How to Covertly and Uniformly Scramble the 15 Puzzle and Rubik's Cube

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

Kazuki Kanai
National Institute of Technology, Kure College, Hiroshima, Japan

Kengo Miyamoto
Ibaraki University, 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

A combination puzzle is a puzzle consisting of a set of pieces that can be rearranged into various combinations, such as the 15 Puzzle and Rubik's Cube. Suppose a speedsolving competition for a combination puzzle is to be held. To make the competition fair, we need to generate an instance (i.e., a state having a solution) that is chosen uniformly at random and unknown to anyone. We call this problem a secure random instance generation of the puzzle. In this paper, we construct secure random instance generation protocols for the 15 Puzzle and for Rubik's Cube. Our method is based on uniform cyclic group factorizations for finite groups, which is recently introduced by the same authors, applied to permutation groups for the puzzle instances. Specifically, our protocols require 19 shuffles for the 15 Puzzle and 43 shuffles for Rubik's Cube.

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

Keywords and phrases Card-based cryptography, Uniform cyclic group factorization, Secure random instance generation, The 15 Puzzle, Rubik's Cube

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.30

Funding *Kazumasa Shinagawa*: This work was supported in part by JSPS KAKENHI Grant Numbers 21K17702 and 23H00479, and JST CREST Grant Number MJCR22M1.
Kengo Miyamoto: This work was supported in part by JSPS KAKENHI Grant Numbers 20K14302 and 23H00479.
Koji Nuida: This work was supported in part by JSPS KAKENHI Grant Number JP19H01109.

1 Introduction

1.1 Secure Random Instance Generation Problem

A *combination puzzle* is a puzzle that consists of a set of pieces that can be rearranged into various combinations, such as the 15 Puzzle and Rubik's Cube. Suppose we want to hold a speedsolving competition for a combination puzzle. For the competition, we need to generate an *instance* (i.e., a state having a solution) of the puzzle. It must be chosen uniformly at random from all instances, since otherwise some players may predict the instance. In an actual speedsolving competition, an instance is chosen randomly by a computer program, and a person called the *scrambler* applies the corresponding scrambling procedure to the puzzle. An obvious drawback of this method is that the scrambler, who should know the chosen instance, cannot fairly participate in the competition. One solution to this problem could be to use a robot that generates a uniformly random instance of the puzzle, but verifying the correctness of the robot's behavior is not easy in general.

From this background, we need to generate an instance of the puzzle that satisfies the following conditions:

- It is chosen uniformly at random from all instances.
- It is hidden from all players, *including the scrambler*, until the competition starts.
- It is arranged by human hands without electronic devices (such as a robot).

We call this problem a *secure random instance generation problem* of the puzzle.

For the 15 Puzzle, the problem might seem to be trivial at first glance, because the structure of an ordinary 15 Puzzle board allows taking out the 15 blocks, turning them face-down, and scramble them randomly. However, there is actually an issue; a completely random arrangement of blocks may have no solution. Precisely[1], it is well-known that when the blocks are arranged according to a permutation σ in the symmetric group S_{15}, a solution exists if and only if σ is in the alternating group A_{15}, which happens only with probability 1/2 for a uniformly random σ. Therefore, we need to scramble the blocks in a way that only permutations in A_{15} can appear, which is a non-trivial task.

For Rubik's Cube, similar to the 15 Puzzle, we need to covertly and uniformly generate a permutation of the *Rubik's Cube group* R, which is a subgroup of the 48th symmetric group S_{48}, but the situation is more complicated than for the 15 Puzzle (not just because the group R is more complicated than A_{15}). First, unlike the 15 Puzzle, each piece of the cube cannot be "face-down." Second, since all pieces are mechanically connected, applying a permutation to the cube is a non-trivial task.

In this paper, we consider the secure random instance generation problem for the first time and solve the problem for the 15 Puzzle and Rubik's Cube. For both puzzles, we need to covertly and uniformly generate a permutation from a finite group. A similar problem has been studied in the research area of *card-based cryptography.*

1.2 Card-Based Cryptography

Card-based cryptography [3, 5, 14] is a research area within cryptography, which is based on physical cards; cryptographic protocols such as secure computation protocols and zero-knowledge proof protocols are implemented by a deck of cards without electronic devices.

A *shuffle* is an operation to rearrange a card sequence covertly and randomly. Formally, a shuffle for a sequence of n cards is defined by a pair of a set of permutations $\Pi \subseteq S_n$, where S_n denotes the n-th symmetric group, and a probability distribution $\mathcal{F}$ on Π, which is denoted by $(\mathsf{shuffle}, \Pi, \mathcal{F})$. For a sequence $\vec{c} := (c_1, c_2, \ldots, c_n)$ of face-down cards, it chooses a permutation $\pi \in \Pi$ according to $\mathcal{F}$ covertly, and rearranges it into the sequence $\pi(\vec{c}) = (c_{\pi^{-1}(1)}, c_{\pi^{-1}(2)}, \ldots, c_{\pi^{-1}(n)})$. Here, it is assumed that no player (including the player who actually operates the shuffle operation) can guess which permutation π was chosen beyond the fact that it was chosen according to $\mathcal{F}$. A shuffle $(\mathsf{shuffle}, \Pi, \mathcal{F})$ is said to be *uniform* if $\mathcal{F}$ is the uniform distribution on Π, *closed* if Π is a subgroup of S_n, and *uniform closed* if both conditions hold. Hereinafter, for a uniform shuffle $(\mathsf{shuffle}, \Pi, \mathcal{F})$, we write it as $(\mathsf{shuffle}, \Pi)$ and call it the Π-*shuffle.*

Although the definition of shuffles allows for an arbitrary permutation set Π and an arbitrary distribution $\mathcal{F}$, how to physically implement a shuffle $(\mathsf{shuffle}, \Pi, \mathcal{F})$ given Π and $\mathcal{F}$ is quite non-trivial. In the literature on card-based cryptography, five shuffles – a *random cut*, a *random bisection cut*, a *pile-shifting shuffle*, a *complete shuffle*, and a *pile-scramble shuffle*

[1] We assume that an instance of the 15 Puzzle always has the empty square placed at the bottom right (as well as the default instance of the puzzle), therefore an arrangement of the puzzle corresponds to a permutation in S_{15} rather than S_{16}.

– are considered easy to implement. This is because the first three shuffles are performed by random cyclic shifting and the last two shuffles are performed by completely random scrambling. (See also Ueda et al. [23] for how to implement a random cut, a random bisection cut, and a pile-shifting shuffle.) In this paper, we call these five shuffles *practical shuffles*.

From this context, there is a line of research to implement a class of shuffles from practical shuffles with the use of "helping cards". Saito et al. [20] showed that any shuffle $(\mathsf{shuffle}, \Pi, \mathcal{F})$ can be implemented by three practical shuffles (one random cut, one pile-shifting shuffle, and one pile-scramble shuffle) with helping cards if every probability of $\mathcal{F}$ is a rational number. Although it is very general and important work, it requires at least $n \cdot |\Pi|$ helping cards; it is inefficient in terms of the number of cards when $|\Pi|$ is large. Another important example of previous work is the result by Koch [11]. He showed that any uniform closed shuffle $(\mathsf{shuffle}, \Pi)$ can be implemented by pile-shifting shuffles. However, it requires $O(|\Pi|)$ shuffles; it is inefficient in terms of the number of shuffles when $|\Pi|$ is large. In summary, up until now, there is no efficient implementation of a shuffle when $|\Pi|$ is large.

1.3 Our Contribution

In this paper, we propose physical protocols for secure random instance generation of the 15 Puzzle and Rubik's Cube. Let G be the permutation group corresponding to the set of all instances of the puzzle. We divide the problem into the following problems:

- How to implement a G-shuffle from practical shuffles?
- How to apply a G-shuffle to the puzzle physically?

For the first problem, we utilize a decomposition of a finite group into cyclic subgroups called a *uniform cyclic group factorization* (UCF), which is recently proposed by the authors [10] (see Section 3.1). We show that if a group G has a UCF of length k, a G-shuffle can be implemented by a sequence of k *cyclic group shuffles*, which are C-shuffles for some cyclic group C, with no additional cards. This is an efficient method to implement a G-shuffle from practical shuffles since k is considerably smaller than $|G|$. Up until now, it is known that all solvable groups have a UCF [10] but whether any finite group has a UCF or not is still open. Fortunately, the alternating group and the Rubik's Cube group R have a UCF. Based on the UCF, the A_{15}-shuffle and the R-shuffle can be implemented by a sequence of 19 and 43 cyclic group shuffles, respectively. In addition, these cyclic group shuffles are practical shuffles: random cuts, random bisection cuts, and pile-shifting shuffles. Note that since $|A_{15}| = 653837184000$ and $|R| = 43252003274489856000$, the existing methods [11, 20] require a large number of cards or a large number of shuffles at least these numbers.

For the 15 Puzzle, the second problem is almost trivial: Turn the blocks face-down and apply the A_{15}-shuffle to the sequence of face-down blocks just like a sequence of cards. Thus, the secure random instance generation problem of the 15 Puzzle is solved.

For Rubik's Cube, the second problem is more complicated than that of the 15 Puzzle because there are two obstacles: (1) since each piece of the cube cannot be "face-down", how to hide an instance of the cube is non-trivial; (2) since all pieces are mechanically connected, how to apply a permutation to the cube is non-trivial. To solve (1) and (2), we propose two methods for the construction of a secure random instance generation of Rubik's Cube.

The first solution requires *color stickers*, which are concealed by peelable *films*: The faces of the cube are hidden by stickers and films, as a solution to (1), and a permutation is applied to a sequence of color stickers instead of the cube, as a solution to (2). The protocol proceeds as follows. First, an R-shuffle is applied to a sequence of color stickers. Then, each sticker is placed on each face of the cube. Finally, just before the competition starts, all films are peeled off. We call it a protocol in the *free permutation model* since it applies a permutation freely. The protocol flow is summarized as Figure 1.

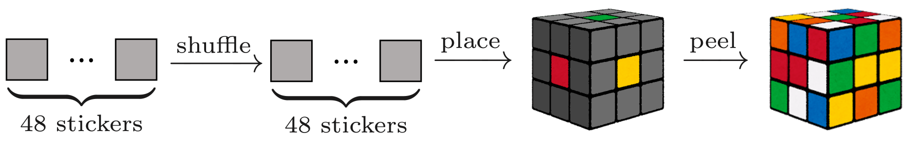


Figure 1 The flow of our protocol for Rubik's Cube in the free permutation model.

The second solution requires a piece of cloth: The cube is covered by a piece of cloth, as a solution to (1), and standard operations of Rubik's Cube (i.e., $\mathsf{F}, \mathsf{B}, \mathsf{U}, \mathsf{D}, \mathsf{L}$, and R) are applied from under the cloth, as a solution to (2). The protocol proceeds as follows. First, cover the solved cube by a large piece of cloth. Then, apply a sequence of standard operations repeatedly until the player who scrambles the puzzle cannot remember how many times it has been repeated (just like a *Hindu cut* of the playing card) from under the cloth. Finally, just before the competition starts, the cloth is removed. We call it a protocol in the *restricted permutation model* since the permutations applied to the puzzle is restricted to the standard operations.

In summary, we propose three protocols for secure random instance generation of the 15 Puzzle and Rubik's Cube. For the 15 Puzzle, we propose a protocol with 19 practical shuffles (7 random cuts, 7 random bisection cuts, and 5 pile-shifting shuffles). For Rubik's Cube, our protocol in the free permutation model applies 43 practical shuffles (22 random bisection cuts and 21 pile-shifting shuffles) to the sequence of color stickers, and one in the restricted permutation model applies a sequence of standard operations to the cube directly.

1.4 Related Work

Mathematics of the 15 Puzzle. It is NP-hard to determine whether an instance of the generalized 15 Puzzle can be solved at most k moves for a given integer k [17,18]. It is shown that the length of shortest solutions ranges from 0 to 80 single-tile moves [1] or 43 multi-tile moves [16]. For the generalized 15 Puzzle with an $n \times n$ puzzle board, it is shown that the asymptotic mixing time is $O(n^4 \log n)$ when random moves are made [2].

Mathematics of Rubik's Cube. It is NP-complete to determine whether an instance of the generalized $n \times n \times n$ Rubik's Cube can be solved at most k moves for a given integer k [4]. It is shown that the minimum number of steps required to solve any instance of Rubik's Cube called *God's Number* is 20 [19].

Card-Based Cryptography. There is a line of research to implement somewhat complicated shuffles from practical shuffles. There are several studies on generating a *derangement*, which is a permutation without fixed points, covertly and uniformly at random [3,6,8,9]. It can be seen as a uniform shuffle whose permutation set is the set of all derangements. Hashimoto et al. [7] proposed a *secure grouping protocol* that generates a random permutation with some conditions, which has an application to the Werewolf game. Also, it can be seen as a kind of uniform shuffle. Miyamoto and Shinagawa [12,21] constructed a protocol for implementing *graph shuffles*, which is a class of uniform closed shuffles whose permutation set is the automorphism group of a graph, from pile-scramble shuffles.

Secure Computation Using the 15 Puzzle. Mizuki, Kugimoto, and Sone [13] showed that secure computation can be done by the use of the 15 Puzzle. In particular, they showed that any 4-variable Boolean function and any 14-variable Boolean symmetric function are securely computed by using the 15 Puzzle.

2 Preliminaries

2.1 Notations

Throughout this paper, any groups are assumed to be finite. We denote the n-th symmetric group by S_n and the n-th alternating group by A_n. We assume to multiply permutations from left to right, e.g., $(1,2)(1,3) = (1,2,3)$. Note that it is the convention used by the GAP and the community of Rubik's Cube.

2.2 Shuffle

A *shuffle* is an operation that rearranges a sequence of cards covertly and randomly (see also Section 1.2). In Sections 4 and 5, we use random cuts, random bisection cuts, and pile-shifting shuffles. These shuffles are special cases of cyclic group shuffles.

Cyclic Group Shuffle. A *cyclic group shuffle* is a G-shuffle for some cyclic group $G = \langle g \rangle$. Let k be the order of G. Then it applies a permutation g^r to a sequence of cards, where $r \in \{0, 1, \ldots, k-1\}$ is chosen uniformly at random. For example, applying a cyclic group shuffle $(\mathsf{shuffle}, \langle g \rangle)$, where $g = (1,2)(3,4,5,6) \in S_6$, to a sequence of six cards yields the following result:

$$\vec{c} = \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \longmapsto \pi(\vec{c}) = \begin{cases} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\ \overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} & \text{if } \pi = g^0 \text{ (with prob. } 1/4); \\ \overset{2}{\boxed{?}}\overset{1}{\boxed{?}}\ \overset{6}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}} & \text{if } \pi = g^1 \text{ (with prob. } 1/4); \\ \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\ \overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}} & \text{if } \pi = g^2 \text{ (with prob. } 1/4); \\ \overset{2}{\boxed{?}}\overset{1}{\boxed{?}}\ \overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\overset{3}{\boxed{?}} & \text{if } \pi = g^3 \text{ (with prob. } 1/4). \end{cases}$$

Random Cut. A *random cut* [5] (of length k) is a cyclic group shuffle whose generator g is a conjugate of $(1, 2, \ldots, k) \in S_n$, i.e., there exists a permutation $\tau \in S_n$ such that $g = \tau^{-1}(1, 2, \ldots, k)\tau$. For example, applying a random cut $(\mathsf{shuffle}, \langle g \rangle)$, where $g = (2,1,4)$, to a sequence of four cards yields the following result:

$$\vec{c} = \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}} \longmapsto \pi(\vec{c}) = \begin{cases} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}} & \text{if } \pi = g^0 \text{ (with prob. } 1/3); \\ \overset{2}{\boxed{?}}\overset{4}{\boxed{?}}\overset{3}{\boxed{?}}\overset{1}{\boxed{?}} & \text{if } \pi = g^1 \text{ (with prob. } 1/3); \\ \overset{4}{\boxed{?}}\overset{1}{\boxed{?}}\overset{3}{\boxed{?}}\overset{2}{\boxed{?}} & \text{if } \pi = g^2 \text{ (with prob. } 1/3). \end{cases}$$

Random Bisection Cut. A *random bisection cut* [15] (of size ℓ) is a cyclic group shuffle whose generator is a conjugate of $(1, \ell+1)(2, \ell+2) \cdots (\ell, 2\ell) \in S_n$. For example, applying a random bisection cut $(\mathsf{shuffle}, g\rangle)$, where $g = (1,4)(2,5)(3,6)$, to a sequence of six cards yields the following result:

$$\vec{c} = \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \longmapsto \pi(\vec{c}) = \begin{cases} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\ \overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} & \text{if } \pi = g^0 \text{ (with prob. } 1/2); \\ \overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\ \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}} & \text{if } \pi = g^1 \text{ (with prob. } 1/2). \end{cases}$$

Pile-Shifting Shuffle. A *pile-shifting shuffle* [22] (of size ℓ with k piles) is a cyclic group shuffle whose generator g is a conjugate of the following permutation:

$$\vec{c} = (1, \ell{+}1, 2\ell{+}1, \ldots, (k{-}1)\ell{+}1)(2, \ell{+}2, 2\ell{+}2, \ldots, (k{-}1)\ell{+}2) \cdots (\ell, 2\ell, 3\ell, \ldots, k\ell) \in S_n.$$

For example, applying a pile-shifting shuffle $(\mathsf{shuffle}, \langle g \rangle)$, where $g = (1,3,5)(2,4,6)$, to a sequence of six cards yields the following result:

$$\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} \longmapsto \pi(\vec{c}) = \begin{cases} \overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\;\overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\;\overset{5}{\boxed{?}}\overset{6}{\boxed{?}} & \text{if } \pi = g^0 \text{ (with prob. } 1/3); \\ \overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\;\overset{1}{\boxed{?}}\overset{2}{\boxed{?}}\;\overset{3}{\boxed{?}}\overset{4}{\boxed{?}} & \text{if } \pi = g^1 \text{ (with prob. } 1/3); \\ \overset{3}{\boxed{?}}\overset{4}{\boxed{?}}\;\overset{5}{\boxed{?}}\overset{6}{\boxed{?}}\;\overset{1}{\boxed{?}}\overset{2}{\boxed{?}} & \text{if } \pi = g^2 \text{ (with prob. } 1/3). \end{cases}$$

3 Uniform Cyclic Group Factorization and Its Application to Shuffle

In this section, we recall the notion of uniform cyclic group factorizations of a finite group [10] and show that if a group G has a uniform cyclic group factorization, the G-shuffle can be implemented by a sequence of cyclic group shuffles with no additional cards.

3.1 Uniform Cyclic Group Factorization

Let G be a group. Let $\mathcal{H} = (H_1, H_2, \ldots, H_k)$ be an ordered tuple of subsets of G. Define the multiplication map $\mathsf{mult}_{\mathcal{H}} : H_1 \times H_2 \times \cdots \times H_k \to G$ by $\mathsf{mult}_{\mathcal{H}}(h_1, h_2, \ldots, h_k) := h_1 h_2 \cdots h_k$. $\mathcal{H}$ is called a *factorization* of G if $\mathsf{mult}_{\mathcal{H}}$ is surjective. The integer k is called the *length* of $\mathcal{H}$. If $H_1, H_2, \ldots, H_k$ are proper subsets of G, then $\mathcal{H}$ is called a *proper factorization* of G.

▶ **Definition 1** (Definition 2.1 in [10]). *Let G be a group and $\mathcal{H} = (H_1, H_2, \ldots, H_k)$ a factorization of G.*
(1) *The factorization $\mathcal{H}$ is a* uniform factorization *of G if $|\mathsf{mult}_{\mathcal{H}}^{-1}(g)|$ does not depend on $g \in G$. The integer $t := |\mathsf{mult}_{\mathcal{H}}^{-1}(g)|$ is called the* multiplicity *of $\mathcal{H}$.*
(2) *The factorization $\mathcal{H}$ is a* uniform group factorization (UGF) *of G if $\mathcal{H}$ is a uniform factorization of G and all $H_1, H_2, \ldots, H_k$ are subgroups of G.*
(3) *The factorization $\mathcal{H}$ is a* uniform cyclic group factorization (UCF) *of G if $\mathcal{H}$ is a uniform group factorization of G and all $H_1, H_2, \ldots, H_k$ are cyclic subgroups of G.*

When $(H_1, H_2, \ldots, H_k)$ is a UGF (or a UCF) of G, we write $G = H_1 H_2 \ldots H_k$.

We give some examples of UCF. For the n-th symmetric group S_n, by letting $H_i := \langle (1, \ldots, i+1) \rangle$, the tuple $\mathcal{H}_1 = (H_1, H_2, \ldots, H_{n-1})$ is a UCF of S_n. The length of $\mathcal{H}_1$ is $n-1$ and the multiplicity of $\mathcal{H}_1$ is 1. For the n-th dihedral group $D_n = \langle \sigma, \tau \mid \sigma^n = 1, \tau^2 = 1, \tau^{-1}\sigma\tau = \sigma^{-1} \rangle$, the tuple $\mathcal{H}_2 = (\langle \sigma \rangle, \langle \tau \rangle)$ is a UCF of D_n. The length of $\mathcal{H}_2$ is 2 and the multiplicity of $\mathcal{H}_2$ is 1.

▶ **Lemma 2.** *Let G_1, G_2 be groups. If G_1 and G_2 have UGF (resp. UCF), then the direct product $G_1 \times G_2$ and the semi-direct product $G_1 \rtimes G_2$ also have UGF (resp. UCF).*

Proof. Suppose that $G_1 = H_1 H_2 \ldots H_k$ and $G_2 = M_1 M_2 \cdots M_\ell$ where $H_i = \langle h_i \rangle$ and $M_j = \langle m_i \rangle$. Then the direct product $G_1 \times G_2 = \{(g_1, g_2) \mid g_i \in G_i\}$ has a UCF $G_1 \times G_2 = \widehat{H_1}\widehat{H_2} \ldots \widehat{H_k}\widehat{M_1}\widehat{M_2} \cdots \widehat{M_\ell}$ where $\widehat{H_i} := \langle (h_i, \mathsf{id}_{G_2}) \rangle$ and $\widehat{M_j} := \langle (\mathsf{id}_{G_1}, m_j) \rangle$. Since every element $x \in G_1 \rtimes G_2$ is uniquely represented by $x = g_1 g_2$ for $g_i \in G_i$, the semi-direct product $G_1 \rtimes G_2$ has a UCF $G_1 \rtimes G_2 = H_1 H_2 \ldots H_k M_1 M_2 \cdots M_\ell$. ◀

Any solvable group has a UCF (Theorem 3.3 in [10]). It is an open problem whether any group has a UCF or not. The authors showed that any group has a UCF whenever any group has a proper UGF (Theorem 3.4 in [10]), i.e., the open problem can be affirmatively solved if any group has a UGF.

3.2 Shuffles Based on Uniform Cyclic Group Factorization

We show that if a group G has a uniform cyclic group factorization, the G-shuffle can be implemented by a sequence of cyclic group shuffles with no additional cards.

First, we define the equivalence between a shuffle and a sequence of shuffles. Intuitively, they are said to be equivalent if the resultant probability distributions are the same.

▶ **Definition 3.** *Let $G, H_1, \dots, H_s$ be subgroups of S_n. A shuffle $(\mathsf{shuffle}, G, \mathcal{F})$ is said to be* equivalent *to a sequence of shuffles $(\mathsf{shuffle}, H_1, \mathcal{F}_1), \dots, (\mathsf{shuffle}, H_s, \mathcal{F}_s)$ if for all $g \in G$,*

$$\mathcal{F}(g) = \sum_{\substack{(h_1,h_2,\dots,h_s)\in H_1\times H_2\times\cdots\times H_s \\ h_1h_2\cdots h_s=g}} \prod_{i=1}^{s} \mathcal{F}_i(h_i), \tag{1}$$

where the summation is taken over all $(h_1, \dots, h_s) \in H_1 \times \cdots \times H_s$ such that $h_1 \cdots h_s = g$.

The following lemma is easy but important for implementing shuffles.

▶ **Lemma 4.** *Let G be a group and $\mathcal{H} = (H_1, H_2, \dots, H_k)$ a factorization of G. Let $\mathcal{F}_{\mathcal{H}}$ be a probability distribution on G defined as follows:*

$$\mathcal{F}_{\mathcal{H}}(g) = \frac{|\mathsf{mult}_{\mathcal{H}}^{-1}(g)|}{\prod_{i=1}^{k} |H_i|}.$$

Then a shuffle $(\mathsf{shuffle}, G, \mathcal{F}_{\mathcal{H}})$ is equivalent to a sequence of k uniform shuffles as follows:

$$(\mathsf{shuffle}, H_1), (\mathsf{shuffle}, H_2), \dots, (\mathsf{shuffle}, H_k).$$

Moreover, if $\mathcal{H}$ is a uniform factorization, $\mathcal{F}_{\mathcal{H}}$ is a uniform distribution on G.

Proof. Since the probability that $h_i \in H_i$ is chosen by $(\mathsf{shuffle}, H_i)$ is $\frac{1}{|H_i|}$ and the number of $(h_1, \dots, h_k) \in H_1 \times \cdots \times H_k$ such that $h_1 \cdots h_k = g$ is $|\mathsf{mult}_{\mathcal{H}}^{-1}(g)|$, the right hand side of Eq. (1) is $\frac{|\mathsf{mult}_{\mathcal{H}}^{-1}(g)|}{\prod_{i=1}^{k} |H_i|}$ which is equal to $\mathcal{F}_{\mathcal{H}}(g)$. Therefore, $(\mathsf{shuffle}, G, \mathcal{F}_{\mathcal{H}})$ is equivalent to the sequence of k uniform shuffles. If $\mathcal{H}$ is a uniform factorization, $\mathcal{F}_{\mathcal{H}}$ is a uniform distribution on G since $|\mathsf{mult}_{\mathcal{H}}^{-1}(g)|$ does not depend on g. ◀

▶ **Corollary 5.** *Let G be a group. If G has a uniform cyclic group factorization of length k, a uniform closed shuffle $(\mathsf{shuffle}, G)$ is equivalent to a sequence of k cyclic group shuffles.*

Proof. Follows from Lemma 4. ◀

4 Secure Random Instance Generation of the 15 Puzzle

In this section, we propose a secure random instance generation protocol of the 15 Puzzle. In Section 4.1, we give a UCF of the alternating group A_n based on the construction given by the authors [10]. In Section 4.2, we construct a protocol based on the UCF.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 2 Cells of the 15 Puzzle.

4.1 Uniform Cyclic Group Factorization of the Alternating Group

The 15 Puzzle is a sliding block puzzle consisting of a *board* and 15 *blocks* from $\boxed{1}$ to $\boxed{15}$. The backs of all blocks are identical and denoted by $\boxed{?}$. The board has 4×4 *cells*, which are numbered from 1 to 16 as in Figure 2. Initially, all blocks are placed on cells from 1 to 15 in a random order. A cell having no block is called a *blank cell.* The aim of the puzzle is to make a board with $\boxed{1}$ to $\boxed{15}$ in cells from 1 to 15 in the order, by sliding blocks into the blank cell. An arrangement of the puzzle is identified with a permutation $\sigma \in S_{15}$ such that $\sigma(i) = j$ if the block $\boxed{j}$ is on the cell i. It is known that a permutation σ has a solution if and only if σ is an even permutation, thus the set of all instances of the 15 Puzzle forms the alternating group A_{15}.

Now we construct a UCF of A_n based on Proposition 5.2 in [10]. If n is odd, A_n is decomposed by $A_n = HK$ where $H \simeq A_{n-1}$ is the stabilizer fixing the point n and $K = \langle(1, 2, 3, \ldots, n)\rangle$ is a cyclic group. If n is even (i.e., $n = 2m$), A_n is decomposed by $A_n = HK_1K_2$ where $H \simeq A_{n-1}$ is the stabilizer fixing the point n and $K_1 = \langle(1, 2, \ldots, m)(m+1, m+2, \ldots, 2m)\rangle$ and $K_2 = \langle(1, m+1)(m, 2m)\rangle$ are cyclic groups.

From the above, we can construct a UCF of A_n for any n. Note that all subgroups correspond to practical shuffles: $K = \langle(1, 2, 3, \ldots, n)\rangle$ corresponds to a random cut, $K_1 = \langle(1, 2, \ldots, m)(m+1, m+2, \ldots, 2m)\rangle$ corresponds to a pile-shifting shuffle, and $K_2 = \langle(1, m+1)(m, 2m)\rangle$ corresponds to a random bisection cut. The above discussion is summarized by the following lemma.

▶ **Lemma 6.** *Let $n \geq 4$ and A_n be the set of all even permutations in S_n. If $n = 2m$, an A_n-shuffle is equivalent to a sequence of $3m - 3$ shuffles: $m - 1$ random cuts, m random bisection cuts, and $m - 2$ pile-shifting shuffles. If $n = 2m + 1$, it is equivalent to a sequence of $3m - 2$ shuffles: m random cuts, m random bisection cuts, and $m - 2$ pile-shifting shuffles.*

A UCF of A_{15} is given by $A_{15} = H_1H_2 \cdots H_{19}$ as follows:

- $H_1 = \langle(1, 2, 3)\rangle$;
- $H_2 = \langle(1, 3)(2, 4)\rangle$;
- $H_3 = \langle(1, 2)(3, 4)\rangle$;
- $H_4 = \langle(1, 2, 3, 4, 5)\rangle$;
- $H_5 = \langle(1, 4)(3, 6)\rangle$;
- $H_6 = \langle(1, 2, 3)(4, 5, 6)\rangle$;
- $H_7 = \langle(1, 2, 3, 4, 5, 6, 7)\rangle$;
- $H_8 = \langle(1, 5)(4, 8)\rangle$;
- $H_9 = \langle(1, 2, 3, 4)(5, 6, 7, 8)\rangle$;
- $H_{10} = \langle(1, 2, 3, 4, 5, 6, 7, 8, 9)\rangle$;
- $H_{11} = \langle(1, 6)(5, 10)\rangle$;
- $H_{12} = \langle(1, 2, 3, 4, 5)(6, 7, 8, 9, 10)\rangle$;
- $H_{13} = \langle(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)\rangle$;
- $H_{14} = \langle(1, 7)(6, 12)\rangle$;

- $H_{15} = \langle(1,2,3,4,5,6)(7,8,9,10,11,12)\rangle$;
- $H_{16} = \langle(1,2,3,4,5,6,7,8,9,10,11,12,13)\rangle$;
- $H_{17} = \langle(1,8)(7,14)\rangle$;
- $H_{18} = \langle(1,2,3,4,5,6,7)(8,9,10,11,12,13,14)\rangle$;
- $H_{19} = \langle(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)\rangle$.

4.2 Our Protocol for the 15 Puzzle

Based on the UCF of A_{15}, we can construct a secure random instance generation protocol for the 15 Puzzle. From Lemma 6, our protocol requires 19 practical shuffles consisting of 7 random cuts, 7 random bisection cuts, and 5 pile-shifting shuffles. The protocol proceeds as follows:

1. Arrange 15 blocks as follows:

$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}\boxed{7}\boxed{8}\boxed{9}\boxed{10}\boxed{11}\boxed{12}\boxed{13}\boxed{14}\boxed{15}.$$

2. Turn all blocks face-down as follows:

$$\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

3. Apply a shuffle $(\mathsf{shuffle}, H_i)$ for $i = 1, 2, \ldots, 19$ as follows:

$$\underbrace{\boxed{?}\cdots\boxed{?}}_{\text{15 blocks}} \xrightarrow{H_1} \underbrace{\boxed{?}\cdots\boxed{?}}_{\text{15 blocks}} \xrightarrow{H_2} \underbrace{\boxed{?}\cdots\boxed{?}}_{\text{15 blocks}} \xrightarrow{H_3} \cdots \xrightarrow{H_{19}} \underbrace{\boxed{?}\cdots\boxed{?}}_{\text{15 blocks}}.$$

4. Place the i-th block to the i-th cell of the board for $1 \le i \le 15$. The face-down blocks on the board is a random instance of the 15 Puzzle.

5 Secure Random Instance Generation of Rubik's Cube

In this section, we propose a secure random instance generation protocol of Rubik's Cube. In Section 5.1, we give a UCF of the Rubik's Cube group R. In Section 5.2, we construct a protocol in the free permutation model. In Section 5.3, we give a UCF of R whose generators are written by a product of the standard generators. In Section 5.4, we construct a protocol in the restricted permutation model.

5.1 Uniform Cyclic Group Factorization of the Rubik's Cube Group

Rubik's Cube consists of a number of pieces called *cubies*. A face of a cubie is called a *facelet*. There are 6 center cubies, 8 corner cubies, and 12 edge cubies in Rubik's Cube. By fixing the center cubies, any instance can be regarded as a permutation of the 48 facelets, and thus the Rubik's Cube group R is a subgroup of S_{48}. It is generated by the following permutations:

- $\mathsf{F} := (17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11)$;
- $\mathsf{B} := (33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27)$;
- $\mathsf{U} := (1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19)$;
- $\mathsf{D} := (41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40)$;
- $\mathsf{L} := (9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35)$;
- $\mathsf{R} := (25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24)$.

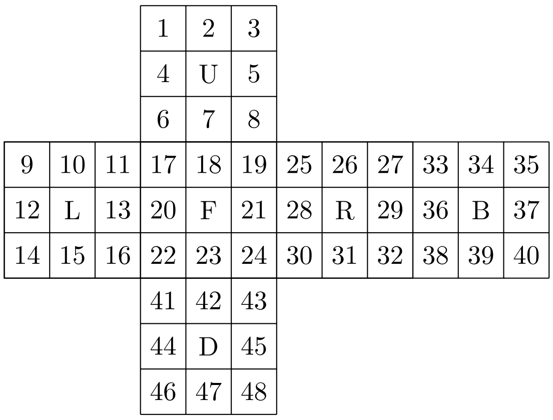


Figure 3 Facelets of Rubik's Cube.

Here, the facelets are numbered from 1 to 48 as in Figure 3 and the above six permutations correspond to the clockwise 90° rotation of the front, back, up, down, left, and right faces of the cube, respectively. We call them the *standard generators* of R.

Edge cubies are represented by a pair of two indices, denoted by $[\alpha, \beta]$, as follows: $e_1 := [2, 34]$, $e_2 := [4, 10]$, $e_3 := [5, 26]$, $e_4 := [7, 18]$, $e_5 := [12, 37]$, $e_6 := [13, 20]$, $e_7 := [15, 44]$, $e_8 := [21, 28]$, $e_9 := [23, 42]$, $e_{10} := [29, 36]$, $e_{11} := [31, 45]$, and $e_{12} := [39, 47]$. Corner cubies are represented by a tuple of three indices, denoted by $[\gamma, \delta, \epsilon]$, as follows: $c_1 := [1, 35, 9]$, $c_2 := [3, 27, 33]$, $c_3 := [6, 11, 17]$, $c_4 := [8, 19, 25]$, $c_5 := [14, 40, 46]$, $c_6 := [16, 41, 22]$, $c_7 := [24, 43, 30]$, and $c_8 := [32, 48, 38]$. For an edge cubie $e_i = [\alpha_i, \beta_i]$, a permutation representing a flip of the cubie is denoted by $\tau_i := (\alpha_i, \beta_i)$. For a corner cubie $c_i = [\gamma_i, \delta_i, \epsilon_i]$, a permutation representing the counterclockwise 120° rotation of the cubie is denoted by $\sigma_i := (\gamma_i, \delta_i, \epsilon_i)$. For a permutation $\pi \in S_{12}$, a permutation $\mathsf{e}(\pi) \in S_{48}$ is defined as a permutation that moves the edge cubies according to π, i.e., the i-th edge cubie e_i is moved to the $\pi(i)$-th edge cubie. For example, $\mathsf{e}((1, 2, 3)) = (2, 4, 5)(34, 10, 26)$. For a permutation $\pi \in S_8$, a permutation $\mathsf{c}(\pi) \in S_{48}$ is defined as a permutation that moves the corner cubies according to π, i.e., the i-th corner cubie c_i is moved to the $\pi(i)$-th corner cubie. For example, $\mathsf{c}((1, 2, 3)) = (1, 3, 6)(35, 27, 11)(9, 33, 17)$.

The group structure of R is given by $R \simeq (\mathbb{Z}_2^{11} \times \mathbb{Z}_3^7) \rtimes ((A_8 \times A_{12}) \rtimes \mathbb{Z}_2)$ as follows:

- The subgroup $\mathbb{Z}_2^{11}$ of R is given by:

$$\mathbb{Z}_2^{11} = \{\tau_1^{b_1} \tau_2^{b_2} \cdots \tau_{12}^{b_{12}} \mid b_1 + b_2 + \cdots + b_{12} \equiv 0 \mod 2\}.$$

  It represents the group of all flippings of 12 edge cubies with the restriction that the number of flipped cubies is even. A UCF of $\mathbb{Z}_2^{11}$ is given by $\mathbb{Z}_2^{11} = H_1 H_2 \ldots H_{11}$, where $H_i = \langle \tau_1 \tau_{i+1} \rangle$ for $1 \leq i \leq 11$.
- The subgroup $\mathbb{Z}_3^7$ of R is given by:

$$\mathbb{Z}_3^7 = \{\sigma_1^{b_1} \sigma_2^{b_2} \cdots \sigma_8^{b_8} \mid b_1 + b_2 + \cdots + b_8 \equiv 0 \mod 3\}.$$

  It represents the group of all rotations of 8 corner cubies, where all but one may be rotated freely, but they determine the orientation of the last corner cubie. A UCF of $\mathbb{Z}_3^7$ is given by $\mathbb{Z}_3^7 = H_{12} H_{13} \ldots H_{18}$, where $H_{11+i} = \langle (\sigma_1)^{-1} \sigma_{i+1} \rangle$ for $1 \leq i \leq 7$.
- The subgroup A_8 of R is given by:

$$A_8 = \{\mathsf{c}(\pi) \mid \pi \in A_8 \leq S_8\}.$$

  It represents the group of even permutations over 8 corner cubies. A UCF of A_8 is given by $A_8 = H_{19} H_{20} \cdots H_{27}$, where $H_{18+i} = \langle \mathsf{c}(\sigma_i') \rangle$ for $1 \leq i \leq 9$ and σ_i' is a generator of the UCF of A_8 given in Section 4.1.

- The subgroup A_{12} of R is given by:

$$A_{12} = \{\mathsf{e}(\pi) \mid \pi \in A_{12} \leq S_{12}\}.$$

It represents the group of even permutations over 12 edge cubies. A UCF of A_{12} is given by $A_{12} = H_{28}H_{29} \cdots H_{42}$, where $H_{27+i} = \langle \mathsf{e}(\sigma'_i) \rangle$ for $1 \leq i \leq 15$ and σ'_i is a generator of the UCF of A_{12} given in Section 4.1.
- The subgroup Z_2 of R is given by:

$$Z_2 = \langle \mathsf{e}((1,2))\mathsf{c}((1,2)) \rangle.$$

It is a group generated by the composition of exchanging two corner cubies and exchanging two edge cubies. Since Z_2 is a cyclic group, it has a trivial UCF $H_{43} := Z_2$.

From the above and Lemma 2, a UCF of R is given by $R = H_1H_2 \cdots H_{43}$ as follows:
- $H_1 = \langle (2,34)(4,10) \rangle$;
- $H_2 = \langle (2,34)(5,26) \rangle$;
- $H_3 = \langle (2,34)(7,18) \rangle$;
- $H_4 = \langle (2,34)(12,37) \rangle$;
- $H_5 = \langle (2,34)(13,20) \rangle$;
- $H_6 = \langle (2,34)(15,44) \rangle$;
- $H_7 = \langle (2,34)(21,28) \rangle$;
- $H_8 = \langle (2,34)(23,42) \rangle$;
- $H_9 = \langle (2,34)(29,36) \rangle$;
- $H_{10} = \langle (2,34)(31,45) \rangle$;
- $H_{11} = \langle (2,34)(39,47) \rangle$;
- $H_{12} = \langle (1,9,35)(3,27,33) \rangle$;
- $H_{13} = \langle (1,9,35)(6,11,17) \rangle$;
- $H_{14} = \langle (1,9,35)(8,19,25) \rangle$;
- $H_{15} = \langle (1,9,35)(14,40,46) \rangle$;
- $H_{16} = \langle (1,9,35)(16,41,22) \rangle$;
- $H_{17} = \langle (1,9,35)(24,43,30) \rangle$;
- $H_{18} = \langle (1,9,35)(32,48,38) \rangle$;
- $H_{19} = \langle (1,3,6)(35,27,11)(9,33,17) \rangle$;
- $H_{20} = \langle (1,6)(35,11)(9,17)(3,8)(27,19)(33,25) \rangle$;
- $H_{21} = \langle (1,3)(35,27)(9,33)(6,8)(11,19)(17,25) \rangle$;
- $H_{22} = \langle (1,3,6,8,41)(35,27,11,19,22)(9,33,17,25,16) \rangle$;
- $H_{23} = \langle (1,8)(35,19)(9,25)(6,43)(11,30)(17,24) \rangle$;
- $H_{24} = \langle (1,3,6)(35,27,11)(9,33,17)(8,41,43)(19,22,30)(25,16,24) \rangle$;
- $H_{25} = \langle (1,3,6,8,41,43,46)(35,27,11,19,22,30,14)(9,33,17,25,16,24,40) \rangle$;
- $H_{26} = \langle (1,41)(35,22)(9,16)(8,48)(19,38)(25,32) \rangle$;
- $H_{27} = \langle (1,3,6,8)(35,27,11,19)(9,33,17,25)(41,43,46,48)(22,30,14,38)(16,24,40,32) \rangle$;
- $H_{28} = \langle (2,4,5)(34,10,26) \rangle$;
- $H_{29} = \langle (2,5)(34,26)(4,7)(10,18) \rangle$;
- $H_{30} = \langle (2,4)(34,10)(5,7)(26,18) \rangle$;
- $H_{31} = \langle (2,4,5,7,12)(34,10,26,18,37) \rangle$;
- $H_{32} = \langle (2,7)(34,18)(5,20)(26,13) \rangle$;
- $H_{33} = \langle (2,4,5)(34,10,26)(7,12,20)(18,37,13) \rangle$;
- $H_{34} = \langle (2,4,5,7,12,20,44)(34,10,26,18,37,13,15) \rangle$;
- $H_{35} = \langle (2,12)(34,37)(7,28)(18,21) \rangle$;

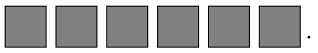
- $H_{36} = \langle (2,4,5,7)(34,10,26,18)(12,20,44,28)(37,13,15,21) \rangle$;
- $H_{37} = \langle (2,4,5,7,12,20,44,28,42)(34,10,26,18,37,13,15,21,23) \rangle$;
- $H_{38} = \langle (2,20)(34,13)(12,36)(37,29) \rangle$;
- $H_{39} = \langle (2,4,5,7,12)(34,10,26,18,37)(20,44,28,42,36)(13,15,21,23,29) \rangle$;
- $H_{40} = \langle (2,4,5,7,12,20,44,28,42,36,45)(34,10,26,18,37,13,15,21,23,29,31) \rangle$;
- $H_{41} = \langle (2,44)(34,15)(20,47)(13,39) \rangle$;
- $H_{42} = \langle (2,4,5,7,12,20)(34,10,26,18,37,13)(44,28,42,36,45,47)(15,21,23,29,31,39) \rangle$;
- $H_{43} = \langle (2,4)(34,10)(1,3)(35,27)(9,33) \rangle$.

5.2 Our Protocol for Rubik's Cube in the Free Permutation Model

In this section, we construct a secure random instance generation protocol of Rubik's Cube in the free permutation model. It requires color stickers along with a cube.

A *color sticker* is a sticker with one of the six colors as follows:

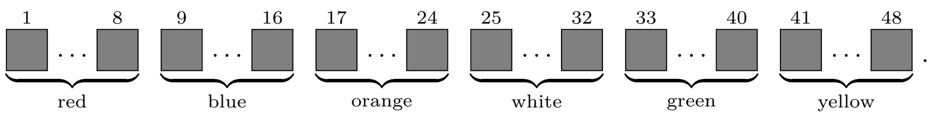
Note that each letter (R, B, O, W, G, and Y) is indicated for clarity and is not actually written. They are the same size as the facelet of the cube and can be placed on facelets. At the beginning, all stickers are concealed by *films* that can be peeled off as follows:



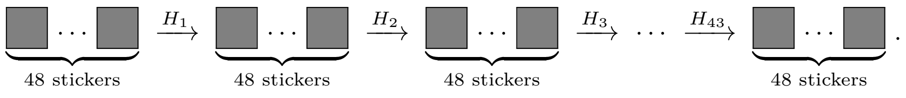
All stickers are assumed to be indistinguishable when their films are not peeled off.

Our protocol requires 48 color stickers, consisting of 8 color stickers of each color, and 43 practical shuffles, consisting of 22 random bisection cuts and 21 pile-shifting shuffles. The protocol proceeds as follows:

1. Arrange a sequence of color stickers as follows:



2. Apply an H_i-shuffle to the sequence for $i = 1, 2, \ldots, 43$ as follows:



3. Place the i-th color sticker to the i-th facelet of Rubik's Cube for $1 \leq i \leq 48$. The cube concealed by films is a random instance of Rubik's Cube.

5.3 Uniform Cyclic Group Factorization with the Standard Generators

For each $\mathsf{O} \in \{\mathsf{F}, \mathsf{B}, \mathsf{U}, \mathsf{D}, \mathsf{L}, \mathsf{R}\}$, we denote O^3 by O'. All generators of the UCF in Section 5.1 can be expressed by a product of the standard generators $\mathsf{F}, \mathsf{B}, \mathsf{U}, \mathsf{D}, \mathsf{L}$, and R. For example, $(2,34)(4,10) = \mathsf{RLFU^2F'RL'UB^2U'F^2L^2U'F^2R^2B^2D'}$. All generators of the subgroups H_i $(1 \leq i \leq 43)$ are expressed by a product of the standard generators as follows:

- $H_1 = \langle \mathsf{RLFU^2F'RL'UB^2U'F^2L^2U'F^2R^2B^2D'} \rangle$;
- $H_2 = \langle \mathsf{RUR'U'R'U'RURB'U'R^2URB} \rangle$;
- $H_3 = \langle \mathsf{U^2LFUF'UL'B'R'U^2RBU'LU'L'} \rangle$;
- $H_4 = \langle \mathsf{LBDL^2D'L'BD^2R^2F^2U^2F^2R^2DB^2D} \rangle$;

- $H_5 = \langle \mathsf{LB'D'B^2DLBDL^2DF^2R^2U^2R^2F^2D^2} \rangle$;
- $H_6 = \langle \mathsf{RL'FU^2F'R'LDR^2U^2R^2F^2R^2U^2R^2D'} \rangle$;
- $H_7 = \langle \mathsf{RB'D'R^2DRB'U^2F^2D'L^2DF^2UB^2U} \rangle$;
- $H_8 = \langle \mathsf{URLF'R^2F'RL'UR^2U^2B^2L^2F^2DL^2B^2U'} \rangle$;
- $H_9 = \langle \mathsf{UBU'B'U'B'UBUR'B'U^2BUR} \rangle$;
- $H_{10} = \langle \mathsf{RLF'U^2FRL'U^2R^2L^2DF^2D'R^2L^2U^2R^2} \rangle$;
- $H_{11} = \langle \mathsf{B^2LUBU'BL'D'R'B^2RDB'LB'L'} \rangle$;
- $H_{12} = \langle \mathsf{R^2U'F^2UB^2U'F^2UB^2DR'URD'R'U'R'} \rangle$;
- $H_{13} = \langle \mathsf{L^2D'L^2DF^2U'FUL^2UL^2U'F} \rangle$;
- $H_{14} = \langle \mathsf{F^2U'F^2U'R^2DR^2DB^2D^2FDB^2D'F'U^2} \rangle$;
- $H_{15} = \langle \mathsf{R^2D'B^2U'F^2UB^2U'F^2UR'URDR'U'R'} \rangle$;
- $H_{16} = \langle \mathsf{L^2D'B^2DB^2U'L^2UL^2D^2RD'L^2DR'D^2} \rangle$;
- $H_{17} = \langle \mathsf{R^2UF^2U'L^2UF^2U'R^2UFD'B^2DF'U'} \rangle$;
- $H_{18} = \langle \mathsf{B^2DB^2D'R^2UR^2U'B^2U^2F'UB^2U'FU^2} \rangle$;
- $H_{19} = \langle \mathsf{R^2DB^2D'F^2DB^2D'F^2R^2} \rangle$;
- $H_{20} = \langle \mathsf{RBR'F^2RB'RB^2U'F^2DL^2B^2D'F^2UF^2} \rangle$;
- $H_{21} = \langle \mathsf{RU^2R^2DR^2U^2RB^2UR^2UR^2U^2B^2D'R^2} \rangle$;
- $H_{22} = \langle \mathsf{UR^2UF^2U^2F^2U'R^2U'LR'F^2L'R} \rangle$;
- $H_{23} = \langle \mathsf{URLD^2R'L'F^2U'L^2U'B^2UL^2DR^2D'} \rangle$;
- $H_{24} = \langle \mathsf{U^2F^2R^2F^2R^2UR^2F^2R^2F^2U} \rangle$;
- $H_{25} = \langle \mathsf{L^2UR'LF^2R'L'F^2UB^2L^2D^2R^2DF^2DB^2} \rangle$;
- $H_{26} = \langle \mathsf{RLU^2RL'U^2R^2F^2U^2B^2D^2R^2B^2U^2L^2} \rangle$;
- $H_{27} = \langle \mathsf{URLU^2D^2RLU'F^2R^2UF^2D^2R^2UB^2L^2} \rangle$;
- $H_{28} = \langle \mathsf{R^2U^2F^2L^2B^2DB^2L^2F^2U'R^2} \rangle$;
- $H_{29} = \langle \mathsf{URL'U^2R'LUF^2UF^2UF^2UF^2UF^2} \rangle$;
- $H_{30} = \langle \mathsf{UFB'U^2F'BUL^2UL^2UL^2UL^2UL^2} \rangle$;
- $H_{31} = \langle \mathsf{L^2FL^2U^2L^2FUL^2B^2R^2U'R^2UDR^2B^2} \rangle$;
- $H_{32} = \langle \mathsf{R'U^2F^2U^2RU^2F^2R^2U^2F^2U^2F^2R^2U^2F^2} \rangle$;
- $H_{33} = \langle \mathsf{L^2U^2F^2R^2B^2U'L^2B^2R^2UFUF'D'ULU'L} \rangle$;
- $H_{34} = \langle \mathsf{B^2U'B^2L^2UL^2B^2L^2D'U^2L'B'LDL^2BLB} \rangle$;
- $H_{35} = \langle \mathsf{R^2F^2U^2FU^2D^2BU^2D^2F^2D^2B^2L^2U^2D^2} \rangle$;
- $H_{36} = \langle \mathsf{DUB^2L^2R^2D^2F^2UL^2DBDF'L'FLBR^2F'} \rangle$;
- $H_{37} = \langle \mathsf{U^2L^2DR^2F^2R^2UF^2R^2U^2FD'LR'FL'FL'RU'} \rangle$;
- $H_{38} = \langle \mathsf{RDBD'R'U'L'B'LUBU'DL'D'U} \rangle$;
- $H_{39} = \langle \mathsf{R^2D'R^2UF^2L^2D'B^2R^2DFL'U'B^2UL'R^2BR^2U^2} \rangle$;
- $H_{40} = \langle \mathsf{L^2B^2L^2D^2UR^2U'B^2U^2B'L^2D'U'B^2R'D^2R^2U'F^2U'} \rangle$;
- $H_{41} = \langle \mathsf{D'RU^2F^2U^2R'F^2U^2L^2B^2U^2F^2D^2R^2B^2D'} \rangle$;
- $H_{42} = \langle \mathsf{DF^2U'F^2L^2U^2F^2U^2F^2UFR^2U'L^2R^2UF^2RF^2U'} \rangle$;
- $H_{43} = \langle \mathsf{UR^2U'R^2DR^2D'F^2UF^2R^2} \rangle$.

We compute them by Online Rubik's Cube Solver [24]. We remark that all of the generators as above are of length at most 20, which is God's Number of Rubik's Cube.

5.4 Our Protocol for Rubik's Cube in the Restricted Permutation Model

In this section, we construct a secure random instance generation protocol of Rubik's Cube in the restricted permutation model.

Our protocol uses a *piece of cloth*, which is sufficiently large to hide a cube completely. A player can hold a cube through his/her hand from under the cloth, and can apply a sequence of the standard generators. During this operation, which permutation is applied to the cube is completely hidden from other players.

Now we introduce a repetitive shuffle, which is a shuffling operation for a cube. Let $\mathsf{seq} \in \{\mathsf{F}, \mathsf{B}, \mathsf{U}, \mathsf{D}, \mathsf{L}, \mathsf{R}\}^*$ be a sequence of the standard generators such that the order of seq is k, i.e., applying seq k times is equal to the identity permutation. For a cube covered with a piece of cloth, a *repetitive shuffle* of seq covertly applies seq^r to the cube for a uniformly random number $r \in \{0, 1, \ldots, k-1\}$. Here, r is completely hidden from all players.

We give two physical implementations of repetitive shuffles. The first method is performed by a single player: The player holding a cube through his hand from under the cloth applies seq a sufficiently large number of times until he loses the number of times. The second method is performed by (at least) two players: The first player holding a cube through his hand from under the cloth applies seq^{r_1} to the cube for a uniformly random number $r_1 \in \{0, 1, \ldots, k-1\}$ which is generated by his mind; Then the cube is passed to the second player, and she applies seq^{r_2} to the cube for a uniformly random number $r_2 \in \{0, 1, \ldots, k-1\}$ which is generated by her mind. Since $r := r_1 + r_2 \bmod k$ is distributed uniformly at random and r is completely hidden from the both players, the second method correctly implements the repetitive shuffle.

Our protocol requires 43 repetitive shuffles. The protocol proceeds as follows:

1. Prepare a solved cube. Cover it with a piece of cloth.
2. Apply 43 repetitive shuffles of the generators of cyclic groups given in Section 5.3. The cube covered with a piece of cloth is a random instance of Rubik's Cube.

6 Conclusion

In this paper, we introduced the secure random instance generation problem for the first time and designed three protocols for the 15 Puzzle and Rubik's Cube. We left as an open problem to achieve the same task with a smaller number of practical shuffles. Note that our protocols are based on UCFs with multiplicity 1, but it may be possible to construct a shorter UCF with multiplicity 2 or more. Another research direction is to design a secure random instance generation protocol for other combination puzzles.

References

1 Adrian Brüngger, Ambros Marzetta, K. Fukuda, and Jürg Nievergelt. The parallel search bench ZRAM and its applications. *Ann. Oper. Res.*, 90:45–63, 1999.

2 Yang Chu and Robert Hough. Solution of the 15 puzzle problem, 2019. `arXiv:1908.07106`.

3 Claude Crépeau and Joe Kilian. Discreet solitary games. In *Advances in Cryptology—CRYPTO' 93*, volume 773 of *LNCS*, pages 319–330. Springer, 1994.

4 Erik D. Demaine, Sarah Eisenstat, and Mikhail Rudoy. Solving the rubik's cube optimally is NP-complete. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018*, volume 96 of *LIPIcs*, pages 24:1–24:13, 2018.

5 Bert Den Boer. More efficient match-making and satisfiability the five card trick. In *EUROCRYPT 1989*, volume 434 of *LNCS*, pages 208–217. Springer, 1990.

6 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.*, E101.A(9):1503–1511, 2018.

7 Yuji Hashimoto, Kazumasa Shinagawa, Koji Nuida, Masaki Inamura, and Goichiro Hanaoka. Secure grouping protocol using a deck of cards. In *Information Theoretic Security*, volume 10681 of *LNCS*, pages 135–152. Springer, 2017.

8 Takuya Ibaraki and Yoshifumi Manabe. A more efficient card-based protocol for generating a random permutation without fixed points. In *Mathematics and Computers in Sciences and in Industry (MCSI)*, pages 252–257, 2016.

9 Rie Ishikawa, Eikoh Chida, and Takaaki Mizuki. Efficient card-based protocols for generating a hidden random permutation without fixed points. In *Unconventional Computation and Natural Computation*, volume 9252 of *LNCS*, pages 215–226. Springer, 2015.

10 Kazuki Kanai, Kengo Miyamoto, Koji Nuida, and Kazumasa Shinagawa. Uniform cyclic group factorizations of finite groups. *Communications in Algebra*, 2023.

11 Alexander Koch and Stefan Walzer. Foundations for actively secure card-based cryptography. In *Fun with Algorithms*, volume 157 of *LIPIcs*, pages 17:1–17:23, 2020.

12 Kengo Miyamoto and Kazumasa Shinagawa. Graph automorphism shuffles from pile-scramble shuffles. *New Gener. Comput.*, 40:199–223, 2022.

13 Takaaki Mizuki, Yoshinori Kugimoto, and Hideaki Sone. Secure multiparty computations using the 15 puzzle. In *Combinatorial Optimization and Applications*, volume 4616 of *LNCS*, pages 255–266. Springer, 2007.

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

15 Takaaki Mizuki and Hideaki Sone. Six-card secure AND and four-card secure XOR. In *Frontiers in Algorithmics*, volume 5598 of *LNCS*, pages 358–369. Springer, 2009.

16 Bruce Norskog and Morley Davidson. The fifteen puzzle can be solved in 43 "moves", 2010. URL: `http://cubezzz.duckdns.org/drupal/?q=node/view/223`.

17 Daniel Ratner and Manfred K. Warmuth. Finding a shortest solution for the N $\times$ N extension of the 15-puzzle is intractable. In *Proceedings of the 5th National Conference on Artificial Intelligence. Volume 1: Science*, pages 168–172, 1986.

18 Daniel Ratner and Manfred K. Warmuth. N $\times$ N puzzle and related relocation problem. *J. Symb. Comput.*, 10(2):111–138, 1990.

19 Tomas Rokicki, Palo Alto, Herbert Kociemba, Morley Davidson, and John Dethridge. God's number is 20. URL: `https://www.cube20.org/`.

20 Takahiro Saito, Daiki Miyahara, Yuta Abe, Takaaki Mizuki, and Hiroki Shizuya. How to implement a non-uniform or non-closed shuffle. In *Theory and Practice of Natural Computing*, volume 12494 of *LNCS*, pages 107–118. Springer, 2020.

21 Kazumasa Shinagawa and Kengo Miyamoto. Automorphism shuffles for graphs and hypergraphs and its applications. *IEICE Trans. Fundam.*, E106.A(3):306–314, 2023.

22 Kazumasa Shinagawa, Takaaki Mizuki, Jacob C. N. Schuldt, Koji Nuida, Naoki Kanayama, Takashi Nishide, Goichiro Hanaoka, and Eiji Okamoto. Multi-party computation with small shuffle complexity using regular polygon cards. In *Provable Security*, volume 9451 of *LNCS*, pages 127–146. Springer, 2015.

23 Itaru Ueda, Daiki Miyahara, Akihiro Nishimura, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. Secure implementations of a random bisection cut. *Int. J. Inf. Secur.*, 19(4):445–452, 2020.

24 Online rubik's cube solver. URL: `https://rubiks-cube-solver.com/`.