entropy

MDPI

Article

From Random Numbers to Random Objects

Behrouz Zolfaghari [1], Khodakhast Bibak [2,*] and Takeshi Koshiba [3]

[1] Cyber Science Lab, School of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada; behrouz@cybersciencelab.org
[2] Department of Computer Science and Software Engineering, Miami University, Oxford, OH 45056, USA
[3] Department of Mathematics, Faculty of Education and Integrated Arts and Sciences, Waseda University, Tokyo 169-8050, Japan; tkoshiba@waseda.jp
* Correspondence: bibakk@miamioh.edu

Abstract: Many security-related scenarios including cryptography depend on the random generation of passwords, permutations, Latin squares, CAPTCHAs and other types of non-numerical entities. Random generation of each entity type is a different problem with different solutions. This study is an attempt at a unified solution for all of the mentioned problems. This paper is the first of its kind to pose, formulate, analyze and solve the problem of *random object generation* as the general problem of generating random non-numerical entities. We examine solving the problem via connecting it to the well-studied *random number generation* problem. To this end, we highlight the challenges and propose solutions for each of them. We explain our method using a case study; random Latin square generation.

Keywords: integer compositions; Linear Feedback Shift Registers (LFSRs); parallel LFSRs; random number generation; random object generation; S-restricted random number generator

MSC: 65C10; 94A60; 97P60

check for updates

Citation: Zolfaghari, B.; Bibak, K.; Koshiba, T. From Random Numbers to Random Objects. *Entropy* **2022**, *24*, 928. https://doi.org/10.3390/e24070928

Academic Editor: Éloi Bossé

Received: 30 May 2022
Accepted: 1 July 2022
Published: 4 July 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction and Basic Concepts

Random password generation [1], random CAPTCHA generation (Completely Automated Public Turing test to tell Computers and Humans Apart) [2,3], random permutation generation [4,5] and random Latin square generation [6,7] are critical cryptographic problems. There are many other similar problems that play critical roles in different branches of science and technology. Each of these problems is about random generation of *non-numeric entity*, to all of which we refer using the general name *object* in this paper. Although the mentioned problems may appear conceptually similar, solutions proposed for each of them may not be applicable to others. In this paper, we unify these problems and formalize the general problem of *random object generation* as the problem of generating random instances of any *non-numeric* entity type. Afterwards, we examine solving the newly-posed problem via connecting it to the well-known problem of *random number generation*. The primary idea behind our proposed approach is to assign numeric codes to objects, and generate random object codes using Random Number Generators (RNGs) (Please see more details in Section 2.3). However, there are challenges that need to be resolved. We analyze and resolve each of these challenges (Section 2.2).

In this study, we first formalize the problem of *random object generation*. Then, we introduce the notion of S-restricted RNGs as RNGs capable of generating random numbers derived from an arbitrary set S. We show how the use of S-restricted RNGs along with proper encoding schemes can lead to a solution to the problem of *random object generation*. We present our proposed *random object generation* method based on these two components. In the next step, we will propose a method based on integer compositions for automatic design of parallel LFSRs. Furthermore, we present an architecture based on parallel LFSRs

(as parallel RNGs) for designing S-restricted RNGs. Lastly, we present a case study for illustrating the proposed method. We design a circuit for generating random Latin squares of order 4 with the help of a novel encoding scheme.

Before formalizing the ROG problem and discussing our approach towards its solution, we need to introduce some concepts in the next subsection.

1.1. Basic Concepts

In the following, we define integer composition, parallel LFSRs, S-restricted RNGs and Latin squares, which will be used later while describing our solution to the problem of *random object generation*.

1.1.1. Integer Compositions

A composition C of a positive integer n is a sequence of positive integers called parts (summands) that add up to n. Different aspects and applications of integer compositions have been studied by researchers [8]. In this paper, we represent compositions by tuples, and denote the set of compositions of a positive integer n by $\mathcal{C}(n)$. As an example, $\mathcal{C}(3) = \{(1,1,1),(1,2),(2,1),(3)\}$. It can easily be shown that $|\mathcal{C}(n)| = 2^{n-1}$ for every $n \in \mathbb{N}$. For a composition $C = (c_1, c_2, \ldots, c_{l-1}, c_l)$ and $i \in \{1, 2, \ldots, l\}$, we define $\tilde{C} = (c_l, c_{l-1}, \ldots, c_2, c_1)$, $\lambda(C, i) = c_i$, $\text{first}(C) = \lambda(C, 1) = c_1$, $last(C) = \lambda(C, l) = c_l$, $\text{length}(C) = l$, $f^-(C) = (c_2, \ldots, c_{l-1}, c_l)$ and $l^-(C) = (c_1, c_2, \ldots, c_{l-1})$. Moreover, for a positive integer x, we represent $(x, c_1, c_2, \ldots, c_{l-1}, c_l)$ and $(c_1, c_2, \ldots, c_{l-1}, c_l, x)$ by $x; C$ and $C; x$, respectively. It is immediate that $f^-(C) = l^-(C) = \varphi = ()$ if $\text{length}(C) = 1$, and $x; C = C; x = (x)$ if $C = \varphi$.

An S-restricted composition of n is a composition in which all parts are chosen from a given set $S \subset \{1, 2, \ldots, n\}$. Several properties of S-restricted compositions as well as related problems have been investigated in various research works [9]. We use $\mathcal{C}^{(S)}(n)$ to represent the set of S-restricted compositions of n. For example, $\mathcal{C}^{(\{1,2\})}(3) = \{(1,1,1),(1,2),(2,1)\}$. Different types and aspects of S-restricted compositions have been investigated by researchers [10]. However, to the best of our knowledge, there is no closed form solution for calculating the number of S-restricted compositions.

A palindromic composition (a palindrome) of n is a composition of n that is read in the same way from the left and the right. The notation $\mathcal{C}_P(n)$ is used in this paper to represent the set of palindromic compositions of n. For example, $\mathcal{C}_P(3) = \{(1,1,1),(3)\}$. It can easily be shown that $|\mathcal{C}_P(n)| = 2^{\lfloor \frac{n}{2} \rfloor}$. An S-restricted palindromic composition (S-restricted palindrome) of n is a palindrome in which parts are chosen from a given set $S \subset \{1, 2, \ldots, n\}$. For example, $\mathcal{C}_P^{\{1,2\}}(3) = \{(1,1,1)\}$.

1.1.2. Parallel LFSRs

An LFSR is constructed of a shift register for keeping the state of the LFSR, along with a feedback loop, which controls the state transition. We denote an LFSR with n flip-flops and the generating polynomial G by $\mathcal{P}_n(G, M)$, where M is the input string. There are two common representations for $\mathcal{P}_n(G, M)$; Fibonacci representation and Galois representation. Galois and Fibonacci representations are mathematically equivalent in the sense that every sequence generated by a Fibonacci LFSR can be generated by a Galois LFSR, and vice versa [11][12]. Since the delay in the corresponding logic circuit is not dependent on the size of the LFSR, we choose to use Galois LFSR defined by Equation (1):

$$F_i(k+1) = \begin{cases} F_n(k) + M_k, & i = 1, \\ F_{i-1}(k) + G_{i-1}F_n(k), & i \in \{2, 3, \ldots, n\}. \end{cases} \tag{1}$$

In Equation (1), n is referred to as the size of the LFSR, and the vector $\mathcal{S}^k(\mathcal{P}_n(G, M)) = [F_1(k), F_2(k), \ldots, F_n(k)]$ is called the k^{th} state of the LFSR. Especially, $\mathcal{S}^0(\mathcal{P}_n(G, M)) = [F_1(0), F_2(0), \ldots, F_n(0)]$ is referred to as the initial state of the LFSR. An implementation of an LFSR of size n is called a programmable implementation if it is capable of working with

any arbitrary generating polynomial of degree n. A programmable LFSR implemented on the basis of Equation (1) is shown in Figure 1.

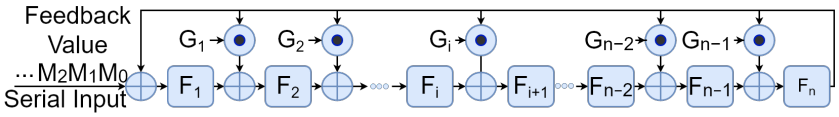

Figure 1. A programmable LFSR of size n.

In the programmable LFSR shown in Figure 1, $\oplus$ and $\odot$ represent $GF(2)$ addition and multiplication, respectively, which can be implemented at the hardware layer using XOR and AND gates. Given a fixed generating polynomial, the $\odot$ operations will obviously be no longer needed. Moreover, in an implementation with a fixed generating polynomial, the output of each $\odot$ operation will be equal to 0 if its G input is 0. This eliminates the need for the corresponding $\oplus$ operation. As an example, an LFSR with generating polynomial $G = x^8 + x^4 + x^3 + x^2 + 1$ is shown in Figure 2. In this paper, we use $\mathcal{P}$ to represent programmable LFSRs and $\mathcal{L}$ for LFSRs that work with a fixed generating polynomial.

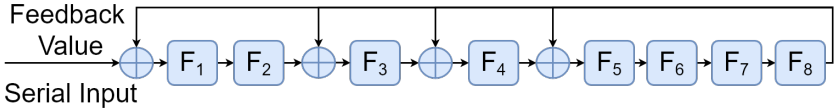

Figure 2. Galois and Fibonacci representations of an LFSR with generating polynomial $G = x^8 + x^4 + x^3 + x^2 + 1$.

An LFSR with n flip-flops and a primitive polynomial G guarantees to generate $2^n - 1$ different numbers $\{1, 2, \ldots, 2^n - 1\}$ in each $2^n - 1$ consecutive clock cycles by an order $r_1, r_2, \ldots, r_{2^n-1}$, which depends on G. This capability makes it possible to use LFSRs in the design of pseudo-random number generators.

The definition of parallel LFSR (as suggested by the related literature) is a little tricky. A parallel LFSR with generating polynomial G and a sampling rate equal to j generates the sequence $r_1, r_{j+1}, r_{2j+1}, \ldots$ where $r_1, r_2, \ldots, r_{2^n-1}$ is the sequence generated by a serial LFSR with the same generating polynomial, and r_1 is equal to the seed. As stated by the definition, a parallel LFSR skips $j - 1$ consecutive random numbers and outputs the j^{th} one in each invocation or clock cycle.

For a positive integer j, an n-bit j-parallel LFSR $\mathcal{L}_n^j(G, M)$ is defined by Equation (2):

$$\forall k \geq 0 : \mathcal{S}^k(\mathcal{L}_n^j(G, M)) = \mathcal{S}^{kj}(\mathcal{L}_n(G, M)). \tag{2}$$

In Equation (2), j is referred to as the sampling rate or the degree of parallelism. Parallel LFSRs are designed to achieve higher performance [13].

The LFSR $\mathcal{L}_n^1(G, M) = \mathcal{L}_n(G, M)$ is sometimes called a serial LFSR in order to distinguish it from parallel LFSRs. Similar to the case of serial LFSRs, parallel LFSRs can be implemented in a programmable way. We represent a programmable n-bit j-parallel LFSR by $\mathcal{P}_n^j(G, M))$. Programmable LFSRS have been of particular interest for designers during the last few decades [14]. Figure 3 shows the block diagram of $\mathcal{P}_n^j(G, M))$.

In Figure 3, the state transition logic calculates $\mathcal{S}^1(\mathcal{P}_n^j(G, M)) = \mathcal{S}^j(\mathcal{P}_n(G, M))$ using $\mathcal{S}^0(\mathcal{P}_8(G, M))$, G and M in the first clock cycle. Afterwards, the state calculated in each cycle is considered as the initial state for the next cycle. In the rest of this paper, the term *"LFSR"* refers to non-programmable serial Galois-type LFSR, unless we clearly specify another type of LFSR. An LFSR with n flip-flops and a primitive polynomial G guarantees to generate $2^n - 1$ different numbers $\{1, 2, \ldots, 2^n - 1\}$ in each $2^n - 1$ consecutive clock cycles by an order $r_1, r_2, \ldots, r_{2^n-1}$, which depends on G. This helps LFSRs be used as random number generators. Figure 4 shows how LFSRs and parallel LFSRs can be used to build a parallel RNG.

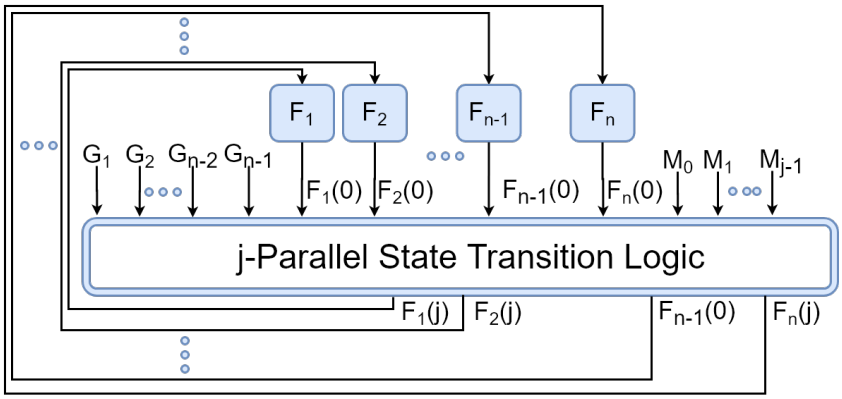

Figure 3. The block diagram of a programmable n-Bit j-parallel LFSR.

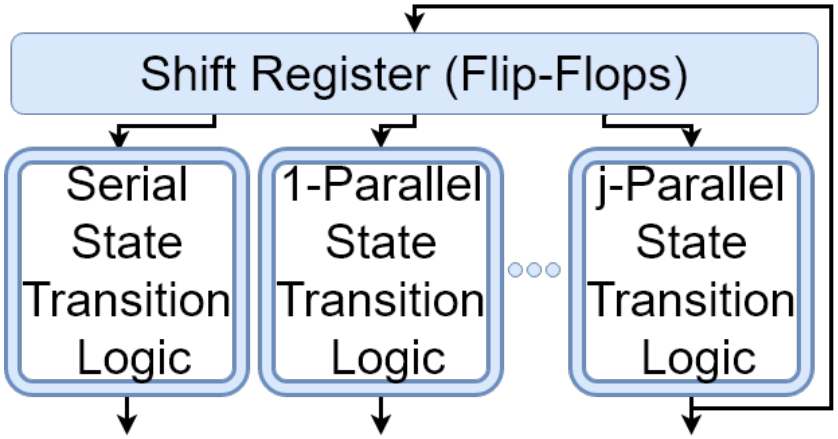

Figure 4. A j-parallel RNG based on parallel LFSRs.

1.1.3. S-Restricted RNGs

Assume the set S of positive integers, not necessarily consisting of consecutive integers. Furthermore, assume a random sequence of positive integers $\mathcal{I} : (i_l)_{l=1}^{t}$ in a way that $\{i_l | i_l \in \mathcal{I}\} = \mathcal{S}$. Any RNG capable of generating $\mathcal{I}$ is referred to as an S-restricted RNG. Simply put, S-restricted RNG, which takes an arbitrary set S of integers as the input, and randomly generates the elements of S without generating any other random number. The notion of S-restricted RNG is introduced for the first time in this paper. We use it as part of our solution to the problem of *random object generation*.

1.1.4. Latin Squares

A Latin square of order q contains $1, 2, \ldots, q$ in each row and each column in a way that no number is repeated in a row or a column. Latin squares are of many applications in cryptography [15–17] and related areas [18]. Table 1 shows a Latin square of order 10.

1.2. Organization

The rest of this paper is organized as follows: Section 2 formalizes the problem of *random object generation*, discusses the challenges raised by the problem and presents our proposed solution based on S-restricted RNGs and encoding schemes. Section 3 reviews related research works. This section compares the most relevant works with our work in this paper. Section 4 presents a novel method based on integer compositions for designing parallel LFSRs. Section 5 proposes an architecture for designing S-restricted RNGs using parallel LFSRs. Section 6 presents the case study. The first subsection in this section presents a novel encoding scheme for Latin squares. The second subsection designs a circuit for generating random Latin squares of order 4. Lastly, Section 7 concludes the paper and suggests further research.

Table 1. A sample Latin square of order 10.

1	8	9	10	2	4	6	3	5	7
7	2	8	9	10	3	5	4	6	1
6	1	3	8	9	10	4	5	7	2
5	7	2	4	8	9	10	6	1	3
10	6	1	3	5	8	9	7	2	4
9	10	7	2	4	6	8	1	3	5
8	9	10	1	3	5	7	2	4	6
2	3	4	5	6	7	1	8	9	10
3	4	5	6	7	1	2	10	8	9
4	5	6	7	1	2	3	9	10	8

2. Preliminaries

In this section, we preset some preliminary discussions. In Section 2.1, we state the problem of *random object generation*. Section 2.2 discusses the challenges raised by the formulated problem. Section 2.3 introduces our approach to solving the problem. Lastly, Section 2.4 explains the novelties and achievements of this paper.

2.1. Problem Statement

For $n \in \mathbb{N}$, let $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ and $\mathcal{S} = \{s_1, s_2, \ldots, s_n\} \subset \mathbb{Z}^+$ represent a set of objects and a set of non-negative integers, respectively. Furthermore, let $\mathcal{M} : \mathcal{O} \mapsto \mathcal{S}$ be an injective and surjective map. In the presence of $\mathcal{M}$, we refer to each element of S as a *valid object code*, and any other integer is considered as an *invalid object code*. The problem of *random object generation* is defined as the problem of generating a random sequence of positive integers $\mathcal{I} : (i_l)_{l=1}^{t}$ in a way that $\{i_l | i_l \in \mathcal{I}\} = \mathcal{S}$. Solving this problem requires an RNG that generates the sequence $\mathcal{I}$. For $n \in \mathbb{N}$, let $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ and $\mathcal{S} = \{s_1, s_2, \ldots, s_n\} \subset \mathbb{Z}^+$ representing a set of objects and a set of non-negative integers, respectively. Furthermore, let $\mathcal{M} : \mathcal{O} \mapsto \mathcal{S}$ be an injective and surjective one-to-one map, playing the role of an encoding scheme. In the presence of $\mathcal{M}$, we refer to each element of S as a *valid object code*, and any other integer is considered as an *invalid object code*. The problem of *random object generation* is defined as the problem of generating a random sequence of positive integers $\mathcal{I} : (i_l)_{l=1}^{t}$ in a way that $\{i_l | i_l \in \mathcal{I}\} = \mathcal{S}$. Solving this problem requires an RNG that generates the sequence $\mathcal{I}$.

2.2. Challenges

A naive solution to the problem of *random object generation* is to assign numerical codes to objects, let an RNG generate random numbers and interpret the generated random numbers as object codes. However, there are some challenges. These challenges are discussed below.

1. The first challenge here is to find or to build an RNG for which the set of possible outputs is exactly equal to the set of codes assigned to the objects. For example, for random permutations, we require encoding schemes that assign codes from $S = \{1, 2, \ldots, q!\}$ to permutations of q objects [19]. Each of the mentioned codes can be represented by $l = \lceil \log q! \rceil$ binary digits. However, a l-bit RNG usually generates all 2^l elements of $\{0, 1, \ldots, 2^l - 1\}$, while $2^l - q! = 2^{\lceil \log q! \rceil} - 2^{\log q!}$ of them are *invalid*. We address this challenge by introducing S-restricted RNGs, which generate random numbers drawn from a given set S.
2. The second challenge is that the encoding scheme will most likely be different from one problem to another. For example, an encoding scheme proposed for passwords

may not be applicable to CAPTCHAs as the statistical properties of valid passwords are totally different from those of valid CAPTCHAs.

3. Third, the S-restricted RNG will be dependent on the encoding scheme and consequently on the target set of objects. This component will vary from Latin squares of order n to the same squares of order $m \neq n$. We address this challenge as well as the above one via proposing the use of reconfigurable S-restricted RNGs. In our case study, this challenge is resolved in two ways. First, our proposed architecture for designing S-restricted RNG is capable of adopting any kind of parallel RNG. Second, we use programmable parallel LFSRs instead of fixed-polynomial parallel LFSRs to improve the reconfigurability of the design. Existing methods for designing parallel LFSRs work only with a fixed generating polynomial [20,21]. In addition to inadequate reconfigurability, fixed-polynomial LFSRs make the system more vulnerable against some well-known security attacks [22].

2.3. Problem Solving Approach

Figure 5 shows our solution to the problem of *random object generation*.

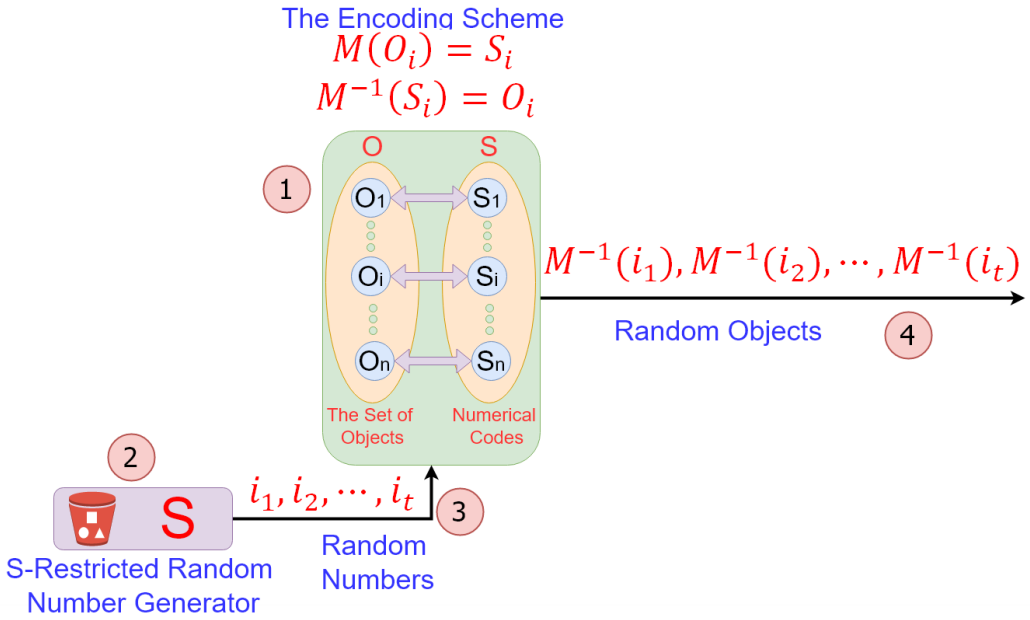

Figure 5. A solution to the problem of *random object generation*. Using S-Restricted RNGs: (1) An encoding scheme is created that assigns the set S of numeric codes to the set O of objects using a one-to-one (reversible) mapping. (2) An S-restricted random number generator is designed that is capable of generating elements of S in a random way. (3) The S-restricted random number generator generates random numbers. (4) The generated random numbers are converted to random objects using the reverse of the encoding map.

The solution illustrated in Figure 5 depends on an encoding scheme and an S-restricted RNG. The encoding scheme is inevitably dependent on the specific set of target objects. We compensate this dependence via the use of flexible S-restricted RNGs. This way, our method can be used for solving any kind of random non-numerical object generation problem with the help of a proper encoding scheme and an s-restricted RNG that generates exactly the set of numeric codes assigned to the objects.

In our proposal, the S-restricted RNG is constructed of a few parallel RNGs, which simultaneously generate multiple random numbers, along with a selection mechanism that chooses a *valid* object code among the generated random numbers. As a case study, we will show later how S-restricted RNGs can be used to generate Latin squares. We use parallel LFSRs to build parallel RNGs and subsequently S-restricted RNGs for generating Latin squares of order 4. We have chosen LFSRs due to their lightweight implementation.

Figure 6 demonstrates our proposed architecture for the design of S-restricted RNGs. This architecture is based on parallel RNGs. A parallel RNG is capable of creating multiple simultaneous random numbers at its output [23]. In addition to parallel RNGs, the proposed architecture uses an *Invalid Run length calculation module* as well as a *feedback selection module*. In Figure 6, the delay module is initialized to d_l. In each clock, the parallel RNG generates $\{d_l, r_{l+1}, d_{l+2}, \ldots, d_{l+j-1}\}$ where j is the sampling rate of the parallel RNG. $r_i = r$. An invalid run length equal to k allows $i_l = d_{l+k}$ in the output and loads it on the feedback loop as well.

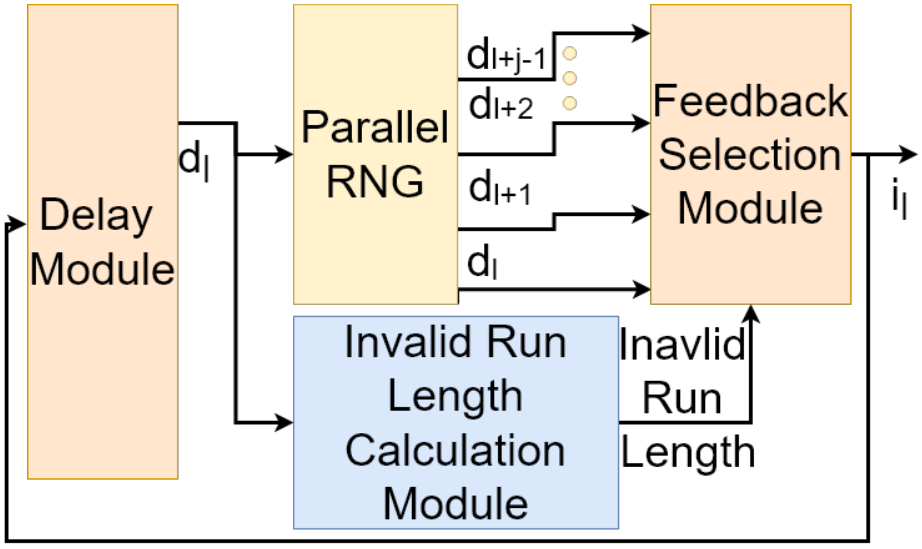

Figure 6. The architecture of an S-restricted RNG based on a parallel RNG.

2.4. Novelties and Contributions

The contributions of this paper can be listed as follows:

1. In this paper, we unify all problems related to random generation of non-numerical entities for the first time. We bring all these problems under a single umbrella via posing and formulating the general problem of *random object generation* (Section 2.1).
2. This paper is the first to propose a solution suitable for generating random instances of any kind of non-numerical entity. Our solution depends on two core components. The first component is a proper encoding scheme assigning a unique code to every individual object. The second component is an RNG capable of generating random numbers restricted to the set of assigned numeric codes (Section 2.3).
3. In this paper, we propose a novel approach based on integer compositions for automatic design of programmable parallel LFSRs (Section 4);
4. In this paper, we introduce the notion of S-restricted, RNGs for the first time. Moreover, we present a novel method for designing S-restricted RNGs using parallel LFSRs (Section 5);
5. This paper presents the first encoding scheme for Latin squares. This encoding is essentially en extended variant of Lehmer's code previously proposed for encoding permutations of a set of objects (Section 6.1);
6. We propose the first circuit for generating random Latin squares of degree 4 (Section 6.2).

Figure 7 shows the achievements of this paper and how they are connected to each other.

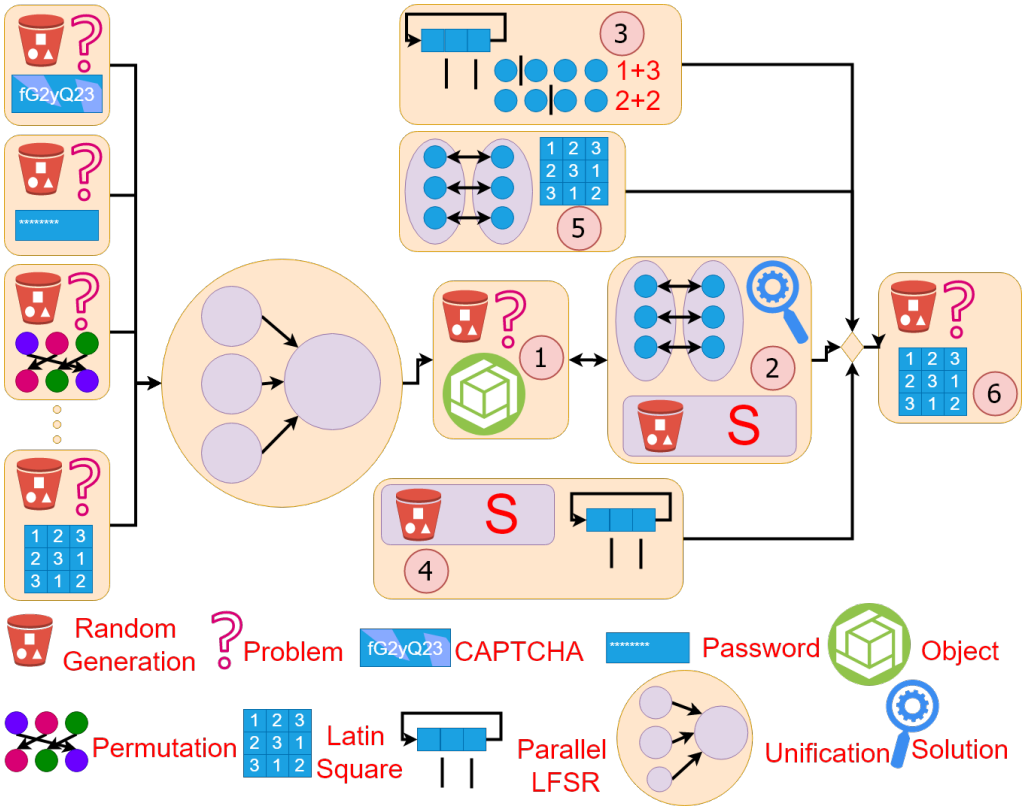

Figure 7. The achievements of this paper: (1) We unify problems like random CAPTCHA generation, random password generation, random permutation generation and random Latin square generation. We formulate the unified problem as the *random object generation* problem. (2) We present a solution based on proper encoding and S-restricted random number generators for the problem of *random object generation*. (3) We present an encoding scheme for Latin squares. (4) We propose a method based on integer compositions for designing parallel LFSRs. We propose a method based on parallel LFSRs for designing S-restricted random number generators. (6) As a case study, we propose a logic circuit for generating Latin squares of order 4.

3. Background and Related Works

In this section, we take a quick look at the background of research on random number generation as well as the random generation of non-numerical entities. We compare the most relevant research works with our work in this paper.

3.1. Random Numbers

Random number generation is an old problem. Random bit generation [24], random sequence generation [25], and random vector generation [26] can be mentioned as variants of this problem. There are two main types of random numbers, namely true-random numbers and pseudo-random numbers.

True random numbers are generated using an unpredictable physical object, phenomenon or process referred to as the source of randomness. Among these sources, one may refer to the following:

- Noises [27,28];
- Waves [29,30];
- Hardware Sources [31].

Pseudo-random numbers are completely computer-generated. They are generated using computer algorithms or devices. Pseudo-random numbers are used in a variety of applications ranging from sensor networks [32] to cryptography [33]. Different approaches have been used for designing pseudo-random number generators [34,35]. Moreover, several

enabling technologies including artificial intelligence [36,37], fuzzy logic [38] and chaos theory [39] are used for this purpose.

3.2. LFSRs and Parallel LFSRs

Among existing random umber generators, LFSRs are most relevant as we are using them in our case study. The research community has considered LFSRs as promising choices for random number generation and cryptographic purposes because of their low area and power consumption as well as their high throughput [40,41]. They have been widely used for both pseudo-random [42] and true-random number [43,44] generation. Different variants of LFSRs have been used for this purpose [22,45,46]. LFSRs are particularly used for random key generation in stream ciphers [47]. In addition to serial LFSRs, parallel LFSRs have been of interest to the research community in recent years [48]. They have been widely used in random number generation [49] and many other applications [50,51].

3.3. Random Non-Numerical Entities

Generating random non-numeric entities dates back to the last few decades [52–54]. Random network coding [55], random decision trees [56] or random device IDs [57] are required in different domains. Just as an example, random deadlines [58], random power levels [59] and random linear network codes [60] are required to be generated in IoT applications for traffic management, attack resistance and bandwidth management purposes, respectively. Particularly, random permutations [61], random passwords [1], and random CAPTCHAs [2] play significant roles in cryptography.

Different Approaches can be used to generate random non-numeric entities [62,63]. A variety of enabling technologies including chaos theory [64], information theory [65] and artificial intelligence [66] are utilized in this area.

Random Latin Squares

Latin squares appear as part of our case study in this paper. This is why we would like to discuss them separately. Random Latin squares have been widely used in IoT environments for channel access arbitration [67], encryption [15], secret sharing [18], etc. Generating random Latin squares is a critical problem in the realm of combinatorial cryptography [6,7,68,69]. Currently, there is no systematic solution for this problem.

3.4. Most Relevant Works

According to the above discussions, many existing research reports are somehow relevant to this study. However, the most relevant research works are those focusing on restricted random number generation or generating random non-numerical entities using RNGs. These works are studied in the following:

3.4.1. Restricted RNGs

By a *restricted RNG*, we mean an RNG that generates a certain subset of $\{0, 1, 2, \ldots, 2^m - 1\}$. The literature comes with some RNGs of this type.

As an example, one may refer to *constrained random number generators*, which have been of interest to researchers in recent years [70]. A constrained random number generator is defined as follows [71]. Let x be an element of the Cartesian product χ^n of a given set χ. Constrained RNG uses a sequence of random numbers subject to a distribution v on χ^n defined by Equation (3):

$$v(x) = \begin{cases} \frac{\mu(x)}{\mu(\{x:Ax=c\})}, & Ax = c; \\ 0, & Otherwise. \end{cases} \tag{3}$$

In Equation (3), μ is a probability distribution on χ^n, A is a function $A : \chi^n \to \{Ax : x \in \chi^n\}$, and $c \in \{Ax : x \in \chi^n\}$. Constrained RNGs are used in channel coding [72].

A constrained RNG is similar to an S-restricted RNG in that both of them filter the output of a traditional RNG. However, the difference is that a constrained RNG uses the value of a function to filter the generated random numbers, while an S-restricted RNG checks them against a given set S of valid numbers.

Another relevant research has been reported in [23], which presents a VLSI (Very Large Scale Integration) design for a parallel RNG, and uses it to generate random numbers drawn from an interval of integers. This work is different from ours in three aspects. First, the parallel RNG designed in [23] generates multiple streams of random numbers, while our proposed S-restricted RNG architecture depends on parallel RNGs that generate multiple random numbers from the same stream as their output each time. Second, their designed RNG generates random numbers is a contiguous interval of integers, while our S-restricted RNG can generate elements of any arbitrary set S. Third, in our work, parallel RNGs are based on parallel LFSRs and can be automatically designed with arbitrary generating polynomials.

3.4.2. RNGs and Random Non-Numerical Entities

Some researchers have proposed methods based on random number generators for generating random decision trees [56], random permutations, and random device IDs [73]. However, none of these methods are capable of being applied on all kinds of non-numerical entities.

3.5. Motivations

There are a wide range of problems, each of which can be considered as a spacial instance of the problem posed in this paper: *random object generation*. However, they have not been formulated as a single general problem. The reason is the lack of a general-purpose solution applicable to all of these problems. This is the gap we are addressing in this paper. We formalize the general problem and provide a general solution for it.

4. Automatic Design of Parallel LFSRs Using Integer Compositions

Existing methods for designing parallel LFSRs work only with a fixed generating polynomial [20,21]. However, in security-related applications, we are usually interested in randomizing the generating polynomial. Thus, we need an automatic method for designing programmable parallel LFSRs.

In this section, we present a framework for automatic design of parallel LFSRs using the mathematical properties of integer compositions. This framework consists of two parts. In the first part, we present a method for automatic derivation of Reed–Muller expressions describing programmable parallel LFSRs using compositions and palindromes. Afterwards, we modify the framework to describe (non-programmable A.K.A fixed-polynomial) parallel LFSRs by Reed–Muller expressions using S-restricted palindromes. In the second part of the framework, we present procedures for automatic generation of compositions, palindromes and S-restricted palindromes. For $j \in \{0, 1, \ldots, n-1\}$, let $b_j(i)$ represent the $(n-j)^{th}$ most significant digit of $B_n(i) = b_{n-1}(i)b_{n-2}(i)\ldots b_j(i)\ldots b_1(i)b_0(i)$ being the binary representation of $i \in \mathbb{Z}^+$ in n digits. The Reed–Muller canonical form represents a Boolean function $f(x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$ in the form of Equation (4),

$$f(x_{n-1}, x_{n-2}, \ldots, x_1, x_0) = \sum_{i=0}^{2^n-1} \left(a_i \prod_{b_j(i)=1} x_j \right). \tag{4}$$

In Equation (4), the addition and multiplication operations are defined over $GF(2)$. A Boolean function described in the Reed–Muller canonical form can be uniquely converted to other canonical forms such as CNF (Conjunctive Normal Form) and DNF (Disjunctive Normal Form). An advantage of Reed–Muller canonical form is that it does not need any NOT gates to be implemented. This canonical form is commonly used in the description and design of logic circuits [74].

In the second part of our framework for designing parallel LFSRs, we present procedures for automatic generation of compositions, palindromes and S-restricted palindromes.

4.1. Expression Derivation

In this subsection, we take an inductive approach to derive a Boolean expression for describing a programmable parallel LFSR. We begin with Equation (1) and expand it. Induction helps us describe consecutive expansions of this equation using palindromic integer composition.

Before beginning to derive the expressions, let us prove Lemma 1, which presents a procedure for creating $\mathcal{C}(i+1)$ (the set of compositions of $i+1$) given $\mathcal{C}(i)$ (the set of compositions of i).

Lemma 1. *Procedure CreateNextComp in Algorithm 1 returns* $\mathcal{C}_{i+1} = \mathcal{C}(i+1)$ *provided that* $\mathcal{C}_i = \mathcal{C}(i)$.

Algorithm 1 create $\mathcal{C}(i+1)$ given $\mathcal{C}(i)$

Requires: $\mathcal{C}_i = \mathcal{C}(i)$.
Ensures: $\mathcal{C}_{i+1} = \mathcal{C}(i+1)$.

```
1: procedure CreateNextComp(C_i)
2:     C_{i+1} = ∅
3:     for all C ∈ C_i do
4:         C_{i+1} ← C_{i+1} ∪ {(first(c) + 1); f^-(C)}
5:         C_{i+1} ← C_{i+1} ∪ {1; C}
6:     end for
7:     return C_{i+1}
8: end procedure
```

Proof. Since $\mathcal{C}_i = \mathcal{C}(i)$, it is obvious that $\forall C \in \mathcal{C}_i : (\text{first}(c)+1); f^-(C) \in \mathcal{C}(i+1) \wedge 1; C \in \mathcal{C}(i+1)$ Thus, $\mathcal{C}_{i+1} \subset \mathcal{C}(i+1)$. On the other hand, $\forall C' \in \mathcal{C}(i+1) : (\text{first}(C') = 1 \wedge f^-(C') \in \mathcal{C}(i)) \vee (\text{first}(C') - 1); f^-(C') \in \mathcal{C}(i)$. Thus, $\mathcal{C}(i+1) \subset \mathcal{C}_{i+1}$, and the lemma is proved. □

Equation (5) is immediate from Lemma 1:

$$\forall i > 0 : \mathcal{C}(i+1) = \{(\text{first}(c)+1); f^-(C) | C \in \mathcal{C}(i)\} \cup \{1; C | C \in \mathcal{C}(i)\} \tag{5}$$

Now, let us continue by proving Theorem 1, which gives a Reed–Muller description of $\mathcal{P}_n^j(G, M)$ using integer compositions.

Theorem 1. *A Reed–Muller expression of* $\mathcal{P}_n^j(G, M)$ *(shown in Figure 3) is given by Equation (6), wherein* $C' = f^-(C)$, $G_t = 0$ *for* $t \leq 0$ *and* $F_t = M_{-t}$ *for* $t \leq 0$.

$$F_i(j) = F_{i-j}(0) + \sum_{r=0}^{j-1} \sum_{C \in \mathcal{C}(j-r)} \left(G_{i-\text{first}(C)} \left(\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)} \right) F_{n-r}(0) \right), \tag{6}$$

where $i \in \{1, 2, \ldots, n\}$.

Proof. We take an induction-based approach to prove Theorem 1. For $j = 1$, considering $\mathcal{C}(1) = \{(1)\}$, $\text{first}((1)) = 1$ and $\text{length}(f^-((1))) = 0$, Equation (6) is converted to

Equation (1). Thus, Equation (6) holds for $j = 1$. On the other hand, if this equation holds for $j = q$, we can use it to derive Equation (7) via replacing i by $i - 1$.

$$F_{i-1}(q) = F_{i-(q+1)}(0) + \sum_{r=0}^{q-1} \sum_{C \in \mathcal{C}(q-r)} \left(G_{i-(\text{first}(C)+1)} \left(\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)} \right) F_{n-r}(0) \right), \tag{7}$$

where $i \in \{2, 3, \ldots, n+1\}$.

In the derivation of Equation (7), we have replaced $i - 1 - q$ by $i - (q+1)$, and $i - 1 - \text{first}(C)$ by $i - (\text{first}(c) + 1)$. Furthermore, Equation (8) can be obtained by substituting n for i in Equation (6) (rewritten for q instead of j) and multiplying the equation by G_{i-1}:

$$G_{i-1}F_n(q) = G_{i-1}F_{n-q}(0) + \sum_{r=0}^{q-1} \sum_{C \in \mathcal{C}(q-r)} \left(G_{i-1} \left(G_{n-\text{first}(C)} \prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)} \right) F_{n-r}(0) \right). \tag{8}$$

Considering Equations (1) and (5), we can add Equations (7) and (8) together to obtain Equation (9) via some simple algebraic operations:

$$F_i(q+1) = F_{i-(q+1)}(0) + \sum_{r=0}^{q} \sum_{C \in \mathcal{C}(q+1-r)} \left(G_{i-\text{first}(C)} \left(\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)} \right) F_{n-r}(0) \right), \tag{9}$$

where $i \in \{1, 2, \ldots, n\}$.

Equation (9) states that, if Equation (6) holds for $j = q$, it will hold for $j = q + 1$ as well, and the theorem is proved.

□

It is obvious that $\mathcal{P}_n^j(G, M)$ can be implemented using logical AND and XOR gates on the basis of the Reed–Muller representation given in Equation (6). Corollary 1 presents an equivalent Reed–Muller description of $\mathcal{P}_n^j(G, M)$ using palindromes. This can be considered as a simplification to the representation of Theorem 1 because $|\mathcal{C}_P(x)| < |\mathcal{C}(x)|$ for every $x > 2$ and $|\mathcal{C}_P(x)| = |\mathcal{C}(x)|$ for every $x \in \{1, 2\}$.

Corollary 1. *A Reed–Muller expression of $\mathcal{P}_n^j(G, M)$ (shown in Figure 3) is given by Equation (10), wherein $C' = f^-(C)$.*

$$F_i(j) = F_{i-j}(0) + \sum_{r=0}^{j-1} \sum_{C \in \mathcal{C}_P(j-r)} \left(G_{i-\text{first}(C)} \left(\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)} \right) F_{n-r}(0) \right), \tag{10}$$

where $i \in \{1, 2, \ldots, n\}$.

Proof. In order to prove Corollary 1, we note that

$$\forall C \in \mathcal{C}(j-r) : C \notin \mathcal{C}_P(j-r) \Rightarrow \text{first}(C); C' = \text{first}(\tilde{C}); \tilde{C}' \Rightarrow$$

$$G_{i-\text{first}(C)}\left(\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)}\right)+$$
$$G_{i-\text{first}(\tilde{C})}\left(\prod_{u=1}^{\text{length}(\tilde{C}')} G_{n-\lambda(\tilde{C}',u)}\right)=0.$$

Thus,

$$\sum_{C\in\mathcal{C}(j-r)}\left(G_{i-\text{first}(C)}\left(\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)}\right)F_{n-r}(0)\right)=$$
$$\sum_{C\in\mathcal{C}_P(j-r)}\left(G_{i-\text{first}(C)}\left(\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)}\right)F_{n-r}(0)\right).$$

□

Given a fixed generating polynomial, $\mathcal{P}_n^j(G,M)$ can be converted to $\mathcal{L}_n^j(G,M)$. To show how this can be done, let us prove Corollary 2, which gives a Reed–Muller representation of $\mathcal{L}_n^j(G,M)$ using S-restricted palindromes.

Corollary 2. *A Reed–Muller expression of $\mathcal{L}_n^j(G,M)$ is given by*

$$F_i(j)=F_{i-j}(0)+$$
$$\sum_{r=0}^{j-1}\sum_{C\in\mathcal{C}_P^{S(j-r)}(j-r)}\left(G_{i-\text{first}(C)}\left(\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)}\right)F_{n-r}(0)\right), \tag{11}$$

where $i\in\{1,2,\ldots,n\}$ and $C'=f^-(C)$.

Proof. In Equation (10), it is obvious that $\exists i\in\mathcal{C}_P(j-r): G_{n-\lambda(C',u)}=0\Rightarrow\prod_{u=1}^{\text{length}(C')} G_{n-\lambda(C',u)}=0$. Thus, given a fixed generating polynomial G, the set $\mathcal{C}_P(j-r)$ can be replaced with $\mathcal{X}(j-r,G)=\{C|C\in\mathcal{C}_P(j-r)\wedge\forall u\in\{1,1,\ldots,\text{length}(C')\}: G_{n-\lambda(C',u)}=1\}$. On the other hand, for each $C\in\mathcal{C}_P(j-r)$, the set of summands in C is equal to that of C', except for $(j-r)$. Thus, $\mathcal{X}(j-r,G)=\mathcal{C}_P^{S(j-r,G)}(j-r)$, where $S(j-r,G)=\{(j-r)\}\cup\{x|G_{n-x}=1\}$. □

4.2. Generation Procedures

In this subsection, we complete our automatic parallel LFSR design framework by presenting procedures for generating compositions, palindromes and S-restricted palindromes of a positive integer n.

Let us begin with systematic generation of $\mathcal{C}(n)$. It can be done as shown by procedure *CreateComp* in Algorithm 2 via the use of procedure *CreateComp* in Algorithm 1, which was proved to be correct in Section 4.1.

Algorithm 2 Create $\mathcal{C}(n)$

Requires: n is a positive integer.
Ensures: $\mathcal{C}_n = \mathcal{C}(n)$.

```
1: procedure CreateComp(n)
2:     if n == 1 then
3:         C_n ← {(1)}
4:     else
5:         C_{n-1} ← CreateComp(n − 1)
6:         C_n ← CreateNextComp(C_{n-1})
7:     end if
8:     return C_n
9: end procedure
```

Lemma 2 introduces a procedure for systematic creation of $\mathcal{C}_P(n)$ for a positive integer n.

Lemma 2. *Procedure CreatePal in Algorithm 3 returns $\mathcal{C}_P(n)$ for a positive integer n.*

Algorithm 3 Create $\mathcal{C}_P(n)$

Requires: n is a positive integer.
Ensures: $\mathcal{C}_{pn} = \mathcal{C}(n)$.

```
1: procedure CreatePal(n)
2:     if n == 1||n == 2 then
3:         C_pn ← CreateComp(n)
4:     else
5:         C_pn ← ∅
6:         C_{pn-2} ← CreatePal(n − 2)
7:         for all C ∈ C_{pn-2} do
8:             T = (first(c) + 1); f⁻(C)
9:             C_pn ← C_pn ∪ {l⁻(T); (last(T) + 1)}
10:            C_pn ← C_pn ∪ {1; C; 1}
11:        end for
12:    end if
13:    return C_pn
14: end procedure
```

Proof. This lemma can be proved in a very similar way to the case of Lemma 1. □

Lemma 3 introduces a procedure for generating $\mathcal{C}_P^S(n)$ for a positive integer n and a set S of positive integers.

Lemma 3. *For a positive integer n and a set $S \subset$ of positive integers, procedure CreateSRestPal in Algorithm 4 returns $\mathcal{C}_P^S(n)$ if $\mathcal{C}_P^S(n) \neq \emptyset$ and $\{()\}$ (a set consisting of only an empty composition) if $\mathcal{C}_P^S(n) = \emptyset$.*

Algorithm 4 Create S-restricted palindromic compositions

Requires: n a positive integer, S a set of positive integers.

Ensures: $\mathcal{C}_P^S = \mathcal{C}_P^S(n)$ if $\mathcal{C}_P^S(n) \neq \emptyset$, $\mathcal{C}_P^S = \{()\}$ if $\mathcal{C}_P^S(n) = \emptyset$.

1: **procedure** CREATESRESTPAL(n,S)
2: $G \leftarrow 0$
3: $S_1 \leftarrow S$
4: **if** $S_1 == \emptyset$ **then**
5: $\mathcal{C}_P^S \leftarrow \{()\}$ ▷ A set consisting of an empty composition
6: **return**
7: **end if**
8: **if** $n \in S_1$ **then**
9: $\mathcal{C}_P^S \leftarrow \mathcal{C}_P^S \cup \{(n)\}$
10: $G \leftarrow 1$
11: **end if**
12: $\mathcal{C}_P^S \leftarrow \emptyset$
13: **for all** $s \in S_1 \setminus \{n\}$ **do**
14: **for all** $c \in CreateSRestPal(n - 2s, S_1 \setminus \{n\})$ **do**
15: $c' \leftarrow (s); c; (s)$ ▷ Concatenate s to the left and the right of c
16: $\mathcal{C}_P^S \leftarrow \mathcal{C}_P^S \cup \{c'\}$
17: $G \leftarrow 1$
18: **end for**
19: **end for**
20: **if** $G == 0$ **then**
21: $\mathcal{C}_P^S \leftarrow \{()\}$
22: **end if**
23: **return**
24: **end procedure**

Proof. In order to prove this lemma, we prove the following statements:

1. $\mathcal{C}_P^S \subset \mathcal{C}_P^S(n)$ if $\mathcal{C}_P^S(n) \neq \emptyset$.
   There are only two statements in Procedure *CreateSRestPa* that add compositions to $\mathcal{C}_P^S$: Statement 9 and Statement 16. Statement 9 adds (n), which is definitely an element of $\mathcal{C}_P^S(n)$ considering $n \in S$. Moreover, Statement 16 adds $c' = (s); c; (s)$, which is an element of $\mathcal{C}_P^S(n)$ since $c \in \mathcal{C}_P^S(n - 2s)$, and $s \in S$. Thus, whatever Procedure *CreateSRestPa* adds to C is an element of $\mathcal{C}_P^S(n)$.
2. $\mathcal{C}_P^S(n) \subset \mathcal{C}_P^S$ if $\mathcal{C}_P^S(n) \neq \emptyset$. $P_s(n, S)$ may consist of two kinds of compositions.
   - $c_1 = (n)$: This composition is added by Statement 9 if $n \in S$.
   - $\{c_2 | \text{first}(c_2) = \text{last}(c_2) = s \in S \setminus \{n\}, f^-(l^-(c_2)) \in \mathcal{C}_P^{S\setminus\{n\}}(n - 2s)\}$: All of these compositions are added by Statement 16.

   Thus, every element of $\mathcal{C}_P^S(n)$ is guaranteed to be generated by Procedure *CreateSRestPa*.
   converted to $()$.
3. $\mathcal{C}_P^S = \{()\}$ if and only if $\mathcal{C}_P^S(n) = \emptyset$.
   The above two statements show that $\mathcal{C}_P^S = \mathcal{C}_P^S(n)$. It is immediate that, if $\mathcal{C}_P^S(n) = \emptyset$, Procedure *CreateSRestPa* will generate no composition. Since 1 is assigned to G only after adding compositions to $\mathcal{C}_P^S$ (by Statements 9 and 16), Statement 21 sets $\mathcal{C}_P^S = \{()\}$ if $\mathcal{C}_P^S(n) = \emptyset$. On the other hand, $\mathcal{C}_P^S$ will not be equal to $\{()\}$ if $\mathcal{C}_P^S(n) \neq \emptyset$ because $G = 1$ is executed after generating each composition.

□

5. Designing the *S*-Restricted Random Number Generator

In this section, we propose an architecture for designing *S*-restricted RNGs. The architecture is based on parallel RNGs that generate multiple simultaneous random numbers. While any kind of parallel RNGs can be used in the proposed architecture, we use parallel RNGs constructed of parallel LFSRs to explain the design procedure. The rationale underlying this choice is that LFSRs (and parallel LFSRs) are simple devices with lightweight implementations using a few flip-flops along with a few logical gates, the layout of which is defined by a generating polynomial.

An LFSR $\mathcal{L}_n(G)$ of degree n with generating polynomial G can be used to implement an RNG $\mathcal{R}_n(G)$ that periodically generates a sequence of 2^n-1 mutually different values $C_y(\mathcal{R}) = r_0, r_1, \ldots, r_{2^n-3}, r_{2^n-2}$ provided that $S^0(\mathcal{L}_n(G)) \neq 000\ldots00$ [75]. In this case, each state of $\mathcal{L}_n(G)$ is considered as a random number generated by $\mathcal{R}_n(G)$. We define $\Delta(\mathcal{R}_n(G)) = \{r_0, r_1, \ldots, r_{2^n-3}, r_{2^n-2}\}$. Moreover, we define the period length of $\mathcal{R}_n(G)$ as the length of the sequence $C_y(\mathcal{R})$, and represent it $|C_y(\mathcal{R}_n(G))| = |\Delta(\mathcal{R}_n(G))|$.

For a positive integer number n and a set $S \subset \{1, 2, \ldots, 2^{n-1}\}$, we define an (LFSR-based) *S*-restricted RNG $\mathcal{R}_n^S(G)$ as an RNG, for which $\Delta(\mathcal{R}_n^S(G)) = S$. We refer to S as the set of valid states for $\mathcal{R}_n^S(G)$. Moreover, we refer to $I(\mathcal{R}_n^S(G))) = \{1, 2, \ldots, 2^{n-1}\} \setminus S$ as the set of invalid states of $\mathcal{R}_n^S(G))$.

A traditional LFSR-based RNG $\mathcal{R}_n(G)$ is obviously incapable of serving as an *S*-restricted RNG as it will generate some invalid outputs (outputs which are not valid numeric codes assigned to the objects), and $C_y(\mathcal{R}_n(G))$ will consist of alternate runs of valid and invalid numbers (valid runs and invalid runs) unless $S = \{1, 2, \ldots, 2^n - 1\}$. Figure 8 compares $\mathcal{R}_n(G)$ with $\mathcal{R}_n^S(G))$.

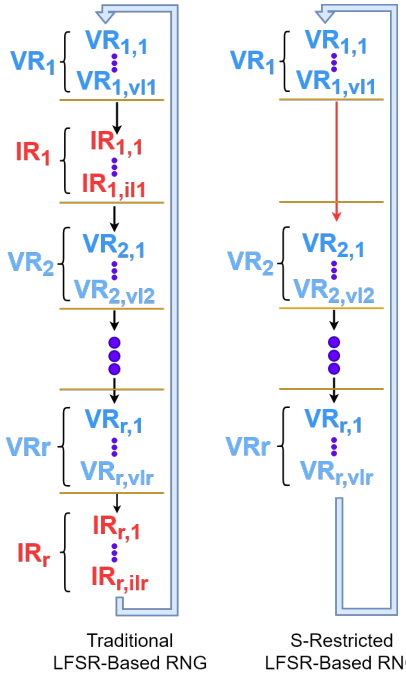

Figure 8. A comparison between traditional and *S*-restricted LFSR-based RNGs.

In Figure 8, valid runs $(VR_1, VR_2, \ldots, VR_r)$ and invalid runs $(IR_1, IR_2, \ldots, IR_r)$ are highlighted by blue and red colors, respectively. In this figure, vl_i and il_i represent the length of the i^{th} valid run and that of the i^{th} invalid run, respectively. This figure shows how invalid runs are bypassed by $\mathcal{R}_n^S(G))$.

A parallel LFSR $\mathcal{L}_n^j(G)$ with a sampling rate of j can bypass $j-1$ consecutive states generated by $\mathcal{L}_n^j(G)$ in each clock cycle. Thus, a number of parallel LFSRs with different degrees of parallelism can be used to bypass invalid runs with different lengths. In this

section, we use this idea for the design of S-restricted RNGs. Figure 9 shows our proposed architecture for the design of $\mathcal{R}_n^S(G)$).

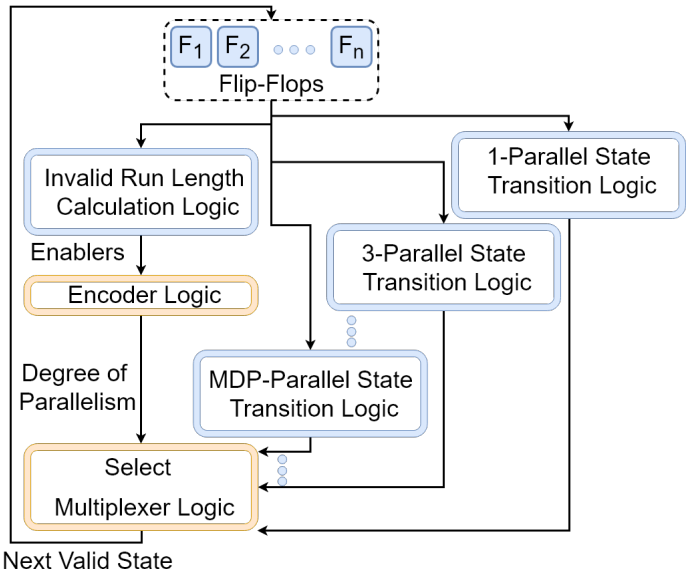

Figure 9. The proposed architecture for S-restricted LFSR-based RNG.

As shown in Figure 9, the proposed architecture consists of a serial (1-parallel) LFSR ($\mathcal{L}_n(G)$) and a set of parallel LFSRs ($\mathcal{L}_n^2(G)$, $\mathcal{L}_n^3(G)$,...,$\mathcal{L}_n^{MIRL}(G)$), which share n flip-flops along with an *Invalid Run Length Calculation Logic* (ILCL) and a multiplexing logic, which control the feedback loop. In this figure, MDP (Maximum Degree of Parallelism) is calculated as $MDP = MIRL + 1$, and MIRL (Maximum Invalid Run Length) is calculated as $MIRL = Max\{il_1, il_2, \ldots, il_r\}$. The architecture includes a j-parallel state transition logic if and only if $\exists v \in \{1, 2, \ldots, r\} : il_v = j$.

In the architecture of Figure 9, each state transition logic circuit is fed by the outputs of the flip-flops $F_1, F_2, \ldots, F_n$ (the current state of the circuit). In each clock cycle, the ILCL generates (at most) MDP enabling signals (enablers), which we denote by $e_1, e_2, \ldots, e_{MDP}$. The signal e_1 will carry the logic value 1 if the current state of the circuit is a valid state. For $i \in \{2, 3, \ldots, MDP\}$, the enabler e_i will be active if the current state is the beginning of an invalid run of length $i - 1$. The enablers are encoded to the proper degree of parallelism using an encoding logic to provide proper select signals for the multiplexing logic. Finally, we have the multiplexing logic Multiplexes among the outputs of the state transition logic modules to feed the next valid state back into the flip-flops.

$IR_{1,1}, IR_{2,1}, \ldots, IR_{r,1}$ can be used to design logical signals $e_2, e_3, \ldots, e_{MDP}$, where e_t (if equal to 1) indicates that the RNG should bypass $t - 1$ invalid states. An encoder converts these signals to proper select inputs for the multiplexer.

6. Case Study: Random Latin Squares of Order 4

In this section, we first present a method for assigning numerical codes to Latin squares of an arbitrary order q. Afterwards, we design an S-restricted RNG that generates random valid codes for Latin codes of order 4.

6.1. Encoding Latin Squares

A Latin square λ of order q can be modeled by $\langle \lambda_1, \lambda_2, \ldots, \lambda_q \rangle$, where $\forall i, j \in \{1, 2, \ldots, q\} : \lambda(i) = \lambda_i \in \Gamma(\{1, 2, \ldots, q\}) \wedge \lambda_i \in \mathcal{D}(\lambda_j)$. Thus, the methods that are used for encoding permutations [19] can be modified to encode Latin squares. In this paper, we modify Lehmer's encoding scheme [19] to assign numerical codes to Latin squares of degree q. For every $\Xi \subset \{1, 2, \ldots, q\}$, and every $\xi \in \Xi$, let us define $\Phi(\xi, \Xi) = |\{x | x \in \Xi \wedge x < \xi\}|$. Lehmer's

encoding scheme assigns a numerical code $0 \leq \mathcal{H}(\gamma) \leq q! - 1$ to each $\gamma \in \Gamma(\{1, 2, \ldots, q\})$ on the basis of Equation (12),

$$\mathcal{H}(\gamma) = \sum_{i=1}^{q} (q-i)! \mathcal{H}(\gamma(i), \{1, 2, \ldots, q\} \setminus \{\gamma(j) | 0 < j < i\}). \tag{12}$$

As an example, consider $\mathcal{H}(\langle 3, 1, 2 \rangle) = 2! * 2 + 1! * 0 + 0! * 0 = 4$. Now let us define the total order $\leqslant$ on $\Gamma(\{1, 2, \ldots, q\})$ as $\forall \gamma_1, \gamma_2 \in \Gamma(\{1, 2, \ldots, q\}) : \gamma_1 \leqslant \gamma_2 \Leftrightarrow \mathcal{H}(\gamma_1) \leq \mathcal{H}(\gamma_2)$. Using the total order $\leqslant$ on $\Gamma(\{1, 2, \ldots, q\})$, we define $\preceq$ on Λ_q as $\forall \lambda_1, \lambda_2 \in \Lambda_q : \lambda_1 \preceq \lambda_2 \Leftrightarrow \exists i \in \{1, 1, \ldots, q\} : (\forall\, 1 \leq j \leq i : \lambda_1(j) = \lambda_2(j) \wedge \lambda_2(i+1) \leqslant \lambda_1(i+1))$. In our proposed encoding scheme, the numerical code of each $\lambda \in \Lambda_q$ is calculated as shown in Equation (13),

$$\mathcal{F}(\lambda) = |\{x | x \in \Lambda_q \wedge x \preceq \lambda\}| + 1. \tag{13}$$

The codes assigned to Latin squares of order 4 can be seen in Appendix A.

6.2. S-Restricted RNG for $S = \{\mathcal{F}(\lambda) | \lambda \in \Lambda_4\}$

The first step towards the design of an S-restricted RNG is to list the set S of valid states, and to decide the size of the LFSRs (the number of flip-flops) accordingly. It is well known that $|\Lambda_q| = 576$. Therefore, using the method explained above, the elements of Λ_4 can be encoded by the elements of $S = \{1, 2, \ldots, 576\}$. Moreover, the size of each LFSR should be $\lceil \log 576 \rceil = 10$.

The second step is to decide the generating polynomial for the serial and parallel LFSRs. There are 60 different primitive polynomials of degree 10 over $GF(2)$. In this case study, we have chosen $G = x^{10} + x^3 + 1$ because it has the minimum number of non-zero coefficients, and this reduces the number of gates required to implement the state transition logic modules. One can obviously choose polynomials with more non-zero coefficients in order to improve resistance against algebraic attacks.

Now we need to analyze the period of $\mathcal{L}_{10}(G)$ to identify the beginning state and the length of each invalid run to decide the set of sampling rates and design the corresponding state transition logic modules. We have analyzed a whole period of $\mathcal{L}_{10}(G)$ beginning from the initial state 0000000001, and identified 221 invalid runs. These runs are shown in the supplementary file. Runs of lengths 1 through 8 as well as 9 and 10 have been identified in the period. The beginning states of the runs are shown in Table 2 along with the corresponding run lengths. In Table 2, the columns labeled "L" show the run lengths, and the ones labeled "Begin" show the beginning states of the runs.

Table 2. Run beginners.

L	Begin
1	580 608 834 656 596 706 752 686 694 848 602 726 654 632 640 578 720 598 650 738 760 588 836 610 728 630 688 666 742 584 612 832 582 626 732 638 674 744 622 646 854 642 736 700 684 842 658 740 862 680 618 730 758 604 696 702 644 592 594 724 710 600 668 676 704 664 692 648 616 614 850 660 708 590 718 860 698 750 712 620 838 624 716 844 852 714 754 764 856 670 858 662 652 840 628 606 682 746 762 766 672 586 722 756 634 734 690 748 846 636 678
2	585 749 653 641 577 589 665 757 637 685 645 725 705 621 669 593 733 597 761 717 649 721 845 697 849 693 737 601 661 833 857 765 837 677 753 673 613 841 605 689 625 581 617 709 729 609 701 861 853 681 657 629 741 633 745 713
3	747 739 859 643 635 651 851 731 587 843 619 627 835 667 715 603 723 755 699 691 763 595 707 683 659 611 579 675
4	599 615 679 583 663 631 647 695 743 727 711 759 839 855

L	Begin	L	Begin	L	Begin	L	Begin	L	Begin	L	Begin
5	623 591 655 751 687 719 847	6	671 735 863 607	7	703	8	767	9	-	10	639

Now we can design the components of the S-restricted RNG in the form of black boxes. These components are shown in Figure 10.

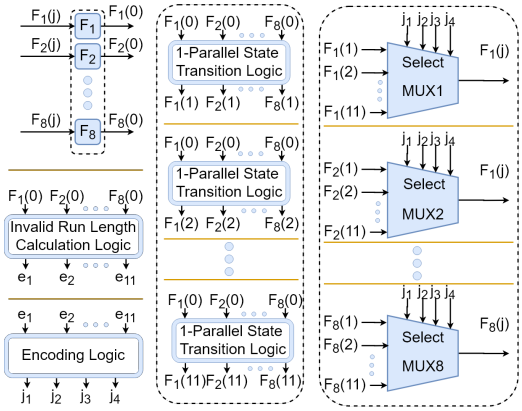

Figure 10. The components of S-restricted RNG for generating random Latin squares of order 4.

In the next step, the state transition logic should be designed. This can be done using the method presented in Section 4. Tables 3 and 4 show the logic expressions describing the output of the state transition logic modules.

Table 3. Logic descriptions of the state transition logic modules in the S-restricted RNG for generating random Latin squares of order 4 (Part 1).

j	Equation
2	$\begin{cases} F_1(2) = F_8(0), F_2(2) = F_9(0), F_3(2) = F_{10}(0) \\ F_4(2) = F_1(0) + F_8(0), F_5(2) = F_2(0) + F_9(0) \\ F_6(2) = F_3(0), F_7(2) = F_4(0), F_8(2) = F_5(0) \\ F_9(2) = F_6(0), F_{10}(2) = F_7(0), \end{cases}$
3	$\begin{cases} F_1(3) = F_7(0), F_2(3) = F_8(0), F_3(3) = F_9(0) \\ F_4(3) = F_{10}(0) + F_7(0), F_5(3) = F_1(0) + F_8(0) \\ F_6(3) = F_2(0) + F_9(0), F_7(3) = F_3(0) \\ F_8(3) = F_4(0), F_9(3) = F_5(0), F_{10}(3) = F_6(0) \end{cases}$
4	$\begin{cases} F_1(4) = F_6(0), F_2(4) = F_7(0), F_3(4) = F_8(0) \\ F_4(4) = F_9(0) + F_6(0), F_5(4) = F_{10}(0) + F_7(0) \\ F_6(4) = F_1(0) + F_8(0), F_7(4) = F_2(0) + F_9(0) \\ F_8(4) = F_3(0), F_9(4) = F_4(0), F_{10}(4) = F_5(0) \end{cases}$
5	$\begin{cases} F_1(5) = F_5(0), F_2(5) = F_6(0), F_3(5) = F_7(0) \\ F_4(5) = F_8(0) + F_5(0), F_5(5) = F_9(0) + F_6(0) \\ F_6(5) = F_{10}(0) + F_7(0), F_7(5) = F_1(0) + F_8(0) \\ F_8(5) = F_2(0) + F_9(0), F_9(5) = F_3(0), F_{10}(5) = F_4(0) \end{cases}$
6	$\begin{cases} F_1(6) = F_4(0), F_2(6) = F_5(0), F_3(6) = F_6(0) \\ F_4(6) = F_7(0) + F_4(0), F_5(6) = F_8(0) + F_5(0) \\ F_6(6) = F_9(0) + F_6(0), F_7(6) = F_{10}(0) + F_7(0) \\ F_8(6) = F_1(0) + F_8(0), F_9(6) = F_2(0) + F_9(0) \\ F_{10}(6) = F_3(0) \end{cases}$

Table 3. *Cont.*

j	**Equation**
7	$\begin{cases} F_1(7) = F_3(0), F_2(7) = F_4(0), F_3(7) = F_5(0) \\ F_4(7) = F_6(0) + F_3(0), F_5(7) = F_7(0) + F_4(0) \\ F_6(7) = F_8(0) + F_5(0), F_7(7) = F_9(0) + F_6(0) \\ F_8(7) = F_{10}(0) + F_7(0), F_9(7) = F_1(0) + F_8(0) \\ F_{10}(7) = F_2(0) + F_9(0) \end{cases}$

Table 4. Logic descriptions of the state transition logic modules in the S-restricted RNG for generating random Latin squares of order 4 (Part 2).

j	**Equation**
8	$\begin{cases} F_1(8) = F_2(0) + F_9(0), F_2(8) = F_3(0), F_3(8) = F_4(0), \\ F_4(8) = F_5(0) + F_2(0) + F_9(0), \\ F_5(8) = F_6(0) + F_3(0), F_6(8) = F_7(0) + F_4(0), \\ F_7(8) = F_8(0) + F_5(0), F_8(8) = F_9(0) + F_6(0), \\ F_9(8) = F_{10}(0) + F_7(0), F_{10}(8) = F_1(0) + F_8(0) \end{cases}$
9	$\begin{cases} F_1(9) = F_1(0) + F_8(0), \\ F_2(9) = F_2(0) + F_9(0), F_3(9) = F_3(0), \\ F_4(9) = F_4(0) + F_1(0) + F_8(0), \\ F_5(9) = F_5(0) + F_2(0) + F_9(0), \\ F_6(9) = F_6(0) + F_3(0), F_7(9) = F_7(0) + F_4(0), \\ F_8(9) = F_8(0) + F_5(0), \\ F_9(9) = F_9(0) + F_6(0) F_{10}(9) = F_{10}(0) + F_7(0) \end{cases}$
10	$\begin{cases} F_1(9) = F_1(0) + F_8(0), F_2(9) = F_2(0) + F_9(0), \\ F_3(9) = F_3(0), F_4(9) = F_4(0) + F_1(0) + F_8(0), \\ F_5(9) = F_5(0) + F_2(0) + F_9(0), \\ F_6(9) = F_6(0) + F_3(0), F_7(9) = F_7(0) + F_4(0), \\ F_8(9) = F_8(0) + F_5(0), \\ F_9(9) = F_9(0) + F_6(0) F_{10}(9) = F_{10}(0) + F_7(0) \end{cases}$
11	$\begin{cases} F_1(11) = F_9(0) + F_6(0), F_2(11) = F_{10}(0) + F_7(0), \\ F_3(11) = F_1(0) + F_8(0), \\ F_4(11) = F_2(0) + F_9(0) + F_9(0) + F_6(0), \\ F_5(11) = F_3(0) + F_{10}(0) + F_7(0), \\ F_6(11) = F_4(0) + F_1(0) + F_8(0), \\ F_7(11) = F_5(0) + F_2(0) + F_9(0) F_8(11) = F_6(0) + F_3(0), \\ F_9(11) = F_7(0) + F_4(0), F_{10}(11) = F_8(0) + F_5(0) \end{cases}$

Now, let us design the ILCL. It can be designed according to the information given in Table 2. As an example, to show how the ILCL is designed, let us consider the row labeled "L = 6" in Table 2. The beginning states in this row can be represented as 10-digit binary numbers by 1010011111, 1011011111, 1101011111 and 1001011111, respectively. This

corresponds to the logic expression given by Equation (14) (In the rest of this section, we simply use 'F_i' instead of '$F_i(0)$' for $i \in \{1, 2, \ldots, 10\}$).

$$\begin{aligned} e_7 = {} & F_1F_2'F_3F_4'F_5'F_6F_7F_8F_9F_{10} \dagger F_1F_2'F_3F_4F_5'F_6F_7F_8F_9F_{10} \dagger \\ & F_1F_2F_3'F_4F_5'F_6F_7F_8F_9F_{10} \dagger F_1F_2'F_3'F_4F_5'F_6F_7F_8F_9F_{10}. \end{aligned} \tag{14}$$

In Equation (14), $\dagger$ represents logical OR (to separate it from logical XOR or $GF(2)$ addition). The logical expression in Equation (14) can be simplified as shown by Equation (15) using standard logic function simplification methods:

$$e_7 = F_1F_2'F_3F_5'F_6F_7F_8F_9F_{10} \dagger F_1F_3'F_4F_5'F_6F_7F_8F_9F_{10}. \tag{15}$$

Non-simplified logic expressions of the enablers are listed in the Supplementary File. Simplified enabler expressions are shown in Table 5.

Table 5. Simplified logic expressions for enablers in the S-restricted RNG for generating random Latin squares of order 4.

j	**Simplified Logic Expression**
2	$e_2 = F_1F_2'F_3F_{10}' \dagger F_1F_2'F_4F_5F_{10}' \dagger F_1F_2F_3'F_4F_5'F_{10}' \dagger F_1F_3'F_4F_5'F_6F_{10}' \dagger F_1F_3'F_4F_5'F_7F_{10}' \dagger F_1F_3'F_4F_5'F_8F_{10}' \dagger F_1F_3'F_4F_5'F_9F_{10}'$
3	$e_3 = F_1F_2'F_3F_9'F_{10} \dagger F_1F_2'F_4F_9'F_{10} \dagger F_1F_3'F_4F_5'F_9'F_{10}$
4	$e_4 = F_1F_2'F_3F_8'F_9F_{10} \dagger F_1F_2'F_4F_8'F_9F_{10} \dagger F_1F_3'F_4F_5'F_8'F_9F_{10}$
5	$e_5 = F_1F_2'F_3F_7'F_8F_9F_{10} \dagger F_1F_2'F_4F_7'F_8F_9F_{10} \dagger F_1F_3'F_4F_5'F_7'F_8F_9F_{10}$
6	$e_6 = F_1F_2'F_3F_6'F_7F_8F_9F_{10} \dagger F_1F_2'F_4F_6'F_7F_8F_9F_{10} \dagger F_1F_3'F_4F_5'F_6'F_7F_8F_9F_{10}$
7	$e_7 = F_1F_2'F_3F_5'F_6F_7F_8F_9F_{10} \dagger F_1F_3'F_4F_5'F_6F_7F_8F_9F_{10}$
8	$e_8 = F_1F_2'F_3F_4'F_5F_6F_7F_8F_9F_{10}$
9	$e_9 = F_1F_2'F_3F_4F_5F_6F_7F_8F_9F_{10}$
11	$e_{11} = F_1F_2'F_3'F_4F_5F_6F_7F_8F_9F_{10}$
1	$e_1 = (e_2 \dagger e_3 \dagger e_4 \dagger e_5 \dagger e_6 \dagger e_7 \dagger e_8 \dagger e_9 \dagger e_{11})'$

The invalid runs and the enablers of the circuit designed in this section can be seen in Appendixes B and C, respectively.

7. Conclusions and Further Works

In this paper, we unified the problems of random password generation, random captcha generation and random permutation generation as well as several similar problems into a generalized problem, which we call *random object generation*. The advantage of this generalization is that every solution proposed for the generalized problem will work for all the aforementioned problems. Moreover, we proposed a solution to the *random object generation* problem via introducing the notion of S-restricted RNG, which generates random numbers restricted to be drawn from a given set S. Our solution requires S to be the set of numeric codes assigned to objects of a specific type. We demonstrated how S-restricted RNGs can be constructed using parallel RNGs. We illustrated the construction procedure using parallel LFSRs. Moreover, we examined random generation of Latin squares as a case study. The most critical limitation of our work is the dependence of the coding scheme, and consequently the S-restricted RNG, on the set of target objects. This issue can be addressed via further research on encoding schemes for different kinds of non-numerical entities. Moreover, research can be continued by applying other types of parallel RNGs or Non-Linear Feedback Shift Registers (NFSRs) to the construction of S-restricted RNGs. Moreover, interested researchers can analyze the impact of the object encoding scheme

as well as the LFSR generating polynomials on the performance and complexity of the S-restricted RNGs proposed in this paper.

Author Contributions: Formal analysis, B.Z.; Writing—original draft, B.Z., K.B. and T.K. All authors have read and agreed to the published version of the manuscript.

Funding: T.K. acknowledges support from MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant No. JPMXS0118067285 and No. JPMXS0120319794 and JSPS KAKENHI Grant No. 19K22849 and 21H04879.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the four referees for carefully reading the paper, and for their useful comments which helped improve the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

LFSR	Linear Feedback Shift Register
Completely Automated Public Turing test to tell Computers and Humans Apart	CAPTCHA generation
RNG	Random Number Generator
VLSI	Very Large Scale Integration
CNF	Conjunctive Normal Form
DNF	Disjunctive Normal Form
ILCL	Invalid Run Length Calculation Logic
MDP	Maximum Degree of Parallelism
MIRL	Maximum Invalid Run Length
NFSR	Nonlinear Feedback Shift Register

Appendix A. Codes Assigned to Latin Squares of Order 4

Codes are separated from the corresponding Latin squares using ":". Each Latin square is represented as a permutation of permutations. Each permutation is denoted using a pair of "⟨" and "⟩". Elements of each permutation are separated using ",".

1: $\langle\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle,\langle 3,4,1,2\rangle,\langle 4,3,2,1\rangle\rangle$, 2: $\langle\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle,\langle 3,4,2,1\rangle,\langle 4,3,1,2\rangle\rangle$
3: $\langle\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle,\langle 4,3,1,2\rangle,\langle 3,4,2,1\rangle\rangle$, 4: $\langle\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle,\langle 4,3,2,1\rangle,\langle 3,4,1,2\rangle\rangle$
5: $\langle\langle 1,2,3,4\rangle,\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle,\langle 4,3,2,1\rangle\rangle$, 6: $\langle\langle 1,2,3,4\rangle,\langle 3,1,4,2\rangle,\langle 4,3,2,1\rangle,\langle 2,4,1,3\rangle\rangle$
7: $\langle\langle 1,2,3,4\rangle,\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle,\langle 3,4,1,2\rangle\rangle$, 8: $\langle\langle 1,2,3,4\rangle,\langle 4,1,2,3\rangle,\langle 3,4,1,2\rangle,\langle 2,3,4,1\rangle\rangle$
9: $\langle\langle 1,2,3,4\rangle,\langle 2,3,4,1\rangle,\langle 4,1,2,3\rangle,\langle 3,4,1,2\rangle\rangle$, 10: $\langle\langle 1,2,3,4\rangle,\langle 2,3,4,1\rangle,\langle 3,4,1,2\rangle,\langle 4,1,2,3\rangle\rangle$
11: $\langle\langle 1,2,3,4\rangle,\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle,\langle 4,3,2,1\rangle\rangle$, 12: $\langle\langle 1,2,3,4\rangle,\langle 2,4,1,3\rangle,\langle 4,3,2,1\rangle,\langle 3,1,4,2\rangle\rangle$
13: $\langle\langle 1,2,3,4\rangle,\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle,\langle 4,3,2,1\rangle\rangle$, 14: $\langle\langle 1,2,3,4\rangle,\langle 3,4,1,2\rangle,\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle\rangle$
15: $\langle\langle 1,2,3,4\rangle,\langle 3,4,1,2\rangle,\langle 2,3,4,1\rangle,\langle 4,1,2,3\rangle\rangle$, 16: $\langle\langle 1,2,3,4\rangle,\langle 3,4,1,2\rangle,\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle\rangle$
17: $\langle\langle 1,2,3,4\rangle,\langle 3,4,2,1\rangle,\langle 2,1,4,3\rangle,\langle 4,3,1,2\rangle\rangle$, 18: $\langle\langle 1,2,3,4\rangle,\langle 3,4,2,1\rangle,\langle 4,3,1,2\rangle,\langle 2,1,4,3\rangle\rangle$
19: $\langle\langle 1,2,3,4\rangle,\langle 4,3,1,2\rangle,\langle 2,1,4,3\rangle,\langle 3,4,2,1\rangle\rangle$, 20: $\langle\langle 1,2,3,4\rangle,\langle 4,3,1,2\rangle,\langle 3,4,2,1\rangle,\langle 2,1,4,3\rangle\rangle$
21: $\langle\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle,\langle 3,4,1,2\rangle\rangle$, 22: $\langle\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle,\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle\rangle$
23: $\langle\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle,\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle\rangle$, 24: $\langle\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle,\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle\rangle$
25: $\langle\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle,\langle 3,4,1,2\rangle,\langle 4,3,2,1\rangle\rangle$, 26: $\langle\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle,\langle 3,4,2,1\rangle,\langle 4,3,1,2\rangle\rangle$
27: $\langle\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle,\langle 4,3,1,2\rangle,\langle 3,4,2,1\rangle\rangle$, 28: $\langle\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle,\langle 4,3,2,1\rangle,\langle 3,4,1,2\rangle\rangle$
29: $\langle\langle 1,2,4,3\rangle,\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle,\langle 4,3,1,2\rangle\rangle$, 30: $\langle\langle 1,2,4,3\rangle,\langle 3,1,2,4\rangle,\langle 4,3,1,2\rangle,\langle 2,4,3,1\rangle\rangle$
31: $\langle\langle 1,2,4,3\rangle,\langle 4,1,3,2\rangle,\langle 2,3,1,4\rangle,\langle 3,4,2,1\rangle\rangle$, 32: $\langle\langle 1,2,4,3\rangle,\langle 4,1,3,2\rangle,\langle 3,4,2,1\rangle,\langle 2,3,1,4\rangle\rangle$
33: $\langle\langle 1,2,4,3\rangle,\langle 2,3,1,4\rangle,\langle 4,1,3,2\rangle,\langle 3,4,2,1\rangle\rangle$, 34: $\langle\langle 1,2,4,3\rangle,\langle 2,3,1,4\rangle,\langle 3,4,2,1\rangle,\langle 4,1,3,2\rangle\rangle$
35: $\langle\langle 1,2,4,3\rangle,\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle,\langle 4,3,1,2\rangle\rangle$, 36: $\langle\langle 1,2,4,3\rangle,\langle 2,4,3,1\rangle,\langle 4,3,1,2\rangle,\langle 3,1,2,4\rangle\rangle$
37: $\langle\langle 1,2,4,3\rangle,\langle 3,4,1,2\rangle,\langle 2,1,3,4\rangle,\langle 4,3,2,1\rangle\rangle$, 38: $\langle\langle 1,2,4,3\rangle,\langle 3,4,1,2\rangle,\langle 4,3,2,1\rangle,\langle 2,1,3,4\rangle\rangle$

39: $\langle\langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle, \langle 2,1,3,4\rangle, \langle 4,3,1,2\rangle\rangle$, 40: $\langle\langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle, \langle 4,1,3,2\rangle, \langle 2,3,1,4\rangle\rangle$
41: $\langle\langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle, \langle 2,3,1,4\rangle, \langle 4,1,3,2\rangle\rangle$, 42: $\langle\langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle, \langle 4,3,1,2\rangle, \langle 2,1,3,4\rangle\rangle$
43: $\langle\langle 1,2,4,3\rangle, \langle 4,3,1,2\rangle, \langle 2,1,3,4\rangle, \langle 3,4,2,1\rangle\rangle$, 44: $\langle\langle 1,2,4,3\rangle, \langle 4,3,1,2\rangle, \langle 3,1,2,4\rangle, \langle 2,4,3,1\rangle\rangle$
45: $\langle\langle 1,2,4,3\rangle, \langle 4,3,1,2\rangle, \langle 2,4,3,1\rangle, \langle 3,1,2,4\rangle\rangle$, 46: $\langle\langle 1,2,4,3\rangle, \langle 4,3,1,2\rangle, \langle 3,4,2,1\rangle, \langle 2,1,3,4\rangle\rangle$
47: $\langle\langle 1,2,4,3\rangle, \langle 4,3,2,1\rangle, \langle 2,1,3,4\rangle, \langle 3,4,1,2\rangle\rangle$, 48: $\langle\langle 1,2,4,3\rangle, \langle 4,3,2,1\rangle, \langle 3,4,1,2\rangle, \langle 2,1,3,4\rangle\rangle$
49: $\langle\langle 2,1,3,4\rangle, \langle 1,2,4,3\rangle, \langle 3,4,1,2\rangle, \langle 4,3,2,1\rangle\rangle$, 50: $\langle\langle 2,1,3,4\rangle, \langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle, \langle 4,3,1,2\rangle\rangle$
51: $\langle\langle 2,1,3,4\rangle, \langle 1,2,4,3\rangle, \langle 4,3,1,2\rangle, \langle 3,4,2,1\rangle\rangle$, 52: $\langle\langle 2,1,3,4\rangle, \langle 1,2,4,3\rangle, \langle 4,3,2,1\rangle, \langle 3,4,1,2\rangle\rangle$
53: $\langle\langle 2,1,3,4\rangle, \langle 1,3,4,2\rangle, \langle 4,2,1,3\rangle, \langle 3,4,2,1\rangle\rangle$, 54: $\langle\langle 2,1,3,4\rangle, \langle 1,3,4,2\rangle, \langle 3,4,2,1\rangle, \langle 4,2,1,3\rangle\rangle$
55: $\langle\langle 2,1,3,4\rangle, \langle 1,4,2,3\rangle, \langle 3,2,4,1\rangle, \langle 4,3,1,2\rangle\rangle$, 56: $\langle\langle 2,1,3,4\rangle, \langle 1,4,2,3\rangle, \langle 4,3,1,2\rangle, \langle 3,2,4,1\rangle\rangle$
57: $\langle\langle 2,1,3,4\rangle, \langle 3,2,4,1\rangle, \langle 1,4,2,3\rangle, \langle 4,3,1,2\rangle\rangle$, 58: $\langle\langle 2,1,3,4\rangle, \langle 3,2,4,1\rangle, \langle 4,3,1,2\rangle, \langle 1,4,2,3\rangle\rangle$
59: $\langle\langle 2,1,3,4\rangle, \langle 4,2,1,3\rangle, \langle 1,3,4,2\rangle, \langle 3,4,2,1\rangle\rangle$, 60: $\langle\langle 2,1,3,4\rangle, \langle 4,2,1,3\rangle, \langle 3,4,2,1\rangle, \langle 1,3,4,2\rangle\rangle$
61: $\langle\langle 2,1,3,4\rangle, \langle 3,4,1,2\rangle, \langle 1,2,4,3\rangle, \langle 4,3,2,1\rangle\rangle$, 62: $\langle\langle 2,1,3,4\rangle, \langle 3,4,1,2\rangle, \langle 4,3,2,1\rangle, \langle 1,2,4,3\rangle\rangle$
63: $\langle\langle 2,1,3,4\rangle, \langle 3,4,2,1\rangle, \langle 1,2,4,3\rangle, \langle 4,3,1,2\rangle\rangle$, 64: $\langle\langle 2,1,3,4\rangle, \langle 3,4,2,1\rangle, \langle 1,3,4,2\rangle, \langle 4,2,1,3\rangle\rangle$
65: $\langle\langle 2,1,3,4\rangle, \langle 3,4,2,1\rangle, \langle 4,2,1,3\rangle, \langle 1,3,4,2\rangle\rangle$, 66: $\langle\langle 2,1,3,4\rangle, \langle 3,4,2,1\rangle, \langle 4,3,1,2\rangle, \langle 1,2,4,3\rangle\rangle$
67: $\langle\langle 2,1,3,4\rangle, \langle 4,3,1,2\rangle, \langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle\rangle$, 68: $\langle\langle 2,1,3,4\rangle, \langle 4,3,1,2\rangle, \langle 1,4,2,3\rangle, \langle 3,2,4,1\rangle\rangle$
69: $\langle\langle 2,1,3,4\rangle, \langle 4,3,1,2\rangle, \langle 3,2,4,1\rangle, \langle 1,4,2,3\rangle\rangle$, 70: $\langle\langle 2,1,3,4\rangle, \langle 4,3,1,2\rangle, \langle 3,4,2,1\rangle, \langle 1,2,4,3\rangle\rangle$
71: $\langle\langle 2,1,3,4\rangle, \langle 4,3,2,1\rangle, \langle 1,2,4,3\rangle, \langle 3,4,1,2\rangle\rangle$, 72: $\langle\langle 2,1,3,4\rangle, \langle 4,3,2,1\rangle, \langle 3,4,1,2\rangle, \langle 1,2,4,3\rangle\rangle$
73: $\langle\langle 2,1,4,3\rangle, \langle 1,2,3,4\rangle, \langle 3,4,1,2\rangle, \langle 4,3,2,1\rangle\rangle$, 74: $\langle\langle 2,1,4,3\rangle, \langle 1,2,3,4\rangle, \langle 3,4,2,1\rangle, \langle 4,3,1,2\rangle\rangle$
75: $\langle\langle 2,1,4,3\rangle, \langle 1,2,3,4\rangle, \langle 4,3,1,2\rangle, \langle 3,4,2,1\rangle\rangle$, 76: $\langle\langle 2,1,4,3\rangle, \langle 1,2,3,4\rangle, \langle 4,3,2,1\rangle, \langle 3,4,1,2\rangle\rangle$
77: $\langle\langle 2,1,4,3\rangle, \langle 1,3,2,4\rangle, \langle 4,2,3,1\rangle, \langle 3,4,1,2\rangle\rangle$, 78: $\langle\langle 2,1,4,3\rangle, \langle 1,3,2,4\rangle, \langle 3,4,1,2\rangle, \langle 4,2,3,1\rangle\rangle$
79: $\langle\langle 2,1,4,3\rangle, \langle 1,4,3,2\rangle, \langle 3,2,1,4\rangle, \langle 4,3,2,1\rangle\rangle$, 80: $\langle\langle 2,1,4,3\rangle, \langle 1,4,3,2\rangle, \langle 4,3,2,1\rangle, \langle 3,2,1,4\rangle\rangle$
81: $\langle\langle 2,1,4,3\rangle, \langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle, \langle 4,3,2,1\rangle\rangle$, 82: $\langle\langle 2,1,4,3\rangle, \langle 3,2,1,4\rangle, \langle 4,3,2,1\rangle, \langle 1,4,3,2\rangle\rangle$
83: $\langle\langle 2,1,4,3\rangle, \langle 4,2,3,1\rangle, \langle 1,3,2,4\rangle, \langle 3,4,1,2\rangle\rangle$, 84: $\langle\langle 2,1,4,3\rangle, \langle 4,2,3,1\rangle, \langle 3,4,1,2\rangle, \langle 1,3,2,4\rangle\rangle$
85: $\langle\langle 2,1,4,3\rangle, \langle 3,4,1,2\rangle, \langle 1,2,3,4\rangle, \langle 4,3,2,1\rangle\rangle$, 86: $\langle\langle 2,1,4,3\rangle, \langle 3,4,1,2\rangle, \langle 1,3,2,4\rangle, \langle 4,2,3,1\rangle\rangle$
87: $\langle\langle 2,1,4,3\rangle, \langle 3,4,1,2\rangle, \langle 4,2,3,1\rangle, \langle 1,3,2,4\rangle\rangle$, 88: $\langle\langle 2,1,4,3\rangle, \langle 3,4,1,2\rangle, \langle 4,3,2,1\rangle, \langle 1,2,3,4\rangle\rangle$
89: $\langle\langle 2,1,4,3\rangle, \langle 3,4,2,1\rangle, \langle 1,2,3,4\rangle, \langle 4,3,1,2\rangle\rangle$, 90: $\langle\langle 2,1,4,3\rangle, \langle 3,4,2,1\rangle, \langle 4,3,1,2\rangle, \langle 1,2,3,4\rangle\rangle$
91: $\langle\langle 2,1,4,3\rangle, \langle 4,3,1,2\rangle, \langle 1,2,3,4\rangle, \langle 3,4,2,1\rangle\rangle$, 92: $\langle\langle 2,1,4,3\rangle, \langle 4,3,1,2\rangle, \langle 3,4,2,1\rangle, \langle 1,2,3,4\rangle\rangle$
93: $\langle\langle 2,1,4,3\rangle, \langle 4,3,2,1\rangle, \langle 1,2,3,4\rangle, \langle 3,4,1,2\rangle\rangle$, 94: $\langle\langle 2,1,4,3\rangle, \langle 4,3,2,1\rangle, \langle 1,4,3,2\rangle, \langle 3,2,1,4\rangle\rangle$
95: $\langle\langle 2,1,4,3\rangle, \langle 4,3,2,1\rangle, \langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle\rangle$, 96: $\langle\langle 2,1,4,3\rangle, \langle 4,3,2,1\rangle, \langle 3,4,1,2\rangle, \langle 1,2,3,4\rangle\rangle$
97: $\langle\langle 1,3,2,4\rangle, \langle 2,1,4,3\rangle, \langle 4,2,3,1\rangle, \langle 3,4,1,2\rangle\rangle$, 98: $\langle\langle 1,3,2,4\rangle, \langle 2,1,4,3\rangle, \langle 3,4,1,2\rangle, \langle 4,2,3,1\rangle\rangle$
99: $\langle\langle 1,3,2,4\rangle, \langle 3,1,4,2\rangle, \langle 2,4,1,3\rangle, \langle 4,2,3,1\rangle\rangle$, 100: $\langle\langle 1,3,2,4\rangle, \langle 3,1,4,2\rangle, \langle 2,4,3,1\rangle, \langle 4,2,1,3\rangle\rangle$
101: $\langle\langle 1,3,2,4\rangle, \langle 3,1,4,2\rangle, \langle 4,2,1,3\rangle, \langle 2,4,3,1\rangle\rangle$, 102: $\langle\langle 1,3,2,4\rangle, \langle 3,1,4,2\rangle, \langle 4,2,3,1\rangle, \langle 2,4,1,3\rangle\rangle$
103: $\langle\langle 1,3,2,4\rangle, \langle 4,1,3,2\rangle, \langle 3,2,4,1\rangle, \langle 2,4,1,3\rangle\rangle$, 104: $\langle\langle 1,3,2,4\rangle, \langle 4,1,3,2\rangle, \langle 2,4,1,3\rangle, \langle 3,2,4,1\rangle\rangle$
105: $\langle\langle 1,3,2,4\rangle, \langle 3,2,4,1\rangle, \langle 4,1,3,2\rangle, \langle 2,4,1,3\rangle\rangle$, 106: $\langle\langle 1,3,2,4\rangle, \langle 3,2,4,1\rangle, \langle 2,4,1,3\rangle, \langle 4,1,3,2\rangle\rangle$
107: $\langle\langle 1,3,2,4\rangle, \langle 2,4,1,3\rangle, \langle 3,1,4,2\rangle, \langle 4,2,3,1\rangle\rangle$, 108: $\langle\langle 1,3,2,4\rangle, \langle 2,4,1,3\rangle, \langle 4,1,3,2\rangle, \langle 3,2,4,1\rangle\rangle$
109: $\langle\langle 1,3,2,4\rangle, \langle 2,4,1,3\rangle, \langle 3,2,4,1\rangle, \langle 4,1,3,2\rangle\rangle$, 110: $\langle\langle 1,3,2,4\rangle, \langle 2,4,1,3\rangle, \langle 4,2,3,1\rangle, \langle 3,1,4,2\rangle\rangle$
111: $\langle\langle 1,3,2,4\rangle, \langle 2,4,3,1\rangle, \langle 3,1,4,2\rangle, \langle 4,2,1,3\rangle\rangle$, 112: $\langle\langle 1,3,2,4\rangle, \langle 2,4,3,1\rangle, \langle 4,2,1,3\rangle, \langle 3,1,4,2\rangle\rangle$
113: $\langle\langle 1,3,2,4\rangle, \langle 4,2,1,3\rangle, \langle 3,1,4,2\rangle, \langle 2,4,3,1\rangle\rangle$, 114: $\langle\langle 1,3,2,4\rangle, \langle 4,2,1,3\rangle, \langle 2,4,3,1\rangle, \langle 3,1,4,2\rangle\rangle$
115: $\langle\langle 1,3,2,4\rangle, \langle 4,2,3,1\rangle, \langle 2,1,4,3\rangle, \langle 3,4,1,2\rangle\rangle$, 116: $\langle\langle 1,3,2,4\rangle, \langle 4,2,3,1\rangle, \langle 3,1,4,2\rangle, \langle 2,4,1,3\rangle\rangle$
117: $\langle\langle 1,3,2,4\rangle, \langle 4,2,3,1\rangle, \langle 2,4,1,3\rangle, \langle 3,1,4,2\rangle\rangle$, 118: $\langle\langle 1,3,2,4\rangle, \langle 4,2,3,1\rangle, \langle 3,4,1,2\rangle, \langle 2,1,4,3\rangle\rangle$
119: $\langle\langle 1,3,2,4\rangle, \langle 3,4,1,2\rangle, \langle 2,1,4,3\rangle, \langle 4,2,3,1\rangle\rangle$, 120: $\langle\langle 1,3,2,4\rangle, \langle 3,4,1,2\rangle, \langle 4,2,3,1\rangle, \langle 2,1,4,3\rangle\rangle$
121: $\langle\langle 1,3,4,2\rangle, \langle 2,1,3,4\rangle, \langle 4,2,1,3\rangle, \langle 3,4,2,1\rangle\rangle$, 122: $\langle\langle 1,3,4,2\rangle, \langle 2,1,3,4\rangle, \langle 3,4,2,1\rangle, \langle 4,2,1,3\rangle\rangle$
123: $\langle\langle 1,3,4,2\rangle, \langle 3,1,2,4\rangle, \langle 2,4,1,3\rangle, \langle 4,2,3,1\rangle\rangle$, 124: $\langle\langle 1,3,4,2\rangle, \langle 3,1,2,4\rangle, \langle 2,4,3,1\rangle, \langle 4,2,1,3\rangle\rangle$
125: $\langle\langle 1,3,4,2\rangle, \langle 3,1,2,4\rangle, \langle 4,2,1,3\rangle, \langle 2,4,3,1\rangle\rangle$, 126: $\langle\langle 1,3,4,2\rangle, \langle 3,1,2,4\rangle, \langle 4,2,3,1\rangle, \langle 2,4,1,3\rangle\rangle$
127: $\langle\langle 1,3,4,2\rangle, \langle 4,1,2,3\rangle, \langle 3,2,1,4\rangle, \langle 2,4,3,1\rangle\rangle$, 128: $\langle\langle 1,3,4,2\rangle, \langle 4,1,2,3\rangle, \langle 2,4,3,1\rangle, \langle 3,2,1,4\rangle\rangle$
129: $\langle\langle 1,3,4,2\rangle, \langle 3,2,1,4\rangle, \langle 4,1,2,3\rangle, \langle 2,4,3,1\rangle\rangle$, 130: $\langle\langle 1,3,4,2\rangle, \langle 3,2,1,4\rangle, \langle 2,4,3,1\rangle, \langle 4,1,2,3\rangle\rangle$
131: $\langle\langle 1,3,4,2\rangle, \langle 2,4,1,3\rangle, \langle 3,1,2,4\rangle, \langle 4,2,3,1\rangle\rangle$, 132: $\langle\langle 1,3,4,2\rangle, \langle 2,4,1,3\rangle, \langle 4,2,3,1\rangle, \langle 3,1,2,4\rangle\rangle$
133: $\langle\langle 1,3,4,2\rangle, \langle 2,4,3,1\rangle, \langle 3,1,2,4\rangle, \langle 4,2,1,3\rangle\rangle$, 134: $\langle\langle 1,3,4,2\rangle, \langle 2,4,3,1\rangle, \langle 4,1,2,3\rangle, \langle 3,2,1,4\rangle\rangle$
135: $\langle\langle 1,3,4,2\rangle, \langle 2,4,3,1\rangle, \langle 3,2,1,4\rangle, \langle 4,1,2,3\rangle\rangle$, 136: $\langle\langle 1,3,4,2\rangle, \langle 2,4,3,1\rangle, \langle 4,2,1,3\rangle, \langle 3,1,2,4\rangle\rangle$
137: $\langle\langle 1,3,4,2\rangle, \langle 4,2,1,3\rangle, \langle 2,1,3,4\rangle, \langle 3,4,2,1\rangle\rangle$, 138: $\langle\langle 1,3,4,2\rangle, \langle 4,2,1,3\rangle, \langle 3,1,2,4\rangle, \langle 2,4,3,1\rangle\rangle$
139: $\langle\langle 1,3,4,2\rangle, \langle 4,2,1,3\rangle, \langle 2,4,3,1\rangle, \langle 3,1,2,4\rangle\rangle$, 140: $\langle\langle 1,3,4,2\rangle, \langle 4,2,1,3\rangle, \langle 3,4,2,1\rangle, \langle 2,1,3,4\rangle\rangle$
141: $\langle\langle 1,3,4,2\rangle, \langle 4,2,3,1\rangle, \langle 3,1,2,4\rangle, \langle 2,4,1,3\rangle\rangle$, 142: $\langle\langle 1,3,4,2\rangle, \langle 4,2,3,1\rangle, \langle 2,4,1,3\rangle, \langle 3,1,2,4\rangle\rangle$
143: $\langle\langle 1,3,4,2\rangle, \langle 3,4,2,1\rangle, \langle 2,1,3,4\rangle, \langle 4,2,1,3\rangle\rangle$, 144: $\langle\langle 1,3,4,2\rangle, \langle 3,4,2,1\rangle, \langle 4,2,1,3\rangle, \langle 2,1,3,4\rangle\rangle$
145: $\langle\langle 3,1,2,4\rangle, \langle 1,2,4,3\rangle, \langle 2,4,3,1\rangle, \langle 4,3,1,2\rangle\rangle$, 146: $\langle\langle 3,1,2,4\rangle, \langle 1,2,4,3\rangle, \langle 4,3,1,2\rangle, \langle 2,4,3,1\rangle\rangle$

147: $\langle\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle,\langle 2,4,1,3\rangle,\langle 4,2,3,1\rangle\rangle$, 148: $\langle\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle,\langle 2,4,3,1\rangle,\langle 4,2,1,3\rangle\rangle$
149: $\langle\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle,\langle 4,2,1,3\rangle,\langle 2,4,3,1\rangle\rangle$, 150: $\langle\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle,\langle 4,2,3,1\rangle,\langle 2,4,1,3\rangle\rangle$
151: $\langle\langle 3,1,2,4\rangle,\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle,\langle 4,2,1,3\rangle\rangle$, 152: $\langle\langle 3,1,2,4\rangle,\langle 1,4,3,2\rangle,\langle 4,2,1,3\rangle,\langle 2,3,4,1\rangle\rangle$
153: $\langle\langle 3,1,2,4\rangle,\langle 2,3,4,1\rangle,\langle 1,4,3,2\rangle,\langle 4,2,1,3\rangle\rangle$, 154: $\langle\langle 3,1,2,4\rangle,\langle 2,3,4,1\rangle,\langle 4,2,1,3\rangle,\langle 1,4,3,2\rangle\rangle$
155: $\langle\langle 3,1,2,4\rangle,\langle 2,4,1,3\rangle,\langle 1,3,4,2\rangle,\langle 4,2,3,1\rangle\rangle$, 156: $\langle\langle 3,1,2,4\rangle,\langle 2,4,1,3\rangle,\langle 4,2,3,1\rangle,\langle 1,3,4,2\rangle\rangle$
157: $\langle\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle,\langle 1,2,4,3\rangle,\langle 4,3,1,2\rangle\rangle$, 158: $\langle\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle,\langle 1,3,4,2\rangle,\langle 4,2,1,3\rangle\rangle$
159: $\langle\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle,\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle\rangle$, 160: $\langle\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle,\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle\rangle$
161: $\langle\langle 3,1,2,4\rangle,\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle,\langle 2,4,3,1\rangle\rangle$, 162: $\langle\langle 3,1,2,4\rangle,\langle 4,2,1,3\rangle,\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle\rangle$
163: $\langle\langle 3,1,2,4\rangle,\langle 4,2,1,3\rangle,\langle 2,3,4,1\rangle,\langle 1,4,3,2\rangle\rangle$, 164: $\langle\langle 3,1,2,4\rangle,\langle 4,2,1,3\rangle,\langle 2,4,3,1\rangle,\langle 1,3,4,2\rangle\rangle$
165: $\langle\langle 3,1,2,4\rangle,\langle 4,2,3,1\rangle,\langle 1,3,4,2\rangle,\langle 2,4,1,3\rangle\rangle$, 166: $\langle\langle 3,1,2,4\rangle,\langle 4,2,3,1\rangle,\langle 2,4,1,3\rangle,\langle 1,3,4,2\rangle\rangle$
167: $\langle\langle 3,1,2,4\rangle,\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle,\langle 2,4,3,1\rangle\rangle$, 168: $\langle\langle 3,1,2,4\rangle,\langle 4,3,1,2\rangle,\langle 2,4,3,1\rangle,\langle 1,2,4,3\rangle\rangle$
169: $\langle\langle 3,1,4,2\rangle,\langle 1,2,3,4\rangle,\langle 2,4,1,3\rangle,\langle 4,3,2,1\rangle\rangle$, 170: $\langle\langle 3,1,4,2\rangle,\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle,\langle 2,4,1,3\rangle\rangle$
171: $\langle\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle,\langle 2,4,1,3\rangle,\langle 4,2,3,1\rangle\rangle$, 172: $\langle\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle,\langle 2,4,3,1\rangle,\langle 4,2,1,3\rangle\rangle$
173: $\langle\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle,\langle 4,2,1,3\rangle,\langle 2,4,3,1\rangle\rangle$, 174: $\langle\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle,\langle 4,2,3,1\rangle,\langle 2,4,1,3\rangle\rangle$
175: $\langle\langle 3,1,4,2\rangle,\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle,\langle 4,2,3,1\rangle\rangle$, 176: $\langle\langle 3,1,4,2\rangle,\langle 1,4,2,3\rangle,\langle 4,2,3,1\rangle,\langle 2,3,1,4\rangle\rangle$
177: $\langle\langle 3,1,4,2\rangle,\langle 2,3,1,4\rangle,\langle 1,4,2,3\rangle,\langle 4,2,3,1\rangle\rangle$, 178: $\langle\langle 3,1,4,2\rangle,\langle 2,3,1,4\rangle,\langle 4,2,3,1\rangle,\langle 1,4,2,3\rangle\rangle$
179: $\langle\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle,\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle\rangle$, 180: $\langle\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle,\langle 4,2,3,1\rangle\rangle$
181: $\langle\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle,\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle\rangle$, 182: $\langle\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle,\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle\rangle$
183: $\langle\langle 3,1,4,2\rangle,\langle 2,4,3,1\rangle,\langle 1,3,2,4\rangle,\langle 4,2,1,3\rangle\rangle$, 184: $\langle\langle 3,1,4,2\rangle,\langle 2,4,3,1\rangle,\langle 4,2,1,3\rangle,\langle 1,3,2,4\rangle\rangle$
185: $\langle\langle 3,1,4,2\rangle,\langle 4,2,1,3\rangle,\langle 1,3,2,4\rangle,\langle 2,4,3,1\rangle\rangle$, 186: $\langle\langle 3,1,4,2\rangle,\langle 4,2,1,3\rangle,\langle 2,4,3,1\rangle,\langle 1,3,2,4\rangle\rangle$
187: $\langle\langle 3,1,4,2\rangle,\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle,\langle 2,4,1,3\rangle\rangle$, 188: $\langle\langle 3,1,4,2\rangle,\langle 4,2,3,1\rangle,\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle\rangle$
189: $\langle\langle 3,1,4,2\rangle,\langle 4,2,3,1\rangle,\langle 2,3,1,4\rangle,\langle 1,4,2,3\rangle\rangle$, 190: $\langle\langle 3,1,4,2\rangle,\langle 4,2,3,1\rangle,\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle\rangle$
191: $\langle\langle 3,1,4,2\rangle,\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle,\langle 2,4,1,3\rangle\rangle$, 192: $\langle\langle 3,1,4,2\rangle,\langle 4,3,2,1\rangle,\langle 2,4,1,3\rangle,\langle 1,2,3,4\rangle\rangle$
193: $\langle\langle 1,4,2,3\rangle,\langle 2,1,3,4\rangle,\langle 3,2,4,1\rangle,\langle 4,3,1,2\rangle\rangle$, 194: $\langle\langle 1,4,2,3\rangle,\langle 2,1,3,4\rangle,\langle 4,3,1,2\rangle,\langle 3,2,4,1\rangle\rangle$
195: $\langle\langle 1,4,2,3\rangle,\langle 3,1,4,2\rangle,\langle 2,3,1,4\rangle,\langle 4,2,3,1\rangle\rangle$, 196: $\langle\langle 1,4,2,3\rangle,\langle 3,1,4,2\rangle,\langle 4,2,3,1\rangle,\langle 2,3,1,4\rangle\rangle$
197: $\langle\langle 1,4,2,3\rangle,\langle 4,1,3,2\rangle,\langle 2,3,1,4\rangle,\langle 3,2,4,1\rangle\rangle$, 198: $\langle\langle 1,4,2,3\rangle,\langle 4,1,3,2\rangle,\langle 2,3,4,1\rangle,\langle 3,2,1,4\rangle\rangle$
199: $\langle\langle 1,4,2,3\rangle,\langle 4,1,3,2\rangle,\langle 3,2,1,4\rangle,\langle 2,3,4,1\rangle\rangle$, 200: $\langle\langle 1,4,2,3\rangle,\langle 4,1,3,2\rangle,\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle\rangle$
201: $\langle\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle,\langle 3,1,4,2\rangle,\langle 4,2,3,1\rangle\rangle$, 202: $\langle\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle,\langle 4,1,3,2\rangle,\langle 3,2,4,1\rangle\rangle$
203: $\langle\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle,\langle 3,2,4,1\rangle,\langle 4,1,3,2\rangle\rangle$, 204: $\langle\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle,\langle 4,2,3,1\rangle,\langle 3,1,4,2\rangle\rangle$
205: $\langle\langle 1,4,2,3\rangle,\langle 2,3,4,1\rangle,\langle 4,1,3,2\rangle,\langle 3,2,1,4\rangle\rangle$, 206: $\langle\langle 1,4,2,3\rangle,\langle 2,3,4,1\rangle,\langle 3,2,1,4\rangle,\langle 4,1,3,2\rangle\rangle$
207: $\langle\langle 1,4,2,3\rangle,\langle 3,2,1,4\rangle,\langle 4,1,3,2\rangle,\langle 2,3,4,1\rangle\rangle$, 208: $\langle\langle 1,4,2,3\rangle,\langle 3,2,1,4\rangle,\langle 2,3,4,1\rangle,\langle 4,1,3,2\rangle\rangle$
209: $\langle\langle 1,4,2,3\rangle,\langle 3,2,4,1\rangle,\langle 2,1,3,4\rangle,\langle 4,3,1,2\rangle\rangle$, 210: $\langle\langle 1,4,2,3\rangle,\langle 3,2,4,1\rangle,\langle 4,1,3,2\rangle,\langle 2,3,1,4\rangle\rangle$
211: $\langle\langle 1,4,2,3\rangle,\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle,\langle 4,1,3,2\rangle\rangle$, 212: $\langle\langle 1,4,2,3\rangle,\langle 3,2,4,1\rangle,\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle\rangle$
213: $\langle\langle 1,4,2,3\rangle,\langle 4,2,3,1\rangle,\langle 3,1,4,2\rangle,\langle 2,3,1,4\rangle\rangle$, 214: $\langle\langle 1,4,2,3\rangle,\langle 4,2,3,1\rangle,\langle 2,3,1,4\rangle,\langle 3,1,4,2\rangle\rangle$
215: $\langle\langle 1,4,2,3\rangle,\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle,\langle 3,2,4,1\rangle\rangle$, 216: $\langle\langle 1,4,2,3\rangle,\langle 4,3,1,2\rangle,\langle 3,2,4,1\rangle,\langle 2,1,3,4\rangle\rangle$
217: $\langle\langle 1,4,3,2\rangle,\langle 2,1,4,3\rangle,\langle 3,2,1,4\rangle,\langle 4,3,2,1\rangle\rangle$, 218: $\langle\langle 1,4,3,2\rangle,\langle 2,1,4,3\rangle,\langle 4,3,2,1\rangle,\langle 3,2,1,4\rangle\rangle$
219: $\langle\langle 1,4,3,2\rangle,\langle 3,1,2,4\rangle,\langle 2,3,4,1\rangle,\langle 4,2,1,3\rangle\rangle$, 220: $\langle\langle 1,4,3,2\rangle,\langle 3,1,2,4\rangle,\langle 4,2,1,3\rangle,\langle 2,3,4,1\rangle\rangle$
221: $\langle\langle 1,4,3,2\rangle,\langle 4,1,2,3\rangle,\langle 2,3,1,4\rangle,\langle 3,2,4,1\rangle\rangle$, 222: $\langle\langle 1,4,3,2\rangle,\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle,\langle 3,2,1,4\rangle\rangle$
223: $\langle\langle 1,4,3,2\rangle,\langle 4,1,2,3\rangle,\langle 3,2,1,4\rangle,\langle 2,3,4,1\rangle\rangle$, 224: $\langle\langle 1,4,3,2\rangle,\langle 4,1,2,3\rangle,\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle\rangle$
225: $\langle\langle 1,4,3,2\rangle,\langle 2,3,1,4\rangle,\langle 4,1,2,3\rangle,\langle 3,2,4,1\rangle\rangle$, 226: $\langle\langle 1,4,3,2\rangle,\langle 2,3,1,4\rangle,\langle 3,2,4,1\rangle,\langle 4,1,2,3\rangle\rangle$
227: $\langle\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle,\langle 3,1,2,4\rangle,\langle 4,2,1,3\rangle\rangle$, 228: $\langle\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle,\langle 4,1,2,3\rangle,\langle 3,2,1,4\rangle\rangle$
229: $\langle\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle,\langle 3,2,1,4\rangle,\langle 4,1,2,3\rangle\rangle$, 230: $\langle\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle,\langle 4,2,1,3\rangle,\langle 3,1,2,4\rangle\rangle$
231: $\langle\langle 1,4,3,2\rangle,\langle 3,2,1,4\rangle,\langle 2,1,4,3\rangle,\langle 4,3,2,1\rangle\rangle$, 232: $\langle\langle 1,4,3,2\rangle,\langle 3,2,1,4\rangle,\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle\rangle$
233: $\langle\langle 1,4,3,2\rangle,\langle 3,2,1,4\rangle,\langle 2,3,4,1\rangle,\langle 4,1,2,3\rangle\rangle$, 234: $\langle\langle 1,4,3,2\rangle,\langle 3,2,1,4\rangle,\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle\rangle$
235: $\langle\langle 1,4,3,2\rangle,\langle 3,2,4,1\rangle,\langle 4,1,2,3\rangle,\langle 2,3,1,4\rangle\rangle$, 236: $\langle\langle 1,4,3,2\rangle,\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle,\langle 4,1,2,3\rangle\rangle$
237: $\langle\langle 1,4,3,2\rangle,\langle 4,2,1,3\rangle,\langle 3,1,2,4\rangle,\langle 2,3,4,1\rangle\rangle$, 238: $\langle\langle 1,4,3,2\rangle,\langle 4,2,1,3\rangle,\langle 2,3,4,1\rangle,\langle 3,1,2,4\rangle\rangle$
239: $\langle\langle 1,4,3,2\rangle,\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle,\langle 3,2,1,4\rangle\rangle$, 240: $\langle\langle 1,4,3,2\rangle,\langle 4,3,2,1\rangle,\langle 3,2,1,4\rangle,\langle 2,1,4,3\rangle\rangle$
241: $\langle\langle 4,1,2,3\rangle,\langle 1,2,3,4\rangle,\langle 2,3,4,1\rangle,\langle 3,4,1,2\rangle\rangle$, 242: $\langle\langle 4,1,2,3\rangle,\langle 1,2,3,4\rangle,\langle 3,4,1,2\rangle,\langle 2,3,4,1\rangle\rangle$
243: $\langle\langle 4,1,2,3\rangle,\langle 1,3,4,2\rangle,\langle 3,2,1,4\rangle,\langle 2,4,3,1\rangle\rangle$, 244: $\langle\langle 4,1,2,3\rangle,\langle 1,3,4,2\rangle,\langle 2,4,3,1\rangle,\langle 3,2,1,4\rangle\rangle$
245: $\langle\langle 4,1,2,3\rangle,\langle 1,4,3,2\rangle,\langle 2,3,1,4\rangle,\langle 3,2,4,1\rangle\rangle$, 246: $\langle\langle 4,1,2,3\rangle,\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle,\langle 3,2,1,4\rangle\rangle$
247: $\langle\langle 4,1,2,3\rangle,\langle 1,4,3,2\rangle,\langle 3,2,1,4\rangle,\langle 2,3,4,1\rangle\rangle$, 248: $\langle\langle 4,1,2,3\rangle,\langle 1,4,3,2\rangle,\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle\rangle$
249: $\langle\langle 4,1,2,3\rangle,\langle 2,3,1,4\rangle,\langle 1,4,3,2\rangle,\langle 3,2,4,1\rangle\rangle$, 250: $\langle\langle 4,1,2,3\rangle,\langle 2,3,1,4\rangle,\langle 3,2,4,1\rangle,\langle 1,4,3,2\rangle\rangle$
251: $\langle\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle,\langle 1,2,3,4\rangle,\langle 3,4,1,2\rangle\rangle$, 252: $\langle\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle,\langle 1,4,3,2\rangle,\langle 3,2,1,4\rangle\rangle$
253: $\langle\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle,\langle 3,2,1,4\rangle,\langle 1,4,3,2\rangle\rangle$, 254: $\langle\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle,\langle 3,4,1,2\rangle,\langle 1,2,3,4\rangle\rangle$

255: $\langle\langle 4,1,2,3\rangle, \langle 3,2,1,4\rangle, \langle 1,3,4,2\rangle, \langle 2,4,3,1\rangle\rangle$, 256: $\langle\langle 4,1,2,3\rangle, \langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle, \langle 2,3,4,1\rangle\rangle$
257: $\langle\langle 4,1,2,3\rangle, \langle 3,2,1,4\rangle, \langle 2,3,4,1\rangle, \langle 1,4,3,2\rangle\rangle$, 258: $\langle\langle 4,1,2,3\rangle, \langle 3,2,1,4\rangle, \langle 2,4,3,1\rangle, \langle 1,3,4,2\rangle\rangle$
259: $\langle\langle 4,1,2,3\rangle, \langle 3,2,4,1\rangle, \langle 1,4,3,2\rangle, \langle 2,3,1,4\rangle\rangle$, 260: $\langle\langle 4,1,2,3\rangle, \langle 3,2,4,1\rangle, \langle 2,3,1,4\rangle, \langle 1,4,3,2\rangle\rangle$
261: $\langle\langle 4,1,2,3\rangle, \langle 2,4,3,1\rangle, \langle 1,3,4,2\rangle, \langle 3,2,1,4\rangle\rangle$, 262: $\langle\langle 4,1,2,3\rangle, \langle 2,4,3,1\rangle, \langle 3,2,1,4\rangle, \langle 1,3,4,2\rangle\rangle$
263: $\langle\langle 4,1,2,3\rangle, \langle 3,4,1,2\rangle, \langle 1,2,3,4\rangle, \langle 2,3,4,1\rangle\rangle$, 264: $\langle\langle 4,1,2,3\rangle, \langle 3,4,1,2\rangle, \langle 2,3,4,1\rangle, \langle 1,2,3,4\rangle\rangle$
265: $\langle\langle 4,1,3,2\rangle, \langle 1,2,4,3\rangle, \langle 2,3,1,4\rangle, \langle 3,4,2,1\rangle\rangle$, 266: $\langle\langle 4,1,3,2\rangle, \langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle, \langle 2,3,1,4\rangle\rangle$
267: $\langle\langle 4,1,3,2\rangle, \langle 1,3,2,4\rangle, \langle 3,2,4,1\rangle, \langle 2,4,1,3\rangle\rangle$, 268: $\langle\langle 4,1,3,2\rangle, \langle 1,3,2,4\rangle, \langle 2,4,1,3\rangle, \langle 3,2,4,1\rangle\rangle$
269: $\langle\langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle, \langle 2,3,1,4\rangle, \langle 3,2,4,1\rangle\rangle$, 270: $\langle\langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle, \langle 2,3,4,1\rangle, \langle 3,2,1,4\rangle\rangle$
271: $\langle\langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle, \langle 3,2,1,4\rangle, \langle 2,3,4,1\rangle\rangle$, 272: $\langle\langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle, \langle 3,2,4,1\rangle, \langle 2,3,1,4\rangle\rangle$
273: $\langle\langle 4,1,3,2\rangle, \langle 2,3,1,4\rangle, \langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle\rangle$, 274: $\langle\langle 4,1,3,2\rangle, \langle 2,3,1,4\rangle, \langle 1,4,2,3\rangle, \langle 3,2,4,1\rangle\rangle$
275: $\langle\langle 4,1,3,2\rangle, \langle 2,3,1,4\rangle, \langle 3,2,4,1\rangle, \langle 1,4,2,3\rangle\rangle$, 276: $\langle\langle 4,1,3,2\rangle, \langle 2,3,1,4\rangle, \langle 3,4,2,1\rangle, \langle 1,2,4,3\rangle\rangle$
277: $\langle\langle 4,1,3,2\rangle, \langle 2,3,4,1\rangle, \langle 1,4,2,3\rangle, \langle 3,2,1,4\rangle\rangle$, 278: $\langle\langle 4,1,3,2\rangle, \langle 2,3,4,1\rangle, \langle 3,2,1,4\rangle, \langle 1,4,2,3\rangle\rangle$
279: $\langle\langle 4,1,3,2\rangle, \langle 3,2,1,4\rangle, \langle 1,4,2,3\rangle, \langle 2,3,4,1\rangle\rangle$, 280: $\langle\langle 4,1,3,2\rangle, \langle 3,2,1,4\rangle, \langle 2,3,4,1\rangle, \langle 1,4,2,3\rangle\rangle$
281: $\langle\langle 4,1,3,2\rangle, \langle 3,2,4,1\rangle, \langle 1,3,2,4\rangle, \langle 2,4,1,3\rangle\rangle$, 282: $\langle\langle 4,1,3,2\rangle, \langle 3,2,4,1\rangle, \langle 1,4,2,3\rangle, \langle 2,3,1,4\rangle\rangle$
283: $\langle\langle 4,1,3,2\rangle, \langle 3,2,4,1\rangle, \langle 2,3,1,4\rangle, \langle 1,4,2,3\rangle\rangle$, 284: $\langle\langle 4,1,3,2\rangle, \langle 3,2,4,1\rangle, \langle 2,4,1,3\rangle, \langle 1,3,2,4\rangle\rangle$
285: $\langle\langle 4,1,3,2\rangle, \langle 2,4,1,3\rangle, \langle 1,3,2,4\rangle, \langle 3,2,4,1\rangle\rangle$, 286: $\langle\langle 4,1,3,2\rangle, \langle 2,4,1,3\rangle, \langle 3,2,4,1\rangle, \langle 1,3,2,4\rangle\rangle$
287: $\langle\langle 4,1,3,2\rangle, \langle 3,4,2,1\rangle, \langle 1,2,4,3\rangle, \langle 2,3,1,4\rangle\rangle$, 288: $\langle\langle 4,1,3,2\rangle, \langle 3,4,2,1\rangle, \langle 2,3,1,4\rangle, \langle 1,2,4,3\rangle\rangle$
289: $\langle\langle 2,3,1,4\rangle, \langle 1,2,4,3\rangle, \langle 4,1,3,2\rangle, \langle 3,4,2,1\rangle\rangle$, 290: $\langle\langle 2,3,1,4\rangle, \langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle, \langle 4,1,3,2\rangle\rangle$
291: $\langle\langle 2,3,1,4\rangle, \langle 3,1,4,2\rangle, \langle 1,4,2,3\rangle, \langle 4,2,3,1\rangle\rangle$, 292: $\langle\langle 2,3,1,4\rangle, \langle 3,1,4,2\rangle, \langle 4,2,3,1\rangle, \langle 1,4,2,3\rangle\rangle$
293: $\langle\langle 2,3,1,4\rangle, \langle 1,4,2,3\rangle, \langle 3,1,4,2\rangle, \langle 4,2,3,1\rangle\rangle$, 294: $\langle\langle 2,3,1,4\rangle, \langle 1,4,2,3\rangle, \langle 4,1,3,2\rangle, \langle 3,2,4,1\rangle\rangle$
295: $\langle\langle 2,3,1,4\rangle, \langle 1,4,2,3\rangle, \langle 3,2,4,1\rangle, \langle 4,1,3,2\rangle\rangle$, 296: $\langle\langle 2,3,1,4\rangle, \langle 1,4,2,3\rangle, \langle 4,2,3,1\rangle, \langle 3,1,4,2\rangle\rangle$
297: $\langle\langle 2,3,1,4\rangle, \langle 1,4,3,2\rangle, \langle 4,1,2,3\rangle, \langle 3,2,4,1\rangle\rangle$, 298: $\langle\langle 2,3,1,4\rangle, \langle 1,4,3,2\rangle, \langle 3,2,4,1\rangle, \langle 4,1,2,3\rangle\rangle$
299: $\langle\langle 2,3,1,4\rangle, \langle 4,1,2,3\rangle, \langle 1,4,3,2\rangle, \langle 3,2,4,1\rangle\rangle$, 300: $\langle\langle 2,3,1,4\rangle, \langle 4,1,2,3\rangle, \langle 3,2,4,1\rangle, \langle 1,4,3,2\rangle\rangle$
301: $\langle\langle 2,3,1,4\rangle, \langle 4,1,3,2\rangle, \langle 1,2,4,3\rangle, \langle 3,4,2,1\rangle\rangle$, 302: $\langle\langle 2,3,1,4\rangle, \langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle, \langle 3,2,4,1\rangle\rangle$
303: $\langle\langle 2,3,1,4\rangle, \langle 4,1,3,2\rangle, \langle 3,2,4,1\rangle, \langle 1,4,2,3\rangle\rangle$, 304: $\langle\langle 2,3,1,4\rangle, \langle 4,1,3,2\rangle, \langle 3,4,2,1\rangle, \langle 1,2,4,3\rangle\rangle$
305: $\langle\langle 2,3,1,4\rangle, \langle 3,2,4,1\rangle, \langle 1,4,2,3\rangle, \langle 4,1,3,2\rangle\rangle$, 306: $\langle\langle 2,3,1,4\rangle, \langle 3,2,4,1\rangle, \langle 1,4,3,2\rangle, \langle 4,1,2,3\rangle\rangle$
307: $\langle\langle 2,3,1,4\rangle, \langle 3,2,4,1\rangle, \langle 4,1,2,3\rangle, \langle 1,4,3,2\rangle\rangle$, 308: $\langle\langle 2,3,1,4\rangle, \langle 3,2,4,1\rangle, \langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle\rangle$
309: $\langle\langle 2,3,1,4\rangle, \langle 4,2,3,1\rangle, \langle 3,1,4,2\rangle, \langle 1,4,2,3\rangle\rangle$, 310: $\langle\langle 2,3,1,4\rangle, \langle 4,2,3,1\rangle, \langle 1,4,2,3\rangle, \langle 3,1,4,2\rangle\rangle$
311: $\langle\langle 2,3,1,4\rangle, \langle 3,4,2,1\rangle, \langle 1,2,4,3\rangle, \langle 4,1,3,2\rangle\rangle$, 312: $\langle\langle 2,3,1,4\rangle, \langle 3,4,2,1\rangle, \langle 4,1,3,2\rangle, \langle 1,2,4,3\rangle\rangle$
313: $\langle\langle 2,3,4,1\rangle, \langle 1,2,3,4\rangle, \langle 4,1,2,3\rangle, \langle 3,4,1,2\rangle\rangle$, 314: $\langle\langle 2,3,4,1\rangle, \langle 1,2,3,4\rangle, \langle 3,4,1,2\rangle, \langle 4,1,2,3\rangle\rangle$
315: $\langle\langle 2,3,4,1\rangle, \langle 3,1,2,4\rangle, \langle 1,4,3,2\rangle, \langle 4,2,1,3\rangle\rangle$, 316: $\langle\langle 2,3,4,1\rangle, \langle 3,1,2,4\rangle, \langle 4,2,1,3\rangle, \langle 1,4,3,2\rangle\rangle$
317: $\langle\langle 2,3,4,1\rangle, \langle 1,4,2,3\rangle, \langle 4,1,3,2\rangle, \langle 3,2,1,4\rangle\rangle$, 318: $\langle\langle 2,3,4,1\rangle, \langle 1,4,2,3\rangle, \langle 3,2,1,4\rangle, \langle 4,1,3,2\rangle\rangle$
319: $\langle\langle 2,3,4,1\rangle, \langle 1,4,3,2\rangle, \langle 3,1,2,4\rangle, \langle 4,2,1,3\rangle\rangle$, 320: $\langle\langle 2,3,4,1\rangle, \langle 1,4,3,2\rangle, \langle 4,1,2,3\rangle, \langle 3,2,1,4\rangle\rangle$
321: $\langle\langle 2,3,4,1\rangle, \langle 1,4,3,2\rangle, \langle 3,2,1,4\rangle, \langle 4,1,2,3\rangle\rangle$, 322: $\langle\langle 2,3,4,1\rangle, \langle 1,4,3,2\rangle, \langle 4,2,1,3\rangle, \langle 3,1,2,4\rangle\rangle$
323: $\langle\langle 2,3,4,1\rangle, \langle 4,1,2,3\rangle, \langle 1,2,3,4\rangle, \langle 3,4,1,2\rangle\rangle$, 324: $\langle\langle 2,3,4,1\rangle, \langle 4,1,2,3\rangle, \langle 1,4,3,2\rangle, \langle 3,2,1,4\rangle\rangle$
325: $\langle\langle 2,3,4,1\rangle, \langle 4,1,2,3\rangle, \langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle\rangle$, 326: $\langle\langle 2,3,4,1\rangle, \langle 4,1,2,3\rangle, \langle 3,4,1,2\rangle, \langle 1,2,3,4\rangle\rangle$
327: $\langle\langle 2,3,4,1\rangle, \langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle, \langle 3,2,1,4\rangle\rangle$, 328: $\langle\langle 2,3,4,1\rangle, \langle 4,1,3,2\rangle, \langle 3,2,1,4\rangle, \langle 1,4,2,3\rangle\rangle$
329: $\langle\langle 2,3,4,1\rangle, \langle 3,2,1,4\rangle, \langle 1,4,2,3\rangle, \langle 4,1,3,2\rangle\rangle$, 330: $\langle\langle 2,3,4,1\rangle, \langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle, \langle 4,1,2,3\rangle\rangle$
331: $\langle\langle 2,3,4,1\rangle, \langle 3,2,1,4\rangle, \langle 4,1,2,3\rangle, \langle 1,4,3,2\rangle\rangle$, 332: $\langle\langle 2,3,4,1\rangle, \langle 3,2,1,4\rangle, \langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle\rangle$
333: $\langle\langle 2,3,4,1\rangle, \langle 4,2,1,3\rangle, \langle 3,1,2,4\rangle, \langle 1,4,3,2\rangle\rangle$, 334: $\langle\langle 2,3,4,1\rangle, \langle 4,2,1,3\rangle, \langle 1,4,3,2\rangle, \langle 3,1,2,4\rangle\rangle$
335: $\langle\langle 2,3,4,1\rangle, \langle 3,4,1,2\rangle, \langle 1,2,3,4\rangle, \langle 4,1,2,3\rangle\rangle$, 336: $\langle\langle 2,3,4,1\rangle, \langle 3,4,1,2\rangle, \langle 4,1,2,3\rangle, \langle 1,2,3,4\rangle\rangle$
337: $\langle\langle 3,2,1,4\rangle, \langle 2,1,4,3\rangle, \langle 1,4,3,2\rangle, \langle 4,3,2,1\rangle\rangle$, 338: $\langle\langle 3,2,1,4\rangle, \langle 2,1,4,3\rangle, \langle 4,3,2,1\rangle, \langle 1,4,3,2\rangle\rangle$
339: $\langle\langle 3,2,1,4\rangle, \langle 1,3,4,2\rangle, \langle 4,1,2,3\rangle, \langle 2,4,3,1\rangle\rangle$, 340: $\langle\langle 3,2,1,4\rangle, \langle 1,3,4,2\rangle, \langle 2,4,3,1\rangle, \langle 4,1,2,3\rangle\rangle$
341: $\langle\langle 3,2,1,4\rangle, \langle 1,4,2,3\rangle, \langle 4,1,3,2\rangle, \langle 2,3,4,1\rangle\rangle$, 342: $\langle\langle 3,2,1,4\rangle, \langle 1,4,2,3\rangle, \langle 2,3,4,1\rangle, \langle 4,1,3,2\rangle\rangle$
343: $\langle\langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle, \langle 2,1,4,3\rangle, \langle 4,3,2,1\rangle\rangle$, 344: $\langle\langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle, \langle 4,1,2,3\rangle, \langle 2,3,4,1\rangle\rangle$
345: $\langle\langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle, \langle 2,3,4,1\rangle, \langle 4,1,2,3\rangle\rangle$, 346: $\langle\langle 3,2,1,4\rangle, \langle 1,4,3,2\rangle, \langle 4,3,2,1\rangle, \langle 2,1,4,3\rangle\rangle$
347: $\langle\langle 3,2,1,4\rangle, \langle 4,1,2,3\rangle, \langle 1,3,4,2\rangle, \langle 2,4,3,1\rangle\rangle$, 348: $\langle\langle 3,2,1,4\rangle, \langle 4,1,2,3\rangle, \langle 1,4,3,2\rangle, \langle 2,3,4,1\rangle\rangle$
349: $\langle\langle 3,2,1,4\rangle, \langle 4,1,2,3\rangle, \langle 2,3,4,1\rangle, \langle 1,4,3,2\rangle\rangle$, 350: $\langle\langle 3,2,1,4\rangle, \langle 4,1,2,3\rangle, \langle 2,4,3,1\rangle, \langle 1,3,4,2\rangle\rangle$
351: $\langle\langle 3,2,1,4\rangle, \langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle, \langle 2,3,4,1\rangle\rangle$, 352: $\langle\langle 3,2,1,4\rangle, \langle 4,1,3,2\rangle, \langle 2,3,4,1\rangle, \langle 1,4,2,3\rangle\rangle$
353: $\langle\langle 3,2,1,4\rangle, \langle 2,3,4,1\rangle, \langle 1,4,2,3\rangle, \langle 4,1,3,2\rangle\rangle$, 354: $\langle\langle 3,2,1,4\rangle, \langle 2,3,4,1\rangle, \langle 1,4,3,2\rangle, \langle 4,1,2,3\rangle\rangle$
355: $\langle\langle 3,2,1,4\rangle, \langle 2,3,4,1\rangle, \langle 4,1,2,3\rangle, \langle 1,4,3,2\rangle\rangle$, 356: $\langle\langle 3,2,1,4\rangle, \langle 2,3,4,1\rangle, \langle 4,1,3,2\rangle, \langle 1,4,2,3\rangle\rangle$
357: $\langle\langle 3,2,1,4\rangle, \langle 2,4,3,1\rangle, \langle 1,3,4,2\rangle, \langle 4,1,2,3\rangle\rangle$, 358: $\langle\langle 3,2,1,4\rangle, \langle 2,4,3,1\rangle, \langle 4,1,2,3\rangle, \langle 1,3,4,2\rangle\rangle$
359: $\langle\langle 3,2,1,4\rangle, \langle 4,3,2,1\rangle, \langle 2,1,4,3\rangle, \langle 1,4,3,2\rangle\rangle$, 360: $\langle\langle 3,2,1,4\rangle, \langle 4,3,2,1\rangle, \langle 1,4,3,2\rangle, \langle 2,1,4,3\rangle\rangle$
361: $\langle\langle 3,2,4,1\rangle, \langle 2,1,3,4\rangle, \langle 1,4,2,3\rangle, \langle 4,3,1,2\rangle\rangle$, 362: $\langle\langle 3,2,4,1\rangle, \langle 2,1,3,4\rangle, \langle 4,3,1,2\rangle, \langle 1,4,2,3\rangle\rangle$

363: $\langle\langle 3,2,4,1\rangle,\langle 1,3,2,4\rangle,\langle 4,1,3,2\rangle,\langle 2,4,1,3\rangle\rangle$, 364: $\langle\langle 3,2,4,1\rangle,\langle 1,3,2,4\rangle,\langle 2,4,1,3\rangle,\langle 4,1,3,2\rangle\rangle$
365: $\langle\langle 3,2,4,1\rangle,\langle 1,4,2,3\rangle,\langle 2,1,3,4\rangle,\langle 4,3,1,2\rangle\rangle$, 366: $\langle\langle 3,2,4,1\rangle,\langle 1,4,2,3\rangle,\langle 4,1,3,2\rangle,\langle 2,3,1,4\rangle\rangle$
367: $\langle\langle 3,2,4,1\rangle,\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle,\langle 4,1,3,2\rangle\rangle$, 368: $\langle\langle 3,2,4,1\rangle,\langle 1,4,2,3\rangle,\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle\rangle$
369: $\langle\langle 3,2,4,1\rangle,\langle 1,4,3,2\rangle,\langle 4,1,2,3\rangle,\langle 2,3,1,4\rangle\rangle$, 370: $\langle\langle 3,2,4,1\rangle,\langle 1,4,3,2\rangle,\langle 2,3,1,4\rangle,\langle 4,1,2,3\rangle\rangle$
371: $\langle\langle 3,2,4,1\rangle,\langle 4,1,2,3\rangle,\langle 1,4,3,2\rangle,\langle 2,3,1,4\rangle\rangle$, 372: $\langle\langle 3,2,4,1\rangle,\langle 4,1,2,3\rangle,\langle 2,3,1,4\rangle,\langle 1,4,3,2\rangle\rangle$
373: $\langle\langle 3,2,4,1\rangle,\langle 4,1,3,2\rangle,\langle 1,3,2,4\rangle,\langle 2,4,1,3\rangle\rangle$, 374: $\langle\langle 3,2,4,1\rangle,\langle 4,1,3,2\rangle,\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle\rangle$
375: $\langle\langle 3,2,4,1\rangle,\langle 4,1,3,2\rangle,\langle 2,3,1,4\rangle,\langle 1,4,2,3\rangle\rangle$, 376: $\langle\langle 3,2,4,1\rangle,\langle 4,1,3,2\rangle,\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle\rangle$
377: $\langle\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle,\langle 1,4,2,3\rangle,\langle 4,1,3,2\rangle\rangle$, 378: $\langle\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle,\langle 1,4,3,2\rangle,\langle 4,1,2,3\rangle\rangle$
379: $\langle\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle,\langle 4,1,2,3\rangle,\langle 1,4,3,2\rangle\rangle$, 380: $\langle\langle 3,2,4,1\rangle,\langle 2,3,1,4\rangle,\langle 4,1,3,2\rangle,\langle 1,4,2,3\rangle\rangle$
381: $\langle\langle 3,2,4,1\rangle,\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle,\langle 4,1,3,2\rangle\rangle$, 382: $\langle\langle 3,2,4,1\rangle,\langle 2,4,1,3\rangle,\langle 4,1,3,2\rangle,\langle 1,3,2,4\rangle\rangle$
383: $\langle\langle 3,2,4,1\rangle,\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle,\langle 1,4,2,3\rangle\rangle$, 384: $\langle\langle 3,2,4,1\rangle,\langle 4,3,1,2\rangle,\langle 1,4,2,3\rangle,\langle 2,1,3,4\rangle\rangle$
385: $\langle\langle 2,4,1,3\rangle,\langle 1,2,3,4\rangle,\langle 3,1,4,2\rangle,\langle 4,3,2,1\rangle\rangle$, 386: $\langle\langle 2,4,1,3\rangle,\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle,\langle 3,1,4,2\rangle\rangle$
387: $\langle\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle,\langle 3,1,4,2\rangle,\langle 4,2,3,1\rangle\rangle$, 388: $\langle\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle,\langle 4,1,3,2\rangle,\langle 3,2,4,1\rangle\rangle$
389: $\langle\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle,\langle 3,2,4,1\rangle,\langle 4,1,3,2\rangle\rangle$, 390: $\langle\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle,\langle 4,2,3,1\rangle,\langle 3,1,4,2\rangle\rangle$
391: $\langle\langle 2,4,1,3\rangle,\langle 1,3,4,2\rangle,\langle 3,1,2,4\rangle,\langle 4,2,3,1\rangle\rangle$, 392: $\langle\langle 2,4,1,3\rangle,\langle 1,3,4,2\rangle,\langle 4,2,3,1\rangle,\langle 3,1,2,4\rangle\rangle$
393: $\langle\langle 2,4,1,3\rangle,\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle,\langle 4,2,3,1\rangle\rangle$, 394: $\langle\langle 2,4,1,3\rangle,\langle 3,1,2,4\rangle,\langle 4,2,3,1\rangle,\langle 1,3,4,2\rangle\rangle$
395: $\langle\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle,\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle\rangle$, 396: $\langle\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle,\langle 4,2,3,1\rangle\rangle$
397: $\langle\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle,\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle\rangle$, 398: $\langle\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle,\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle\rangle$
399: $\langle\langle 2,4,1,3\rangle,\langle 4,1,3,2\rangle,\langle 1,3,2,4\rangle,\langle 3,2,4,1\rangle\rangle$, 400: $\langle\langle 2,4,1,3\rangle,\langle 4,1,3,2\rangle,\langle 3,2,4,1\rangle,\langle 1,3,2,4\rangle\rangle$
401: $\langle\langle 2,4,1,3\rangle,\langle 3,2,4,1\rangle,\langle 1,3,2,4\rangle,\langle 4,1,3,2\rangle\rangle$, 402: $\langle\langle 2,4,1,3\rangle,\langle 3,2,4,1\rangle,\langle 4,1,3,2\rangle,\langle 1,3,2,4\rangle\rangle$
403: $\langle\langle 2,4,1,3\rangle,\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle,\langle 3,1,4,2\rangle\rangle$, 404: $\langle\langle 2,4,1,3\rangle,\langle 4,2,3,1\rangle,\langle 1,3,4,2\rangle,\langle 3,1,2,4\rangle\rangle$
405: $\langle\langle 2,4,1,3\rangle,\langle 4,2,3,1\rangle,\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle\rangle$, 406: $\langle\langle 2,4,1,3\rangle,\langle 4,2,3,1\rangle,\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle\rangle$
407: $\langle\langle 2,4,1,3\rangle,\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle,\langle 3,1,4,2\rangle\rangle$, 408: $\langle\langle 2,4,1,3\rangle,\langle 4,3,2,1\rangle,\langle 3,1,4,2\rangle,\langle 1,2,3,4\rangle\rangle$
409: $\langle\langle 2,4,3,1\rangle,\langle 1,2,4,3\rangle,\langle 3,1,2,4\rangle,\langle 4,3,1,2\rangle\rangle$, 410: $\langle\langle 2,4,3,1\rangle,\langle 1,2,4,3\rangle,\langle 4,3,1,2\rangle,\langle 3,1,2,4\rangle\rangle$
411: $\langle\langle 2,4,3,1\rangle,\langle 1,3,2,4\rangle,\langle 3,1,4,2\rangle,\langle 4,2,1,3\rangle\rangle$, 412: $\langle\langle 2,4,3,1\rangle,\langle 1,3,2,4\rangle,\langle 4,2,1,3\rangle,\langle 3,1,4,2\rangle\rangle$
413: $\langle\langle 2,4,3,1\rangle,\langle 1,3,4,2\rangle,\langle 3,1,2,4\rangle,\langle 4,2,1,3\rangle\rangle$, 414: $\langle\langle 2,4,3,1\rangle,\langle 1,3,4,2\rangle,\langle 4,1,2,3\rangle,\langle 3,2,1,4\rangle\rangle$
415: $\langle\langle 2,4,3,1\rangle,\langle 1,3,4,2\rangle,\langle 3,2,1,4\rangle,\langle 4,1,2,3\rangle\rangle$, 416: $\langle\langle 2,4,3,1\rangle,\langle 1,3,4,2\rangle,\langle 4,2,1,3\rangle,\langle 3,1,2,4\rangle\rangle$
417: $\langle\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle,\langle 1,2,4,3\rangle,\langle 4,3,1,2\rangle\rangle$, 418: $\langle\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle,\langle 4,2,1,3\rangle\rangle$
419: $\langle\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle,\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle\rangle$, 420: $\langle\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle,\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle\rangle$
421: $\langle\langle 2,4,3,1\rangle,\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle,\langle 4,2,1,3\rangle\rangle$, 422: $\langle\langle 2,4,3,1\rangle,\langle 3,1,4,2\rangle,\langle 4,2,1,3\rangle,\langle 1,3,2,4\rangle\rangle$
423: $\langle\langle 2,4,3,1\rangle,\langle 4,1,2,3\rangle,\langle 1,3,4,2\rangle,\langle 3,2,1,4\rangle\rangle$, 424: $\langle\langle 2,4,3,1\rangle,\langle 4,1,2,3\rangle,\langle 3,2,1,4\rangle,\langle 1,3,4,2\rangle\rangle$
425: $\langle\langle 2,4,3,1\rangle,\langle 3,2,1,4\rangle,\langle 1,3,4,2\rangle,\langle 4,1,2,3\rangle\rangle$, 426: $\langle\langle 2,4,3,1\rangle,\langle 3,2,1,4\rangle,\langle 4,1,2,3\rangle,\langle 1,3,4,2\rangle\rangle$
427: $\langle\langle 2,4,3,1\rangle,\langle 4,2,1,3\rangle,\langle 1,3,2,4\rangle,\langle 3,1,4,2\rangle\rangle$, 428: $\langle\langle 2,4,3,1\rangle,\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle,\langle 3,1,2,4\rangle\rangle$
429: $\langle\langle 2,4,3,1\rangle,\langle 4,2,1,3\rangle,\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle\rangle$, 430: $\langle\langle 2,4,3,1\rangle,\langle 4,2,1,3\rangle,\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle\rangle$
431: $\langle\langle 2,4,3,1\rangle,\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle,\langle 3,1,2,4\rangle\rangle$, 432: $\langle\langle 2,4,3,1\rangle,\langle 4,3,1,2\rangle,\langle 3,1,2,4\rangle,\langle 1,2,4,3\rangle\rangle$
433: $\langle\langle 4,2,1,3\rangle,\langle 2,1,3,4\rangle,\langle 1,3,4,2\rangle,\langle 3,4,2,1\rangle\rangle$, 434: $\langle\langle 4,2,1,3\rangle,\langle 2,1,3,4\rangle,\langle 3,4,2,1\rangle,\langle 1,3,4,2\rangle\rangle$
435: $\langle\langle 4,2,1,3\rangle,\langle 1,3,2,4\rangle,\langle 3,1,4,2\rangle,\langle 2,4,3,1\rangle\rangle$, 436: $\langle\langle 4,2,1,3\rangle,\langle 1,3,2,4\rangle,\langle 2,4,3,1\rangle,\langle 3,1,4,2\rangle\rangle$
437: $\langle\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle,\langle 2,1,3,4\rangle,\langle 3,4,2,1\rangle\rangle$, 438: $\langle\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle,\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle\rangle$
439: $\langle\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle,\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle\rangle$, 440: $\langle\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle,\langle 3,4,2,1\rangle,\langle 2,1,3,4\rangle\rangle$
441: $\langle\langle 4,2,1,3\rangle,\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle,\langle 2,4,3,1\rangle\rangle$, 442: $\langle\langle 4,2,1,3\rangle,\langle 3,1,2,4\rangle,\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle\rangle$
443: $\langle\langle 4,2,1,3\rangle,\langle 3,1,2,4\rangle,\langle 2,3,4,1\rangle,\langle 1,4,3,2\rangle\rangle$, 444: $\langle\langle 4,2,1,3\rangle,\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle,\langle 1,3,4,2\rangle\rangle$
445: $\langle\langle 4,2,1,3\rangle,\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle,\langle 2,4,3,1\rangle\rangle$, 446: $\langle\langle 4,2,1,3\rangle,\langle 3,1,4,2\rangle,\langle 2,4,3,1\rangle,\langle 1,3,2,4\rangle\rangle$
447: $\langle\langle 4,2,1,3\rangle,\langle 1,4,3,2\rangle,\langle 3,1,2,4\rangle,\langle 2,3,4,1\rangle\rangle$, 448: $\langle\langle 4,2,1,3\rangle,\langle 1,4,3,2\rangle,\langle 2,3,4,1\rangle,\langle 3,1,2,4\rangle\rangle$
449: $\langle\langle 4,2,1,3\rangle,\langle 2,3,4,1\rangle,\langle 3,1,2,4\rangle,\langle 1,4,3,2\rangle\rangle$, 450: $\langle\langle 4,2,1,3\rangle,\langle 2,3,4,1\rangle,\langle 1,4,3,2\rangle,\langle 3,1,2,4\rangle\rangle$
451: $\langle\langle 4,2,1,3\rangle,\langle 2,4,3,1\rangle,\langle 1,3,2,4\rangle,\langle 3,1,4,2\rangle\rangle$, 452: $\langle\langle 4,2,1,3\rangle,\langle 2,4,3,1\rangle,\langle 1,3,4,2\rangle,\langle 3,1,2,4\rangle\rangle$
453: $\langle\langle 4,2,1,3\rangle,\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle\rangle$, 454: $\langle\langle 4,2,1,3\rangle,\langle 2,4,3,1\rangle,\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle\rangle$
455: $\langle\langle 4,2,1,3\rangle,\langle 3,4,2,1\rangle,\langle 2,1,3,4\rangle,\langle 1,3,4,2\rangle\rangle$, 456: $\langle\langle 4,2,1,3\rangle,\langle 3,4,2,1\rangle,\langle 1,3,4,2\rangle,\langle 2,1,3,4\rangle\rangle$
457: $\langle\langle 4,2,3,1\rangle,\langle 2,1,4,3\rangle,\langle 1,3,2,4\rangle,\langle 3,4,1,2\rangle\rangle$, 458: $\langle\langle 4,2,3,1\rangle,\langle 2,1,4,3\rangle,\langle 3,4,1,2\rangle,\langle 1,3,2,4\rangle\rangle$
459: $\langle\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle,\langle 2,1,4,3\rangle,\langle 3,4,1,2\rangle\rangle$, 460: $\langle\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle,\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle\rangle$
461: $\langle\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle,\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle\rangle$, 462: $\langle\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle,\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle\rangle$
463: $\langle\langle 4,2,3,1\rangle,\langle 1,3,4,2\rangle,\langle 3,1,2,4\rangle,\langle 2,4,1,3\rangle\rangle$, 464: $\langle\langle 4,2,3,1\rangle,\langle 1,3,4,2\rangle,\langle 2,4,1,3\rangle,\langle 3,1,2,4\rangle\rangle$
465: $\langle\langle 4,2,3,1\rangle,\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle,\langle 2,4,1,3\rangle\rangle$, 466: $\langle\langle 4,2,3,1\rangle,\langle 3,1,2,4\rangle,\langle 2,4,1,3\rangle,\langle 1,3,4,2\rangle\rangle$
467: $\langle\langle 4,2,3,1\rangle,\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle,\langle 2,4,1,3\rangle\rangle$, 468: $\langle\langle 4,2,3,1\rangle,\langle 3,1,4,2\rangle,\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle\rangle$
469: $\langle\langle 4,2,3,1\rangle,\langle 3,1,4,2\rangle,\langle 2,3,1,4\rangle,\langle 1,4,2,3\rangle\rangle$, 470: $\langle\langle 4,2,3,1\rangle,\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle\rangle$

471: $\langle\langle 4,2,3,1\rangle,\langle 1,4,2,3\rangle,\langle 3,1,4,2\rangle,\langle 2,3,1,4\rangle\rangle$, 472: $\langle\langle 4,2,3,1\rangle,\langle 1,4,2,3\rangle,\langle 2,3,1,4\rangle,\langle 3,1,4,2\rangle\rangle$
473: $\langle\langle 4,2,3,1\rangle,\langle 2,3,1,4\rangle,\langle 3,1,4,2\rangle,\langle 1,4,2,3\rangle\rangle$, 474: $\langle\langle 4,2,3,1\rangle,\langle 2,3,1,4\rangle,\langle 1,4,2,3\rangle,\langle 3,1,4,2\rangle\rangle$
475: $\langle\langle 4,2,3,1\rangle,\langle 2,4,1,3\rangle,\langle 1,3,2,4\rangle,\langle 3,1,4,2\rangle\rangle$, 476: $\langle\langle 4,2,3,1\rangle,\langle 2,4,1,3\rangle,\langle 1,3,4,2\rangle,\langle 3,1,2,4\rangle\rangle$
477: $\langle\langle 4,2,3,1\rangle,\langle 2,4,1,3\rangle,\langle 3,1,2,4\rangle,\langle 1,3,4,2\rangle\rangle$, 478: $\langle\langle 4,2,3,1\rangle,\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle,\langle 1,3,2,4\rangle\rangle$
479: $\langle\langle 4,2,3,1\rangle,\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle,\langle 1,3,2,4\rangle\rangle$, 480: $\langle\langle 4,2,3,1\rangle,\langle 3,4,1,2\rangle,\langle 1,3,2,4\rangle,\langle 2,1,4,3\rangle\rangle$
481: $\langle\langle 3,4,1,2\rangle,\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle,\langle 4,3,2,1\rangle\rangle$, 482: $\langle\langle 3,4,1,2\rangle,\langle 1,2,3,4\rangle,\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle\rangle$
483: $\langle\langle 3,4,1,2\rangle,\langle 1,2,3,4\rangle,\langle 2,3,4,1\rangle,\langle 4,1,2,3\rangle\rangle$, 484: $\langle\langle 3,4,1,2\rangle,\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle\rangle$
485: $\langle\langle 3,4,1,2\rangle,\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle,\langle 4,3,2,1\rangle\rangle$, 486: $\langle\langle 3,4,1,2\rangle,\langle 1,2,4,3\rangle,\langle 4,3,2,1\rangle,\langle 2,1,3,4\rangle\rangle$
487: $\langle\langle 3,4,1,2\rangle,\langle 2,1,3,4\rangle,\langle 1,2,4,3\rangle,\langle 4,3,2,1\rangle\rangle$, 488: $\langle\langle 3,4,1,2\rangle,\langle 2,1,3,4\rangle,\langle 4,3,2,1\rangle,\langle 1,2,4,3\rangle\rangle$
489: $\langle\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle,\langle 1,2,3,4\rangle,\langle 4,3,2,1\rangle\rangle$, 490: $\langle\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle,\langle 1,3,2,4\rangle,\langle 4,2,3,1\rangle\rangle$
491: $\langle\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle,\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle\rangle$, 492: $\langle\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle,\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle\rangle$
493: $\langle\langle 3,4,1,2\rangle,\langle 1,3,2,4\rangle,\langle 2,1,4,3\rangle,\langle 4,2,3,1\rangle\rangle$, 494: $\langle\langle 3,4,1,2\rangle,\langle 1,3,2,4\rangle,\langle 4,2,3,1\rangle,\langle 2,1,4,3\rangle\rangle$
495: $\langle\langle 3,4,1,2\rangle,\langle 4,1,2,3\rangle,\langle 1,2,3,4\rangle,\langle 2,3,4,1\rangle\rangle$, 496: $\langle\langle 3,4,1,2\rangle,\langle 4,1,2,3\rangle,\langle 2,3,4,1\rangle,\langle 1,2,3,4\rangle\rangle$
497: $\langle\langle 3,4,1,2\rangle,\langle 2,3,4,1\rangle,\langle 1,2,3,4\rangle,\langle 4,1,2,3\rangle\rangle$, 498: $\langle\langle 3,4,1,2\rangle,\langle 2,3,4,1\rangle,\langle 4,1,2,3\rangle,\langle 1,2,3,4\rangle\rangle$
499: $\langle\langle 3,4,1,2\rangle,\langle 4,2,3,1\rangle,\langle 2,1,4,3\rangle,\langle 1,3,2,4\rangle\rangle$, 500: $\langle\langle 3,4,1,2\rangle,\langle 4,2,3,1\rangle,\langle 1,3,2,4\rangle,\langle 2,1,4,3\rangle\rangle$
501: $\langle\langle 3,4,1,2\rangle,\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle\rangle$, 502: $\langle\langle 3,4,1,2\rangle,\langle 4,3,2,1\rangle,\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle\rangle$
503: $\langle\langle 3,4,1,2\rangle,\langle 4,3,2,1\rangle,\langle 2,1,3,4\rangle,\langle 1,2,4,3\rangle\rangle$, 504: $\langle\langle 3,4,1,2\rangle,\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle,\langle 1,2,3,4\rangle\rangle$
505: $\langle\langle 3,4,2,1\rangle,\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle,\langle 4,3,1,2\rangle\rangle$, 506: $\langle\langle 3,4,2,1\rangle,\langle 1,2,3,4\rangle,\langle 4,3,1,2\rangle,\langle 2,1,4,3\rangle\rangle$
507: $\langle\langle 3,4,2,1\rangle,\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle,\langle 4,3,1,2\rangle\rangle$, 508: $\langle\langle 3,4,2,1\rangle,\langle 1,2,4,3\rangle,\langle 4,1,3,2\rangle,\langle 2,3,1,4\rangle\rangle$
509: $\langle\langle 3,4,2,1\rangle,\langle 1,2,4,3\rangle,\langle 2,3,1,4\rangle,\langle 4,1,3,2\rangle\rangle$, 510: $\langle\langle 3,4,2,1\rangle,\langle 1,2,4,3\rangle,\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle\rangle$
511: $\langle\langle 3,4,2,1\rangle,\langle 2,1,3,4\rangle,\langle 1,2,4,3\rangle,\langle 4,3,1,2\rangle\rangle$, 512: $\langle\langle 3,4,2,1\rangle,\langle 2,1,3,4\rangle,\langle 1,3,4,2\rangle,\langle 4,2,1,3\rangle\rangle$
513: $\langle\langle 3,4,2,1\rangle,\langle 2,1,3,4\rangle,\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle\rangle$, 514: $\langle\langle 3,4,2,1\rangle,\langle 2,1,3,4\rangle,\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle\rangle$
515: $\langle\langle 3,4,2,1\rangle,\langle 2,1,4,3\rangle,\langle 1,2,3,4\rangle,\langle 4,3,1,2\rangle\rangle$, 516: $\langle\langle 3,4,2,1\rangle,\langle 2,1,4,3\rangle,\langle 4,3,1,2\rangle,\langle 1,2,3,4\rangle\rangle$
517: $\langle\langle 3,4,2,1\rangle,\langle 1,3,4,2\rangle,\langle 2,1,3,4\rangle,\langle 4,2,1,3\rangle\rangle$, 518: $\langle\langle 3,4,2,1\rangle,\langle 1,3,4,2\rangle,\langle 4,2,1,3\rangle,\langle 2,1,3,4\rangle\rangle$
519: $\langle\langle 3,4,2,1\rangle,\langle 4,1,3,2\rangle,\langle 1,2,4,3\rangle,\langle 2,3,1,4\rangle\rangle$, 520: $\langle\langle 3,4,2,1\rangle,\langle 4,1,3,2\rangle,\langle 2,3,1,4\rangle,\langle 1,2,4,3\rangle\rangle$
521: $\langle\langle 3,4,2,1\rangle,\langle 2,3,1,4\rangle,\langle 1,2,4,3\rangle,\langle 4,1,3,2\rangle\rangle$, 522: $\langle\langle 3,4,2,1\rangle,\langle 2,3,1,4\rangle,\langle 4,1,3,2\rangle,\langle 1,2,4,3\rangle\rangle$
523: $\langle\langle 3,4,2,1\rangle,\langle 4,2,1,3\rangle,\langle 2,1,3,4\rangle,\langle 1,3,4,2\rangle\rangle$, 524: $\langle\langle 3,4,2,1\rangle,\langle 4,2,1,3\rangle,\langle 1,3,4,2\rangle,\langle 2,1,3,4\rangle\rangle$
525: $\langle\langle 3,4,2,1\rangle,\langle 4,3,1,2\rangle,\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle\rangle$, 526: $\langle\langle 3,4,2,1\rangle,\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle\rangle$
527: $\langle\langle 3,4,2,1\rangle,\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle,\langle 1,2,4,3\rangle\rangle$, 528: $\langle\langle 3,4,2,1\rangle,\langle 4,3,1,2\rangle,\langle 2,1,4,3\rangle,\langle 1,2,3,4\rangle\rangle$
529: $\langle\langle 4,3,1,2\rangle,\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle,\langle 3,4,2,1\rangle\rangle$, 530: $\langle\langle 4,3,1,2\rangle,\langle 1,2,3,4\rangle,\langle 3,4,2,1\rangle,\langle 2,1,4,3\rangle\rangle$
531: $\langle\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle,\langle 3,4,2,1\rangle\rangle$, 532: $\langle\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle,\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle\rangle$
533: $\langle\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle,\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle\rangle$, 534: $\langle\langle 4,3,1,2\rangle,\langle 1,2,4,3\rangle,\langle 3,4,2,1\rangle,\langle 2,1,3,4\rangle\rangle$
535: $\langle\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle,\langle 1,2,4,3\rangle,\langle 3,4,2,1\rangle\rangle$, 536: $\langle\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle,\langle 1,4,2,3\rangle,\langle 3,2,4,1\rangle\rangle$
537: $\langle\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle,\langle 3,2,4,1\rangle,\langle 1,4,2,3\rangle\rangle$, 538: $\langle\langle 4,3,1,2\rangle,\langle 2,1,3,4\rangle,\langle 3,4,2,1\rangle,\langle 1,2,4,3\rangle\rangle$
539: $\langle\langle 4,3,1,2\rangle,\langle 2,1,4,3\rangle,\langle 1,2,3,4\rangle,\langle 3,4,2,1\rangle\rangle$, 540: $\langle\langle 4,3,1,2\rangle,\langle 2,1,4,3\rangle,\langle 3,4,2,1\rangle,\langle 1,2,3,4\rangle\rangle$
541: $\langle\langle 4,3,1,2\rangle,\langle 3,1,2,4\rangle,\langle 1,2,4,3\rangle,\langle 2,4,3,1\rangle\rangle$, 542: $\langle\langle 4,3,1,2\rangle,\langle 3,1,2,4\rangle,\langle 2,4,3,1\rangle,\langle 1,2,4,3\rangle\rangle$
543: $\langle\langle 4,3,1,2\rangle,\langle 1,4,2,3\rangle,\langle 2,1,3,4\rangle,\langle 3,2,4,1\rangle\rangle$, 544: $\langle\langle 4,3,1,2\rangle,\langle 1,4,2,3\rangle,\langle 3,2,4,1\rangle,\langle 2,1,3,4\rangle\rangle$
545: $\langle\langle 4,3,1,2\rangle,\langle 3,2,4,1\rangle,\langle 2,1,3,4\rangle,\langle 1,4,2,3\rangle\rangle$, 546: $\langle\langle 4,3,1,2\rangle,\langle 3,2,4,1\rangle,\langle 1,4,2,3\rangle,\langle 2,1,3,4\rangle\rangle$
547: $\langle\langle 4,3,1,2\rangle,\langle 2,4,3,1\rangle,\langle 1,2,4,3\rangle,\langle 3,1,2,4\rangle\rangle$, 548: $\langle\langle 4,3,1,2\rangle,\langle 2,4,3,1\rangle,\langle 3,1,2,4\rangle,\langle 1,2,4,3\rangle\rangle$
549: $\langle\langle 4,3,1,2\rangle,\langle 3,4,2,1\rangle,\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle\rangle$, 550: $\langle\langle 4,3,1,2\rangle,\langle 3,4,2,1\rangle,\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle\rangle$
551: $\langle\langle 4,3,1,2\rangle,\langle 3,4,2,1\rangle,\langle 2,1,3,4\rangle,\langle 1,2,4,3\rangle\rangle$, 552: $\langle\langle 4,3,1,2\rangle,\langle 3,4,2,1\rangle,\langle 2,1,4,3\rangle,\langle 1,2,3,4\rangle\rangle$
553: $\langle\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle,\langle 3,4,1,2\rangle\rangle$, 554: $\langle\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle,\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle\rangle$
555: $\langle\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle,\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle\rangle$, 556: $\langle\langle 4,3,2,1\rangle,\langle 1,2,3,4\rangle,\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle\rangle$
557: $\langle\langle 4,3,2,1\rangle,\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle,\langle 3,4,1,2\rangle\rangle$, 558: $\langle\langle 4,3,2,1\rangle,\langle 1,2,4,3\rangle,\langle 3,4,1,2\rangle,\langle 2,1,3,4\rangle\rangle$
559: $\langle\langle 4,3,2,1\rangle,\langle 2,1,3,4\rangle,\langle 1,2,4,3\rangle,\langle 3,4,1,2\rangle\rangle$, 560: $\langle\langle 4,3,2,1\rangle,\langle 2,1,3,4\rangle,\langle 3,4,1,2\rangle,\langle 1,2,4,3\rangle\rangle$
561: $\langle\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle,\langle 1,2,3,4\rangle,\langle 3,4,1,2\rangle\rangle$, 562: $\langle\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle,\langle 1,4,3,2\rangle,\langle 3,2,1,4\rangle\rangle$
563: $\langle\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle,\langle 3,2,1,4\rangle,\langle 1,4,3,2\rangle\rangle$, 564: $\langle\langle 4,3,2,1\rangle,\langle 2,1,4,3\rangle,\langle 3,4,1,2\rangle,\langle 1,2,3,4\rangle\rangle$
565: $\langle\langle 4,3,2,1\rangle,\langle 3,1,4,2\rangle,\langle 1,2,3,4\rangle,\langle 2,4,1,3\rangle\rangle$, 566: $\langle\langle 4,3,2,1\rangle,\langle 3,1,4,2\rangle,\langle 2,4,1,3\rangle,\langle 1,2,3,4\rangle\rangle$
567: $\langle\langle 4,3,2,1\rangle,\langle 1,4,3,2\rangle,\langle 2,1,4,3\rangle,\langle 3,2,1,4\rangle\rangle$, 568: $\langle\langle 4,3,2,1\rangle,\langle 1,4,3,2\rangle,\langle 3,2,1,4\rangle,\langle 2,1,4,3\rangle\rangle$
569: $\langle\langle 4,3,2,1\rangle,\langle 3,2,1,4\rangle,\langle 2,1,4,3\rangle,\langle 1,4,3,2\rangle\rangle$, 570: $\langle\langle 4,3,2,1\rangle,\langle 3,2,1,4\rangle,\langle 1,4,3,2\rangle,\langle 2,1,4,3\rangle\rangle$
571: $\langle\langle 4,3,2,1\rangle,\langle 2,4,1,3\rangle,\langle 1,2,3,4\rangle,\langle 3,1,4,2\rangle\rangle$, 572: $\langle\langle 4,3,2,1\rangle,\langle 2,4,1,3\rangle,\langle 3,1,4,2\rangle,\langle 1,2,3,4\rangle\rangle$
573: $\langle\langle 4,3,2,1\rangle,\langle 3,4,1,2\rangle,\langle 1,2,3,4\rangle,\langle 2,1,4,3\rangle\rangle$, 574: $\langle\langle 4,3,2,1\rangle,\langle 3,4,1,2\rangle,\langle 1,2,4,3\rangle,\langle 2,1,3,4\rangle\rangle$
575: $\langle\langle 4,3,2,1\rangle,\langle 3,4,1,2\rangle,\langle 2,1,3,4\rangle,\langle 1,2,4,3\rangle\rangle$, 576: $\langle\langle 4,3,2,1\rangle,\langle 3,4,1,2\rangle,\langle 2,1,4,3\rangle,\langle 1,2,3,4\rangle\rangle$

Appendix B. Invalid Runs (In Decimal and Binary Forms)

Decimal and binary representations of each state are separated by ":". Consecutive states of each run are separated by ",". Consecutive runs are separated by |.

580: 1001000100 | 608: 1001100000 | 585: 1001001001, 868: 1101100100 | 834: 1101000010 | 656: 1010010000 | 596: 1001010100 | 706: 1011000010 | 752: 1011110000 | 599: 1001010111, 875: 1101101011, 1013: 1111110101, 954: 1110111010 | 686: 1010101110 | 747: 1011101011, 821: 1100110101, 986: 1111011010 | 694: 1010110110 | 749: 1011101101, 822: 1100110110 | 653: 1010001101, 774: 1100000110 | 641: 1010000001, 768: 1100000000 | 577: 1001000001, 864: 1101100000 | 589: 1001001101, 870: 1101100110 | 665: 1010011001, 780: 1100001100 | 848: 1101010000 | 602: 1001011010 | 726: 1011010110 | 757: 1011110101, 826: 1100111010 | 654: 1010001110 | 739: 1011100011, 817: 1100110001, 984: 1111011000 | 637: 1001111101, 894: 1101111110 | 671: 1010011111, 783: 1100001111, 967: 1111000111, 931: 1110100011, 913: 1110010001, 904: 1110001000 | 632: 1001111000 | 615: 1001100111, 883: 1101110011, 1017: 1111111001, 956: 1110111100 | 859: 1101011011, 1005: 1111101101, 950: 1110110110 685: 1010101101, 790: 1100010110 | 645: 1010000101, 770: 1100000010 | 640: 1010000000 | 578: 1001000010 | 720: 1011010000 | 598: 1001010110 | 725: 1011010101, 810: 1100101010 | 650: 1010001010 | 738: 1011100010 | 760: 1011111000 | 623: 1001101111, 887: 1101110111, 1019: 1111111011, 957: 1110111101, 926: 1110011110 | 679: 1010100111, 787: 1100010011, 969: 1111001001, 932: 1110100100 | 705: 1011000001, 800: 1100100000 | 588: 1001001100 | 836: 1101000100 | 610: 1001100010 | 728: 1011011000 | 621: 1001101101, 886: 1101110110 | 669: 1010011101, 782: 1100001110 | 643: 1010000011, 769: 1100000001, 960: 1111000000 | 583: 1001000111, 867: 1101100011, 1009: 1111110001, 952: 1110111000 | 635: 1001111011, 893: 1101111101, 1022: 1111111110 | 703: 1010111111, 799: 1100011111, 975: 1111001111, 935: 1110100111, 915: 1110010011, 905: 1110001001, 900: 1110000100 | 593: 1001010001, 872: 1101101000 | 630: 1001110110 | 733: 1011011101, 814: 1100101110 | 651: 1010001011, 773: 1100000101, 962: 1111000010 | 688: 1010110000 | 597: 1001010101, 874: 1101101010 | 666: 1010011010 | 742: 1011100110 | 761: 1011111001, 828: 1100111100 | 851: 1101010011, 1001: 1111101001, 948: 1110110100 | 717: 1011001101, 806: 1100100110 | 649: 1010001001, 772: 1100000100 | 584: 1001001000 | 612: 1001100100 | 832: 1101000000 | 582: 1001000110 | 721: 1011010001, 808: 1100101000 | 626: 1001110010, 732: 1011011100 | 845: 1101001101, 998: 1111100110 | 697: 1010111001, 796: 1100011100 | 849: 1101010001, 1000: 1111101000 | 638: 1001111110 | 735: 1011011111, 815: 1100101111, 983: 1111010111, 939: 1110101011, 917: 1110010101, 906: 1110001010 | 674: 1010100010 | 744: 1011101000 | 622: 1001101110 | 731: 1011011011, 813: 1100101101, 982: 1111010110 | 693: 1010110101, 794: 1100011010 | 646: 1010000110 | 737: 1011100001, 816: 1100110000 | 601: 1001011001, 876: 1101101100 | 854: 1101010110 | 661: 1010010101, 778: 1100001010 | 642: 1010000010 | 736: 1011100000 | 587: 1001001011, 869: 1101100101, 1010: 1111110010 | 700: 1010111100 | 843: 1101001011, 997: 1111100101, 946: 1110110010 | 684: 1010101100 | 842: 1101001010 | 658: 1010010010 | 740: 1011100100 | 833: 1101000001, 992: 1111100000 | 591: 1001001111, 871: 1101100111, 1011: 1111110011, 953: 1110111001, 924: 1110011100 | 857: 1101011001, 1004: 1111101100 | 862: 1101011110 | 663: 1010010111, 779: 1100001011, 965: 1111000101, 930: 1110100010 | 680: 1010101000 | 618: 1001101010 | 730: 1011011010 | 758: 1011110110 | 765: 1011111101, 830: 1100111110 | 655: 1010001111, 775: 1100000111, 963: 1111000011, 929: 1110100001, 912: 1110010000 | 604: 1001011100 | 837: 1101000101, 994: 1111100010 | 696: 1010111000 | 619: 1001101011, 885: 1101110101, 1018: 1111111010 | 702: 1010111110 | 751: 1011101111, 823: 1100110111, 987: 1111011011, 941: 1110101101, 918: 1110010110 | 677: 1010100101, 786: 1100010010 | 644: 1010000100 | 592: 1001010000 | 594: 1001010010 | 724: 1011010100 | 710: 1011000110 | 753: 1011110001, 824: 1100111000 | 627: 1001110011, 889: 1101111001, 1020: 1111111100 | 863: 1101011111, 1007: 1111101111, 951: 1110110111, 923: 1110011011, 909: 1110001101, 902: 1110000110 | 673: 1010100001, 784: 1100010000 | 600: 1001011000 | 613: 1001100101, 882: 1101110010 | 668: 1010011100 | 841: 1101001001, 996: 1111100100 | 835: 1101000011, 993: 1111100001, 944: 1110110000 | 605: 1001011101, 878: 1101101110 | 667: 1010011011, 781: 1100001101, 966: 1111000110 | 689: 1010110001, 792: 1100011000 | 625: 1001110001, 888: 1101111000 | 631: 1001110111, 891: 1101111011, 1021: 1111111101, 958: 1110111110 | 687: 1010101111, 791: 1100010111, 971: 1111001011, 933: 1110100101, 914: 1110010010 | 676: 1010100100 | 704: 1011000000 | 581: 1001000101, 866: 1101100010 | 664: 1010011000 | 617: 1001101001, 884: 1101110100 | 715: 1011001011, 805: 1100100101, 978: 1111010010 | 692: 1010110100 | 709: 1011000101, 802: 1100100010 | 648: 1010001000 | 616: 1001101000 | 614: 1001100110 | 729: 1011011001, 812: 1100101100 | 850: 1101010010 | 660: 1010010100 | 708: 1011000100 | 609: 1001100001, 880: 1101110000 | 603: 1001011011, 877: 1101101101, 1014: 1111110110 | 701: 1010111101, 798: 1100011110 | 647: 1010000111, 771: 1100000011, 961: 1111000001, 928: 1110100000 | 590: 1001001110 | 723: 1011010011, 809: 1100101001, 980: 1111010100 | 718: 1011001110 | 755: 1011110011, 825: 1100111001, 988: 1111011100 | 861: 1101011101, 1006: 1111101110 | 699: 1010111011, 797: 1100011101, 974: 1111001110 | 691: 1010110011, 793: 1100011001, 972: 1111001100 | 860: 1101011100 | 853: 1101010101, 1002: 1111101010 | 698: 1010111010 | 750: 1011101110 | 763: 1011111011, 829: 1100111101, 990: 1111011110 | 695: 1010110111, 795: 1100011011, 973: 1111001101, 934: 1110100110 | 681: 1010101001, 788: 1100010100 | 712: 1011001000 | 620: 1001101100 | 838: 1101000110 | 657: 1010010001, 776: 1100001000 | 624: 1001110000 | 595: 1001010011, 873: 1101101001, 1012: 1111110100 | 719: 1011001111, 807: 1100100111, 979: 1111010011, 937: 1110101001,

916: 1110010100 | 716: 1011001100 | 844: 1101001100 | 852: 1101010100 | 714: 1011001010 | 754: 1011110010 | 764: 1011111100 | 847: 1101001111, 999: 1111100111, 947: 1110110011, 921: 1110011001, 908: 1110001100 | 856: 1101011000 | 629: 1001110101, 890: 1101111010 | 670: 1010011110 | 743: 1011100111, 819: 1100110011, 985: 1111011001, 940: 1110101100 | 858: 1101011010 | 662: 1010010110 | 741: 1011100101, 818: 1100110010 | 652: 1010001100 | 840: 1101001000 | 628: 1001110100 | 707: 1011000011, 801: 1100100001, 976: 1111010000 | 606: 1001011110 | 727: 1011010111, 811: 1100101011, 981: 1111010101, 938: 1110101010 | 682: 1010101010 | 746: 1011101010 | 762: 1011111010 | 766: 1011111110 | 767: 1011111111, 831: 1100111111, 991: 1111011111, 943: 1110101111, 919: 1110010111, 907: 1110001011, 901: 1110000101, 898: 1110000010 | 672: 1010100000 | 586: 1001001010 | 722: 1011010010 | 756: 1011110100 | 711: 1011000111, 803: 1100100011, 977: 1111010001, 936: 1110101000 | 634: 1001111010 | 734: 1011011110 | 759: 1011110111, 827: 1100111011, 989: 1111011101, 942: 1110101110 | 683: 1010101011, 789: 1100010101, 970: 1111001010 | 690: 1010110010 | 748: 1011101100 | 846: 1101001110 | 659: 1010010011, 777: 1100001001, 964: 1111000100 | 611: 1001100011, 881: 1101110001, 1016: 1111111000 | 639: 1001111111, 895: 1101111111, 1023: 1111111111, 959: 1110111111, 927: 1110011111, 911: 1110001111, 903: 1110000111, 899: 1110000011, 897: 1110000001, 896: 1110000000 | 579: 1001000011, 865: 1101100001, 1008: 1111110000 | 607: 1001011111, 879: 1101101111, 1015: 1111110111, 955: 1110111011, 925: 1110011101, 910: 1110001110 | 675: 1010100011, 785: 1100010001, 968: 1111001000 | 636: 1001111100 | 839: 1101000111, 995: 1111100011, 945: 1110110001, 920: 1110011000 | 633: 1001111001, 892: 1101111100 | 855: 1101010111, 1003: 1111101011, 949: 1110110101, 922: 1110011010 | 678: 1010100110 | 745: 1011101001, 820: 1100110100 | 713: 1011001001, 804: 1100100100 |

Appendix C. Enablers

For $i \in \{1,2,\ldots,10\}$, '$F_i(0)$' has been replaced by 'F_i' for simplicity.

****

1-Run Beginnings (2-Parallel State Transition):

****

$F_1F_2'F_3'F_4F_5'F_6'F_7'F_8F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6'F_7'F_8'F_9'F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6'F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6F_7'F_8'F_9'F_{10}'$ †
$F_1F_2'F_3'F_4F_5'F_6F_7'F_8F_9'F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6'F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6'F_7F_8F_9F_{10}'$ †
$F_1F_2'F_3F_4'F_5F_6F_7'F_8F_9F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6F_7'F_8F_9F_{10}'$ †
$F_1F_2'F_3F_4'F_5'F_6'F_7F_8F_9F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6'F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6'F_7'F_8F_9F_{10}'$ †
$F_1F_2'F_3F_4F_5'F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6'F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6'F_7'F_8'F_9F_{10}'$ †
$F_1F_2'F_3F_4F_5F_6F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6'F_7F_8F_9'F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6'F_7F_8F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6'F_7'F_8'F_9F_{10}'$ †
$F_1F_2'F_3F_4F_5'F_6F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6F_7F_8'F_9F_{10}'$ †
$F_1F_2'F_3F_4F_5F_6'F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6'F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6'F_7F_8F_9'F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6'F_7'F_8'F_9'F_{10}'$ †
$F_1F_2'F_3'F_4F_5'F_6'F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6F_7F_8F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6F_7F_8F_9F_{10}'$ †
$F_1F_2'F_3F_4'F_5F_6'F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6'F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6'F_7F_8F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6'F_7F_8F_9F_{10}'$ †
$F_1F_2F_3'F_4F_5'F_6F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6'F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6'F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6F_7F_8F_9'F_{10}'$ †
$F_1F_2'F_3F_4'F_5F_6'F_7F_8F_9'F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6'F_7F_8F_9'F_{10}'$ †
$F_1F_2F_3'F_4F_5'F_6F_7F_8F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6'F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6'F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6F_7F_8'F_9F_{10}'$ †
$F_1F_2'F_3F_4F_5F_6F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6F_7F_8F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6F_7F_8F_9F_{10}'$ †
$F_1F_2'F_3F_4'F_5'F_6'F_7'F_8F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6F_7F_8F_9'F_{10}'$ †
$F_1F_2'F_3F_4F_5'F_6'F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6'F_7F_8'F_9'F_{10}'$ †
$F_1F_2'F_3F_4F_5'F_6'F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6F_7'F_8F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6'F_7F_8'F_9'F_{10}'$ †
$F_1F_2'F_3'F_4F_5F_6'F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6'F_7'F_8F_9F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6F_7F_8'F_9'F_{10}'$ †
$F_1F_2'F_3F_4F_5'F_6'F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8F_9F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6F_7F_8'F_9'F_{10}'$ †
$F_1F_2'F_3F_4'F_5F_6F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6'F_7F_8F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6'F_7F_8F_9'F_{10}'$ †
$F_1F_2F_3'F_4F_5'F_6'F_7F_8F_9F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8'F_9F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6F_7'F_8F_9'F_{10}'$ †
$F_1F_2F_3'F_4F_5'F_6F_7'F_8F_9'F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6F_7F_8F_9'F_{10}'$ †
$F_1F_2F_3'F_4F_5'F_6F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6F_7F_8F_9F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5'F_6F_7F_8F_9F_{10}'$ †
$F_1F_2'F_3F_4'F_5'F_6'F_7F_8F_9'F_{10}'$ † $F_1F_2F_3'F_4F_5'F_6'F_7F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6F_7'F_8F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6F_7F_8F_9F_{10}'$ †
$F_1F_2'F_3F_4'F_5F_6'F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6F_7'F_8F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6F_7F_8F_9F_{10}'$ †
$F_1F_2'F_3F_4'F_5F_6'F_7'F_8'F_9'F_{10}'$ † $F_1F_2'F_3'F_4F_5'F_6'F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6F_7'F_8F_9'F_{10}'$ †
$F_1F_2'F_3'F_4F_5F_6F_7F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5'F_6F_7F_8F_9F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6F_7'F_8'F_9F_{10}'$ † $F_1F_2'F_3F_4F_5F_6'F_7F_8F_9'F_{10}'$ †
$F_1F_2F_3'F_4F_5'F_6'F_7F_8F_9F_{10}'$ † $F_1F_2'F_3'F_4F_5F_6F_7F_8F_9'F_{10}'$ † $F_1F_2'F_3F_4'F_5F_6'F_7'F_8F_9F_{10}'$

****

2-Run Beginnings (3-Parallel State Transition):

****

$F_1F_2'F_3'F_4F_5'F_6'F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4F_5F_6'F_7F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6'F_7F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6'F_7'F_8'F_9'F_{10}$ †
$F_1F_2'F_3'F_4F_5'F_6'F_7'F_8'F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6'F_7F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4F_5F_6F_7'F_8F_9'F_{10}$ †
$F_1F_2'F_3'F_4F_5F_6F_7F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5F_6'F_7F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6'F_7'F_8F_9'F_{10}$ † $F_1F_2'F_3F_4F_5'F_6F_7'F_8F_9'F_{10}$ †
$F_1F_2'F_3F_4F_5'F_6'F_7'F_8'F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5F_6'F_7F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6F_7F_8F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6F_7'F_8'F_9'F_{10}$ †
$F_1F_2'F_3F_4F_5'F_6F_7F_8F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6F_7'F_8F_9'F_{10}$ † $F_1F_2'F_3F_4F_5F_6F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8F_9'F_{10}$ †
$F_1F_2'F_3F_4'F_5'F_6'F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4F_5'F_6F_7'F_8'F_9'F_{10}$ † $F_1F_2F_3'F_4F_5'F_6'F_7F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5F_6F_7F_8'F_9'F_{10}$ †
$F_1F_2F_3'F_4F_5'F_6F_7'F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5F_6F_7'F_8F_9'F_{10}$ † $F_1F_2'F_3F_4F_5F_6'F_7'F_8'F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6F_7F_8'F_9'F_{10}$ †
$F_1F_2'F_3F_4'F_5'F_6F_7'F_8F_9'F_{10}$ † $F_1F_2F_3'F_4F_5'F_6'F_7'F_8'F_9'F_{10}$ † $F_1F_2F_3'F_4F_5'F_6F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4F_5F_6F_7F_8F_9'F_{10}$ †
$F_1F_2F_3'F_4F_5'F_6'F_7'F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5F_6'F_7'F_8F_9'F_{10}$ † $F_1F_2'F_3F_4F_5F_6F_7'F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5F_6'F_7'F_8'F_9'F_{10}$ †
$F_1F_2'F_3'F_4F_5F_6'F_7'F_8F_9'F_{10}$ † $F_1F_2F_3'F_4F_5'F_6'F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5F_6F_7'F_8'F_9'F_{10}$ †
$F_1F_2'F_3'F_4F_5F_6F_7'F_8'F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6'F_7'F_8F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5F_6'F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4F_5'F_6'F_7'F_8F_9'F_{10}$ †
$F_1F_2'F_3F_4F_5'F_6F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5F_6'F_7'F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5F_6F_7F_8F_9'F_{10}$ † $F_1F_2F_3'F_4F_5'F_6F_7F_8F_9'F_{10}$ †
$F_1F_2F_3'F_4F_5'F_6F_7'F_8F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5F_6'F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6F_7'F_8'F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5F_6F_7'F_8F_9'F_{10}$ †
$F_1F_2'F_3F_4F_5F_6'F_7'F_8F_9'F_{10}$ † $F_1F_2'F_3'F_4F_5F_6F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4F_5F_6'F_7F_8'F_9'F_{10}$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8'F_9'F_{10}$

****

3-Run Beginnings (4-Parallel State Transition):

****

$F_1F_2'F_3F_4F_5F_6'F_7F_8'F_9F_{10}$ † $F_1F_2'F_3F_4F_5F_6'F_7'F_8'F_9F_{10}$ † $F_1F_2F_3'F_4F_5'F_6F_7F_8'F_9F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6'F_7'F_8'F_9F_{10}$ †
$F_1F_2'F_3'F_4F_5F_6F_7F_8'F_9F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6'F_7F_8'F_9F_{10}$ † $F_1F_2F_3'F_4F_5'F_6F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3F_4F_5'F_6F_7F_8'F_9F_{10}$ †
$F_1F_2'F_3'F_4F_5'F_6'F_7F_8'F_9F_{10}$ † $F_1F_2F_3'F_4F_5'F_6'F_7F_8'F_9F_{10}$ † $F_1F_2'F_3'F_4F_5F_6'F_7F_8'F_9F_{10}$ † $F_1F_2'F_3'F_4F_5F_6F_7'F_8'F_9F_{10}$ †
$F_1F_2F_3'F_4F_5'F_6'F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6F_7F_8'F_9F_{10}$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8'F_9F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6F_7F_8'F_9F_{10}$ †
$F_1F_2'F_3F_4F_5'F_6F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3F_4F_5F_6F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3F_4'F_5F_6F_7F_8'F_9F_{10}$ † $F_1F_2'F_3F_4'F_5F_6F_7'F_8'F_9F_{10}$ †
$F_1F_2'F_3F_4F_5F_6F_7F_8'F_9F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3F_4F_5'F_6'F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3F_4'F_5F_6'F_7'F_8'F_9F_{10}$ †
$F_1F_2'F_3F_4'F_5'F_6F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3'F_4F_5F_6'F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6'F_7'F_8'F_9F_{10}$ † $F_1F_2'F_3F_4'F_5F_6'F_7'F_8'F_9F_{10}$

****

4-Run Beginnings (5-Parallel State Transition):

****

$F_1F_2'F_3'F_4F_5'F_6F_7'F_8F_9F_{10}$ † $F_1F_2'F_3'F_4F_5F_6'F_7'F_8F_9F_{10}$ † $F_1F_2'F_3F_4'F_5F_6'F_7'F_8F_9F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6'F_7'F_8F_9F_{10}$ †
$F_1F_2'F_3F_4'F_5'F_6F_7'F_8F_9F_{10}$ † $F_1F_2'F_3'F_4F_5F_6F_7'F_8F_9F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6'F_7'F_8F_9F_{10}$ † $F_1F_2'F_3F_4'F_5F_6F_7'F_8F_9F_{10}$ †
$F_1F_2'F_3F_4F_5F_6'F_7'F_8F_9F_{10}$ † $F_1F_2'F_3F_4F_5'F_6F_7'F_8F_9F_{10}$ † $F_1F_2'F_3F_4F_5'F_6'F_7'F_8F_9F_{10}$ † $F_1F_2'F_3F_4F_5F_6F_7'F_8F_9F_{10}$ †
$F_1F_2F_3'F_4F_5'F_6'F_7'F_8F_9F_{10}$ † $F_1F_2F_3'F_4F_5'F_6F_7'F_8F_9F_{10}$

****

5-Run Beginnings (6-Parallel State Transition):

****

$F_1F_2'F_3'F_4F_5F_6'F_7F_8F_9F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6'F_7F_8F_9F_{10}$ † $F_1F_2'F_3F_4'F_5'F_6'F_7F_8F_9F_{10}$ † $F_1F_2'F_3F_4F_5F_6'F_7F_8F_9F_{10}$ †
$F_1F_2'F_3F_4'F_5F_6'F_7F_8F_9F_{10}$ † $F_1F_2'F_3F_4F_5'F_6'F_7F_8F_9F_{10}$ † $F_1F_2F_3'F_4F_5'F_6'F_7F_8F_9F_{10}$

****

6-Run Beginnings (7-Parallel State Transition):

****

$F_1F_2'F_3F_4'F_5'F_6F_7F_8F_9F_{10}$ † $F_1F_2'F_3F_4F_5'F_6F_7F_8F_9F_{10}$ † $F_1F_2F_3'F_4F_5'F_6F_7F_8F_9F_{10}$ † $F_1F_2'F_3'F_4F_5'F_6F_7F_8F_9F_{10}$

****

7-Run Beginnings (8-Parallel State Transition):

****

$F_1F_2'F_3F_4'F_5F_6F_7F_8F_9F_{10}$

****

8-Run Beginnings (9-Parallel State Transition):

****

$F_1F_2'F_3F_4F_5F_6F_7F_8F_9F_{10}$

****

9-Run Beginnings (10-Parallel State Transition):

*** -

****

10-Run Beginnings (11-Parallel State Transition):

****

$F_1F_2'F_3'F_4F_5F_6F_7F_8F_9F_{10}$

References

1. Cheng, Y.; Xu, C.; Hai, Z.; Li, Y. Deepmnemonic: Password mnemonic generation via deep attentive encoder-decoder model. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 77–90. [CrossRef]
2. Zi, Y.; Gao, H.; Cheng, Z.; Liu, Y. An end-to-end attack on text captchas. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 753–766. [CrossRef]
3. Gao, S.; Mohamed, M.; Saxena, N.; Zhang, C. Emerging-image motion captchas: Vulnerabilities of existing designs, and countermeasures. *IEEE Trans. Dependable Secur. Comput.* **2019**, *16*, 1040–1053. [CrossRef]
4. Lee, J. Indifferentiability of the sum of random permutations toward optimal security. *IEEE Trans. Inf. Theory* **2017**, *63*, 4050–4054. [CrossRef]
5. Zhou, J.; Liu, X.; Au, O.C.; Tang, Y.Y. Designing an efficient image encryption-then-compression system via prediction error clustering and random permutation. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 39–50. [CrossRef]
6. Selvi, D.; Velammal, T.G.; Arockiadoss, T. Modified method of generating randomized latin squares. *IOSR J. Comput. Eng. (IOSR-JCE)* **2014**, *16*, 76–80. [CrossRef]
7. Kwan, M.; Sudakov, B. Intercalates and discrepancy in random latin squares. *arXiv* **2017**, arXiv:1607.04981.
8. Bóna, M.; Knopfmacher, A. On the probability that certain compositions have the same number of parts. *Ann. Comb.* **2010**, *14*, 291–306. [CrossRef]
9. Banderier, C.; Hitczenko, P. Enumeration and asymptotics of restricted compositions having the same number of parts. *Discret. Appl. Math.* **2012**, *160*, 2542–2554. [CrossRef]
10. Sedighi, M.; Fallah, M.S.; Zolfaghari, B. S-restricted compositions revisited. *J. Discret. Math. Theor. Comput. Sci.* **2017**, *19*, 1–19.
11. Goresky, M.; Klapper, A.M. Fibonacci and galois representations of feedback-with-carry shift registers. *IEEE Trans. Inf. Theory* **2002**, *48*, 2826–2836. [CrossRef]
12. Dubrova, E. A transformation from the fibonacci to the galois nlfsrs. *IEEE Trans. Inf. Theory* **2006**, *55*, 5263–5271. [CrossRef]
13. Ayinala, M.; Parhi, K.K. High-speed parallel architectures for linear feedback shift registers. *IEEE Trans. Signal Process.* **2012**, *59*, 4459–4469. [CrossRef]
14. Zolfaghari, B.; Sedighi, M.; Fallah, M.S. Designing programmable parallel lfsr using parallel prefix trees. *J. Eng. Res.* **2019**, *7*, 105–122.
15. Kumar, S.N.; Kumar, H.S.; Panduranga, H.T. Hardware software co-simulation of dual image encryption using latin square image. In Proceedings of the Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, India, 4–6 July 2013.
16. Mou, H.; Li, X.; Li, G.; Lu, D.; Zhang, R. A self-adaptive and dynamic image encryption based on latin square and high-dimensional chaotic system. In Proceedings of the IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, China, 27–29 June 2018.
17. Pal, S.K.; Bhardwaj, D.; Kumar, R.; Bhatia, V. A new cryptographic hash function based on latin squares and non-linear transformations. In Proceedings of the IEEE International Advance Computing Conference, Patiala, India, 6–7 March 2009.
18. Shen, J.; Zhou, T.; Liu, X.; Chang, Y.C. A novel latin-square-based secret sharing for m2m communications. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3659–3668. [CrossRef]
19. Lehmer, D.H. Teaching combinatorial tricks to a computer. In *Proceedings of Symposium on Applied Mathematics*; American Mathematical Society: New York City, NY, USA, 1960.
20. Dubrova, E.; Mansouri, S.S. A bdd-based approach to constructing lfsrs for parallel crc encoding. In Proceedings of the International Symposium on Multiple-Valued Logic, Victoria, BC, Canada, 14–16 May 2012.
21. Cheng, C.; Parhi, K.K. High-speed parallel crc implementation based on unfolding, pipelining, and retiming. *IEEE Trans. Circuits Syst. II Express Briefs* **2006**, *53*, 1017–1021. [CrossRef]
22. Nandi, S.; Krishnaswamy, S.; Zolfaghari, B.; Mitra, P. Key-dependant feedback configuration matrix of σ-lfsr and resistance to some known plaintext attacks. *IEEE Access* **2022**, *10*, 44840–44854. [CrossRef]
23. Kuehnel, R.; Theiler, J.; Wang, Y. Parallel random number generators for sequences uniformly distributed over any range of integers. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2006**, *53*, 1496–1505. [CrossRef]
24. Pae, S.I. A generalization of peres's algorithm for generating random bits from loaded dice. *IEEE Trans. Inf. Theory* **2015**, *61*, 751–757.
25. Zhang, G.H.; Poon, C.C.; Zhang, Y.T. Analysis of using interpulse intervals to generate 128-bit biometric random binary sequences for securing wireless body sensor networks. *IEEE Trans. Inf. Technol. Biomed.* **2012**, *16*, 176–182. [CrossRef]
26. Cauwenberghs, G. Delta-sigma cellular automata for analog vlsi random vector generation. *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* **1999**, *46*, 240–250. [CrossRef]
27. Morgan, D. Analysis of digital random numbers generated from serial samples of correlated gaussian noise (corresp.). *IEEE Trans. Inf. Theory* **1981**, *27*, 235–239. [CrossRef]
28. Chen, X.; Wang, L.; Li, B.; Wang, Y.; Li, X.; Liu, Y.; Yang, H. Modeling random telegraph noise as a randomness source and its application in true random number generation. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2016**, *35*, 1435–1448. [CrossRef]

29. Gao, S.; Zhang, L.; Xu, Y.; Chen, L.; Bao, X. High-speed random bit generation via brillouin random fiber laser with non-uniform fibers. *IEEE Photonics Technol. Lett.* **2017**, *29*, 1352–1355. [CrossRef]
30. Lv, Y.; Zink, B.R.; Wang, J.P. Bipolar random spike and bipolar random number generation by two magnetic tunnel junctions. *IEEE Trans. Electron Devices* **2022**, *69*, 1582–1587. [CrossRef]
31. Ge, Z.; Xiao, Y.; Hao, T.; Li, W.; Li, M. Tb/s fast random bit generation based on a broadband random optoelectronic oscillator. *IEEE Photonics Technol. Lett.* **2021**, *33*, 1223–1226. [CrossRef]
32. Seetharam, D.; Rhee, S. An efficient pseudo random number generator for low-power sensor networks [wireless networks]. In Proceedings of the Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, 16–18 November 2004.
33. Ramesh, A.; Jain, A. Hybrid image encryption using pseudo random number generators, and transposition and substitution techniques. In Proceedings of the International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15), Bangalore, India, 21–22 December 2015.
34. Anghelescu, P.; Sofron, E.; Ionita, S.; Ionescu, L. Fpga implementations of cellular automata for pseudo-random number generation. In Proceedings of the International Semiconductor Conference, Sinaia, Romania, 27–29 September 2006.
35. Tutueva, A.V.; Butusov, D.N.; Pesterev, D.O.; Belkin, D.A.; Ryzhov, N.G. Novel normalization technique for chaotic pseudo-random number generators based on semi-implicit ode solvers. In Proceedings of the International Conference "Quality Management, Transport and Information Security, Information Technologies" (IT&QM&IS), St. Petersburg, Russia, 24–30 September 2017.
36. Desai, V.V.; Deshmukh, V.B.; Rao, D.H. Pseudo random number generator using elman neural network. In Proceedings of the IEEE Recent Advances in Intelligent Computational Systems, Trivandrum, India, 22–24 September 2011.
37. Jeong, Y.S.; Oh, K.; Cho, C.K.; Choi, H.J. Pseudo random number generation using lstms and irrational numbers. In Proceedings of the IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China, 15–17 January 2018.
38. Anikin, I.V.; Alnajjar, K. Pseudo-random number generator based on fuzzy logic. In Proceedings of the International Siberian Conference on Control and Communications (SIBCON), Moscow, Russia, 12–14 May 2016.
39. Yang, H.T.; Huang, J.R.; Chang, T.Y. A chaos-based fully digital 120 mhz pseudo random number generator. In Proceedings of the 2004 IEEE Asia-Pacific Conference on Circuits and Systems, Tainan, Taiwan, 6–9 December 2004.
40. Unde, A.S.; Deepthi, P.P. Design and analysis of compressive sensing-based lightweight encryption scheme for multimedia iot. *IEEE Trans. Circuits Syst. II Express Briefs* **2019**, *67*, 167–171. [CrossRef]
41. Wheeldon, A.; Shafik, R.; Rahman, T.; Lei, J.; Yakovlev, A.; Granmo, O.C. Learning automata based energy-efficient ai hardware design for iot applications. *Philos. Trans. R. Soc. A* **2020**, *378*, 20190593. [CrossRef]
42. Hussain, S.; Chaudhary, A.K.; Verma, S. Enhancing security in iot devices by using pseudo random number generator based on two different lfsr and a comparator. In Proceedings of the IEEE Delhi Section Conference (DELCON), New Delhi, India, 11–13 February 2022.
43. Han, M.; Kim, Y. Unpredictable 16 bits lfsr-based true random number generator. In Proceedings of the International SoC Design Conference (ISOCC), Seoul, Korea, 5–8 November 2017.
44. Zode, P.; Zode, P.; Deshmukh, R. Fpga based novel true random number generator using lfsr with dynamic seed. In Proceedings of the IEEE 16th India Council International Conference (INDICON), Rajkot, India, 13–15 December 2019.
45. Gu,X.; Zhang, M. Uniform random number generator using leap ahead lfsr architecture. In Proceedings of the International Conference on Computer and Communications Security, Hong Kong, China, 5–6 December 2009.
46. Tan, Z.; Guo, W.; Gong, G.; Lu, H. A new pseudo-random number generator based on the leap-ahead lfsr architecture. In Proceedings of the IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA), Beijing, China, 21–23 November 2018.
47. Tuncer, T.; Avaroğlu, E. Random number generation with lfsr based stream cipher algorithms. In Proceedings of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 22–26 May 2017.
48. Hu, G.; Sha, J.; Wang, Z. High-speed parallel lfsr architectures based on improved state-space transformations. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 1159–1163. [CrossRef]
49. Moghadam, I.Z.; Rostami, A.S.; Tanhatalab, M.R. Designing a random number generator with novel parallel lfsr substructure for key stream ciphers. In Proceedings of the International Conference On Computer Design and Applications, Qinhuangdao, China, 25–27 June 2010.
50. Hlawiczka, A. Compression of three-state data serial streams by means of a parallel lfsr signature analyzer. *IEEE Trans. Comput.* **1986**, *C-35*, 732–741.
51. Kongtim, P.; Reungpeerakul, T. Parallel lfsr reseeding with selection register for mixed-mode bist. In Proceedings of the 19th IEEE Asian Test Symposium, Shanghai, China, 1–4 December 2010.
52. Sokal, N.O. Optimum choice of noise frequency band and sampling rate for generating random binary digits from clipped white noise. *IEEE Trans. Comput.* **1971**, *C-21*, 614–615.
53. Alves, J.C.; Martins, A.C. A strategy to generate random binary errors in a data stream. *IEEE Trans. Instrum. Meas.* **1986**, *IM-35*, 42–45.
54. Waicukauski, J.A.; Lindbloom, E.; Eichelberger, E.B.; Forlenza, O.P. A method for generating weighted random test patterns. *IBM J. Res. Dev.* **1989**, *33*, 149–161. [CrossRef]

55. Xu, M.; Liu, J. Double-layered random coding for secret key generation in gaussian wiretap channels. *IEEE Commun. Lett.* **2020**, *24*, 264–267. [CrossRef]
56. Vaidya, J.; Shafiq, B.; Fan, W.; Mehmood, D.; Lorenzi, D. A random decision tree framework for privacy-preserving data mining. *IEEE Trans. Dependable Secur. Comput.* **2014**, *11*, 399–411. [CrossRef]
57. Tang, F.; Chen, D.G.; Wang, B.; Bermak, A.; Amira, A.; Mohamad, S. Cmos on-chip stable true-random id generation using antenna effect. *IEEE Electron Device Lett.* **2014**, *35*, 54–56. [CrossRef]
58. Li, J.; Zhou, Y.; Chen, H. Age of information for multicast transmission with fixed and random deadlines in iot systems. *IEEE Internet Things J.* **2020**, *7*, 8178–8191. [CrossRef]
59. Chung, S.C.; Yu, C.Y.; Lee, S.S.; Chang, H.C.; Lee, C.Y. An improved dpa countermeasure based on uniform distribution random power generator for iot applications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 2522–2531. [CrossRef]
60. Angelopoulos, G.; Paidimarri, A.; Médard, M.; Chandrakasan, A.P. A random linear network coding accelerator in a 2.4ghz transmitter for iot applications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 2582–2590. [CrossRef]
61. Silverman, J.; Vickers, V.; Sampson, J. Statistical estimates of then-bit gray codes by restricted random generation of permutations of 1 to 2^n. *IEEE Trans. Inf. Theory* **1983**, *29*, 894–901. [CrossRef]
62. Gujar, U.; Kavanagh, R. Generation of random signals with specified probability density functions and power density spectra. *IEEE Trans. Autom. Control* **1968**, *13*, 716–719. [CrossRef]
63. Mitchell, R.L.; Stone, C.R. Table-lookup methods for generating arbitrary random numbers. *IEEE Trans. Comput.* **1977**, *C-26*, 1006–1008.
64. Wang, Y.; Li, P.; Zhang, J. Fast random bit generation in optical domain with ultrawide bandwidth chaotic laser. *IEEE Photonics Technol. Lett.* **2010**, *22*, 1680–1682. [CrossRef]
65. Fang, X.; Wetzel, B.; Merolla, J.M.; Dudley, J.M.; Larger, L.; Guyeux, C.; Bahi, J.M. Noise and chaos contributions in fast random bit sequence generated from broadband optoelectronic entropy sources. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 888–901. [CrossRef]
66. Almardeny, Y.; Benavoli, A.; Boujnah, N.; Naredo, E. A reinforcement learning system for generating instantaneous quality random sequences. *IEEE Trans. Artif. Intell.* **2022**. [CrossRef]
67. Boucetta, C.; Nour, B.; Moungla, H.; Lahlou, L. An iot scheduling and interference mitigation scheme in tsch using latin rectangles. In Proceedings of the IEEE Global Communications Conference, Waikoloa, HI, USA, 9–13 December 2019.
68. Fontana, R. Random latin squares and sudoku designs generation. *Electron. J. Stat.* **2014**, *8*, 883–893. [CrossRef]
69. DeSalvo, S. Random sampling of latin squares via binary contingency tablesand probabilistic divide-and-conquer. *arXiv* **2017**, arXiv:1703.08627.
70. Muramatsu, J.; Miyake, S. Channel code using constrained-random-number generator revisited. *IEEE Trans. Inf. Theory* **2019**, *65*, 500–510. [CrossRef]
71. Muramatsu, J. Channel coding and lossy source coding using a generator of constrained random numbers. *IEEE Trans. Inf. Theory* **2014**, *60*, 2667–2686. [CrossRef]
72. Muramatsu, J. Variable-length lossy source code using a constrained-random-number generator. *IEEE Trans. Inf. Theory* **2015**, *61*, 3574–3592. [CrossRef]
73. Moon, S.; Lee, H.S.; Lee, J.W. Sara: Sparse code multiple access-applied random access for iot devices. *IEEE Internet Things J.* **2018**, *5*, 3160–3174. [CrossRef]
74. Khan, M.H.A. Design of reversible synchronous sequential circuits using pseudo reed-muller expressions. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2014**, *22*, 2278–2286. [CrossRef]
75. Zenner, E. On Cryptographic Properties of LFSR-Based Pseudorandom Generators. Ph.D. Thesis, Universitat Mannheim, Mannheim, Germany, 2004.