D}}$ from q to q' exists if and only if there exists a visible sequence $v \in \mathsf{Vis}^{\mathcal{D}}$ such that $A(q, v) = (q', \mathsf{action})$.

See Figure 2 in Section 4.1 for an example of the protocol diagram of the four-card trick [20]. Note that a protocol diagram can be viewed as a reduced version of the Koch–Walzer–Härtel diagram [19]. We remark that any finite-runtime protocol can be easily converted to a (functionally equivalent) protocol whose protocol diagram forms a finite tree.

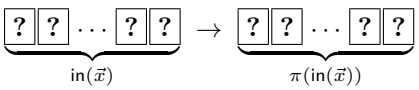
3 Our Conversion for Special Case

In order to explain the essential idea of our proposed conversion before a general description given in Section 4 below, in this section, we first describe our conversion for a special type of card-based protocols which we call single-shuffle full-open protocols. A card-based protocol is said to be *single-shuffle full-open (SSFO)* if it proceeds as follows:

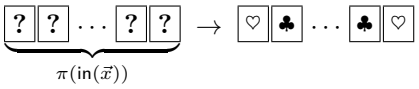
1. The input sequence $\mathsf{in}(\vec{x})$ is given:



2. Apply a shuffle $(\mathsf{shuf}, \Pi, \mathcal{F})$ to the sequence as follows:



   where $\pi \in \Pi$ is chosen according to $\mathcal{F}$.
3. Turn over all cards and determine the output value:

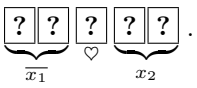


In Section 3.1, we demonstrate our conversion for an SSFO protocol called the five-card trick. In Section 3.2, we give our conversion for any SSFO protocol.

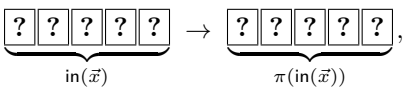
3.1 PSM Protocol from Five-Card Trick

The five-card trick is a five-card AND protocol proposed by den Boer [11]. First, we recall the protocol description of the five-card trick.

1. The input sequence $\mathsf{in}(\vec{x})$ is given as follows, where $\vec{x} = (x_1, x_2) \in \{0,1\}^2$:



2. Apply a shuffle $(\mathsf{shuf}, \langle(1\ 2\ 3\ 4\ 5)\rangle)$ to the sequence as follows:



   where $\pi = (1\ 2\ 3\ 4\ 5)^r$ for a uniformly random $r \in \{0,1,2,3,4\}$.
3. Turn over all cards. Output 0 if they are ♡♣♡♣♡ up to cyclic shifts, and 1 if they are ♡♡♡♣♣ up to cyclic shifts.

Now we convert the above protocol to a PSM protocol. Let Alice and Bob be the players holding $x_1, x_2 \in \{0,1\}$, respectively, and Charlie the referee.

First, following the encoding rule ♣ $= 0$ and ♡ $= 1$, the input sequence $\mathsf{in}(x_1, x_2)$, which is the sequence of symbols for the cards, can be represented by

$$\mathsf{in}(x_1, x_2) := (\overline{x_1}, x_1, 1, x_2, \overline{x_2}) \in \{0,1\}^5.$$

Although neither Alice nor Bob alone can compute the whole of $\mathsf{in}(x_1, x_2)$, Alice can compute

$$s_1(x_1) := (\overline{x_1}, x_1, 1, 0, 0) \in \{0,1\}^5,$$

and Bob can compute

$$s_2(x_2) := (0, 0, 0, x_2, \overline{x_2}) \in \{0,1\}^5.$$

We note that, by taking the component-wise XOR, we have

$$s_1(x_1) \oplus s_2(x_2) = (\overline{x_1}, x_1, 1, x_2, \overline{x_2}) = \mathsf{in}(x_1, x_2).$$

In other words, thanks to the locality of in, Alice and Bob can jointly compute the input sequence $\mathsf{in}(x_1, x_2)$ in a distributed manner.

Next, we notice that Alice and Bob can apply a shuffle to the sequence *in their heads* by sharing a randomness of the shuffle as common randomness. In particular, by sharing a random cyclic permutation $\pi \in \langle(1\ 2\ 3\ 4\ 5)\rangle$, Alice and Bob can compute $\pi(s_1(x_1))$ and $\pi(s_2(x_2))$, respectively. We emphasize that

$$\pi(s_1(x_1)) \oplus \pi(s_2(x_2)) = \pi(s_1(x_1) \oplus s_2(x_2)) = \pi(\mathsf{in}(x_1, x_2))$$

by the property of permutations.

Finally, to hide their inputs, Alice and Bob compute $m_1 := \pi(s_1(x_1)) \oplus r$ and $m_2 := \pi(s_2(x_2)) \oplus r$, respectively, where $r \in \{0,1\}^5$ is also shared as common randomness, and send m_1 and m_2 to Charlie. Thanks to the masking by the random r, each of the messages m_1 and m_2 alone does not leak any information on x_1 and x_2, respectively. Unmasking by Charlie requires computation of XOR for the two messages to cancel the term r:

$$\begin{aligned} m := m_1 \oplus m_2 &= (\pi(s_1(x_1)) \oplus r) \oplus (\pi(s_2(x_2)) \oplus r) \\ &= \pi(s_1(x_1)) \oplus \pi(s_2(x_2)) = \pi(\mathsf{in}(x_1, x_2)), \end{aligned}$$

which only tells Charlie the result $\pi(\mathsf{in}(x_1, x_2))$ of the XOR and does not tell individual sequences $\pi(s_1(x_1))$ nor $\pi(s_2(x_2))$. Then Charlie outputs 0 if m is equal to 10101 up to cyclic shifts, and 1 if m is equal to 11100 up to cyclic shifts.

The resulting PSM protocol is summarized as follows.

1. The common randomness consists of a cyclic permutation $\pi \in \langle(1\ 2\ 3\ 4\ 5)\rangle$ and a 5-bit string $r \in \{0,1\}^5$, both are chosen uniformly at random.
2. Alice (resp. Bob) computes $m_1 = \pi(s_1(x_1)) \oplus r$ (resp. $m_2 = \pi(s_2(x_2)) \oplus r$) and sends it to Charlie.
3. On receiving m_1 and m_2, Charlie determines an output value y as follows:

$$y := \begin{cases} 1 & \text{if } m_1 \oplus m_2 \in \{00111, 10011, 11001, 11100, 01110\}; \\ 0 & \text{if } m_1 \oplus m_2 \in \{01011, 10101, 11010, 01101, 10110\}. \end{cases}$$

The communication complexity of the protocol is 10 bits. The randomness complexity of the protocol is $5 + \log_2 5$ bits.

3.2 PSM Protocol from Any Single-Shuffle Full-Open Protocol

Let $\mathcal{P} = (\mathcal{D}, U, Q, A, \mathsf{in}, \mathsf{out})$ be any SSFO protocol whose shuffle operation is $(\mathsf{shuf}, \Pi, \mathcal{F})$. By a similar observation to Section 3.1 thanks to the locality of the function in, we can define a function $s_i \colon X \to \{0,1\}^d$ $(i \in [n])$ such that $s_1(x_1) \oplus \cdots \oplus s_n(x_n) = \mathsf{in}(\vec{x})$ for any $\vec{x} \in X^n$.

The resulting PSM protocol $\widehat{\mathcal{P}}$ is given as follows, where $H(\mathcal{F})$ denotes the Shannon entropy of the distribution $\mathcal{F}$.

The PSM protocol $\widehat{\mathcal{P}}$ from any SSFO protocol $\mathcal{P}$

Input: $\vec{x} = (x_i)_{i\in[n]} \in X^n$

Common Randomness: $(\pi, (r_i)_{i\in[n]})$ is generated as follows:

- Choose a permutation $\pi \in \Pi$ according to $\mathcal{F}$.
- Choose $r_1, r_2, \ldots, r_{n-1} \in \{0,1\}^d$ uniformly at random, and set $r_n := r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1}$.

Encoding Function: Output $m_i := \pi(s_i(x_i)) \oplus r_i$.

Decoding Function: Compute $m := m_1 \oplus m_2 \oplus \cdots \oplus m_n$. The output value y is equal to the output of the protocol $\mathcal{P}$ when the opened symbol is m.

Communication Complexity: nd

Randomness Complexity: $(n-1)d + H(\mathcal{F})$

▶ **Theorem 1.** *Let $\mathcal{P}$ be an SSFO protocol with correctness and security for $f\colon X^n \to Y$ using d cards. Then the above PSM protocol $\widehat{\mathcal{P}}$ is correct and secure for f with communication complexity nd.*

Proof. From the following computation, we have:

$$\begin{aligned} m &= m_1 \oplus m_2 \oplus \cdots \oplus m_n \\ &= (\pi(s_1(x_1)) \oplus r_1) \oplus (\pi(s_2(x_2)) \oplus r_2) \oplus \cdots \oplus (\pi(s_n(x_n)) \oplus r_n) \\ &= \pi(s_1(x_1)) \oplus \pi(s_2(x_2)) \oplus \cdots \oplus \pi(s_n(x_n)) \oplus r_1 \oplus r_2 \oplus \cdots \oplus r_n. \end{aligned}$$

By the choice of r_n in $\widehat{\mathcal{P}}$, we have $r_1 \oplus r_2 \oplus \cdots \oplus r_n = 0$. Hence we have

$$\begin{aligned} m &= \pi(s_1(x_1)) \oplus \pi(s_2(x_2)) \oplus \cdots \oplus \pi(s_n(x_n)) \\ &= \pi(s_1(x_1) \oplus s_2(x_2) \oplus \cdots \oplus s_n(x_n)) = \pi(\mathsf{in}(\vec{x})), \end{aligned}$$

which is exactly equal to the opened symbols of $\mathcal{P}$. Thus, the correctness of $\widehat{\mathcal{P}}$ follows from the correctness of $\mathcal{P}$.

We prove the security of $\widehat{\mathcal{P}}$ by constructing a simulator Sim for $\widehat{\mathcal{P}}$. Given an output value $y \in Y$, Sim first invokes a simulator for $\mathcal{P}$ on input y, which outputs a visible sequence trace $(?^d, v) \in (\mathsf{Vis}^{\mathcal{D}})^*$ of a protocol execution. Here, $v \in \{0,1\}^d$ represents the opened values at the end of the protocol $\mathcal{P}$, therefore v has the same conditional distribution as m in the protocol $\widehat{\mathcal{P}}$ conditioned on the output value y. Then Sim chooses $r'_1, r'_2, \ldots, r'_{n-1} \in \{0,1\}^d$ uniformly at random, and outputs $(r'_1, r'_2, \ldots, r'_{n-1}, r'_1 \oplus \cdots \oplus r'_{n-1} \oplus v)$. Now the XOR of the n components is v, which (as mentioned above) has the same conditional distribution as m. Moreover, for each of the first $n-1$ component, say the i-th, the message $m_i = \pi(s_i(x_i)) \oplus r_i$ is uniformly random thanks to the random choice of r_i. Hence the distribution of $\mathsf{Sim}(y)$ equals the distribution of the messages, therefore $\widehat{\mathcal{P}}$ is secure. ◀

4 Our Conversion for General Case

In this section, we give our conversion for *simple* protocols. In Section 4.1, we define the class of simple protocols. In Section 4.2, we give an overview of our conversion. In Section 4.3, we describe our conversion for simple protocols.

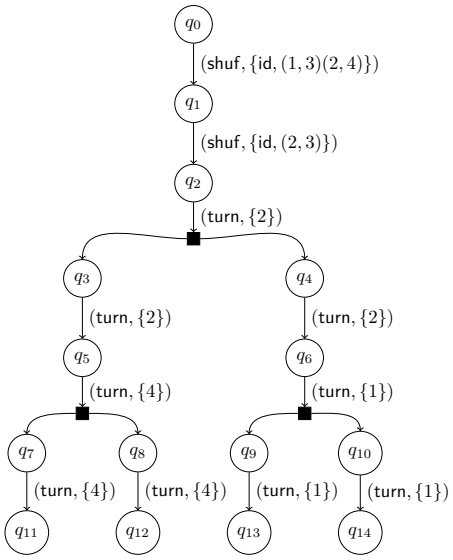


Figure 2 The protocol diagram of the four-card trick.

4.1 Simple Protocols

▶ **Definition 2.** *A card-based protocol* $\mathcal{P} = (\mathcal{D}, U, Q, A, \mathsf{in}, \mathsf{out})$ *is said to be* simple *if* $\mathcal{P}$ *satisfies the following conditions:*

Finite-Tree: *The protocol diagram of* $\mathcal{P}$ *forms a finite tree.*

Face-Down: *Any input sequence in* U *consists of face-down cards.*

Mono-Opening: *Every turn operation* (turn, T) *in* $\mathcal{P}$ *satisfies* $|T| = 1$. *(Intuitively speaking, every turn operation flips a single card.)*

Instant-Turn: *Whenever a turn operation* (turn, T) *was applied to a sequence of all face-down cards at the last step, the next operation is the same turn operation* (turn, T). *(Intuitively speaking, if a face-down card is opened, then it is immediately faced down.)*

Here we note that, from the viewpoint of feasibility, focusing only on the simple card-based protocols does not decrease the generality of our conversion, because any finite-runtime protocol $\mathcal{P}$ can be converted to a simple protocol (see Remark 3 below). We remark that almost all existing finite-runtime card-based protocols are already simple or can be trivially converted to simple (e.g., when some multiple cards are opened simultaneously, decomposing this step into a series of opening and closing of each single card in order to satisfy the mono-opening property). See Figure 2 for an example of the protocol tree of the four-card trick [20].

Let $\mathcal{P} = (\mathcal{D}, U, Q, A, \mathsf{in}, \mathsf{out})$ be a simple protocol. Then any turn operation in $\mathcal{P}$ is classified into the following two types:

- The operation opens a single card from the whole sequence of face-down cards; referred to as an *open operation*. In this case, since we deal with two-colored decks, there are at most two subsequent states after this operation. We call this operation a *branch operation* if there are two subsequent states, and a *non-branch operation* otherwise.
- The operation faces down the card that was opened at the previous step; referred to as a *close operation*.

Now we observe that for each non-final state $q \in Q \setminus Q_{\mathsf{fin}}$, the operation performed at the current step is uniquely determined from the state q. Indeed, when an open operation was performed at the previous step, the operation at the current step must be a close operation

for the unique opened card due to the instant-turn property. On the other hand, for the other case, the current visible sequence is the trivial visible sequence $v = (?, ?, \ldots, ?)$, therefore the operation $A(q, v)$ in fact depends solely on q. This yields the following partitions for the set of non-final states:

$$\begin{aligned} Q \setminus Q_{\mathsf{fin}} &= Q_{\mathsf{turn}} \sqcup Q_{\mathsf{perm}} \sqcup Q_{\mathsf{shuf}} \\ &= (Q^{+}_{\mathsf{turn}} \sqcup Q^{-}_{\mathsf{turn}}) \sqcup Q_{\mathsf{perm}} \sqcup Q_{\mathsf{shuf}} \\ &= ((Q_{\mathsf{branch}} \sqcup Q_{\mathsf{nonbranch}}) \sqcup Q^{-}_{\mathsf{turn}}) \sqcup Q_{\mathsf{perm}} \sqcup Q_{\mathsf{shuf}}, \end{aligned}$$

where Q_{turn}, Q^{+}_{turn}, Q^{-}_{turn}, Q_{branch}, $Q_{\mathsf{nonbranch}}$, Q_{perm}, and Q_{shuf} are the sets of all non-final states for which the next operation is a turn operation turn, an open operation, a close operation (hence $Q_{\mathsf{turn}} = Q^{+}_{\mathsf{turn}} \sqcup Q^{-}_{\mathsf{turn}}$), a branch operation, a non-branch operation (hence $Q^{+}_{\mathsf{turn}} = Q_{\mathsf{branch}} \sqcup Q_{\mathsf{nonbranch}}$), a permutation operation perm, and a shuffle operation shuf, respectively.

The *opening complexity* of $\mathcal{P}$ is defined by $|Q^{+}_{\mathsf{turn}}|$. Note that this notion can be defined similarly for any finite-runtime protocol that is not necessarily simple.

▶ Remark 3. We note that any finite-runtime protocol can be converted into a simple protocol without affecting its correctness and security. As already stated in Section 2.2, finite-tree can be easily obtained. Mono-opening can be obtained by replacing each $(\mathsf{turn}, \{i_1, \ldots, i_k\})$ with $(\mathsf{turn}, \{i_1\}), \ldots, (\mathsf{turn}, \{i_k\})$. Instant-turn can be obtained by appending close operations for every open operations. The most non-trivial part is to obtain face-down. Note that if a protocol is not face-down, then an input sequence may contain some face-up cards depending on the inputs, and some shuffle operations are applied to a sequence containing face-up cards called a *branching shuffle*. To obtain face-down, we modify the protocol as follows:

- We associate a pair of auxiliary cards (faced down in default) to each card, representing one-bit information by the order of two cards (where swapping these two cards corresponds to bit flipping). Then:
  - We change each face-up card in $\mathsf{in}(\vec{x})$ to be faced-down (satisfying instant-opening) while recording this fact to auxiliary cards. At the beginning of the protocol, we read information in auxiliary cards by opening them, and turn suitable cards to recover the original $\mathsf{in}(\vec{x})$.
  - Each step of the protocol is changed to: (1) turn down each face-up main (i.e., non-auxiliary) card while recording to auxiliary cards which main cards are to be face-up; (2a) in permutation/shuffle, main and auxiliary cards are synchronized; (2b) turn for main cards are emulated by updating information in (i.e., permuting) auxiliary cards; and (3) read information in auxiliary cards, turn up main cards suitably, and reset auxiliary cards. This avoids permutation/shuffle operations for face-up cards.

4.2 Overview of Our Conversion

Let $\mathcal{P}$ be a simple protocol for $f\colon X^n \to Y$. From the finite-runtime property, we can assume that the protocol diagram of $\mathcal{P}$ forms a tree.

If all permutations in the shuffle operations are fixed, the order of the sequence of cards at any step in an execution is determined. This is because the probability in an execution is taken only for shuffle operations. So, the basic idea of our conversion is to share the permutations chosen in shuffle operations as common randomness among the players, similar to the protocol in Section 3.

Let $c_1, c_2, \ldots, c_t \in \{0, 1\}$ be the opened symbols when a sequence of permutations $\vec{\pi}$ in the shuffle operations and the input $\vec{x} \in X^n$ are given. Similar to the protocol in Section 3, we can observe that for each c_i, at least one player can compute it if $\vec{\pi}$ is shared

as common randomness. Therefore, each player P_i can compute a message $m_i \in \{0,1\}^t$ such that $m_1 \oplus m_2 \oplus \cdots \oplus m_n = (c_1, c_2, \ldots, c_t)$. Since the output value of $\mathcal{P}$ is determined by the opened values, the referee can compute the output value from the messages and the correctness of the obtained PSM protocol $\widehat{\mathcal{P}}$ can be satisfied.

The remaining thing we need to consider is the security of $\widehat{\mathcal{P}}$. See Figure 2. This protocol has three possible opened symbols c_1, c_2, c_3 corresponding to q_2, q_4, q_8, respectively, but only two symbols are revealed in an execution as follows: If $c_1 = 0$, the left path (c_1, c_2) should be revealed; If $c_1 = 1$, the right path (c_1, c_3) should be revealed. Since any leakage of the values outside the path would compromise the security, the values outside the path must be hidden from the referee. To hide these values, some player can replace the i-th bit of the player's message with a random bit if he notices that c_i is not on the path.

The above idea is summarized as follows.

1. The common randomness consists of a sequence of permutations in all shuffle operations, an n-tuple of t-bit random numbers $(r_1, r_2, \ldots, r_n) \in (\{0,1\}^t)^n$ such that $r_1 \oplus r_2 \oplus \cdots \oplus r_n = 0^t$, and other random bits to hide the values outside the path.
2. Each player P_i computes $m_i \in \{0,1\}^t$ such that $m_1 \oplus m_2 \oplus \cdots \oplus m_n$ equals the opened values $(c_1, c_2, \ldots, c_t)$.
3. Each player P_i updates m_i to hide the values c_i outside the path.
4. Each player P_i sets $m_i \leftarrow m_i \oplus r_i$ and sends it to the referee.
5. On receiving $m_1, m_2, \ldots, m_n$, the referee obtains the opened symbols on the path from $m_1 \oplus m_2 \oplus \cdots \oplus m_n$, and determines an output value y based on simple protocol $\mathcal{P}$.

4.3 Our Conversion for Simple Protocols

Let $\mathcal{P} = (\mathcal{D}, U, Q, A, \mathsf{in}, \mathsf{out})$ be a simple protocol for $f\colon X^n \to Y$. We can assume that the protocol diagram of $\mathcal{P}$ forms a tree.

For each $q \in Q_{\mathsf{shuf}}$, let $(\mathsf{shuf}, \Pi_q, \mathcal{F}_q)$ be the shuffle operation of q. As stated in Section 4.2, given a sequence of permutations $\vec{\pi} = (\pi_q)_{q \in Q_{\mathsf{shuf}}}$ for $\pi_q \in \Pi_q$, the order of the sequence of cards in a protocol execution is determined. In particular, given a sequence of permutations $\vec{\pi}$ and an input $\vec{x} \in X^n$, the opened symbol for $q \in Q^+_{\mathsf{turn}}$ is determined.

Fix a sequence of permutations $\vec{\pi}$ and an input $\vec{x} \in X^n$. We define the *responsibility* for $q \in Q^+_{\mathsf{turn}}$ as follows. We say that the first player P_1 is responsible for q if the opened value at q is a constant bit or depends on $x_1 \in X$. We say that the i-th ($i \neq 1$) player P_i is responsible for q if the opened value at q is not a constant bit and depends on $x_i \in X$. We can observe that, for any $q \in Q^+_{\mathsf{turn}}$, exactly one player P_i is responsible for q.

Let $c_q \in \{0,1\}$ be the opened symbol at $q \in Q^+_{\mathsf{turn}}$ when $\vec{\pi}$ and $\vec{x}$ are given. We define a map $\mathsf{val}_{i,q}$ for $i \in [n]$ and $q \in Q^+_{\mathsf{turn}}$ as follows:

$$\mathsf{val}_{i,q}(\vec{\pi}, x_i) := \begin{cases} c_q & \text{if } P_i \text{ is responsible for } q; \\ 0 & \text{otherwise.} \end{cases}$$

Since exactly one player is responsible for q, we have

$$\sum_{i=1}^{n} \mathsf{val}_{i,q}(\vec{\pi}, x_i) = c_q,$$

i.e., the opened value c_q is computed by the players in a distributed manner.

For a state $q \in Q$, the *descendants* of q is recursively defined as follows: a child of q is a descendant of q, and a child of a descendant of q is a descendant of q. For a state $q \in Q$, we define $\mathsf{descen}(q)$ as follows:

$$\mathsf{descen}(q) := \{q' \in Q^+_{\mathsf{turn}} \mid q' \text{ is a descendant of } q\}.$$

For a branch state $q \in Q_{\mathsf{branch}}$, let $q_b \in Q$ ($b \in \{0,1\}$) be the child of q corresponding to the opened value b, and we define $\mathsf{descen}(q,b)$ as follows:

$$\mathsf{descen}(q,b) := \{q'_b\} \cup \mathsf{descen}(q'_b),$$

where q'_b is either $q'_b = q_b$ if $q_b \in Q^+_{\mathsf{turn}}$ or the nearest descendant of q_b in Q^+_{turn}.

The resulting PSM protocol is given as follows, where $H(\mathcal{F}_q)$ denotes the Shannon entropy of the distribution $\mathcal{F}_q$ for $q \in Q_{\mathsf{shuf}}$ and $t := |Q^+_{\mathsf{turn}}|$ denotes the opening complexity of $\mathcal{P}$.

The PSM protocol $\widehat{\mathcal{P}}$ from any simple protocol $\mathcal{P}$

Input: $\vec{x} = (x_i)_{i\in[n]} \in X^n$

Common Randomness: $(\vec{\pi}, \vec{r}, \vec{s})$ is generated as follows:

- For each $q \in Q_{\mathsf{shuf}}$ whose operation is $(\mathsf{shuf}, \Pi_q, \mathcal{F}_q)$, choose a permutation $\pi_q \in \Pi_q$ according to $\mathcal{F}_q$. Set $\vec{\pi} := (\pi_q)_{q\in Q_{\mathsf{shuf}}}$.
- Choose $r_i = (r_i[q])_{q\in Q^+_{\mathsf{turn}}} \in \{0,1\}^t$ for $1 \le i \le n-1$ uniformly at random, and set $r_n := r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1}$. Set $\vec{r} := (r_i)_{i\in[n]}$.
- For each $q \in Q_{\mathsf{branch}}$ and $q' \in \mathsf{descen}(q)$, choose $s^{(q)}_{q'} \in \{0,1\}$ uniformly at random. Set $\vec{s} := (s^{(q)}_{q'})_{q\in Q_{\mathsf{branch}}, q'\in\mathsf{descen}(q)}$.

Encoding Function: Output $m_i := (m_i[q])_{q\in Q^+_{\mathsf{turn}}} \in \{0,1\}^t$ as follows:

1. For each $q \in Q^+_{\mathsf{turn}}$, set $m_i[q] \leftarrow \mathsf{val}_{i,q}(\vec{\pi}, x_i)$.
2. For each $q \in Q_{\mathsf{branch}}$, if the player i is responsible for q, do the following:
   - Let $c_q \in \{0,1\}$ be the opened value at q.
   - For each $q' \in \mathsf{descen}(q, \overline{c_q})$, compute $m_i[q'] \leftarrow m_i[q'] \oplus s^{(q)}_{q'}$.
3. For each $q \in Q^+_{\mathsf{turn}}$, set $m_i[q] \leftarrow m_i[q] \oplus r_i[q]$.

Decoding Function: Compute $m := m_1 \oplus m_2 \oplus \cdots \oplus m_n$. Since $m = (m[q])_{q\in Q^+_{\mathsf{turn}}}$ represents a path on the tree, the referee outputs the value corresponding to the path. **Communication Complexity:** nt **Randomness Complexity:** $(n-1)t + \sum_{q\in Q_{\mathsf{branch}}} |\mathsf{descen}(q)| + \sum_{q\in Q_{\mathsf{shuf}}} H(\mathcal{F}_q)$

▶ **Theorem 4.** *Let $\mathcal{P}$ be a simple protocol with correctness and security for $f\colon X^n \to Y$ with opening complexity t. Then the above PSM protocol $\widehat{\mathcal{P}}$ is correct and secure for f with communication complexity nt.*

Proof. Let $\vec{x}$ be an input and $\vec{\pi}$ be all permutations in shuffle operations. Given $\vec{x}$ and $\vec{\pi}$, let $Q_{\vec{x},\vec{\pi}}$ be the set of all states $q \in Q^+_{\mathsf{turn}}$ that actually appear during the execution. By the definition of encoding function in $\widehat{\mathcal{P}}$, we can observe that $m_i[q] = \mathsf{val}_{i,q}(\vec{\pi}, x_i) \oplus r_i[q]$ if and only if $q \in Q_{\vec{x},\vec{\pi}}$. For any $q \in Q_{\vec{x},\vec{\pi}}$, we have

$$\begin{aligned} m[q] &= m_1[q] \oplus \cdots \oplus m_n[q] \\ &= (\mathsf{val}_{1,q}(\vec{\pi}, x_1) \oplus r_1[q]) \oplus \cdots \oplus (\mathsf{val}_{n,q}(\vec{\pi}, x_n) \oplus r_n[q]) \\ &= \mathsf{val}_{1,q}(\vec{\pi}, x_1) \oplus \cdots \oplus \mathsf{val}_{n,q}(\vec{\pi}, x_n) \oplus r_1[q] \oplus r_2[q] \oplus \cdots \oplus r_n[q]. \end{aligned}$$

By the choice of r_n in $\widehat{\mathcal{P}}$, we have $r_1[q] \oplus \cdots \oplus r_n[q] = 0$ for any $q \in Q^+_{\mathsf{turn}}$. Hence we have

$$m[q] = \mathsf{val}_{1,q}(\vec{\pi}, x_1) \oplus \cdots \oplus \mathsf{val}_{n,q}(\vec{\pi}, x_n) = c_q,$$

where $c_q \in \{0,1\}$ is the opened symbol at $q \in Q^+_{\mathsf{turn}}$ of $\mathcal{P}$. Thus, the correctness of $\widehat{\mathcal{P}}$ follows from the correctness of $\mathcal{P}$.

We prove the security of $\widehat{\mathcal{P}}$ by constructing a simulator Sim for $\widehat{\mathcal{P}}$. Given an output value $y \in Y$, Sim first invokes a simulator for $\mathcal{P}$ on input y, which outputs a visible sequence trace $\vec{v} \in (\mathsf{Vis}^{\mathcal{D}})^*$ of $\mathcal{P}$. From $\vec{v}$, the path of the execution in the protocol tree of $\mathcal{P}$ is determined. Let $c_1, c_2, \ldots, c_k \in \{0,1\}$ be the opened values in $\vec{v}$ and $q_1, q_2, \ldots, q_k \in Q^+_{\mathsf{turn}}$ be the states corresponding to them. Define $m' = (m'[q])_{q \in Q^+_{\mathsf{turn}}} \in \{0,1\}^t$ as follows:

$$m'[q] := \begin{cases} c_q & \text{if } q \in \{q_1, q_2, \ldots, q_k\}; \\ 0 & \text{otherwise.} \end{cases}$$

Then the simulator Sim chooses $r'_1, r'_2, \ldots, r'_{n-1} \in \{0,1\}^t$ uniformly at random and outputs $(r'_1, \ldots, r'_{n-1}, r'_1 \oplus \cdots \oplus r'_{n-1} \oplus m')$. For $q \in Q^+_{\mathsf{turn}}$, the message $(m_1[q], \ldots, m_n[q])$ is uniformly random conditioned on $\sum_{i=1}^n m_i[q] = c_q$ thanks to the random choice of $(r_1[q], \ldots, r_n[q])$. For $q \notin Q^+_{\mathsf{turn}}$, $(m_1[q], \ldots, m_n[q])$ is uniformly random thanks to the random choice of $\vec{s}$. Hence the distribution of $\mathsf{Sim}(y)$ equals the distribution of the messages, therefore $\widehat{\mathcal{P}}$ is secure. ◀

4.4 Extension of Our Conversion to Up-Down Cards

A deck of *up-down cards* [22] consists of two types of cards $\boxed{\uparrow}$ and $\boxed{\downarrow}$, which are transformed to each other by 180° rotation. The extension of our conversion to up-down cards is somewhat straightforward. The only difference is that random permutations $\pi_q \in \Pi_q$ in the the common randomness are replaced with *extended permutations* defined below.

Let ρ be the 180° rotation operation, and define $\mathsf{CMap} := \{\mathsf{id}, \rho\}$ as the set of rotation operations. An *extended permutation* (over d cards) is defined by a pair of d rotation operations $(\rho_1, \ldots, \rho_d) \in \mathsf{CMap}^d$ and a permutation $\pi \in S_d$. For a sequence of d cards $\Gamma := (\alpha_1, \alpha_2, \ldots, \alpha_d)$, we define an action of an extended permutation $\omega := ((\rho_1, \ldots, \rho_d), \pi) \in \mathsf{CMap}^d \times S_d$ by

$$\omega(\Gamma) := (\rho_1(\alpha_{\pi^{-1}(1)}), \ldots, \rho_d(\alpha_{\pi^{-1}(d)})),$$

i.e., it first permutes the sequence of cards according to π and then applies the rotation operations. Here, by defining the operation "$\circ$" as

$$\omega \circ \omega' := ((\rho_1 \circ \rho'_{\pi^{-1}(1)}, \ldots, \rho_d \circ \rho'_{\pi^{-1}(d)}), \pi\pi')$$

for $\omega = \big((\rho_j)_{j\in[d]}, \pi\big), \omega' = \big((\rho'_j)_{j\in[d]}, \pi'\big) \in \mathsf{CMap}^d \times S_d$, the set $\mathsf{CMap}^d \times S_d$ forms a monoid with $\mathsf{id} := ((\mathsf{id}, \ldots, \mathsf{id}), \mathsf{id})$ being the identity element, which is so-called the wreath product $\mathsf{CMap} \wr S_d$. In this extended model, perm and shuffle operations are extended to the set of extended permutations $\mathsf{CMap} \wr S_d$ instead of S_d. Other definitions are the same as the Mizuki–Shizuya model.

Theorems 1 and 4 also hold for up-down cards. In particular, an SSFO protocol using d up-down cards implies a PSM protocol with communication complexity nd, and a simple protocol using up-down cards with opening complexity t implies a PSM protocol with communication complexity nt. In Section 5.2, we demonstrate our conversion for a card-based protocol using up-down cards.

5 Application

5.1 Lower Bounds of Card-Based Protocols

From our conversion method in Section 4, lower bounds on the opening complexity for card-based protocols are immediately implied to lower bounds on the communication complexity of PSM protocols. This is summarized by Corollary 5.

▶ **Corollary 5.** *Let ℓ be a lower bound on the communication complexity of PSM protocols for a function $f : X^n \to Y$. Then for any simple protocol computing f, the opening complexity t must satisfy $t \geq \ell/n$. In particular, for any SSFO protocol computing f, the number of cards d must satisfy $d \geq \ell/n$.*

In 1994, Feige, Kilian, and Naor [12] constructed a two-player PSM protocol for any function $f : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ with communication complexity $O(2^k)$. After 20 years, Beimel, Ishai, Kumaresan, and Kushilevitz [6] improved it to $O(2^{k/2})$. For multi-player setting, in 2018, Beimel, Kushilevitz, and Nissim [7] constructed an n-player PSM protocol for any function $f : (\{0,1\}^k)^n \to \{0,1\}$ with communication complexity $O(2^{kn/2})$. In 2021, Assouline and Liu [3] improved it to $O(2^{k(n-1)/2})$ for infinitely many n. Although the communication complexity has gradually improved, the communication complexity of general-purpose protocols is still exponential with respect to the total input length nk, and improving it to subexponential seems difficult at least based on the existing techniques. We summarize this observation in the following conjecture.

▶ **Conjecture 6.** *There exists no PSM protocol for $f : (\{0,1\}^k)^n \to \{0,1\}$ with subexponential communication complexity $2^{o(kn)}$.*

From Conjecture 6, we obtain a lower bound on the opening complexity.

▶ **Corollary 7.** *Assume Conjecture 6 holds. Then there exists no simple protocol for $f : (\{0,1\}^k)^n \to \{0,1\}$ with opening complexity $2^{o(kn)}$. In particular, there exists no SSFO protocol for $f : (\{0,1\}^k)^n \to \{0,1\}$ with $2^{o(kn)}$ cards.*

Proof. If there exists a card-based protocol with $2^{o(kn)}$, there exists a PSM protocol with $n \cdot 2^{o(kn)} = 2^{o(kn)}$, contradicting to Conjecture 6. ◀

Of course, whether Corollary 7 holds is an open problem, and thus whether such a subexponential lower bound holds is not yet proven. However, we believe that the previous work of PSM protocols provides a piece of evidence that such a subexponential lower bound might hold in card-based cryptography.

One might imagine that a non-trivial lower bound for card-based protocols could be obtained from existing lower bounds for PSM protocols. In 2022, Ball and Randlph [5] showed a quadratic lower bound of PSM protocols based on the *modified Nečiporuk measure* $G^*(f)$ for a function $f\colon (\{0,1\}^k)^n \to \{0,1\}$, which is a measure of function complexity. They proved that the communication complexity of PSM protocols computing $f\colon (\{0,1\}^k)^n \to \{0,1\}$ is greater than or equal to $G^*(f)/2$, and a random function f has $G^*(f) = \Omega(\frac{kn^2}{\log_2(kn)})$, implying a quadratic lower bound of PSM protocols.

Now we try to obtain a lower bound of card-based protocols from the Ball–Randlph's lower bound. From Corollary 5, we obtain a lower bound $d = \Omega(\frac{kn}{\log_2(kn)})$ on the number of cards d of SSFO protocols, but this is not a strong bound because a card-based protocol naturally requires $\Omega(kn)$ cards for representing a kn-bit input. Unfortunately, since Ball and Randlph also showed that $G^*(f) \leq \frac{kn^2}{\log_2(kn)}$ for any f, we cannot obtain any better lower bound using the Nečiporuk measure. In order to obtain a non-trivial lower bound of card-based protocols, we need to obtain a super-quadratic lower bound $\omega(kn^2)$ of PSM protocols. As far as we know, no super-quadratic lower bound of PSM protocols has been proven so far, but our result provides a new motivation for obtaining such a lower bound.

5.2 PSM Protocol for Indirect Storage Access Function

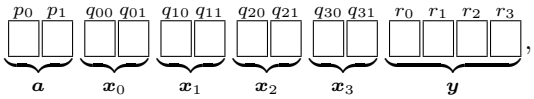
An *indirect storage access* (ISA, in short) function [23] is a function $f_{\mathsf{ISA}}^{k,\ell} : \{0,1\}^{k+\ell K+L} \to \{0,1\}$, where $K = 2^k$ and $L = 2^\ell$, defined as follows:

$$f_{\mathsf{ISA}}^{k,\ell}(\boldsymbol{a}, \boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{K-1}, \boldsymbol{y}) := y_{|\boldsymbol{x}_{|\boldsymbol{a}|}|},$$

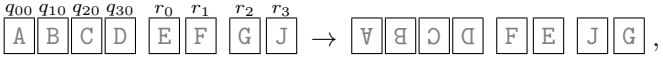
where $\boldsymbol{a} = (a_0, \ldots, a_{k-1}) \in \{0,1\}^k$, $\boldsymbol{x}_j = (x_{j,0}, \ldots, x_{j,\ell-1}) \in \{0,1\}^\ell$ $(0 \le j \le K-1)$, $\boldsymbol{y} = (y_{00\cdots0}, y_{00\cdots1}, \ldots, y_{11\cdots1}) \in \{0,1\}^L$, and for a bit string $b = (b_0, b_1, \ldots, b_{t-1}) \in \{0,1\}^t$, $|b|$ represents an integer $\sum_{i=0}^{t-1} b_i 2^i$.

We describe our protocol for $f_{\mathsf{ISA}}^{k,\ell}$ using up-down cards when $k = \ell = 2$. The general case follows similarly. The protocol proceeds as follows:

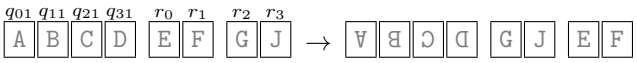
1. Arrange the input sequence as follows:



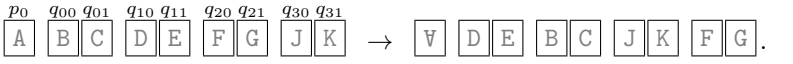
   where p_i, $q_{i,j}$, and r_i are labels of positions.
2. Apply a shuffle $(\mathsf{shuf}, \{\mathsf{id}, \omega_0\})$ for ω_0 defined as follows:



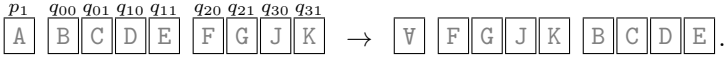
   where $\mathtt{A}, \mathtt{B}, \ldots, \mathtt{J}$ are alphabets with no 180°-rotational symmetry. (Note that these characters are only used to represent the rotation and permutation, and are not actually written on the cards. In the following, we will use the same notation in the protocol.)
3. Apply a shuffle $(\mathsf{shuf}, \{\mathsf{id}, \omega_1\})$ for ω_1 defined as follows:



4. Apply a shuffle $(\mathsf{shuf}, \{\mathsf{id}, \omega_0'\})$ for ω_0' defined as follows:



5. Apply a shuffle $(\mathsf{shuf}, \{\mathsf{id}, \omega_1'\})$ for ω_1' defined as follows:



6. Open the first k cards. Let $\widetilde{\boldsymbol{a}} \in \{0,1\}^k$ be the k-tuple of the opened values.
7. Open the ℓ consecutive cards from the $(k+1+|\widetilde{\boldsymbol{a}}|\ell)$-th card. Let $\widetilde{\boldsymbol{x}} \in \{0,1\}^\ell$ be the ℓ-tuple of the opened values.
8. Open the $(k+\ell K+1+|\widetilde{\boldsymbol{x}}|)$-th card and output the opened value.

For the general case, since the number of opened cards is $k+\ell+1$, the opening complexity is $2^{k+\ell+1} = 2KL$. Thus the communication complexity of the resulting PSM protocol is $2KL(k+\ell K+L) = O(KL^2 + K^2L\ell)$. The communication complexity of Ishai–Kushilevitz's protocol for $f_{\mathsf{ISA}}^{k,\ell}$ is $O(K^2L^3 + K^3L^2\ell)$ since $f_{\mathsf{ISA}}^{k,\ell}$ can be computed by a BP of size $O(KL)$, and that of Kolesnikov's protocol for $f_{\mathsf{ISA}}^{k,\ell}$ is $O(K^2L^2(k+\ell)^2)$ since $f_{\mathsf{ISA}}^{k,\ell}$ can be computed by a formula of depth $2(k+\ell)$. Therefore, our protocol is more efficient than those protocols.

6 Conclusion

In this paper, we showed a generic conversion from card-based to PSM protocols. The significance of our conversion is to show a direct relationship from card-based to conventional cryptography for the first time, which was previously thought to be of little relevance.

Finally, we list open problems and future research directions as follows:

Deriving unconditional lower bounds for card-based protocols: An open problem is to derive unconditional lower bounds for card-based protocols from those of PSM protocols. To obtain such a lower bound for card-based protocols, it is sufficient to obtain a super-quadratic lower bound $\omega(kn^2)$ on the communication complexity of the PSM protocols for $f\colon (\{0,1\}^k)^n \to \{0,1\}$. However, as already mentioned in Section 5.1, a lower bound from the Nečiporuk measure will never be super-quadratic. Thus, it seems essential to develop new techniques to obtain super-quadratic lower bounds.

Constructing efficient PSM protocols: An open problem is to determine a class of functions for which our conversion yields efficient PSM protocols. Since the resulting PSM protocol has communication complexity linear to the opening complexity of the underlying protocol (or the number of cards for SSFO protocols), the following questions are worthy towards obtaining efficient PSM protocols with polynomial communication:

- What is the class of functions for which a simple card-based protocol exists with opening complexity polynomial in the input length?
- What is the class of functions for which an SSFO card-based protocol exists with polynomial number of cards in the input length?

Further relations among card-based and PSM protocols: An open problem is to develop deeper connections between card-based protocols and PSM or other kinds of conventional cryptographic protocols as follows:

- Is it possible to improve the efficiency (i.e., the communication complexity of the resulting PSM protocol) of our conversion?
- Can we establish a conversion in the opposite direction, i.e., from a certain subclass of PSM protocols back to card-based protocols?

References

1 Benny Applebaum, Barak Arkis, Pavel Raykov, and Prashant Nalini Vasudevan. Conditional disclosure of secrets: Amplification, closure, amortization, lower-bounds, and separations. *SIAM J. Comput.*, 50(1):32–67, 2021. doi:10.1137/18M1217097.

2 Benny Applebaum, Thomas Holenstein, Manoj Mishra, and Ofer Shayevitz. The communication complexity of private simultaneous messages, revisited. *J. Cryptol.*, 33(3):917–953, 2020. doi:10.1007/S00145-019-09334-Y.

3 Léonard Assouline and Tianren Liu. Multi-party psm, revisited: Improved communication and unbalanced communication. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography – 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 194–223. Springer, 2021. doi:10.1007/978-3-030-90453-1_7.

4 Marshall Ball, Justin Holmgren, Yuval Ishai, Tianren Liu, and Tal Malkin. On the complexity of decomposable randomized encodings, or: How friendly can a garbling-friendly PRF be? In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 86:1–86:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ITCS.2020.86.

5 Marshall Ball and Tim Randolph. A note on the complexity of private simultaneous messages with many parties. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, volume 230 of *LIPIcs*, pages 7:1–7:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.ITC.2022.7.

6 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 317–342. Springer, 2014. doi:10.1007/978-3-642-54242-8_14.

7 Amos Beimel, Eyal Kushilevitz, and Pnina Nissim. The complexity of multiparty PSM protocols and related models. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 287–318. Springer, 2018. doi:10.1007/978-3-319-78375-8_10.

8 Claude Crépeau and Joe Kilian. Discreet solitary games. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 319–330. Springer, 1993. doi:10.1007/3-540-48329-2_27.

9 László Csirmaz. The size of a share must be large. *J. Cryptol.*, 10(4):223–231, 1997. doi:10.1007/S001459900029.

10 Deepesh Data, Manoj Prabhakaran, and Vinod M. Prabhakaran. On the communication complexity of secure computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 199–216. Springer, 2014. doi:10.1007/978-3-662-44381-1_12.

11 Bert den Boer. More efficient match-making and satisfiability: *The Five Card Trick*. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, volume 434 of *Lecture Notes in Computer Science*, pages 208–217. Springer, 1989. doi:10.1007/3-540-46885-4_23.

12 Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563. ACM, 1994. doi:10.1145/195058.195408.

13 Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 485–502. Springer, 2015. doi:10.1007/978-3-662-48000-7_24.

14 Yuji Hashimoto, Koji Nuida, Kazumasa Shinagawa, Masaki Inamura, and Goichiro Hanaoka. Toward finite-runtime card-based protocol for generating a hidden random permutation without fixed points. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 101-A(9):1503–1511, 2018. doi:10.1587/TRANSFUN.E101.A.1503.

15 Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Fifth Israel Symposium on Theory of Computing and Systems, ISTCS 1997, Ramat-Gan, Israel, June 17-19, 1997, Proceedings*, pages 174–184. IEEE Computer Society, 1997. doi:10.1109/ISTCS.1997.595170.

16 Julia Kastner, Alexander Koch, Stefan Walzer, Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. The minimum number of cards in practical card-based protocols. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017 – 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 126–155. Springer, 2017. doi:10.1007/978-3-319-70700-6_5.

17 Alexander Koch. The landscape of optimal card-based protocols. *IACR Cryptol. ePrint Arch.*, page 951, 2018. URL: https://eprint.iacr.org/2018/951.

18 Alexander Koch, Michael Schrempp, and Michael Kirsten. Card-based cryptography meets formal verification. *New Gener. Comput.*, 39(1):115–158, 2021. doi:10.1007/S00354-020-00120-0.

19 Alexander Koch, Stefan Walzer, and Kevin Härtel. Card-based cryptographic protocols using a minimal number of cards. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015 – 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 – December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 783–807. Springer, 2015. doi:10.1007/978-3-662-48797-6_32.

20 Takaaki Mizuki, Michihito Kumamoto, and Hideaki Sone. The five-card trick can be done with four cards. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012 – 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 598–606. Springer, 2012. doi:10.1007/978-3-642-34961-4_36.

21 Takaaki Mizuki and Hiroki Shizuya. A formalization of card-based cryptographic protocols via abstract machine. *Int. J. Inf. Sec.*, 13(1):15–23, 2014. doi:10.1007/S10207-013-0219-4.

22 Takaaki Mizuki and Hiroki Shizuya. Practical card-based cryptography. In Alfredo Ferro, Fabrizio Luccio, and Peter Widmayer, editors, *Fun with Algorithms – 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1-3, 2014. Proceedings*, volume 8496 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2014. doi:10.1007/978-3-319-07890-8_27.

23 John E. Savage. *Models of computation – exploring the power of computing.* Addison-Wesley, 1998.