A Single Shuffle Is Enough for Secure Card-Based Computation of Any Circuit

Kazumasa Shinagawa[*][†][‡], Koji Nuida[§]

April 10, 2019

Abstract

It is known that information-theoretically secure computation can be done by using a deck of physical cards. In card-based protocols, *shuffles*, which covertly rearrange the order of cards according to a permutation chosen randomly, are the heaviest operations. Due to this, the number of shuffles in a protocol is desired to be small. However, as far as we know, there are no general-purpose protocols with a constant number of shuffles. In this paper, we construct a general-purpose protocol with a constant number of shuffles; surprisingly, just *one* (somewhat complicated) shuffle. This is achieved by introducing the *garbled circuit* methodology. Moreover, to make our shuffle simpler, we also develop a novel technique for aggregating several pile-scramble shuffles (which are efficiently implementable) into one pile-scramble shuffle. Consequently, we also construct a general-purpose protocol with two pile-scramble shuffles. Both techniques may lead to many other applications in this area.

Key words: Card-based protocols; Secure computations; Garbled circuits

1 Introduction

Suppose n players with secret input bits $x_1, \cdots, x_n \in \{0,1\}$, respectively, wish to compute some function $f : \{0,1\}^n \to \{0,1\}^m$ while hiding their inputs from each other. While there are several approaches for this secure computation problem, we try to solve it by using a *deck of physical cards.* A protocol using a deck of physical cards is called a *card-based protocol* [2–27]. As in the previous works, we use a deck of cards with two symbols $\boxed{\clubsuit}$, $\boxed{\heartsuit}$ whose back sides are the same pattern $\boxed{?}$. All cards having the same symbol are assumed to be indistinguishable from one another. A secret binary information $b \in \{0,1\}$ is encoded

[*]Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo, 152-8552, Japan
[†]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo Waterfront Bio-IT Research Building 2-4-7 Aomi, Koto-ku, Tokyo, 135-0064, Japan
[‡]Corresponding author: `shinagawakazumasa@gmail.com`
[§]The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8654, Japan

by a pair of two face-down cards $\boxed{?}\boxed{?}$, called a *commitment to* b, whose front-side is $\boxed{\clubsuit}\boxed{\heartsuit}$ if $b = 0$, otherwise $\boxed{\heartsuit}\boxed{\clubsuit}$. The goal of a card-based protocol is, given a collection of n commitments to $x = (x_1, \cdots, x_n)$, to produce a collection of m commitments to $f(x) = y = (y_1, \cdots, y_m)$ without compromising the secrecy of the inputs.

In the standard (computer-based) cryptography, security of protocols relies on *random numbers*, whereas in card-based protocols, security relies on *shuffle operations*, which covertly rearrange the order of cards according to a permutation chosen randomly. A practical way to securely realize such shuffles (without a trusted third-party as a dealer) has been investigated (when the distributions of permutations are fairly simple), but the shuffles are still a non-trivial and the heaviest operation in card-based protocols. Due to this, the number of shuffles in a card-based protocol is desired to be small.

In 2009, Mizuki and Sone [14] constructed protocols for elementary functions using a shuffle known as a *random bisection cut.* It yields a general-purpose protocol with q random bisection cuts, where q is the number of gates in a circuit. In 2015, Shinagawa et al. [23] constructed a general-purpose protocol with n shuffles, where n is the number of input bits. In these existing protocols, the numbers of shuffles are dependent on the number of gates or input bits, therefore practically inefficient when one wants to compute a complicated function.

Our motivation in this work is to develop a general-purpose protocol that requires only a constant number of shuffles (independent of the function to be computed). As far as we know, no such protocols have been proposed. Our question is summarized as follows:

Can we have a general-purpose protocol with a constant number of shuffles?

1.1 Our Results and Techniques

In this paper, we affirmatively answer the question by constructing a finite-runtime protocol that securely computes an arbitrary function $f : \{0,1\}^n \to \{0,1\}^m$ with one shuffle. This is achieved by introducing the *garbled circuit* methodology [28] into the area of card-based cryptography. This technique enables to construct a protocol whose shuffles are executed in a way that a shuffle for computing a gate is independent of a shuffle for another gate; then we convert it to a protocol with one shuffle by aggregating all shuffles.

A drawback of our result above is that, though the shuffles in the naive construction of our card-based garbled circuit are pile-scramble shuffles which are known to be efficiently implementable, the single shuffle after aggregating the shuffles becomes more complicated. To resolve the issue, we also develop a new technique, which we call a *batching technique*, which enables to combine multiple independent pile-scramble shuffles into a single pile-scramble shuffle (using some additional auxiliary cards). Combining our garbled circuit technique and the batching technique, we obtain a protocol with two pile-scramble shuffles. While the number shuffles is increased by 1, a pile-scramble shuffle is known to be implementable efficiently.

In Section 2, we summarize basic definitions. In Section 3, we introduce a card-based variant of the garbled circuit technique. In Section 4, based on the garbled circuit technique, we construct a general-purpose protocol with one shuffle. In Section 5, we present the batching technique. In Section 6, based on the garbled circuit technique and the batching technique, we construct a general-purpose protocol with two pile-scramble shuffles.

Garbled circuit technique. Roughly speaking, the usual garbled circuit technique to securely evaluate a circuit proceeds as follows; (I) represent each gate in the circuit as the truth table of the associated function $\{0,1\}^2 \to \{0,1\}$; (II) randomly permute the four input-output pairs in the truth table, in order to prevent leakage of the output value when the gate is evaluated; (III) randomly encode each of the inputs and outputs in the truth table (in a consistent manner between the output of the previous gate and the corresponding input(s) of the subsequent gate(s)), in order to hide the input values; (IV) then successively open one true output value among the four in the randomly encoded truth table of each gate, from bottom to top. Protocol 1 in Section 3 is a translation of the process described above into a card-based protocol, where the random permutations in (II) and the random encoding in (III) are realized by shuffle operations (the aforementioned consistency in (III) between the gates are assured by the property of pile-scramble shuffles). See Section 3 for details.

Based on the garbled circuit construction, we obtain a general-purpose protocol with one shuffle immediately. This is done by aggregating all shuffles in the garbled circuit construction into one shuffle. This strategy works because all shuffles in the garbled circuit construction are successively applied.

To the authors' best knowledge, our present work is the first attempt to effectively adapt the garbled circuit methodology to card-based protocols. The reason of why the application of garbled circuits to card-based protocols has not been investigated so far can be presumed as follows; in the ordinary (i.e., not card-based) situation, secure multi-party computation based on garbled circuits seems to be considered fairly inefficient compared to the methodology of successively evaluating fundamental gates based on the secret sharing; and the techniques for securely evaluating each fundamental gate have been well established in the card-based setting as well (cf., [14]). One of the main contributions of this work is to reveal, as opposed to the intuition described above, that even a naive application of the garbled circuit technique to card-based protocols is not very inefficient compared to the aforementioned successive gate evaluation methodology.

The reason of efficiency improvement by moving to the card-based setting can be explained as follows. In the ordinary setting of garbled circuits, in order to prevent an attack to open more than one output value at some gate, the input values in the truth table should be (randomly) encoded into a significantly large string, as otherwise the adversarial party who is *locally* evaluating the garbled circuit can guess the encoded inputs other than that given by the previous gates. In contrast, in the card-based setting, a protocol is supposed to be jointly executed by all parties in a *public* environment, therefore such a dishonest attempt to open more than one output value can be automatically prevented and consequently the garbled truth table may be as small as the original truth table. This phenomenon reflects the typical property of card-based protocols.

Moreover, the more important property of the card-based garbled circuit technique which we find in this work is the compatibility with parallel processing of shuffles. Namely, among the four steps in the garbled circuit technique described above, the random permutations (shuffles) for the truth tables in step (II) can be performed *in parallel* for all gates, and the random encoding (shuffles) of the truth tables in step (III) can be performed *in parallel* for all input/output bits of the gates, too. The parallel executability of shuffles combined with our batching technique explained in Section 5 achieves a protocol with two efficiently implementable shuffles described in Section 6. In contrast, the existing methodology of successively evaluating fundamental gates is not compatible with the parallel processing; in fact, in this methodology, a shuffle for evaluating a gate cannot be performed until the inputs to the gate are determined by the previously evaluated gates.

Batching technique. Another main contribution of this work is to develop a novel technique to convert a number of shuffles performed in parallel into a *single* shuffle. This is a key tool to construct our protocol with two pile-scramble shuffles, which is an operation of applying a hidden and uniformly random permutation to a sequence of piled cards. Here the number of cards in a pile is supposed to be equal for all the piles in order to make the shuffled piles indistinguishable from each other. Pile-scramble shuffles are used in our protocols as well as in many existing card-based protocols. We also note that this is a kind of general technique, so that this technique is expected to lead to many other applications in card-based cryptography, which will be future research topics.

To explain the batching technique, here we use a small example of combining a pile-scramble shuffle of k piles and a pile-scramble shuffle of ℓ piles. The underlying idea is to first apply a pile-scramble shuffle to the whole of $k+\ell$ piles and then divide the resulting piles into the first set of k piles and the second set of ℓ piles. Now both of the first k piles and the second ℓ piles are individually shuffled uniformly at random whenever the shuffle for the whole of $k+\ell$ piles is uniformly random. However, this naive idea does not work in general when the piles consist of face-down cards and the symbols on the front sides of the cards cannot be revealed; in fact, it is impossible in this case to detect the k piles in the first set among the $k+\ell$ shuffled piles. To overcome this issue, before performing the shuffle, we append some auxiliary face-down cards to the top of each pile, where the auxiliary cards for each of the first k piles (respectively, the second ℓ piles) encode the information that this pile belongs to the first (respectively, second) set of piles. Then, even after the shuffle, the piles in the two sets are still distinguishable from each other while keeping the front sides of the original cards secret, by opening the auxiliary cards for each pile only.

We note that this technique requires two kinds of additional cards. One is as mentioned above to make the piles from different sets of piles distinguishable from each other. The other is to equalize the numbers of cards in the piles from different sets of piles, which is required in general since the numbers of cards in the piles must be equal in a pile-scramble shuffle. See Section 5 for details. Therefore, our batching technique increases the total number of cards used in a protocol; but we emphasize that the number of cards in our resulting protocol with two pile-scramble shuffles is still not very large (in comparison to, e.g., the previous general-purpose protocol in [23] with n shuffles, which requires an exponentially (in n) large number of cards). See Section 6 for details.

1.2 Related Works

The Five-Card Trick, which is a card-based AND protocol using five cards, is proposed by den Boer [3]. Crépeau and Kilian [2] achieved to securely compute any function by constructing protocols for fundamental gates and successively evaluating them. While they are *Las-Vegas*[1] protocols, Mizuki and Sone [14] constructed *finite-runtime* protocols for fundamental gates with improving the numbers of cards and shuffles. It yields a general-purpose protocol with q shuffles, where q is the number of gates in a circuit. Shinagawa et al. [23] constructed a general-purpose protocol with $2n$ shuffles, where n is the number of the input bits. However, up until now, it is still unknown that a *constant* number of shuffles is sufficient to securely compute any function. In this paper, we show that only one shuffle is sufficient to securely compute any function $f : \{0,1\}^n \to \{0,1\}$ with small number of cards. Our work achieves the first general-purpose protocol with a constant number of shuffles.

Our main protocol (Section 4) uses a uniform and closed shuffle. On the other hand, some existing protocols used non-uniform and/or non-closed shuffles [8, 22]. It is known that uniform and closed shuffles are easy to implement compared to non-uniform and/or non-closed shuffles. See Section 2.3 for details.

Our protocol with two shuffles uses two pile-scramble shuffles. This kind of shuffles is first proposed by Ishikawa et al. [5] to give a protocol which securely generates a random permutation without fixed points. One of our contributions is to develop a new use of pile-scramble shuffles. Specifically, we use them as a tool to combine a number of shuffles into a *single* shuffle while the previous works [4, 5] use them in order to treat a *permutation* naturally.

2 Preliminaries

For an integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. We use S_k to denote the k-th symmetric group for an integer $k \geq 1$, i.e., S_k is the set of all permutations on the set $[k]$.

[1]We say that a protocol is a Las-Vegas protocol if the *expected* running time of the protocol is finite.

2.1 Circuits

In this paper, we use the following formulation for the circuits given in [1]. A *circuit* C is defined as a six-tuple $C = (n, m, q, L, R, G)$. Here, $n \geq 1$ is the number of input bits, $m \geq 1$ is the number of output bits, $q \geq 1$ is the number of gates. We assume that each gate has two incoming wires and one outgoing wire, and an outgoing wire that is not an output wire of the protocol may then branch and go into several gates as the incoming wires. Accordingly, the outgoing wire of a gate and the corresponding incoming wire(s) of the subsequent gate(s) are identified with each other. We also allow a case where the two incoming wires of a gate come from the same previous gate, in order to realize by a gate a single-input function such as the NOT function.

Now we associate indices to the input bits, gates, wires, and the output bits as follows: $\mathsf{Inputs} = \{1, \cdots, n\}$, $\mathsf{Gates} = \{n+1, \cdots, n+q\}$, $\mathsf{Wires} = \{1, \cdots, n+q\}$, $\mathsf{Outputs} = \{n+q-m+1, \cdots, n+q\}$. Every wire $w \in \mathsf{Wires}$ is either an input wire or an outgoing wire of some gate, which has the same index as the wire itself. Then, $L, R : \mathsf{Gates} \to \mathsf{Wires} \setminus \mathsf{Outputs}$ are functions that map a gate to its left (respectively, right) incoming wire. Moreover, for each $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$, we write $L^{-1}(w), R^{-1}(w)$ to denote the set of the gates g satisfying $L(g) = w$ (respectively, $R(g) = w$). Finally, $G : \mathsf{Gates} \times \{0,1\}^2 \to \{0,1\}$ is a function that determines the functionality of each gate; given $g \in \mathsf{Gates}$ and $b_1, b_2 \in \{0,1\}$, we often write $G_g(b_1, b_2) = G(g, (b_1, b_2)) \in \{0,1\}$ to simplify the description. We require $L(g) \leq R(g) < g$ for all $g \in$ Gates.

Example 1 *We consider a function $f : \{0,1\}^3 \to \{0,1\}^2$ given by $f(x_1, x_2, x_3) = ((x_1 \wedge x_2) \oplus x_3, (x_1 \wedge x_2) \vee x_3)$. A circuit for f can be defined by $n = 3$, $m = 2$, $q = 3$, $G_4(b_1, b_2) = b_1 \wedge b_2$, $G_5(b_1, b_2) = b_2 \oplus b_1$, $G_6(b_1, b_2) = b_2 \vee b_1$, $L(4) = 1$, $R(4) = 2$, $L(5) = 3$, $R(5) = 4$, $L(6) = 3$, and $R(6) = 4$.*

2.2 Card-based Protocols

In this section, we introduce several definitions about card-based protocols for describing our protocol. We follow the formalization proposed by Mizuki and Shizuya [12]. We also follow some notations described by Koch, Walzer, and Härtel [8]. Here, we only treat a finite-runtime protocol (i.e., a protocol always terminates in a finite number of steps) and our protocol only needs *uniform shuffles*, which are believed to be easy to implement compared to *non-uniform shuffles*.

A *deck* $\mathcal{D}$ is a finite multiset of symbols $\heartsuit$ and $\clubsuit$. For instance, $\mathcal{D} = [\heartsuit, \heartsuit, \clubsuit, \clubsuit]$ is a deck. Intuitively, the deck is the set of front-side symbols of the physical cards used during a protocol. For a symbol $c \in \mathcal{D}$, $\frac{c}{?}$ denotes a *face-up card* and $\frac{?}{c}$ a *face-down card* with symbol c, respectively. For a card α (i.e., $\alpha = \frac{c}{?}$ or $\alpha = \frac{?}{c}$ for some symbol c), $\mathsf{top}(\alpha)$ and $\mathsf{atom}(\alpha)$ denote the symbol in the upper side and the symbol distinct from '?', respectively. For instance, $\mathsf{top}(\frac{?}{\heartsuit}) = ?$ and $\mathsf{atom}(\frac{?}{\heartsuit}) = \heartsuit$. A *card sequence* Γ of $\mathcal{D}$ is a vector of $|\mathcal{D}|$ cards $(\alpha_1, \cdots, \alpha_{|\mathcal{D}|})$ such that $[\mathsf{atom}(\alpha_1), \cdots, \mathsf{atom}(\alpha_{|\mathcal{D}|})] = \mathcal{D}$

(as a multiset). We define $\mathsf{Seq}^{\mathcal{D}}$ to be the set of all sequences of a deck $\mathcal{D}$, i.e., $\mathsf{Seq}^{\mathcal{D}} = \{(\alpha_1, \cdots, \alpha_{|\mathcal{D}|}) : [\mathsf{atom}(\alpha_1), \cdots, \mathsf{atom}(\alpha_{|\mathcal{D}|})] = \mathcal{D}\}$. For a card sequence $\Gamma = (\alpha_1, \cdots, \alpha_t)$, $\mathsf{top}(\Gamma)$ and $\mathsf{atom}(\Gamma)$ denote $(\mathsf{top}(\alpha_1), \cdots, \mathsf{top}(\alpha_t))$ and $(\mathsf{atom}(\alpha_1), \cdots, \mathsf{atom}(\alpha_t))$, respectively. For a card α, $\mathsf{swap}(\alpha)$ denotes the flipped card, i.e., $\mathsf{swap}(\frac{c}{?}) = \frac{?}{c}$ and $\mathsf{swap}(\frac{?}{c}) = \frac{c}{?}$. A *commitment* of $x \in \{0,1\}^*$ is a face-down card sequence whose arrangement represents x. As in the previous works, we define $\mathrm{Com}(0) = (\frac{?}{\clubsuit}, \frac{?}{\heartsuit})$ and $\mathrm{Com}(1) = (\frac{?}{\heartsuit}, \frac{?}{\clubsuit})$. (Note that $\mathsf{top}(\mathrm{Com}(0)) = \mathsf{top}(\mathrm{Com}(1)) = (?, ?)$. Intuitively, it means that $\mathrm{Com}(0)$ and $\mathrm{Com}(1)$ are indistinguishable without turning over them.) For a bit string $x = (x_1, \cdots, x_t) \in \{0,1\}^t$, $\mathrm{Com}(x) = (\mathrm{Com}(x_1), \cdots, \mathrm{Com}(x_t))$.

Intuitively, a protocol execution is a sequential process of, given an input card sequence, transforming the current card sequence step by step where the action at each step is adaptively determined according to the results of previous steps (this is an analogy of the standard formalization of algorithms in terms of Turing machines). To formalize the idea, we define a *sequence trace* $(\Gamma_0, \Gamma_1, \cdots, \Gamma_t)$ to be a tuple of card sequences such that Γ_0 is an input card sequence and Γ_t is the current card sequence, and define the corresponding *visible sequence trace* to be $(\mathsf{top}(\Gamma_0), \mathsf{top}(\Gamma_1), \cdots, \mathsf{top}(\Gamma_t))$. We define an *action* to be an operation that transforms the current card sequence Γ_t into a card sequence Γ_{t+1} (and then appends Γ_{t+1} to the end of the sequence trace). Then a protocol is formalized as a quadruple $\mathcal{P} = (\mathcal{D}, U, Q, A)$ consisting of the following objects: $\mathcal{D}$ is a deck, $U \subseteq \mathsf{Seq}^{\mathcal{D}}$ is a set of input card sequences, Q is a set of states having an initial state $q_0 \in Q$ and a final state $q_{\mathrm{f}} \in Q$, and $A : (Q \setminus \{q_{\mathrm{f}}\}) \times \mathsf{Vis} \to Q \times \mathsf{Action}$ is an action function, where Vis is the set of possible current visible sequence traces and Action consists of the following actions:

- (perm, π) for $\pi \in S_{|\mathcal{D}|}$. This transforms a sequence $(\alpha_1, \cdots, \alpha_{|\mathcal{D}|})$ into a permuted sequence $(\alpha_{\pi^{-1}(1)}, \cdots, \alpha_{\pi^{-1}(|\mathcal{D}|)})$.
- $(\mathsf{shuffle}, \Pi)$ for $\Pi \subseteq S_{|\mathcal{D}|}$. This transforms a sequence $(\alpha_1, \cdots, \alpha_{|\mathcal{D}|})$ into $(\alpha_{\pi^{-1}(1)}, \cdots, \alpha_{\pi^{-1}(|\mathcal{D}|)})$, where π is uniformly and independently chosen from Π.
- (turn, P) for $P \subseteq [|\mathcal{D}|]$. This transforms a sequence $(\alpha_1, \cdots, \alpha_{|\mathcal{D}|})$ into a sequence $(\beta_1, \cdots, \beta_{|\mathcal{D}|})$, where $\beta_i = \mathsf{swap}(\alpha_i)$ if $i \in P$, otherwise $\beta_i = \alpha_i$.
- $(\mathsf{result}, P_1, \cdots, P_m)$ for disjoint m sets having two positions $P_i = \{j_{2i-1}, j_{2i}\} \subset [|\mathcal{D}|]$. This halts the protocol with outputs $(\alpha_{j_1}, \alpha_{j_2}), \cdots, (\alpha_{j_{2m-1}}, \alpha_{j_{2m}})$, where $(\alpha_1, \cdots, \alpha_{|\mathcal{D}|})$ is the current sequence.

Definition 1 (Correctness) *Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function. We say that a protocol $\mathcal{P} = (\mathcal{D}, U, Q, A)$ computes f if the following holds:*

- *It always terminates in a finite number of steps.*
- $U = \{\Gamma^x : x \in \{0,1\}^n\}$ *for* $\Gamma^x = (\alpha_1, \cdots, \alpha_{|\mathcal{D}|}) \in \mathsf{Seq}^{\mathcal{D}}$, *where* $(\alpha_{2i-1}, \alpha_{2i}) = \mathrm{Com}(x_i)$ *and the remaining* $|\mathcal{D}| - 2n$ *helping cards* $\alpha_{2n+1}, \cdots, \alpha_{|\mathcal{D}|}$ *do not depend on the input* x.

- *For an execution starting from Γ^b for $b \in \{0,1\}^n$, the protocol ends (i.e., entering the final state q_{f}) with the action* $(\mathsf{result}, p_1, p_2, \cdots, p_{2m-1}, p_{2m})$ *such that* $(\beta_{p_{2i-1}}, \beta_{p_{2i}}) = \mathrm{Com}(f_i(b))$*, where* $\Gamma = (\beta_1, \cdots, \beta_{|\mathcal{D}|})$ *is the final sequence and* $f_i(b)$ *is the i-th output bit of* $f(b)$*.*

Definition 2 (Security) *Let $\mathcal{P} = (\mathcal{D}, U, Q, A)$ be a protocol. Let $\Gamma_{\mathcal{M}}$ be a random variable of the input sequences U with an input distribution $\mathcal{M}$. Let V be a random variable of the visible sequence trace at the end of the protocol execution. We say that $\mathcal{P}$ is secure if for any distribution $\mathcal{M}$, $\Gamma_{\mathcal{M}}$ and V are stochastically independent.*

2.3 Miscellaneous Definitions

Outcome. When P is a set of positions of size ℓ pointing commitments $\mathrm{Com}(a_1), \cdots, \mathrm{Com}(a_\ell)$, we say that $(a_1, \cdots, a_\ell)$ is an *outcome* of the turning operation. For example, the following operation is $(\mathsf{turn}, \{1,2\})$ and the outcome of it is 1 since $\mathrm{Com}(1) = (\frac{?}{\heartsuit}, \frac{?}{\clubsuit})$.

$$\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?} \longrightarrow \boxed{\heartsuit}\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

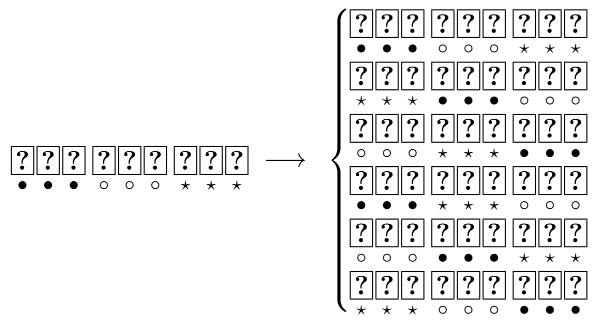
Uniform and closed shuffles. Shuffles of our model are said to be *uniform* because in a shuffle, a permutation is uniformly chosen. A shuffle $(\mathsf{shuffle}, \Pi)$ is said to be *closed* if Π is closed under the composition, i.e., for any $\pi, \pi' \in \Pi$, the composite permutation $\pi \circ \pi'$ is also contained in Π. A uniform and closed shuffle can be implementable under the honest-but-curious assumption as follows: Each party randomly chooses a permutation from Π, and *covertly* rearranges the order of cards according to the permutation. Due to the uniform and closed property, the resulting sequence is equivalently distributed to the sequence applied by the shuffle $(\mathsf{shuffle}, \Pi)$. Our main protocol uses a uniform and closed shuffle.

Pile-scramble shuffle. Our protocol with two shuffles uses uniform and closed shuffles of special type called *pile-scramble shuffles* [5]. Let $P_1, \cdots, P_k \subset [|\mathcal{D}|]$ be k disjoint subsets of $[|\mathcal{D}|]$ such that each P_i has exactly ℓ elements, i.e., $P_i = \{p_{i,1}, \cdots, p_{i,\ell}\}$ where $p_{i,1} < \cdots < p_{i,\ell}$ for every $i \in [k]$. A pile-scramble shuffle for $P_1, \cdots, P_k$ denoted by $(\mathsf{pileShuffle}, P_1, \cdots, P_k)$ is defined by $(\mathsf{shuffle}, \Pi)$ for $\Pi = \{\mathsf{Pile}(\pi) : \pi \in S_k\}$ where we define $\mathsf{Pile} : S_k \to S_{|\mathcal{D}|}$ by

$$\mathsf{Pile}(\pi)(i') = \begin{cases} p_{\pi(i),j} & \text{if } i' = p_{i,j} \text{ for some } i \in [k] \text{ and } j \in [\ell] \\ i' & \text{otherwise.} \end{cases}$$

For example, a pile-scramble shuffle $(\mathsf{pileShuffle}, \{1,2,3\}, \{4,5,6\}, \{7,8,9\})$

for a card sequence of nine cards is as follows.



It generates one of six sequences with probability exactly 1/6.

A pile-scramble shuffle is known to be implementable by physical envelopes; each pile is put into an envelope, and then the order of envelopes is scrambled. The number of envelopes required is the same as the number of piles.

3 Card-based Garbled Circuits

In this section, we introduce a card-based variant of the garbled circuit technique.

Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function and let $C = (n, m, q, L, R, G)$ be a circuit computing f. For each gate $g \in \mathsf{Gates}$, let $t_g \in \{0,1\}^{12}$ be the string representing the truth table of g defined by

$$t_g = (0, 0, G_g(0,0), 0, 1, G_g(0,1), 1, 0, G_g(1,0), 1, 1, G_g(1,1))\,.$$

For instance, $t_g = (0,0,0,0,1,0,1,0,0,1,1,1)$ represents the truth table of an AND gate. The initial sequence Γ^x for the inputs $x = (x_1, \cdots, x_n)$ is the concatenation of the input commitments $\mathrm{Com}(x_1), \cdots, \mathrm{Com}(x_n)$ and commitments $\mathrm{Com}(t_{n+1}), \cdots, \mathrm{Com}(t_{n+q})$ representing the truth tables of gates:

$$\Gamma^x = \underbrace{\boxed{?}\boxed{?}}_{x_1} \cdots \underbrace{\boxed{?}\boxed{?}}_{x_n} \underbrace{\overbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}^{\text{24 cards}}}_{t_{n+1}} \cdots \underbrace{\overbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}^{\text{24 cards}}}_{t_{n+q}}\,.$$

We note that $\mathrm{Com}(t_{n+1}), \cdots, \mathrm{Com}(t_{n+q})$ can be produced in *public* since all truth tables are publicly known.

Before presenting our construction, we define some notations. A *position* is defined by a subset of indices $[2n + 24q]$. (Note that $2n + 24q$ is the number of cards in Γ^x.) For $i \in [n]$, the *position of the i-th input commitment* is defined by $P_i = \{2i - 1, 2i\}$. Let $g \in \mathsf{Gates}$ be index of a gate. (Recall that $\mathsf{Gates} = \{n+1, \cdots, n+q\}$.) For $j \in [12]$, the *position of the j-th commitment in the gate g* is defined by $P_j^{(g)} = \{p_j^{(g)}, p_j^{(g)} + 1\}$, where $p_j^{(g)} = 2n + 24(g - (n + 1)) + 2j - 1$. For $k \in [4]$, the *position of the k-th row in the gate g* is defined by $\tilde{P}_k^{(g)} = P_{3k-2}^{(g)} \cup P_{3k-1}^{(g)} \cup P_{3k}^{(g)}$. Let $w \in \mathsf{Wires}$ be index of a wire. (Recall that $\mathsf{Wires} = \{1, \cdots, n+q\}$.) The *position of the first cards corresponding to the left (resp. right) wire* w is defined by $P_L^{(w)} = \{p_j^{(g)} : g \in L^{-1}(w), j \in \{1,4,7,10\}\}$ (resp. $P_R^{(w)} = \{p_j^{(g)} : g \in R^{-1}(w), j \in \{2,5,8,11\}\}$). The *position of the first cards corresponding to the wire* w is defined by

$$P_{\text{first}}^{(w)} = \begin{cases} \{2w-1\} \cup P_L^{(w)} \cup P_R^{(w)} & \text{if } 1 \le w \le n \\ \{p_j^{(w)} : j \in \{3,6,9,12\}\} \cup P_L^{(w)} \cup P_R^{(w)} & \text{otherwise.} \end{cases}$$

Similarly, the *position of the second cards corresponding to the wire* w is defined by $P_{\text{second}}^{(w)} = \{j+1 : j \in P_{\text{first}}^{(w)}\}$.

Example 2 *Let $C = (2,1,1,L,R,G)$ be an AND circuit such that $L(3) = 1$, $R(3) = 2$, and $G_3(x_1,x_2) = x_1 \wedge x_2$. The initial sequence $\Gamma^{x_1x_2}$ is given by*

$$\Gamma^{x_1x_2} = \underbrace{\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}}_{x_1}\underbrace{\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}}_{x_2}\;\underbrace{\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}}_{0}\underbrace{\overset{7}{\boxed{?}}\overset{8}{\boxed{?}}}_{0}\underbrace{\overset{9}{\boxed{?}}\overset{10}{\boxed{?}}}_{0}\;\underbrace{\overset{11}{\boxed{?}}\overset{12}{\boxed{?}}}_{0}\underbrace{\overset{13}{\boxed{?}}\overset{14}{\boxed{?}}}_{1}\underbrace{\overset{15}{\boxed{?}}\overset{16}{\boxed{?}}}_{0}\;\underbrace{\overset{17}{\boxed{?}}\overset{18}{\boxed{?}}}_{1}\underbrace{\overset{19}{\boxed{?}}\overset{20}{\boxed{?}}}_{0}\underbrace{\overset{21}{\boxed{?}}\overset{22}{\boxed{?}}}_{0}\;\underbrace{\overset{23}{\boxed{?}}\overset{24}{\boxed{?}}}_{1}\underbrace{\overset{25}{\boxed{?}}\overset{26}{\boxed{?}}}_{1}\underbrace{\overset{27}{\boxed{?}}\overset{28}{\boxed{?}}}_{1}.$$

Here the numbers (from 1 *to* 28*) in the upper side denote positions of cards. The positions of the 1st, 2nd, 3rd, and 4th rows in the gate* 3 *are $\tilde{P}_1^{(3)} = \{5,6,7,8,9,10\}$, $\tilde{P}_2^{(3)} = \{11,12,13,14,15,16\}$, $\tilde{P}_3^{(3)} = \{17,18,19,20,21,22\}$, and $\tilde{P}_4^{(3)} = \{23,24,25,26,27,28\}$, respectively. The positions of the first cards corresponding to the left wire* 1 *and the right wire* 2 *are $P_L^{(1)} = \{5,11,17,23\}$ and $P_R^{(2)} = \{7,13,19,25\}$, respectively. The positions of the first cards corresponding to the wires* 1*,* 2*, and* 3 *are $P_{\text{first}}^{(1)} = \{1,5,11,17,23\}$, $P_{\text{first}}^{(2)} = \{3,7,13,19,25\}$, and $P_{\text{first}}^{(3)} = \{9,15,21,27\}$, respectively.*

Our garbled circuit construction proceeds as follows.

Protocol 1 (Card-based Garbled Circuit)

Garbling *Given an initial sequence Γ^x, it proceeds as follows:*

1. *For every $g \in \mathsf{Gates}$, perform $(\mathsf{pileShuffle}, \tilde{P}_1^{(g)}, \tilde{P}_2^{(g)}, \tilde{P}_3^{(g)}, \tilde{P}_4^{(g)})$.*
2. *For every $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$, perform $(\mathsf{pileShuffle}, P_{\text{first}}^{(w)}, P_{\text{second}}^{(w)})$.*
3. *Output the current sequence as the gabled sequence.*

Evaluation *Given a gabled sequence, it proceeds as follows:*

1. *For every $i \in \mathsf{Inputs}$, perform (turn, P_i), and define $x'_i \in \{0,1\}$ by the outcome of the turning cards.*
2. *For every gate $g \in \mathsf{Gates}$ (in order from $n+1$ to $n+q$), perform the following:*

   (a) For every $i \in \{1,2,4,5,7,8,10,11\}$, perform $(\mathsf{turn}, P_i^{(g)})$, and

define $a_i \in \{0,1\}$ by the outcome of the turning cards.

(b) *Define an index $k_g \in \{3,6,9,12\}$ as follows:*

(A) *If $(a_1, a_2) = (x'_{L(g)}, x'_{R(g)})$, then define $k_g = 3$.*

(B) *Else if $(a_4, a_5) = (x'_{L(g)}, x'_{R(g)})$, then define $k_g = 6$.*

(C) *Else if $(a_7, a_8) = (x'_{L(g)}, x'_{R(g)})$, then define $k_g = 9$.*

(D) *Otherwise define $k_g = 12$.*

(c) *If $g \le n+q-m$, perform $(\mathsf{turn}, P^{(g)}_{k_g})$, and define $x'_g \in \{0,1\}$ by the outcome of the turning cards.*

3. *Perform $(\mathsf{result}, P^{(\alpha_1)}_{k_{\alpha_1}}, \cdots, P^{(\alpha_m)}_{k_{\alpha_m}})$, where $\alpha_i = n+q-m+i$ for $i \in [m]$.*

Example 3 *Before proving the correctness and the security of our construction, we give an example of an execution of it for the circuit given in Example 2. In Step 1 of the garbling stage, a pile-scramble shuffle is performed for every $g \in \mathsf{Gates}$. In this example, $\mathsf{Gates} = \{3\}$. For $g = 3$, for $\tilde{P}^{(g)}_1 = \{5,6,7,8,9,10\}$ ("$\bullet$" group in the following), $\tilde{P}^{(g)}_2 = \{11,12,13,14,15,16\}$ ("$\circ$" group), $\tilde{P}^{(g)}_3 = \{17,18,19,20,21,22\}$ ("$\star$" group), $\tilde{P}^{(g)}_4 = \{23,24,25,26,27,28\}$ ("$\diamond$" group).*

$$\begin{array}{cccccccccccccccccccccccccccc}
1&2&3&4&5&6&7&8&9&10&11&12&13&14&15&16&17&18&19&20&21&22&23&24&25&26&27&28\\
\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}\\
&&&&\bullet&\bullet&\bullet&\bullet&\bullet&\bullet&\circ&\circ&\circ&\circ&\circ&\circ&\star&\star&\star&\star&\star&\star&\diamond&\diamond&\diamond&\diamond&\diamond&\diamond
\end{array}.$$

In Step 2 of the garbling stage, a pile-scramble shuffle is performed for each wire $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$. In this example, $\mathsf{Wires} \setminus \mathsf{Outputs} = \{1,2\}$. For $w = 1$ (corresponding to the input wire of x_1), a pile-scramble shuffle is performed for $P^{(w)}_{\text{first}} = \{1,5,11,17,23\}$ ("$\bullet$" group in the following), $P^{(w)}_{\text{second}} = \{2,6,12,18,24\}$ ("$\circ$" group).

$$\begin{array}{cccccccccccccccccccccccccccc}
1&2&3&4&5&6&7&8&9&10&11&12&13&14&15&16&17&18&19&20&21&22&23&24&25&26&27&28\\
\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}\\
\bullet&\circ&&&\bullet&\circ&&&&&\bullet&\circ&&&&&\bullet&\circ&&&&&\bullet&\circ&&&&
\end{array}.$$

Similarly, for $w = 2$ (corresponding to the input wire of x_2), a pile-scramble shuffle is performed for $P^{(w)}_{\text{first}} = \{3,7,13,19,25\}$ ("$\bullet$" group), $P^{(w)}_{\text{second}} = \{4,8,14,20,26\}$ ("$\circ$" group).

$$\begin{array}{cccccccccccccccccccccccccccc}
1&2&3&4&5&6&7&8&9&10&11&12&13&14&15&16&17&18&19&20&21&22&23&24&25&26&27&28\\
\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}\\
&&\bullet&\circ&&&\bullet&\circ&&&&&\bullet&\circ&&&&&\bullet&\circ&&&&&\bullet&\circ&&
\end{array}.$$

For every $i \in \mathsf{Inputs}$, perform (turn, P_i), and define $x'_i \in \{0,1\}$ by the outcome of the turning cards.

In Step 1 of the evaluation stage, turning operations (turn, P_i) for every $i \in \mathsf{Inputs}$ are performed as follows:

$$\begin{array}{cccccccccccccccccccccccccccc}
1&2&3&4&5&6&7&8&9&10&11&12&13&14&15&16&17&18&19&20&21&22&23&24&25&26&27&28\\
\boxed{\heartsuit}&\boxed{\clubsuit}&\boxed{\clubsuit}&\boxed{\heartsuit}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}&\boxed{?}
\end{array}.$$

The above sequence shows one *possible outcome. In this case,* $x'_1 = 1$ *and* $x'_2 = 0$. *In Step 2 (a), turning operations* $(\mathsf{turn}, P^{(g)}_i)$ *for every* $i \in \{1, 2, 4, 5, 7, 8, 10, 11\}$ *are performed as follows:*

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
♡	♣	♣	♡	♡	♣	♡	♣	?	?	♣	♡	♡	♣	?	?	♡	♣	♣	♡	?	?	♣	♡	♣	♡	?	?.

In this case, $a_1 = 1, a_2 = 1, a_4 = 0, a_5 = 1, a_7 = 1, a_8 = 0, a_{10} = 0,$ *and* $a_{11} = 0$. *In Step 2 (b), the index* k_g *is defined. In this case,* k_g *is defined to be* 9 *because* $(a_7, a_8) = (x'_1, x'_2) = (1, 0)$. *In this example, Step 2 (c) is skipped because it holds that* $g\,(= 3) > n + q - m\,(= 2)$. *In Step 3, the commitment at position* $P^{(g)}_{k_g}$ *is outputted. In this case,* $P^{(g)}_{k_g} = P^{(g)}_9 = \{21, 22\}$ *as follows:*

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
♡	♣	♣	♡	♡	♣	♡	♣	?	?	♣	♡	♡	♣	?	?	♡	♣	♣	♡	?	?	♣	♡	♣	♡	?	?.
																				●	●						

Correctness We show the correctness of the above protocol. In Step 1 of the garbling stage, for each gate $g \in \mathsf{Gates}$, a pile-scramble shuffle is performed over the four sets of positions $\tilde{P}^{(g)}_1, \tilde{P}^{(g)}_2, \tilde{P}^{(g)}_3$ and $\tilde{P}^{(g)}_4$. It just permutes the four rows in the truth table t_g because the cards on $\tilde{P}^{(g)}_i$ are the commitments to the i-th row in the truth table. In other words, it preserves the functionality of the truth table. In Step 2 of the garbling stage, for each wire $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$, a pile-scramble shuffle is performed over two positions $P^{(w)}_{\mathrm{first}}$ and $P^{(w)}_{\mathrm{second}}$. Recall that the position $P^{(w)}_{\mathrm{first}}$ (resp. $P^{(w)}_{\mathrm{second}}$) designates the first (resp. second) cards of the commitments corresponding to the wire w. Thus, it is equivalent to masking the values of the commitments by an independent and uniformly random value $r_w \in \{0, 1\}$. Therefore, after applying it, each row $(a, b, G_g(a, b))$ corresponding to the gate g turns out to be $(a \oplus r_{L(g)}, b \oplus r_{R(g)}, G_g(a, b) \oplus r_g)$. Finally, we conclude the proof by showing that the evaluation stage works correctly. If $(a_1, a_2) = (x'_{L(g)}, x'_{R(g)})$ (i.e. it is the case (A) in Step 2(b) of the evaluation stage), then the first row is $(a_1, a_2, G_g(a_1 \oplus r_{L(g)}, a_2 \oplus r_{R(g)}) \oplus r_g)$; thus the value of the k_g-th (in this case $k_g = 3$) commitment is $G_g(a_1 \oplus r_{L(g)}, a_2 \oplus r_{R(g)}) \oplus r_g = G_g(x'_{L(g)} \oplus r_{L(g)}, x'_{R(g)} \oplus r_{R(g)}) \oplus r_g = G_g(x_{L(g)}, x_{R(g)}) \oplus r_g$, where $x_{L(g)}$ and $x_{R(g)}$ are the intermediate values corresponding to the wires $L(g)$ and $R(g)$, respectively. Similarly, we can observe that all four cases in Step 2(b) of the evaluation stage work correctly. Thus, in any of the final gates g whose output is an output bit of the circuit, the value of the k_g-th commitments is the correct output $G_g(x_{L(g)}, x_{R(g)})$. (Recall that the output wires are not masked by randomness in Step 2 of the garbling stage.) Therefore, the above protocol correctly computes the circuit.

Security In order to prove the security, we show that for any input distribution, a random variable of the visible sequence and a random variable of the input sequence are stochastically independent. We can observe that the random variable of the visible sequence is essentially equivalent to a random variable of the *outcome* obtained by turning operations in Steps 1, 2(a) and 2(c) of the evaluation stage. This is because the visible sequence (resp. the outcome) is efficiently computable from the outcome (resp. the visible sequence). Thus, it is sufficient to show that the random variable of the outcome and the random variable of the input sequence are stochastically independent. This follows from the fact that the outcome distribution is computable without knowing the input. We construct such a "simulator" as follows. First, it generates, for each gate g, a uniformly random element π_g from S_4. Then, the four tuple of outcomes $(a_1, a_2), (a_4, a_5), (a_7, a_8), (a_{10}, a_{11})$ is set to $(0, 0), (0, 1), (1, 0), (1, 1)$ according to the permutation π_g, i.e., if $\pi_g(1) = 1$ then $(a_1, a_2) = (0, 0)$, if $\pi_g(2) = 4$ then $(a_4, a_5) = (1, 1)$ and so on. It can be seen that the distribution of the simulated outcome is the same as the distribution of the real outcome due to the shuffles applied in the garbling stage. Thus, the random variable of the outcome and the random variable of the input sequence are stochastically independent. Therefore, the protocol is secure.

4 Our Protocol with One Shuffle

In this section, we construct our protocol with one shuffle. The key observation is that all shuffles in Protocol 1 can be aggregated into one shuffle because they are applied successively.

For every $g \in \mathsf{Gates}$, there exists a closed set of permutations $\Pi_1^{(g)}$ such that

$$(\mathsf{shuffle}, \Pi_1^{(g)}) = (\mathsf{pileShuffle}, \tilde{P}_1^{(g)}, \tilde{P}_2^{(g)}, \tilde{P}_3^{(g)}, \tilde{P}_4^{(g)}).$$

(Recall that $\mathsf{pileShuffle}$ is a sugar syntax of $\mathsf{shuffle}$.) Similarly, for every $w \in \mathsf{Wires} \setminus \mathsf{Outputs}$, there exists a closed set of permutations $\Pi_2^{(w)}$ such that

$$(\mathsf{shuffle}, \Pi_2^{(w)}) = (\mathsf{pileShuffle}, P_{\text{first}}^{(w)}, P_{\text{second}}^{(w)}).$$

Define Π to be the following set of permutations:

$$\Pi = \{\pi_2^{(n+q-m)} \circ \cdots \circ \pi_2^{(1)} \circ \pi_1^{(n+q)} \circ \cdots \circ \pi_1^{(n+1)} : \pi_1^{(g)} \in \Pi_1^{(g)}, \pi_2^{(w)} \in \Pi_2^{(w)}\},$$

where "$\circ$" is the composition of permutations. We can observe that Π is closed under the composition of permutations. This is because a permutation $\pi_1^{(g)} \in \Pi_1^{(g)}$ corresponds to permuting four input-output pairs of the truth table t_g, a permutation $\pi_2^{(w)} \in \Pi_2^{(w)}$ corresponds to masking values of the wire w by a random bit, and they are commutative.

By aggregating all shuffles in Protocol 1 into a single shuffle $(\mathsf{shuffle}, \Pi)$, our main protocol is obtained.

Protocol 2 (Protocol with One Shuffle)

1. Arrange a sequence to be the initial sequence Γ^x as in Protocol 1.

2. *Apply* (shuffle, Π) *to the sequence.*

3. *Perform the evaluation stage as in Protocol 1.*

The number of cards is $2n + 24q$. The correctness and security are easily derived from those of Protocol 1.

5 Batching Technique

In this section, we introduce a new technique, the *batching technique*, which converts multiple pile-scramble shuffles that are executable in parallel into a single pile-scramble shuffle. First, we observe the simplest case such that two pile-scramble shuffles are executable in parallel.

Suppose that we wish to perform two pile-scramble shuffle: one is between "$\bullet$" and "$\circ$" (two piles), and the other is among "$\star$", "$\diamond$" and "$*$" (three piles).

$$\underset{\bullet}{\boxed{?}}\underset{\bullet}{\boxed{?}}\underset{\bullet}{\boxed{?}}\,\underset{\circ}{\boxed{?}}\underset{\circ}{\boxed{?}}\underset{\circ}{\boxed{?}}\quad\underset{\star}{\boxed{?}}\underset{\star}{\boxed{?}}\underset{\star}{\boxed{?}}\,\underset{\diamond}{\boxed{?}}\underset{\diamond}{\boxed{?}}\underset{\diamond}{\boxed{?}}\,\underset{*}{\boxed{?}}\underset{*}{\boxed{?}}\underset{*}{\boxed{?}}.$$

The batching technique enables to combine them into a *single* pile-scramble shuffle by using additional cards. First, two clubs and three hearts are inserted in the sequence as follows:

$$\boxed{\clubsuit}\underset{\bullet}{\boxed{?}}\underset{\bullet}{\boxed{?}}\underset{\bullet}{\boxed{?}}\,\boxed{\clubsuit}\underset{\circ}{\boxed{?}}\underset{\circ}{\boxed{?}}\underset{\circ}{\boxed{?}}\quad\boxed{\heartsuit}\underset{\star}{\boxed{?}}\underset{\star}{\boxed{?}}\underset{\star}{\boxed{?}}\,\boxed{\heartsuit}\underset{\diamond}{\boxed{?}}\underset{\diamond}{\boxed{?}}\underset{\diamond}{\boxed{?}}\,\boxed{\heartsuit}\underset{*}{\boxed{?}}\underset{*}{\boxed{?}}\underset{*}{\boxed{?}}.$$

Then, the inserted cards are turned to be face-down cards. Then, a pile-scramble shuffle among "$\bullet$", "$\circ$", '$\star$", "$\diamond$" and "$*$" (five piles) is performed as follows.

$$\underset{\bullet}{\boxed{?}}\underset{\bullet}{\boxed{?}}\underset{\bullet}{\boxed{?}}\underset{\bullet}{\boxed{?}}\,\underset{\circ}{\boxed{?}}\underset{\circ}{\boxed{?}}\underset{\circ}{\boxed{?}}\underset{\circ}{\boxed{?}}\quad\underset{\star}{\boxed{?}}\underset{\star}{\boxed{?}}\underset{\star}{\boxed{?}}\underset{\star}{\boxed{?}}\,\underset{\diamond}{\boxed{?}}\underset{\diamond}{\boxed{?}}\underset{\diamond}{\boxed{?}}\underset{\diamond}{\boxed{?}}\,\underset{*}{\boxed{?}}\underset{*}{\boxed{?}}\underset{*}{\boxed{?}}\underset{*}{\boxed{?}}.$$

After applying it, open the first cards of all piles as follows:

$$\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}.$$

(The above figure shows an example of outcomes of the turning operation.) Finally, rearrange five piles so as to the former two piles have $\clubsuit$ and the latter piles have $\heartsuit$. In this case, the fourth pile (the underlined pile in the following) is moved to the front of the second pile without changing the order of cards in each pile.

$$\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\,\underline{\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}.$$

By ignoring opened cards (two $\clubsuit$ and three $\heartsuit$), the result sequence in the following is equivalent (as probabilistic distribution) to the result sequence obtained by applying two pile-scramble shuffle sequentially.

$$\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\,\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}.$$

This is the core idea of our batching technique.

Now we explain the batching technique in general case. Suppose that we wish to perform N pile-scramble shuffles: the i-th pile-scramble shuffle is among ℓ_i piles of n_i cards. (Thus, the i-th shuffle treats $\ell_i \cdot n_i$ cards and there are $\sum_{i=1}^{N} \ell_i \cdot n_i$ cards in total.) Suppose that they are executable in parallel. Let $\sigma : [N] \to \{\clubsuit, \heartsuit\}^{\lceil \log_2 N \rceil}$ be an arbitrary injective function. The batching technique proceeds as follows.

1. (**Indexing**) For every $i \in [N]$, insert $\lceil \log_2 N \rceil$ cards representing $\sigma(i)$ to each pile in the i-th shuffle as follows.

   $$\cdots \qquad \underbrace{\boxed{?}\boxed{?}\boxed{?}\ \boxed{?}\boxed{?}\boxed{?}}_{\text{corresponding to the } i\text{-th shuffle}} \qquad \cdots \longrightarrow \cdots \underbrace{\boxed{?}\boxed{?}}_{\sigma(i)}\boxed{?}\boxed{?}\boxed{?}\ \underbrace{\boxed{?}\boxed{?}}_{\sigma(i)}\boxed{?}\boxed{?}\boxed{?} \cdots$$

   (In total, $\sum_{i=1}^{N} \ell_i \cdot \lceil \log_2 N \rceil$ cards are inserted.)

2. (**Padding**) If $n_1 = \cdots = n_N$, skip this step. Otherwise, each pile of the i-th shuffle is appended with $(n_{\max} - n_i)$ cards, where $n_{\max} = \max(n_1, \cdots, n_N)$.

3. (**Shuffle**) Perform a pile-scramble shuffle among all piles of $(\lceil \log_2 N \rceil + n_{\max})$ cards. (Note that the number of piles is $\sum_{i=1}^{N} \ell_i$.)

4. (**Turning**) Turn the indexes of all piles. (In total, $\sum_{i=1}^{N} \ell_i \cdot \lceil \log_2 N \rceil$ cards are turned.)

5. (**Rearrangement**) Rearrange all the cards so as to the first ℓ_1 piles are those having the index $\sigma(1)$, the next ℓ_2 piles are those having the index $\sigma(2)$, and so on. Finally, the inserted cards in the Indexing step are removed. (They can be used in future steps as free cards.)

The total number Δ of additional cards for the batching technique is:

$$\Delta = \sum_{i=1}^{N} \ell_i \cdot \left(\lceil \log_2 N \rceil + n_{\max} - n_i \right).$$

6 Our Protocol with Two Pile-Scramble Shuffles

In this section, we construct our protocol with two shuffles. It is obtained by applying the batching technique to our garbled circuit construction.

Protocol 3 (Protocol with Two Shuffles)

1. *Arrange a sequence to be the initial sequence Γ^x as in Protocol 1.*

2. *Apply the batching technique for all shuffles in Step 1 of the garbling stage in Protocol 1.*

3. *Apply the batching technique for all shuffles in Step 2 of the garbling stage in Protocol 1.*

4. *Perform the evaluation stage as in Protocol 1.*

We note that in the former batching technique, the Padding step is not needed because all piles are having the same number of cards. The number Δ_1 of additional cards in the former batching technique is $\Delta_1 = 4q\lceil \log_2 q \rceil$. The number Δ_2 of additional cards in the latter batching technique is:

$$\Delta_2 = 2(n+q-m)\lceil \log_2(n+q-m) \rceil + \sum_{w=1}^{n+q-m} 2 \cdot (n_{\max} - |P_{\text{first}}^{(w)}|)$$

where $n_{\max} = \max_{1 \leq w \leq n+q-m} |P_{\text{first}}^{(w)}|$. Because the additional cards used in the former batching are used in the latter batching, the number of additional cards is $\max(\Delta_1, \Delta_2)$. Thus, the total number M of cards used in our two-shuffle protocol is $M = 2n + 24q + \max(\Delta_1, \Delta_2)$.

Correctness The correctness follows from that of our two-round protocol and the fact that the batching techniques do essentially the same as Steps 1 and 2 of our two-round protocol.

Security The security proof is similar to the security proof of our two-round protocol. The only difference is that the simulator has to generate the outcome obtained by the batching technique (Steps 1 and 2). This part of the simulation can be easily done. The other part of the simulation is the same as that of our two-round protocol. Thus, the protocol is secure.

Acknowledgments

The authors would like to thank members of the study group "Shin-Akarui-Angou-Benkyou-Kai" for the valuable discussions and helpful comments. Among them, the authors specially thanks Goichiro Hanaoka for his worthful suggestions on this study. The first author was supported during this work by JSPS KAKENHI Grant Number 17J01169, Japan. The second author was supported during this work by JST CREST Grant Number JPMJCR14D6, Japan.

References

[1] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pp. 784–796, 2012.

[2] C. Crépeau and J. Kilian. Discreet solitary games. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference,*

Santa Barbara, California, USA, August 22-26, 1993, Proceedings, pp. 319–330, 1993.

[3] B. 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*, pp. 208–217, 1989.

[4] Y. Hashimoto, K. Shinagawa, K. Nuida, M. Inamura, and G. Hanaoka. Secure grouping protocol using a deck of cards. In *Information Theoretic Security - 10th International Conference, ICITS 2017, Hong Kong, China, November 29 - December 2, 2017, Proceedings*, pp. 135–152, 2017.

[5] R. Ishikawa, E. Chida, and T. Mizuki. Efficient card-based protocols for generating a hidden random permutation without fixed points. In *Unconventional Computation and Natural Computation - 14th International Conference, UCNC 2015, Auckland, New Zealand, August 30 - September 3, 2015, Proceedings*, pp. 215–226, 2015.

[6] J. Kastner, A. Koch, S. Walzer, D. Miyahara, Y. Hayashi, T. Mizuki, and H. 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, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, pp. 126–155, 2017.

[7] A. Koch and S. Walzer. Foundations for actively secure card-based cryptography. *IACR Cryptology ePrint Archive*, 2017:423, 2017.

[8] A. Koch, S. Walzer, and K. 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, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, pp. 783–807, 2015.

[9] T. Mizuki. Efficient and secure multiparty computations using a standard deck of playing cards. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, pp. 484–499, 2016.

[10] T. Mizuki, I. K. Asiedu, and H. 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*, pp. 162–173, 2013.

[11] T. Mizuki, M. Kumamoto, and H. Sone. The five-card trick can be done with four cards. In *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*, pp. 598–606, 2012.

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

[13] T. Mizuki and H. Shizuya. Practical card-based cryptography. In *Fun with Algorithms - 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1-3, 2014. Proceedings*, pp. 313–324, 2014.

[14] T. Mizuki and H. 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*, pp. 358–369, 2009.

[15] T. Mizuki, F. Uchiike, and H. Sone. Securely computing XOR with 10 cards. *The Australasian Journal of Combinatorics*, 36:279–293, 2006.

[16] T. Nakai, S. Shirouchi, M. Iwamoto, and K. Ohta. Four cards are sufficient for a card-based three-input voting protocol utilizing private permutations. In *Information Theoretic Security - 10th International Conference, ICITS 2017, Hong Kong, China, November 29 - December 2, 2017, Proceedings*, pp. 153–165, 2017.

[17] T. Nakai, Y. Tokushige, Y. Misawa, M. Iwamoto, and K. Ohta. Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, pp. 500–517, 2016.

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

[19] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone. Card-based protocols for any boolean function. In *Theory and Applications of Models of Computation - 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings*, pp. 110–121, 2015.

[20] T. Nishida, T. Mizuki, and H. Sone. Securely computing the three-input majority function with eight cards. In *Theory and Practice of Natural Computing - Second International Conference, TPNC 2013, Cáceres, Spain, December 3-5, 2013, Proceedings*, pp. 193–204, 2013.

[21] A. Nishimura, Y. Hayashi, T. Mizuki, and H. Sone. An implementation of non-uniform shuffle for secure multi-party computation. In *Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography, AsiaPKC@AsiaCCS, Xi'an, China, May 30 - June 03, 2016*, pp. 49–55, 2016.

[22] A. Nishimura, T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone. Five-card secure computations using unequal division shuffle. In *Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings*, pp. 109–120, 2015.

[23] K. Shinagawa, T. Mizuki, J. C. N. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto. Multi-party computation with small shuffle complexity using regular polygon cards. In *Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings*, pp. 127–146, 2015.

[24] K. Shinagawa, T. Mizuki, J. C. N. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto. Secure multi-party computation using polarizing cards. In *Advances in Information and Computer Security - 10th International Workshop on Security, IWSEC 2015, Nara, Japan, August 26-28, 2015, Proceedings*, pp. 281–297, 2015.

[25] K. Shinagawa, T. Mizuki, J. C. N. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto. Secure computation protocols using polarizing cards. *IEICE Transactions*, 99-A(6):1122–1131, 2016.

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

[27] I. Ueda, A. Nishimura, Y. Hayashi, T. Mizuki, and H. Sone. How to implement a random bisection cut. In *Theory and Practice of Natural Computing - 5th International Conference, TPNC 2016, Sendai, Japan, December 12-13, 2016, Proceedings*, pp. 58–69, 2016.

[28] A. C. Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pp. 162–167, 1986.