An Implementation of Non-Uniform Shuffle for Secure Multi-Party Computation*

Akihiro Nishimura
Graduate School of Information Sciences, Tohoku University
6-3 Aramaki-Aza-Aoba, Aoba, Sendai 980-8578, Japan

Yu-ichi Hayashi
Faculty of Engineering, Tohoku Gakuin University
1-13-1 Chuo, Tagajo, Miyagi 985-8537, Japan

Takaaki Mizuki
Cyberscience Center, Tohoku University
6-3 Aramaki-Aza-Aoba, Aoba, Sendai 980-8578, Japan
tm-paper+cardshimw[at]g-mail.tohoku-university.jp

Hideaki Sone
Cyberscience Center
Tohoku University
6-3 Aramaki-Aza-Aoba, Aoba, Sendai 980-8578, Japan

ABSTRACT

Card-based cryptographic protocols provide secure multi-party computation using a deck of physical cards. The most important primitive of those protocols is the shuffling operation, and most known protocols rely on uniform shuffles (such as the random cut and random bisection cut) in which each possible outcome is equally likely. However, several protocols with non-uniform shuffles have recently been proposed by Koch et al. Compared to other protocols, their protocols require fewer cards to securely produce a hidden AND value, although implementation of the non-uniform shuffle appearing in their protocols remains an open problem. This paper presents a secure implementation of their non-uniform shuffle. To implement the shuffle, we utilize physical cases that can store piles of cards, such as boxes and envelopes. Therefore, humans are able to perform the non-uniform shuffle using these everyday objects.

Keywords

Card-based cryptographic protocols; Secure multi-party computations; Real-life hand-on cryptography

1. INTRODUCTION

Card-based cryptographic protocols provide secure multi-party computation using a deck of physical cards. A protocol consists of a combination of rearranging, shuffling, and turning-over operations for a sequence of cards along with some encoding scheme. Many protocols have been proposed to perform AND computation (e.g. [1, 2, 4, 6, 8, 12]), XOR computation (e.g. [1, 6, 7]), and any Boolean function (e.g. [9]). In particular, there has been considerable research on designing a committed AND protocol that produces a hidden AND value. Table 1 shows some of the known committed AND protocols. For example, on one hand, the protocol proposed in [12] requires eight cards and uses a random cut, in which the card sequence is randomly shifted; on the other hand, the protocol proposed in [6] requires six cards and uses a random bisection cut, in which the card sequence is bisected into two piles that are then randomly switched. Thus, the number of required cards differs depending on individual protocols. Specifically, shuffling operations play an important role in designing efficient protocols; that is, the number of required cards could be decreased by adopting other shuffling operations.

Actually, a four-card committed AND protocol was recently proposed by Koch et al. at Asiacrypt 2015 [3]. They succeeded in reducing the number of cards required to securely generate a committed AND value through the use of an unconventional shuffle operation, namely a *non-uniform* shuffle. In this technique, they consider a shuffle operation that has a non-uniform probability distribution. However, finding a feasible implementation (for humans) of the non-uniform shuffle remains an open problem [3].

This paper provides a solution to the problem; i.e., we present a secure implementation of the non-uniform shuffle appearing in their protocols. To implement the shuffle, we use physical cases that can store a pile of cards, such as boxes and envelopes. Therefore, it is possible for humans to perform the shuffle using such everyday objects.

The remainder of this paper is organized as follows. The rest of this section presents several notations related to card-based cryptographic protocols. Section 2 introduces the two AND protocols proposed by Koch et al. at Asiacrypt 2015 [3]. Section 3 describes thechniques to utilize the cases that we assume. Section 4 proposes a general description of our implementation. Section 5 explicitly shows that the AND protocols by Koch et al. [3] can be executed by humans. Section 6 concludes our paper.

*This is the Accepted Version.

AsiaPKC'16, May 30-29 2016, Xi'an, China

ISBN 978-1-4503-4286-5/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2898420.2898425

Table 1: Committed AND protocol

Reference	# of colors	# of cards	Type of shuffle	Avg. # of trials
[1]	4	10	RC	6
[8]	2	12	RC	2.5
[12]	2	8	RC	2
[6]	2	6	RBC	1

RC: Random Cut, RBC: Random Bisection Cut

1.1 Preliminary Notations

Here, we provide preliminary notations regarding card-based cryptographic protocols.

First, we assume that cards satisfy the following properties.

1. All cards of the same type (black $\boxed{\clubsuit}$ or red $\boxed{\heartsuit}$) are indistinguishable from one another.
2. The back side of each card $\boxed{?}$ is the same, and hence, all face-down cards are indistinguishable from one another.

Next, the encoding scheme to handle Boolean values is defined as:

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

Given a bit $x \in \{0, 1\}$, when a pair of face-down cards $\boxed{?}\boxed{?}$ describes the value of x with encoding scheme (1), it is called a *commitment* to x and is expressed as

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

Given commitments to $a, b \in \{0, 1\}$, the *committed* AND protocols listed in Table 1 produce their output as a commitment to $a \wedge b$.

1.2 Computational Model

Here, we briefly introduce the computational model and the method of describing a protocol [3, 5].

A tuple

$$\Gamma = (\alpha_1, \alpha_2, \cdots, \alpha_d)$$

represents a *sequence* of d cards for some $d \in \mathbf{N}$ where α_i is a face-down or face-up card, $1 \leq i \leq d$. The operations that can be applied in a protocol are defined below.

Turning Over

Given a *turn set* $T \subseteq \{1, 2, \cdots, d\}$ for a sequence $\Gamma = (\alpha_1, \alpha_2, \cdots, \alpha_d)$, we write (turn, T) to express a *turning-over* operation where all cards whose positions in T are turned over. For example, given a sequence of two cards

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

operation $(\mathsf{turn}, \{1, 2\})$ results in

$$\underbrace{\boxed{?}\boxed{?}}_{0},$$

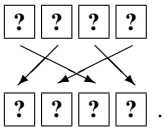
which is a commitment to 0.

Rearrangement

By S_d, we denote the symmetric group of degree d. Given a permutation $\pi \in S_d$ for a sequence $\Gamma = (\alpha_1, \alpha_2, \cdots, \alpha_d)$, we write (perm, π) to express a *rearrangement* operation after which we have the resulting sequence:

$$(\alpha_{\pi^{-1}(1)}, \alpha_{\pi^{-1}(2)}, \cdots, \alpha_{\pi^{-1}(d)}).$$

Consider cyclic permutation (1 3 4 2) as an example. For a sequence of four face-down cards, operation $(\mathsf{perm}, (1\,3\,4\,2))$ works as



Shuffling

Given a probability distribution $\mathcal{F}$ on a permutation set $\Pi \subseteq S_d$ for a sequence $\Gamma = (\alpha_1, \alpha_2, \cdots, \alpha_d)$, we write

$$(\mathsf{shuffle}, \Pi, \mathcal{F})$$

to represent a *shuffling* operation that probabilistically applies (perm, π) to Γ where π is chosen from Π according to $\mathcal{F}$.

Hereafter, id denotes the identify permutation. For example, for two commitments

$$\underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{b},$$

operation

$$(\mathsf{shuffle}, \{\mathsf{id}, (1\,3)(2\,4)\}, \mathsf{id} \mapsto 1/2, (1\,3)(2\,4) \mapsto 1/2)$$

results in either

$$\underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{b} \text{ or } \underbrace{\boxed{?}\boxed{?}}_{b}\underbrace{\boxed{?}\boxed{?}}_{a}$$

where each occurs with a probability of exactly 1/2. Such a shuffling operation is called a random bisection cut [6].

The example above is a type of "uniform" shuffle; i.e., possible permutations id and $(1\,3)(2\,4)$ are uniformly distributed. In this case, we simply write $(\mathsf{shuffle}, \{\mathsf{id}, (1\,3)(2\,4)\})$. Generally, we write $(\mathsf{shuffle}, \Pi)$ instead of $(\mathsf{shuffle}, \Pi, \mathcal{F})$ if $\mathcal{F}$ is a uniform distribution.

The random bisection cut is a popular shuffling operation, and used in many known protocols (e.g. [3, 4, 6, 9]). Another popular type of uniform shuffle is the random cut, which means cyclic shifting; for example,

$$(\mathsf{shuffle}, \{\mathsf{id}, (1\,2\,3), (1\,3\,2)\}).$$

In the next section, we will describe a "non-uniform" shuffle.

Result

The operation (result, i, j) implies that the i-th and j-th cards constitute the output commitment.

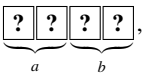
2. KOCH'S AND PROTOCOLS USING NON-UNIFORM SHUFFLE

This section introduces the descriptions of the AND protocols proposed by Koch et al [3]. They proposed two AND protocols: the first is a four-card Las Vegas protocol, and the second is a five-card

finite-runtime protocol. Hereafter, we refer to the former as Koch's four-card AND protocol and the latter as Koch's five-card AND protocol; these are described in Sections 2.1 and 2.2, respectively.

2.1 Koch's Four-Card Las Vegas AND Protocol

Given two commitments



Koch's four-card AND protocol produces a commitment to $a \wedge b$. That is, it requires only four cards to perform a committed AND computation. It contains a "cycle" because of the "goto" operations, and hence, it is a Las Vegas algorithm.

Koch's four card AND protocol.

(shuffle, {id, (1 3)(2 4)})
(shuffle, {id, (2 3)})
(turn, {2})
if (visible seq. is ? ♣ ? ?) then
  (turn, {2})
  (shuffle, {id, (1 3)})
• (shuffle, {id, (1 2)(3 4)}, id ↦ 1/3, (1 2)(3 4) ↦ 2/3)
  (turn, {4})
  if (visible seq. is ? ? ? ♣) then
    (result, 1, 2)
  else if (visible seq. is ? ? ? ♡) then
    (turn, {4})
    (shuffle, {id, (1 3)})
    (perm, (1 3 4 2))
    goto ∘
else if (visible seq. is ? ♡ ? ?) then
  turn,{2})
  (shuffle, {id, (3 4)})
∘ (shuffle, {id, (1 3)(2 4)}, id ↦ 1/3, (1 3)(2 4) ↦ 2/3)
  (turn, {1})
  if (visible seq. is ♡ ? ? ?) then
    (result, 2, 4)
  else if (visible seq. is ♣ ? ? ?) then
    (turn, {1})
    (shuffle, {id, (3 4)})
    (perm, (1 2 4 3))
    goto •

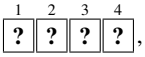
As seen above, this protocol relies on two types of non-uniform shuffle:

$$(\mathsf{shuffle}, \mathsf{id} \mapsto 1/3, (1\,2)(3\,4) \mapsto 2/3) \tag{2}$$

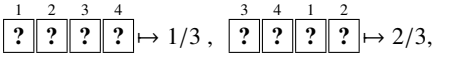
and

$$(\mathsf{shuffle}, \mathsf{id} \mapsto 1/3, (1\,3)(2\,4) \mapsto 2/3), \tag{3}$$

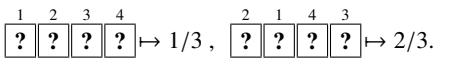
where we omit the descriptions of permutation sets $\{\mathsf{id}, (1\,2)(3\,4)\}$ and $\{\mathsf{id}, (1\,3)(2\,4)\}$. (Hereafter, we simply write $(\mathsf{shuffle}, \mathcal{F})$ instead of $(\mathsf{shuffle}, \Pi, \mathcal{F})$ as above if $\mathcal{F}$ is a non-uniform distribution.) The latter shuffle (3) means, given



we have



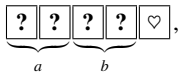
and hence, it is a sort of random bisection cut although the two possible outcomes are not uniformly distributed. The former shuffle (2) means



Because an implementation of these shuffling operations has not appeared in the literature, we will provide a solution in Sections 4 and 5.

2.2 Koch's Five-Card Finite-Runtime AND Protocol

Koch et al. slightly modified the four-card protocol to create a five-card finite-runtime AND protocol. Using five cards, it allows a "break-out" of the cycle, that is, given



the protocol proceeds as follows.

Koch's five-card AND protocol.

(turn, {5})
(shuffle, {id, (1 3)(2 4)})
(shuffle, {id, (2 3)})
(turn, {2})
if (visible seq. is ? ♣ ? ? ?) then
  (turn, {2})
  (shuffle, {id, (1 3)})
• (perm, (1 5 2 4))
  (shuffle, id ↦ 2/3, (5 4 3 2 1) ↦ 1/3)
  (turn, {5})
  if (visible seq. is ? ? ? ? ♣) then
    (result, 4, 3)
  else if (visible seq. is ? ? ? ? ♡) then
    (result, 3, 1)
else if (visible seq. is ? ♡ ? ? ?) then
  (turn,{2})
  (shuffle, {id, (3 4)})
  (shuffle, id ↦ 1/3, (1 3)(2 4) ↦ 2/3)
  (turn, {1})
  if (visible seq. is ♡ ? ? ? ?) then
    (result, 2,4)
  else if (visible seq. is ♣ ? ? ? ?) then
    (turn, {1})
    (shuffle, {id, (3 4)})
    (perm, (1 2 4 3))
    goto •

The first non-uniform shuffle in this protocol is

$$(\mathsf{shuffle}, \mathsf{id} \mapsto 2/3, (5\,4\,3\,2\,1) \mapsto 1/3). \tag{4}$$

On the other hand, the second non-uniform shuffle already appears in Koch's four card AND protocol presented in the previous subsection.

As mentioned above, finding an implementation of the non-uniform shuffle remains an open problem. We will solve it in the succeeding sections.

3. CARD CASES FOR NON-UNIFORM SHUFFLE

In this section, we discuss how to implement a non-uniform shuffle through the use of "card cases."

3.1 How to Utilize Cases

Nishimura et al. considered the card cases shown in Figure 1 to implement unequal division shuffles [10]. We also utilize the same type of cases to implement non-uniform shuffles. Each case can store a deck of cards and has two sliding covers: an upper cover and a lower cover. We assume that the weight of a deck of cards is negligible compared to the case.

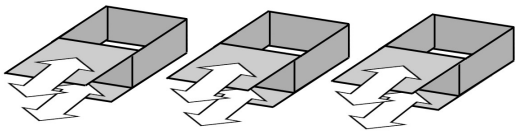


Figure 1: Card cases having two sliding covers

Now, consider the non-uniform shuffle (3) appearing in Koch's four-card AND protocol (Section 2.1); remember that, given a sequence of four face-down cards

$$\overset{1}{?}\,\overset{2}{?}\,\overset{3}{?}\,\overset{4}{?},$$

we must obtain

$$\overset{1}{?}\,\overset{2}{?}\,\overset{3}{?}\,\overset{4}{?} \mapsto 1/3\,, \quad \overset{3}{?}\,\overset{4}{?}\,\overset{1}{?}\,\overset{2}{?} \mapsto 2/3.$$

To implement this, we use three cases, referred to as C_1, C_2, and C_3 for the sake of convenience.

1. Divide the sequence into two piles: $\overset{1}{?}\,\overset{2}{?}$ and $\overset{3}{?}\,\overset{4}{?}$.
2. Store the first pile in C_1, store the second pile in C_3, and leave C_2 empty, as illustrated in Figure 2.
3. Apply a "random cut" to the case sequence (C_1, C_2, C_3); in other words, randomly shift these three cases. We call such an operation a *pile-shifting scramble*.
4. Stack the three cases as illustrated in Figure 3.
5. Remove all sliding covers except for the top-most and bottom-most covers simultaneously, as illustrated in Figure 4.

Then, we have a sequence of four face-down cards. Table 2 shows all three possibilities of case sequences together with card sequences and probabilities.

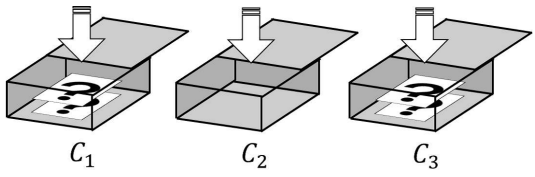


Figure 2: The first and second piles are stored in the first and third cases, respectively.

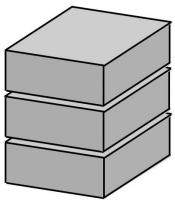


Figure 3: Stacking the three cases

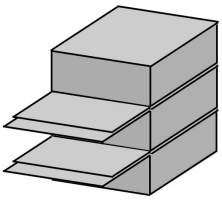


Figure 4: Removing the covers

Table 2: All possibilities to confirm shuffle (3)

Case sequence	Card sequence	Probability
(C_1, C_2, C_3)	$\overset{1}{?}\,\overset{2}{?}\,\overset{3}{?}\,\overset{4}{?}$	1/3
(C_3, C_1, C_2)	$\overset{3}{?}\,\overset{4}{?}\,\overset{1}{?}\,\overset{2}{?}$	1/3
(C_2, C_3, C_1)	$\overset{3}{?}\,\overset{4}{?}\,\overset{1}{?}\,\overset{2}{?}$	1/3

From Table 2, one can easily confirm that the non-uniform shuffle

$$\overset{1}{?}\,\overset{2}{?}\,\overset{3}{?}\,\overset{4}{?} \mapsto 1/3\,, \quad \overset{3}{?}\,\overset{4}{?}\,\overset{1}{?}\,\overset{2}{?} \mapsto 2/3$$

has been implemented.

3.2 Necessary Properties for Cases

To be used in the pile-shifting scramble, cases must have certain properties. Thus, we assume physical cases that satisfy the following properties.

1. It is possible to store a pile of cards in a case without changing its order.
2. It is possible to eject a pile of cards from a case without changing its order.
3. It is possible to eject multiple piles without changing their orders by opening multiple cases at the same time. No information leaks will be caused by this action.
4. A number of cases that possibly contain piles of cards are indistinguishable from one another, and we cannot obtain any information about the cards inside. (For instance, the weight of any pile of cards is negligible compared to the weight of the case. Hence, the weights of these cases leak no information about the internal states. Furthermore, any case containing a pile is assumed to make no sound.)

We believe we can use boxes or envelopes as such cases.

4. NON-UNIFORM SHUFFLE VIA PILE-SHIFTING SCRAMBLE

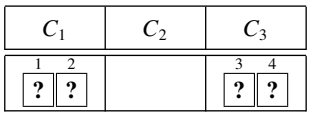
This section proposes a general method to implement a non-uniform shuffle using the pile-shifting scramble.

4.1 Idea

As seen in Section 3.1, a non-uniform shuffle

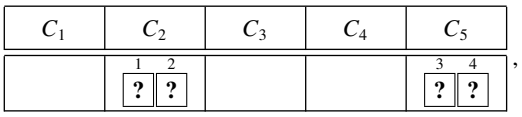
$$(\mathsf{shuffle}, \mathsf{id} \mapsto 1/3, (1\,3)(2\,4) \mapsto 2/3)$$

can be implemented by using three cases:



.

As can be easily noticed, the number of cases and the initial positions of the piles determine the probabilities. For example, if we have five cases and store the piles as



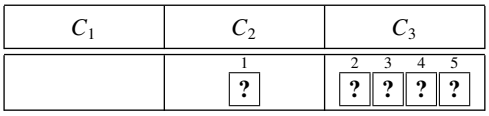
,

then another type of non-uniform shuffle

$$(\mathsf{shuffle}, \mathsf{id} \mapsto 2/5, (1\,3)(2\,4) \mapsto 3/5)$$

can be implemented.

Furthermore, the pile sizes should not necessarily be the same; for instance, consider a one-card pile and a four-card pile along with three cases:



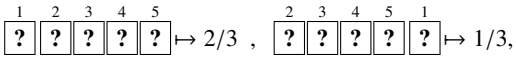
.

Then, after a pile-shifting scramble (randomly shifting the three cases), we have

Case sequence	Card sequence	Probability
(C_1, C_2, C_3)	1 2 3 4 5 ? ? ? ? ?	1/3
(C_3, C_1, C_2)	2 3 4 5 1 ? ? ? ? ?	1/3
(C_2, C_3, C_1)	1 2 3 4 5 ? ? ? ? ?	1/3

.

Therefore, the non-uniform shuffle

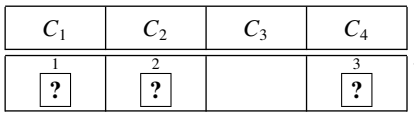


namely

$$(\mathsf{shuffle}, \mathsf{id} \mapsto 2/3, (5\,4\,3\,2\,1) \mapsto 1/3)$$

can be implemented. Actually, this is an exact representation of shuffle (4), which appears in Section 2.2.

Although the card sequence is split into two piles in each of these examples, we can, of course, split it into three or more piles; for instance,



,

achieves

$$(\mathsf{shuffle}, \mathsf{id} \mapsto 1/4, (1\,2\,3) \mapsto 1/4, (2\,3\,1) \mapsto 1/2),$$

which is a "non-uniform" random cut.

In the next subsection, we generalize this idea.

4.2 The Power of a Pile-Shifting Scramble

Here, we discuss the shuffle types that can be implemented by the pile-shifting scramble.

Basically, the pile-shifting scramble achieves a "pile-based" random cut; i.e., if we split a sequence of d cards into k piles where the size of the i-th pile is s_i, then the permutation set of the shuffle consists of k permutations:

$$\left(\begin{array}{c|c|c|c|c|c|c} 1 & \cdots & s_1 & s_1+1 & \cdots & s_1+s_2 & \cdots \\ 1 & \cdots & s_1 & s_1+1 & \cdots & s_1+s_2 & \cdots \end{array} \begin{array}{c|c|c} d-s_k+1 & \cdots & d \\ d-s_k+1 & \cdots & d \end{array}\right)^{-1},$$

$$\left(\begin{array}{ccc|c|ccc|c|ccc} 1 & \cdots & s_k & s_k+1 & \cdots & s_k+s_1 & \cdots & d-s_{k-1}+1 & \cdots & d \\ d-s_k+1 & \cdots & d & 1 & \cdots & s_1 & \cdots & d-s_k-s_{k-1}+1 & \cdots & d-s_k \end{array}\right)^{-1},$$

$$\vdots$$

$$\left(\begin{array}{ccc|c|ccc|c|ccc} 1 & \cdots & s_2 & s_2+1 & \cdots & s_2+s_3+1 & \cdots & d-s_1+1 & \cdots & d \\ s_1+1 & \cdots & s_1+s_2 & s_1+s_2+1 & \cdots & s_1+s_2+s_3 & \cdots & 1 & \cdots & s_1 \end{array}\right)^{-1}.$$

We denote the permutation set defined above by $\Pi^{(s_1,s_2,\cdots,s_k)}$.

If we use exactly k cases to store k piles, we obtain a uniform shuffle $(\mathsf{shuffle}, \Pi^{(s_1,s_2,\cdots,s_k)})$ [i]. As seen in the previous subsection, by preparing empty cases, we are able to obtain a non-uniform shuffle. We now discuss a general treatment.

Assume that we want to implement a non-uniform shuffle

$$(\mathsf{shuffle}, \Pi^{(s_1,s_2,\cdots,s_k)}, \mathcal{F})$$

such that the probabilities of distribution $\mathcal{F}$ are

$$\frac{p_1}{q}, \frac{p_2}{q}, \cdots, \frac{p_k}{q},$$

where each probability is a non-zero rational number and the greatest common divisor of p_i for $1 \le i \le k$ is relatively prime to q. Note that

$$\sum_{i=1}^{k} p_i = q.$$

Then, we require q cases that satisfy the properties mentioned in Section 3.2. The cases are named $C_1, C_2, \cdots, C_q$. The case sequence $C = (C_1, C_2, \cdots, C_q)$ consists of $C_1, C_2, \cdots, C_q$. The j-th pile is stored in case C_{ℓ_j}, where $1 \le j \le k$, according to the following condition:

$$\ell_j = \sum_{i=1}^{j} p_i.$$

[i] Such a shuffling operation is called a cyclic shuffle in [11].

This means that the first pile is stored in the p_1-th case, the second pile is stored in the $(p_1 + p_2)$-th case, and so on. The last pile, namely the k-th pile, is stored in the q-th case (namely, the last case). The other cases are left empty.

After applying a pile-shifting scramble to case sequence C, there are q possibilities for the case sequence; each case sequence is generated with a probability of exactly $1/q$. Then, all sliding covers (except for the top-most and bottom-most covers) are removed to obtain the resulting sequence of cards.

Among the q possibilities mentioned above, there are p_1 possibilities that the sequence starts with the first pile (and ends with the k-th pile). More generally, there are p_j possibilities that the sequence starts with the j-th pile for every j, $1 \leq j \leq k$. Therefore, the probability that the j-th pile leads the sequence is exactly p_j/q, as desired.

Thus, our implementation can be applied if the probability of each possible outcome is a non-zero rational number.

THEOREM 1. *Let $s_1, s_2, \cdots, s_k$ be integers. Any shuffle*

$$(\mathsf{shuffle}, \Pi^{(s_1,s_2,\cdots,s_k)}, \mathcal{F})$$

can be implemented by the pile-shifting scramble if every probability in $\mathcal{F}$ is a non-zero rational number.

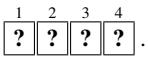
5. REAL EXECUTION OF KOCH'S PROTOCOLS

In this section, we demonstrate how to practically implement Koch's AND protocols, which were introduced in Sections 2.1 and 2.2.

Note that, aside from non-uniform shuffles (2), (3), and (4), the known protocols could be executed by humans. Therefore, our non-uniform shuffle implementation will complete the implementation of their AND protocols.

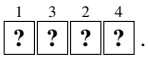
Because we have already shown the realization of shuffles (3) and (4) in Section 4.1, it suffices to focus on shuffle (2), which results in:

It appears that shuffle (2) is not a"pile-based" random cut, but it is implemented by performing additional permutations before and after the pile-shifting scramble. First, apply $(\mathsf{perm}, (2\,3))$ to the sequence



.

Then, we have



.

Next, apply shuffle (3) to the sequence. Finally, apply $(\mathsf{perm}, (2\,3))$ to the sequence.

Table 3 confirms that shuffle (2) is surely implemented.

6. CONCLUSION

In this paper, we proposed a method to securely implement a non-uniform shuffle. That is, we showed that a non-uniform distribution on possible shuffling outcomes can be generated by considering physical cases that could hold a pile of cards. As such physical cases, we can use everyday objects such as boxes and envelopes. Therefore, we believe that humans can practically perform a non-uniform shuffle. Our non-uniform shuffle implementation can be, of course, applied to the two AND protocols designed by Koch et al. [3] (which are the first two protocols that rely on non-uniform shuffles); this means that their excellent protocols can be practically executed by humans using an actual deck of physical cards.

We hope that our implementation will motivate further research on developing new card-based protocols using non-uniform shuffles.

Table 3: All possibilities to confirm shuffle (2)

Case sequence	Card sequence	after (perm,(2 3))	Prob.
(C_1, C_2, C_3)	1 3 2 4 ? ? ? ?	1 2 3 4 ? ? ? ?	1/3
(C_3, C_1, C_2)	2 4 1 3 ? ? ? ?	2 1 4 3 ? ? ? ?	1/3
(C_2, C_3, C_1)	2 4 1 3 ? ? ? ?	2 1 4 3 ? ? ? ?	1/3

7. REFERENCES

[1] C. Crépeau and J. Kilian. Discreet solitary games. In D. R. Stinson, editor, *Advances in Cryptology — CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 319–330. Springer Berlin Heidelberg, 1994.

[2] B. den Boer. More efficient match-making and satisfiability: the five card trick. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 208–217. Springer Berlin Heidelberg, 1990.

[3] A. Koch, S. Walzer, and K. Härtel. Card-based cryptographic protocols using a minimal number of cards. In T. Iwata and J. Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, volume 9452 of *Lecture Notes in Computer Science*, pages 783–807. Springer Berlin Heidelberg, 2015.

[4] T. Mizuki, M. Kumamoto, and H. Sone. The five-card trick can be done with four cards. In X. Wang and K. Sako, editors, *Advances in Cryptology — ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 598–606. Springer Berlin Heidelberg, 2012.

[5] T. Mizuki and H. Shizuya. A formalization of card-based cryptographic protocols via abstract machine. *International Journal of Information Security*, 13(1):15–23, 2014.

[6] T. Mizuki and H. Sone. Six-card secure AND and four-card secure XOR. In X. Deng, J. E. Hopcroft, and J. Xue, editors, *Frontiers in Algorithmics*, volume 5598 of *Lecture Notes in Computer Science*, pages 358–369. Springer Berlin Heidelberg, 2009.

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

[8] V. Niemi and A. Renvall. Secure multiparty computations without computers. *Theoretical Computer Science*, 191(1–2):173–183, 1998.

[9] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone. Card-based protocols for any boolean function. In R. Jain, S. Jain, and F. Stephan, editors, *Theory and Applications of Models of Computation*, volume 9076 of *Lecture Notes in Computer Science*, pages 110–121. Springer International Publishing, 2015.

[10] A. Nishimura, T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone. Five-card secure computations using unequal division shuffle. In A.-H. Dediu, L. Magdalena, and

C. Martín-Vide, editors, *Theory and Practice of Natural Computing*, volume 9477 of *Lecture Notes in Computer Science*, pages 109–120. Springer International Publishing, 2015.

[11] K. Shinagawa, T. Mizuki, J. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto. Multi-party computation with small shuffle complexity using regular polygon cards. In M.-H. Au and A. Miyaji, editors, *Provable Security*, volume 9451 of *Lecture Notes in Computer Science*, pages 127–146. Springer International Publishing, 2015.

[12] A. Stiglic. Computations with a deck of cards. *Theoretical Computer Science*, 259(1–2):671–678, 2001.

Acknowledgments

We thank the anonymous referees, whose comments helped us to improve the presentation of the paper. This work was supported by JSPS KAKENHI Grant Number 26330001.