Cryptography and Communications (2019) 11:921–949
https://doi.org/10.1007/s12095-018-0334-1

R-2 composition tests: a family of statistical randomness tests for a collection of binary sequences

Muhiddin Uğuz[1] · Ali Doğanaksoy[1] · Fatih Sulak[2] · Onur Koçak[3]

Received: 8 June 2017 / Accepted: 2 October 2018 / Published online: 15 October 2018

Abstract
In this article a family of statistical randomness tests for binary strings are introduced, based on Golomb's pseudorandomness postulate R-2 on the number of runs. The basic idea is to construct recursive formulae with computationally tenable probability distribution functions. The technique is illustrated on testing strings of 2^7, 2^8, 2^{10} and 2^{12} bits. Furthermore, the expected value of the number of runs with a specific length is obtained. Finally the tests are applied to several collections of strings arising from different pseudorandom number generators.

Keywords Cryptography · Statistical randomness testing · Golomb's randomness postulates · Compositions · Run test

Mathematics Subject Classification (2010) 05A15 · 05A17 · 94A55 · 94A60

1 Introduction

In cryptography, outputs of block ciphers, stream ciphers, hash functions, and key generators should be indistinguishable from the outputs of random generators. To decide whether a generator is distinguishable from a random generator or not, a common approach is to

✉ Muhiddin Uğuz
muhid@metu.edu.tr

Ali Doğanaksoy
aldoks@metu.edu.tr

Fatih Sulak
fatih.sulak@atilim.edu.tr

Onur Koçak
onur.kocak@tubitak.gov.tr

[1] Department of Mathematics, Middle East Technical University, Ankara, Turkey

[2] Department of Mathematics, Atılım University, Ankara, Turkey

[3] TÜBİTAK BİLGEM, Kocaeli, Turkey

compare certain statistics of the output of the generator with the theoretical expected values computed for the sample space consisting of all strings of the same length.

As in the case of stream ciphers, when the outputs of a given generator are long strings (10^6 or more terms), decision on randomness is based on a sample output (key stream). Under such circumstances where one deals with long strings, asymptotic computations and approximate values of theoretical constants can be employed safely.

On the other hand, outputs of block ciphers, hash functions, and key generators are in general short binary strings (128 to 4096 bits) and statistical randomness tests of such generators are based on a large collection of their output strings. By concatenating all the strings in that collection, one obtains a long string which can be evaluated as if it were the output of a stream cipher. However, a more natural approach is to consider each output string individually and then to evaluate the entire collection. In this case, since we deal with relatively short strings, it is not safe to employ approximations or asymptotic formulations. Main difficulty in this approach is not only to obtain the probability distribution functions, but also to express these functions in such a form that their evaluation is feasible.

In the present article, we design a family of statistical randomness tests for a collection of short binary strings by defining a number of random variables and deriving recursive formulae with computationally feasible probability distribution functions.

Whether a given string is "like" a random string is conventionally decided by means of Golomb's pseudorandomness postulates R-1, R-2 and R-3 [1], which we now state for the sake of completeness albeit the first and the last will be beyond the scope of this work:

R-1 The difference between the number of 1's and the number of 0's should be at most one.

R-2 The number of runs of length one should be at least half of the total number of runs, while the number of runs of length two should be at least one-fourth of the total number of runs, and the number of runs of length three should be at least one-eighth of the total number of runs and so on. Moreover the number of runs of ones and the number of runs of zeros should not differ more than one for each fixed run length.

R-3 The autocorrelation function should be two-valued.

The first postulate considers the frequencies of ones and zeros. The second one is about the number of runs in a string, where a run is defined as an uninterrupted maximal strings of identical bits. Lastly, the third postulate reveals information about the amount of similarities between the string and its shifted versions. A two-valued autocorrelation function is required to assure that the amount of mentioned similarity remains constant for all shifted versions of the string.

Almost all test suites include a test function based on R-1, under the name frequency test or weight test, (see for instance [2, 3, 6, 7]). Similarly, most of the test suites include a test considering certain properties of runs, however none of these tests correspond R-2 completely, [3, 6, 7]. Up to our knowledge, a test function for R-3 is not included in any test suite. The main purpose of this study is to design a family of randomness tests that depend on the R-2 requirements.

In the context of runs, randomness test suites generally consider the following quantities:

– the number of all runs,
– length of the longest run,
– the number of runs of a specified length.

The distribution function related with the number of all runs involves considerably straightforward computations and it is included in almost all test suites. For instance, NIST test suite has "Runs Test", TESTU01 has "Run and gap tests", see [3, 7] respectively. Similarly, length of the longest run test appears in both NIST test suite and TESTU01.

However, obtaining a feasible formulation of the distribution function related with the number of runs with a specified length is a difficult task (especially when specified length is enlarged). Mood obtained the distribution of runs of any length using a combinatorial approach [4]. Later, following similar methods, Doğanaksoy et al. designed randomness tests based on number of runs of lengths one, two and three, [8]. These explicit formulations heavily depend on binomial coefficients and hence as the length grows, calculations become more complex, and the time required for these calculations grows drastically. As a result, these tests are limited to strings of length 256 bits (for runs of length three) and 512 bits (for runs of length at most two).

By making use of Markov chains and transition matrices, Fu and Koutras proposed the statistics of runs of any length and also the length of the longest run [5]. Their work considers the general case where the probability of occurrences of 1's and 0's are not necessarily equal. On the other hand, this method naturally depends on computations performed on a transition matrix dimension of which gets very large depending on the lengths of strings and runs (for instance, for the strings of length 4096 and runs of length 2, dimension of the required transition matrix is 5459). This method is particularly useful for relatively short sequences.

In this work, we use an alternative method which depend on generating functions to obtain simple recursive formulation of related distribution functions. More specifically, we propose a family of test functions depending on the following probabilities:

- the probability that a random string of length n has r runs,
- the probability that a random string of length n has exactly k runs of length a,
- the probability that a random string of length n has exactly k runs of length at least a,
- the probability that a random string of length n has no runs of length longer than L,

where n, r, a, L are positive integers and k is a non-negative integer. Rather than complex explicit expressions, we focus specifically on getting simple recursive relations for the probability functions mentioned above. Different from the previous approaches which use binomial coefficients or transition matrices, all the relations we obtain are linear, thus have low complexities. Employing these relations, we obtained exact probabilities for sufficiently (from cryptographical point of view) long strings such as 2^{14} bits.

The organization of this paper is as follows. In Section 2, compositions and certain restrictions on them are defined, and a correspondence between their numbers and the number of runs is explained. The notations about the number of compositions of certain types and their generating functions are also introduced in this section. In Section 3, generating functions for the number of all compositions of n with exactly r parts and for the number of all compositions of n and finally for the number of all compositions of n whose parts are from a predefined set Γ are computed. Corresponding generating functions are given in examples for some specific choices of Γ . The generating functions for the last three restrictions in Table 1 are given in Theorems 1, 2, and 3. The results obtained in this sections are listed in the Table 2. In Section 4, an alternative method to compute recursions defining the number of certain compositions are introduced. Theorems 4, 5, and 6 are the ones analogous to the theorems given in Section 3. In Theorem 7, the recursion to calculate the number of compositions of n without parts exceeding L is derived. In Section 5, basic probabilities on which our tests depend are introduced in terms of compositions, and generating functions.

Table 1 Notations for the restricted compositions of the positive integer n

Restriction on compositions of n	Number	Generating function
All	$c(n)$	$C(z) = \sum_{n=0}^{\infty} c(n) z^n$
Exactly $r \geq 1$ parts	$c(n, r)$	$C(z, x) = \sum_{n,r=0}^{\infty} c(n, r) z^n x^r$
All parts in a set Γ	$c_\Gamma(n)$	$C_\Gamma(z) = \sum_{n=0}^{\infty} c_\Gamma(n) z^n$
No part exceeding L	$c_L(n)$	$C_L(z) = \sum_{n=0}^{\infty} c_L(n) z^n$
Exactly r parts, with k of them equal to a	$c_a(n, k, r)$	$C_a(z, y, x) = \sum_{n,k,r=0}^{\infty} c_a(n, k, r) z^n y^r x^k$
All with exactly k parts equal to a,	$c_a(n, k)$	$C_a(z, y) = \sum_{n,k=0}^{\infty} c_a(n, k) z^n y^k$
Exactly k parts greater than or equal to a	$c_{\bar{a}}(n, k)$	$C_{\bar{a}}(z, y) = \sum_{n,k=0}^{\infty} c_{\bar{a}}(n) z^n y^k$

Moreover these probability values are derived and then recursions for these probability values are obtained. Theorems 8, 9, 10, and 11 are the main results, and they form the basis of the new randomness tests. In Section 6, expected values and variances for the random variables are computed and they may be used to define new randomness tests. In Section 7.1 we defined the outline of the proposed randomness test. We give a pseudo code for the Algorithm 2 of R-2 Composition Test and an example to evaluate the outputs of AES algorithm with respect to the randomness. We have illustrated the p-values obtained for different types of runs in the Table 5. Section 8 contains several applications of the proposed randomness

Table 2 Generating functions for the number of restricted compositions of the positive integer n

$c(n)$	$C(z) = \sum_{n=0}^{\infty} c(n) z^n = \frac{1-z}{1-2z}$
$c(n, r)$	$C(z, x) = \sum_{n,k=0}^{\infty} c(n, r) z^n x^k = \frac{1-z}{1-z-zx}$
$c_\Gamma(n)$	$C_\Gamma(z) = \sum_{n=0}^{\infty} c_\Gamma(n) z^n = \frac{1}{1-z^\Gamma}$
$c_L(n)$	$C_L(z) = \sum_{n=0}^{\infty} c_L(n) z^n = \frac{1-z}{1-2z+z^{L+1}}$
$c_a(n, k, r)$	$C_a(z, y, x) = \sum_{n,k,r=0}^{\infty} c_a(n, k, r) z^n y^r x^k = \frac{1-z}{1-z(x+1)+z^a x(1-z)(1-y)}$
$c_a(n, k)$	$C_a(z, y) = \sum_{n,k=0}^{\infty} c_a(n, k) z^n y^k = \frac{1-z}{1-2z+z^a(1-z)(1-y)}$
$c_{\bar{a}}(n, k)$	$C_{\bar{a}}(z, y) = \sum_{n,k=0}^{\infty} c_L(n) z^n y^k = \frac{1-z}{1-2z+z^a(1-y)}$

test to different type of binary string collections. Finally, in Section 9, we give a conclusion and a future work plan.

2 Compositions and restricted compositions

Definition 1 A composition of a positive integer n is an ordered array of one or more positive integers whose sum is n.

For example, the number 4 has 8 different compositions:

$$(4)(3,1)(1,3)(2,2)(2,1,1)(1,2,1)(1,1,2)(1,1,1,1)$$

Each term in a composition is called a *part*. As shown below, the number of compositions of a positive integer n is $c(n) = 2^{n-1}$ and $c(0) = 1$ by convention [9].

Let the numbers $l_1, \ldots, l_r$ denote the lengths of the runs of a given binary string of length n. Obviously, the sum of these lengths is equal to the length of the string, thus the ordered array $(l_1, \ldots, l_r)$ is a composition of n. The lengths of runs of a binary string of length n corresponds to a unique composition of n. Conversely, to each composition of n, there corresponds two strings (one starting with a one; the other starting with a zero) so that lengths of runs are equal to the corresponding parts of the composition. Due to this $2-1$ correspondence, any restriction or any problem on the compositions can be visualized as a restriction or a problem on the number of runs of a binary string.

There are numerous studies on compositions with certain restrictions on the parts such as compositions with no occurrence of k (by Chinn et al. [10]), compositions whose parts are from a predefined set (Heubach et al. [11]), compositions with distinct parts (Richmond et al. [12]), compositions with no parts exceeding L (M. E. Malandro [13] and so on [14, 15]. Almost all studies result in recursive relations; Jaklic et al. [16] obtained some results on closed form formulas of restricted compositions.

We now list our notation for the numbers of compositions under certain restrictions:

$c(n)$: number of all compositions of n,
$c(n, r)$: number of compositions of n with r parts,
$c_\Gamma(n)$: number of compositions of n with all parts in a given set $\Gamma \subset Z$,
$c_L(n)$: number of compositions of n with no part larger than L,
$c_a(n, r, k)$: number of compositions of n with r parts and k of these parts equal to a,
$c_a(n, k)$: number of compositions of n with k parts equal to a,
$c_{\bar{a}}(n, k)$: number of compositions of n with k parts larger than or equal to a.

Table 1 below summarizes these notations and introduces the corresponding generating functions.

First two cases in the table are basic cases and can be found in many text books on combinatorics or discrete mathematics (for instance, Lint [9]). $c_\Gamma(n)$ is given in [11] and $c_a(n, k)$ is studied in [10] for only $k = 0$.

The main contribution of this paper is the introduction of recursions for $c_a(n, k, r)$, $c_a(n, k)$ and $c_{\bar{a}}(n, k)$ for $k > 0$ and giving the related generating functions.

The convention $c(0) = 1$ leads to the conventions $c(0, 0) = c_A(0) = c_a(0, 0, 0) = c_a(0, 0) = c_{\bar{a}}(0, 0) = 1$. In addition to these conventions, in order to introduce some ease in computations, we extend the domain of all counting functions (mentioned in the list) to

all nonnegative integers by setting the value as 0 whenever variables are not in the natural domain of the function. For instance, we assume that $c_a(n,k)=0$ when $n<ka$.

3 Generating functions

There are several ways to approach the problem of finding the number of compositions [16]. We firstly concentrate on generating functions. Recall that, for positive integers n, r, to find the number of nonnegative integer solutions of the equation $x_1+x_2+\cdots+x_r=n$, after associating the formal sum $1+z+z^2+\cdots$ with each summand (x_i), it is sufficient to compute the coefficient of z^n in $(1+z+z^2+\cdots)^r$. Any restriction on summands has a natural reflection to the corresponding factors. For instance, the number of solutions of $x_1+x_2+\cdots+x_8=30$ subject to the condition $3\le x_i\le 10$ is given by the coefficient of z^{30} in $(z^3+z^4+\cdots+z^{10})^8$. Similarly, coefficient of z^{30} in $(1+z^2+z^4+\cdots)^4(z+z^3+z^5+\cdots)^4$ is the number of solutions of the same equation, if x_1, x_2, x_3 and x_4 are to be even and the others to be odd.

If we return to the compositions again, we see that $c(n,r)$, the number of compositions of the positive integer n with $r\ge 1$ parts is the number of positive integer solutions of the equation

$$x_1+x_2+\cdots+x_r=n \tag{1}$$

and this number is given by the coefficient of z^n in $(z+z^2+\cdots)^r=\left(\frac{z}{1-z}\right)^r$ or equivalently, $c(n,r)$ is the coefficient of $z^n x^r$ in $1+\sum_{r=1}^{\infty}\left(\frac{z}{1-z}\right)^r x^r=\frac{1}{1-\left(\frac{z}{1-z}\right)x}$ which means

$$C(z,x)=\frac{1-z}{1-z-zx}.$$

On the other hand, as $\frac{z^r}{(1-z)^r}=z^r\sum_{i=0}^{\infty}\binom{-r}{i}(-z)^i=z^r\sum_{i=0}^{\infty}\binom{r+i-1}{r-1}z^i$ by taking $i=n-r$, the coefficient of z^n in $(z+z^2+\cdots)^r$ can be obtained as

$$c(n,r)=\binom{n-1}{r-1}.$$

It follows that the recursion satisfied by $c(n,r)$ is the Pascal's identity

$$c(n,r)=c(n-1,r)+c(n-1,r-1).$$

By definition, for $n\ge 1$ we have $c(n)=\sum_{r=1}^{n}c(n,r)$. In other words, including the case for $n=0$, $c(n)$ is the sum of coefficients of z^n in $1, \frac{z}{1-z}, \frac{z^2}{(1-z)^2}, \ldots$ or equivalently, $c(n)$ is the coefficient of z^n in $1+\sum_{r=1}^{\infty}(\frac{z}{1-z})^r=\frac{1}{1-\frac{z}{1-z}}=\frac{1-z}{1-2z}$. It follows that

$$C(z)=\frac{1-z}{1-2z}.$$

Since

$$C(z) = \frac{1-z}{1-2z} = \frac{1}{1-2z} - \frac{z}{1-2z} = \sum_{n=0}^{\infty} 2^n z^n - \sum_{n=0}^{\infty} 2^n z^{n+1} = 1 + \sum_{n=1}^{\infty} 2^{n-1} z^n$$
$$= 1 + \sum_{n=1}^{\infty} c(n) z^n = \sum_{n=0}^{\infty} c(n) z^n$$

for $n \geq 1$ we get

$$c(n) = 2^{n-1}.$$

In (1), if we require $x_i \in \Gamma, i = 1, \ldots, r$ for some subset Γ of nonnegative integers then $\sum_{\gamma \in \Gamma} z^{\gamma}$ is the factor corresponding to each summand. For simplicity, we will denote the sum $\sum_{\gamma \in \Gamma} z^{\gamma}$ by z^{Γ}. Consequently, the number of compositions with r parts and all parts in Γ, is the coefficient of z^n in$(z^{\Gamma})^r$ (for more details see [11]). Then it follows that, the number of compositions with all parts in Γ, namely $c_{\Gamma}(n)$, is the coefficient of z^n in $\sum_{r=1}^{\infty} (z^{\Gamma})^r$ so that

$$C_{\Gamma}(z) = 1 + \sum_{r=1}^{\infty} (z^{\Gamma})^r = \frac{1}{1 - z^{\Gamma}}.$$

For the following examples recall that the Fibonacci string f_n is defined by the recursion $f_n = f_{n-1} + f_{n-2}$ subject to the initial conditions $f_0 = 0$, $f_1 = 1$. Generating function of this string is $f(z) = \dfrac{z}{1 - z - z^2}$.

Example 1 $\Gamma = \{1, 2\}$: For the number of compositions of n with each part equal to either 1 or 2 we have $z^{\Gamma} = z + z^2$ and generating function is

$$\frac{1}{1 - z - z^2} - 1.$$

Hence the number of such compositions is the Fibonacci number f_{n+1}.

Example 2 $\Gamma = \mathbb{Z} - \{1\}$: the number of compositions of n with no part equal to 1, we have $z^{\Gamma} = z^2 + z^3 + \cdots = \dfrac{z^2}{1 - z}$ and generating function is

$$\frac{1}{1 - \left(\frac{z^2}{1-z}\right)} - 1 = \frac{1 - z}{1 - z - z^2} - 1.$$

It follows that the number of compositions of n without 1 is the Fibonacci number f_{n-1}.

Example 3 $\Gamma = \mathbb{Z} - 2\mathbb{Z}$: For the number of compositions of n with all parts odd we have $z^{\Gamma} = z + z^3 + z^5 + \cdots = \dfrac{z}{1-z^2}$ and generating function is

$$\frac{z}{1-z-z^2}.$$

Hence the number of such compositions is the Fibonacci number f_n.

Example 4 $\Gamma = \mathbb{Z} - \{a\}$: For the number of compositions of n without any occurrence of a, $z^{\Gamma} = \displaystyle\sum_{i=1}^{\infty} z^i - z^a = \frac{z}{1-z} - z^a$ so that the associated generating function is

$$\frac{1-z}{1-2z+z^a-z^{a+1}}.$$

Example 5 $\Gamma = \{0, 1, \ldots, L\}$: For the number of compositions of n with no part exceeding L, $z^{\Gamma} = \displaystyle\sum_{i=1}^{L} z^i = \frac{z(1-z^L)}{1-z}$ so that the associated generating function is

$$C_L(z) = \frac{1-z}{1-2z+z^{L+1}}.$$

We give the generating functions for the last three restrictions in Table 1, namely generating functions of $c_a(n,k,r)$, of $c_a(n,k)$ and of $c_{\bar{a}}(n,k)$ in the following three theorems.

Theorem 1 *If a is a fixed positive integer then*

$$C_a(z,y,x) = \frac{1-z}{1-z(x+1)+z^a x(1-z)(1-y)}.$$

Proof $c_a(n,k,r)$ is the number of positive solutions of the equation $x_1 + x_2 + \cdots + x_r = n$ such that $x_{i_1} = \cdots = x_{i_k} = a$ for some $\{i_1, \ldots, i_k\} \subset \{1, \ldots, r\}$ and $x_i \neq a$ for $i \notin \{i_1, \ldots, i_k\}$. The subset $\{i_1, \ldots, i_k\}$ can be chosen in $\binom{r}{k}$ distinct ways and once this set is determined, the number of solutions of the equation is given by the coefficient of z^n in

$$(z^a)^k \left((z+z^2+\cdots) - z^a\right)^{r-k} = (z^a)^k \left(\frac{z}{1-z} - z^a\right)^{r-k}.$$

It follows that $c_a(n,k,r)$ is the coefficient of $z^n y^k x^r$ in

$$C_a(z,y,x) = 1 + \sum_{k=0}^{\infty}\sum_{r=1}^{\infty} \binom{r}{k} z^{ka} U^{r-k} y^k x^r \tag{2}$$

where $U = \dfrac{z}{1-z} - z^a$. The double sum on the right hand side can be separated for the cases $k = 0$ and $k = 1, 2, \ldots$ to obtain

$$\begin{aligned} C_a(z, y, x) &= 1 + \sum_{r=1}^{\infty} U^r x^r + \sum_{k=1}^{\infty} z^{ka} y^k x^k \sum_{r=1}^{\infty} \binom{r}{k} U^{r-k} x^{r-k} \\ &= \frac{1}{1-Ux} + \sum_{k=1}^{\infty} z^{ka} y^k x^k \sum_{r=1}^{\infty} \binom{r}{k} (Ux)^{r-k} \\ &= \frac{1}{1-Ux} + \frac{1}{1-Ux} \sum_{k=1}^{\infty} \left(\frac{z^a yx}{1-Ux} \right)^k \\ &= \frac{1}{1-Ux-z^a yx} \end{aligned}$$

and finally by substituting $U = \dfrac{z}{1-z} + z^a$, desired expression can be obtained. □

Theorem 2 *If a is a fixed positive integer then*

$$C_a(z, y) = \frac{1-z}{1-2z+z^a(1-z)(1-y)}.$$

Proof From the proof of Theorem 1, we see that $c_a(n, k)$ is the coefficient of z^n in $\sum_{r=1}^{\infty} \binom{r}{k} (z^a)^k U^{r-k}$. It follows that $c_a(n, k)$ is the coefficient of $z^n y^k$ in

$$1 + \sum_{k=0}^{\infty} \sum_{r=1}^{\infty} \binom{r}{k} z^{ka} U^{r-k} y^k.$$

From (2), we observe that this expression is in fact $C_a(z, y, 1)$, thus $C_a(z, y) = C_a(z, y, 1)$. □

Note that, the coefficient of y^0 in $C_a(z, y)$ is $1 + \sum_{r=1}^{\infty} U^r = \dfrac{1}{1-U} = \dfrac{1-z}{1-z+z^a-z^{a+1}}$. This expression is the generating function for the number of strings which do not contain a, as we have previously obtained in Example 4.

Theorem 3 *If a is a fixed positive integer then*

$$C_{\bar{a}}(z, y) = \frac{1-z}{1-2z+z^a(1-y)}.$$

Proof The number of positive solutions of the equation $x_1 + x_2 + \cdots + x_r = n$ such that $x_{i_1} = \cdots = x_{i_k} \geq a$ for some $\{i_1, \ldots, i_k\} \subset \{1, \ldots, r\}$ and $x_i < a$ for $i \notin \{i_1, \ldots, i_k\}$ is given by the coefficient of z^n in

$$\binom{r}{k} (z^a + z^{a+1} + \cdots)^k (z + z^2 + \cdots + z^{a-1})^{r-k} = \binom{r}{k} \left(\frac{z^a}{1-z} \right)^k z^{r-k} \left(\frac{1-z^{a-1}}{1-z} \right)^{r-k}.$$

Then $c_{\bar{a}}(n,k)$ is the coefficient of z^n in $\sum_{r=1}^{\infty}\binom{r}{k}\left(\frac{z^a}{1-z}\right)^k z^{r-k}U^{r-k}y^k$ where $U = \frac{1-z^{a-1}}{1-z}$. But this means that

$$C_{\bar{a}}(z,y) = 1+\sum_{k=0}^{\infty}\sum_{r=1}^{\infty}\binom{r}{k}\left(\frac{z^a}{1-z}\right)^k U^{r-k}z^{r-k}y^k$$

The double sum on the right hand side can be separated for the cases $k=0$ and $k=1,2,\ldots$ to obtain

$$\begin{aligned}
C_{\bar{a}}(z,y) &= 1+\sum_{r=1}^{\infty}U^r z^r + \sum_{k=1}^{\infty}\left(\frac{z^a}{1-z}\right)^k y^k \sum_{r=1}^{\infty}\binom{r}{k}U^{r-k}z^{r-k}\\
&= \frac{1}{1-Uz}+\sum_{k=1}^{\infty}\left(\frac{z^a y}{1-z}\right)^k\sum_{r=1}^{\infty}\binom{r}{k}(Uz)^{r-k}\\
&= \frac{1}{1-Uz}+\frac{1}{1-Uz}\sum_{k=1}^{\infty}\left(\frac{z^a y}{(1-z)(1-Uz)}\right)^k\\
&= \frac{1-z}{(1-z)(1-Uz)-z^a y}
\end{aligned}$$

and substituting $(1-z)U = 1-z^{a-1}$ we obtain $C_{\bar{a}}(z,y) = \dfrac{1-z}{1-2z+z^a(1-y)}$. □

We summarize the results in the following table on the number of restricted compositions and generating functions.

All the generating functions we have obtained are rational functions, hence the strings which they correspond satisfy constant coefficient homogeneous recursive relations. Moreover, these relations can be obtained immediately from the denominators of the generating functions. However, we can obtain these recursions by employing some elementary and straightforward counting arguments.

4 Recursions

When we delete the first part, say $a < n$, of a composition of n we obtain a composition of $n-a$. Thus, by deleting the first parts of all compositions of n we obtain the complete list of all compositions of integers $1,2,\ldots,n-1$.

For instance consider the compositions 4 :

$$(4)\ (3,1)\ (1,3)\ (2,2)\ (2,1,1)\ (1,2,1)\ (1,1,2)\ (1,1,1,1).$$

If the first part of the composition (2, 1, 1) of 4 is deleted, we obtain the composition (1, 1) of 2. By deleting the first parts of all compositions of 4 we obtain the following list

$$(\not{4})\qquad (\not{3},1)\qquad (\not{2},2)\ (\not{2},1,1)\qquad (\not{1},3)\ (\not{1},2,1)\ (\not{1},1,2)\ (\not{1},1,1,1)$$

$$(\square)\qquad \underbrace{(1)}_{compositions\ of\ 1}\qquad \underbrace{(2)(1,1)}_{compositions\ of\ 2}\qquad \underbrace{(3)(2,1)(1,2)(1,1,1)}_{compositions\ of\ 3}$$

Besides the compositions of $1,2,\ldots,n-1$, this list contains one more object, namely the empty composition which is obtained when the first part of composition (n) is deleted. We

may visualize the empty composition to be the unique composition of 0 as an interpretation of $c(0) = 1$. In a similar manner, by appending the positive integer a, as the first part, to any composition of $n - a$, we obtain a unique composition of n. Therefore the set of all compositions of n is equivalent to the set of all compositions of all integers less than n, including 0, hence for any integer $n \geq 1$

$$c(n) = c(n-1) + \cdots + c(1) + c(0) = \sum_{i=1}^{n} c(n-i). \tag{3}$$

Theorem 4 *The string* $\{c(n)\}_{n=0}^{\infty}$ *is determined with the initial conditions* $c(0) = c(1) = 1$*and the recurrence relation* $c(n) = 2c(n-1)$ *for all integers* $n \geq 2$.

Proof By convention we have $c(0) = 1$ and thus $c(1) = 1$ is obvious. We can write the recursion (3) as

$$c(n-1) = \sum_{i=1}^{n-1} c(n-1-i) = \sum_{i=2}^{n} c(n-i) = \left(\sum_{i=1}^{n} c(n-i) \right) - c(n-1)$$

for $n \geq 2$. Comparing this expression to (3), one obtains $c(n) = 2c(n-1)$ for $n \geq 2$. □

Notice that, since the generating function of $c(n)$ is $C(z) = \sum c(n)z^n = \dfrac{1-z}{1-2z}$, from it's denominator one can read the recursion for $c(n)$ as $c(n) = 2c(n-1)$ and this agrees with the recursion obtained above.

A recursion for $c_a(n,k)$ can be obtained in a way analogous to the one used in obtaining (3). First consider the case $k = 0$. By deleting the first part of each composition of n which has no part equal to a, we obtain such compositions of all integers less than n, except $n-a$. Thus, the recursion for $c_a(n,0)$ differs from the recursion for $c(a)$ only by the summand $c(n-a,0)$, that is

$$c_a(n,0) = \sum_{i=1}^{n} c_a(n-i,0) - c_a(n-a,0). \tag{4}$$

Delete the first part of a composition of n which has $k \geq 1$ parts equal to a. If the deleted part is equal to a, then the remaining parts constitute a composition of $n-a$ with $k-1$ parts equal to a. If the deleted term is equal to i $(i \neq a)$, then the remaining parts form a composition of $n-i$ with k parts equal to a. Then we can write

$$c_a(n,k) = \sum_{i=1}^{n} c_a(n-i,k) - c_a(n-a,k) + c_a(n-a,k-1). \tag{5}$$

Theorem 5 *Let* a *be a positive integer. The string* $\{c_a(n,k)\}_{n=0}^{\infty}$ *is determined with the initial conditions;*

– *for* $k = 0$

$$c_a(n,0) = \begin{cases} 1 & if \ \ n = 0 \\ 2^{n-1} & if \ \ 1 \leq n < a \\ 2^{n-1} - 1 & if \ \ n = a \end{cases}$$

– *for* $k \geq 1$

$$c_a(n,k) = \begin{cases} 0 \ if \ \ n \leq ka-1 \\ 1 \ if \ \ n = ka \end{cases}$$

and the recurrence relations are

– *for* $n \geq a+1$:

$$c_a(n,0) = 2c_a(n-1,0) - c_a(n-a,0) + c_a(n-1-a,0) \tag{6}$$

– *for* $k \geq 1$ *and* $n \geq ka+1$:

$$c_a(n,k) = 2c_a(n-1,k) - c_a(n-a,k) + c_a(n-a-1,k) + c_a(n-a,k-1) - c_a(n-a-1,k-1). \tag{7}$$

Proof When $k = 0$, we have $c_a(0,0) = 1$ by convention. If $n < a$, then no composition of n contains a as a part, so $c_a(n,0) = 2^{n-1}$. For $n = a$, only one composition of a contains a, thus $c_a(a,0) = 2^{n-1} - 1$.

When $k \geq 1$, then $c(0,k) = 0$ by convention. If $n < ka$, then no composition of n can contain k parts equal to a, so $c_a(n,k) = 0$ for $n < ka$. Only one composition of $n = ka$ consists of k parts, each equal to a, thus $c_{ka}(a,k) = 1$.

For $n \geq a+1$, recursion (4) can be written as

$$\begin{aligned} c_a(n-1,0) &= \sum_{i=1}^{n-1} c_a(n-1-i,0) - c_a(n-1-a,0) \\ &= \sum_{i=2}^{n} c_a(n-i,0) - c_a(n-1-a,0) \\ &= \sum_{i=1}^{n} c_a(n-i,0) - c_a(n-1,0) - c_a(n-1-a,0). \end{aligned}$$

Comparing this expression to (4), we obtain (6).

In a similar manner, for $n \geq ka+1$ we write the recursion (5) as

$$\begin{aligned} c_a(n-1,k) &= \sum_{i=1}^{n-1} c_a(n-1-i,k) - c_a(n-1-a,k) + c_a(n-1-a,k-1) \\ &= \sum_{i=2}^{n} c_a(n-i,k) - c_a(n-1-a,k) + c_a(n-1-a,k-1) \\ &= \sum_{i=1}^{n} c_a(n-i,k) - c_a(n-1,k) + c_a(n-1-a,k-1) + c_a(n-1-a,k-1). \end{aligned}$$

Comparing this expression to (5), we obtain (7). □

Notice that, since the generating function for $c_a(n,k)$ is $C_a(z,y) = \sum c_a(n,k) z^n y^k = \dfrac{1-z}{1-2z+z^a(1-z)(1-y)} = \dfrac{1-z}{1-2z+z^a-z^{a+1}-z^a y+z^{a+1}y}$, from it's denominator one can read the recursion as $c_a(n,k) = 2c_a(n-1,k) - c_a(n-a,k) + c_a(n-a-1,k) + c_a(n-a,k-1) - c_a(n-a-1,k-1)$ and this agrees with recursion we obtained above.

Finally, for $c_{\bar{a}}(n,k)$ consider the case $k=0$. By deleting the first part of each composition of n which has all parts less than a, we obtain such compositions of all integers $n-1, n-2, \cdots, n-a+1$. Thus, the recursion for $c_{\bar{a}}(n,0)$ differs from the recursion for $c(a)$ only by the summand $c_{\bar{a}}(n-a,0)$, that is

$$c_{\bar{a}}(n,0) = \sum_{i=1}^{a-1} c_{\bar{a}}(n-i,0). \tag{8}$$

Delete the first part of a composition of n which has $k>1$ parts equal to a. If the deleted part is $i \geq a$, then the remaining parts constitute a composition of $n-i$ with $k-1$ parts equal to or larger than a. If the deleted term is $i<a$, then the remaining parts form a composition of $n-i$ with k parts equal to or larger than a. Then we can write

$$c_{\bar{a}}(n,k) = \sum_{i=1}^{a-1} c_{\bar{a}}(n-i,k) + \sum_{i=a}^{n} c_{\bar{a}}(n-i,k-1). \tag{9}$$

Theorem 6 *Let a be a positive integer. The string $\{c_{\bar{a}}(n,k)\}_{n=0}^{\infty}$ is determined with the initial conditions:*

– *for $k=0$*

$$c_{\bar{a}}(n,0) = \begin{cases} 1 & if\ \ n=0 \\ 2^{n-1} & if\ \ a>1 \text{ and } 1 \leq n \leq a-1 \\ 2^{a-1}-1 & if\ \ n=a \end{cases}$$

– *for $k \geq 1$*

$$c_{\bar{a}}(n,k) = \begin{cases} 0 & if\ \ n \leq ka-1 \\ 1 & if\ \ n=ka \end{cases}$$

and the recurrence relations

$$c_{\bar{a}}(n,0) = 2c_{\bar{a}}(n-1,0) - c_{\bar{a}}(n-a,0) \tag{10}$$

and

$$c_{\bar{a}}(n,k) = 2c_{\bar{a}}(n-1,k) - c_{\bar{a}}(n-a,k) + c_{\bar{a}}(n-a,k-1) \tag{11}$$

for $k \geq 1$ and $n \geq ka+1$.

Proof Initial conditions are same with the ones given in Theorem 5. For $n \geq a+1$, recursion (8) can be written as $c_{\bar{a}}(n-1,0) = \sum_{i=1}^{a-1} c_{\bar{a}}(n-1-i,0)$.

Comparing this expression to (8), we obtain (10).

In a similar manner, for $n \geq ka+1$ we write the recursion (9) as

$$\begin{aligned} c_{\bar{a}}(n-1,k) &= \sum_{i=1}^{a-1} c_{\bar{a}}(n-1-i,k) + \sum_{i=a}^{n-1} c_{\bar{a}}(n-1-i,k-1) \\ &= \sum_{i=2}^{a} c_{\bar{a}}(n-i,k) + \sum_{i=a+1}^{n} c_{\bar{a}}(n-i,k-1) \\ &= \sum_{i=1}^{a-1} c_{\bar{a}}(n-i,k) + c_{\bar{a}}(n-a,k) + \sum_{i=a}^{n} c_{\bar{a}}(n-i,k-1) - c_{\bar{a}}(n-a,k-1) \end{aligned}$$

Comparing this expression to (9), we obtain (11). □

Notice that, since the generating function for $c_{\bar{a}}(n,k)$ is $C_{\bar{a}}(z,y) = \sum c_{\bar{a}}(n,k)z^n y^k = \dfrac{1-z}{1-2z+z^a(1-y)}$, from it's denominator one can read the recursion as $c_{\bar{a}}(n,k) = 2c_{\bar{a}}(n-1,k) - c_{\bar{a}}(n-a,k) + c_{\bar{a}}(n-a,k-1)$ and this agrees with recursion we obtained above.

Theorem 7 *Let L be a fixed positive integer. The string $\{c_L(n)\}_{n=0}^{\infty}$ is determined with the initial conditions*

$$c_L(n) = \begin{cases} 1 & if \ \ n = 0 \\ 2^{n-1} & if \ \ n \leq L \end{cases}$$

and the recurrence relation

$$c_L(n) = 2c_L(n-1) - c_L(n-L-1).$$

for $n \geq L+1$.

Proof Call a composition as an L-composition if no part exceeds L. A positive integer n has two type of L-compositions:

Type 1: L-Compositions with the first part equal to 1. Each such composition is equivalent to a unique L-composition of $n-1$. Thus the number of L-compositions of n with the first part equal to 1 is $c_L(n-1)$.

Type 2: L-Compositions with the first part larger than 1. If we decrease the first part of such a composition by 1, we obtain a unique L-composition of $n-1$. By this way all L-compositions of $n-1$, except the ones with the first part equal to L, are obtained. But, the set of exceptional compositions is equivalent to the set of L-compositions of $n-1-L$. Hence the number of Type 2 compositions is $c_L(n-1) - c_L(n-L-1)$.

□

Again notice that, since the generating function for $c_L(n)$ is $C_L(z) = \sum c_L(n)z^n = \dfrac{1-z}{1-2z+z^{L+1}}$, from it's denominator one can read the recursion as $c_L(n) = 2c_L(n-1) - c_L(n-L-1)$ and this agrees with recursion we obtained above.

5 Restricted compositions statistics

5.1 Generating functions for probability sequences

In this section we compute the basic probabilities on which our tests depend. Let Ω_n be the set of binary strings σ of length n. Define the value of nonnegative integer valued random variables X, $X_{\max}$, X_a, $X_{\bar{a}}$ on $\sigma \in \Omega_n$ as

$X(\sigma)$ = number of runs of σ,
$X_{\max}(\sigma)$ = length of a longest run of σ,
$X_a(\sigma)$ = number of runs of length a of σ,
$X_{\bar{a}}(\sigma)$ = number of runs of length at least a of σ.

We denote the probability mass functions of these random variables as

$$\begin{aligned}
&p(n,r) = \text{probability}(X = r),\\
&p_L(n) = \text{probability}(X_{\max} \le L),\\
&p_a(n,r) = \text{probability}(X_a = r),\\
&p_{\bar{a}}(n,r) = \text{probability}(X_{\bar{a}} = r).
\end{aligned}$$

After giving the definition of a composition, we have showed that to each binary string of length n there corresponds a unique composition of n and conversely, to each composition of n there corresponds exactly two binary strings of length 2. Thus, for example, the number of binary strings of length n which have r runs is twice the number of compositions of n with r parts. Since the number of all binary strings of length n is 2^n, we immediately get

$$p(n,r) = \frac{1}{2^n}\,(2c(n,r)) = 2^{1-n}c(n,r)$$

and analogously

$$\begin{aligned}
p_L(n) &= 2^{1-n}c_L(n),\\
p_a(n,k) &= 2^{1-n}c_a(n,k)
\end{aligned}$$

and

$$p_{\bar{a}}(n,k) = 2^{1-n}c_{\bar{a}}(n,k).$$

Let n be a fixed nonnegative integer, then the probability generating function $\sum_{r=0}^{\infty} p(n,r)x^r$ of X is the coefficient of z^n in:

$$\begin{aligned}
P(z,y) &= \sum_{n,r=0}^{\infty} p(n,r)z^n y^r = 2^{1-n}\sum_{n,r=0}^{\infty} c(n,r)z^n y^r\\
&= 2\sum_{n,r=0}^{\infty} c(n,r)\left(\frac{z}{2}\right)^n y^r = 2C(z/2,y)\\
&= \frac{2(2-z)}{2-z-zy}.
\end{aligned}$$

Similarly, probability generating functions of X_a and $X_{\bar{a}}$ are coefficients of z^n in

$$\begin{aligned}
P_a(z,y) &= 2C_a(z/2,y) = \frac{2-z}{1-z+2^{-a-1}z^a(2-z)(1-y)},\\
P_{\bar{a}}(z,y) &= 2C_{\bar{a}}(z/2,y) = \frac{2-z}{1-z+2^{-a}z^a(1-y)}
\end{aligned}$$

respectively.

5.2 Probability calculations

Theorem 8 *The string $\{p(n,r)\}_{n=0}^{\infty}$ is determined with the initial conditions*

$$p(n,r) = \begin{cases} \frac{1}{2^{n-1}} & if\ \ r = n\\ 0 & if\ \ r = 0\ and\ n > 0\\ 0 & if\ \ r > 0\ and\ r > n \end{cases}$$

and the recursion

$$p(n,r) = \frac{1}{2}(p(n-1,r) + p(n-1,r-1))$$

for $n > r > 0$.

Proof We already know that $c(n,r) = \binom{n-1}{r-1}$. Pascal's identity $\binom{n-1}{r-1} = \binom{n-2}{r-1} + \binom{n-2}{r-2}$ leads the desired expression. □

Theorem 9 *Let a be a positive integer. The string $\{p_a(n,r)\}_{n=0}^{\infty}$ is determined with the initial conditions:*

for $r = 0$

$$p_a(n,0) = \begin{cases} 1 & if\ \ n \in \{0,\dots,a-1\} \\ 1-2^{1-a} & if\ \ n \in \{a, a+1\} \end{cases}$$

for $r \geq 1$

$$p_a(n,r) = \begin{cases} 0 & if\ \ n \leq ra-1 \\ 2^{1-n} & if\ \ n \in \{ra, ra+1\} \end{cases}$$

and the recurrence relations

- *for $n \geq a+2$*

$$p_a(n,0) = p_a(n-1,0) - 2^{-a}p_a(n-a,0) + 2^{-a-1}p_a(n-1-a,0)$$

- *for $r \geq 1$ and $n \geq ra+2$*

$$\begin{aligned} p_a(n,r) = p_a(n-1,r) &- 2^{-a}p_a(n-a,r) + 2^{-a-1}p_a(n-a-1,r) \\ &+ 2^{-a}p_a(n-a,r-1) - 2^{-a-1}p_a(n-a-1,r-1). \end{aligned}$$

Proof By convention, $p_a(0,0) = 1$. Initial conditions can be checked by direct computing. For the other cases, just substitute $p_a(n,r) = 2^{1-n}c_a(n,r)$ in Theorem 5. □

Theorem 10 *Let a be a positive integer. The string $\{p_{\bar{a}}(n,r)\}_{n=0}^{\infty}$ is determined with the initial conditions:*

for $r = 0$

$$p_{\bar{a}}(n,0) = \begin{cases} 1 & if\ \ n \leq a-1 \\ 1-2^{1-n} & if\ \ n = a \end{cases}$$

for $r > 1$

$$p_{\bar{a}}(n,r) = \begin{cases} 0 & if\ \ n \leq ra-1 \\ 2^{1-n} & if\ \ n = ra \end{cases}$$

and the recurrence relations

- *for $n \geq a+1$*

$$p_{\bar{a}}(n,0) = p_{\bar{a}}(n-1,0) - 2^{-a}p_{\bar{a}}(n-a,0)$$

- *for $r \geq 1$ and $n \geq ra+1$*

$$p_{\bar{a}}(n,r) = p_{\bar{a}}(n-1,r) - 2^{-a}p_{\bar{a}}(n-a,r) + 2^{-a}p_{\bar{a}}(n-a,r-1).$$

Proof Just substitute $p_{\bar{a}}(n,r) = 2^{1-n}c_{\bar{a}}(n,r)$ in Theorem 6. □

Theorem 11 *Let a be a positive integer. The string $\{p_L(n)\}_{n=0}^{\infty}$ is determined with the initial conditions $p_L(0) = p_L(1) = \cdots = p_L(L) = 1$ and the recurrence relation*

$$p_L(n) = p_L(n-1) - 2^{-L-1}p_L(n-L-1).$$

Proof Just substitute $p_L(n) = 2^{1-n}c_L(n)$ in Theorem 7. □

As for the complexities, to find the complexity of the computations needed in determination of the value of $p_a(n_0, r)$ for a specific value of a, using the recursive formula given in theorem 9, first we assume that all of the values $p_a(n, r)$ are computed for $n \leq n_0$ and $1 \leq r \leq \lfloor \frac{n_0}{a} \rfloor$. Then to compute $p_a(n_0, r)$ we need 4 addition operations for each r. Thus, in total one has to perform $\sum_{n=0}^{n_0} 4\frac{n}{a} = \frac{2n(n+1)}{a}$ additions. This means that the complexity to determine $p_a(n_0, r)$ is $\mathcal{O}(n^2)$. Similar arguments can be given for the formulas stated in the other theorems and the same complexity bound is also valid for them.

6 Expected values and variance

For our purposes we need neither expected values nor variances. On the other hand, these quantities are quite important tools for various similar statistical computations. For the sake of completeness and possible future references here we compute expected values and variances of the random variables X, X_a and $X_{\bar{a}}$.

It is well known that when n is fixed the expected value and the variance of X appear as the coefficient of z^n, respectively in $\frac{\partial P}{\partial x}$ and $\frac{\partial^2 P}{\partial x^2} + \frac{\partial P}{\partial x} - \left(\frac{\partial P}{\partial x}\right)^2$ where all partial derivatives are evaluated at $x = 1$. Similar expressions can be written for X_a and $X_{\bar{a}}$. After some straightforward, but boring computations we obtain the following:

– Coefficient of z^n in

$$\frac{\partial}{\partial x} P(z, x)|_{x=1} = \frac{1}{2}\frac{(2-z)z}{(1-z)^2} = \left(z - \frac{z^2}{2}\right)\sum_{i=0}^{\infty}(i+1)z^i$$

is $\frac{n+1}{2}$.

– Coefficient of z^n in

$$\frac{\partial^2}{\partial x^2} P(z, x)|_{x=1} = \frac{1}{2}\frac{(2-z)z^2}{(1-z)^3} = \frac{1}{2}(2-z)z^2\sum_{i=0}^{\infty}\binom{i+2}{2}z^i$$

is $\frac{(n-1)(n+2)}{4}$.

– Coefficient of z^n in

$$\frac{\partial}{\partial y} P_a(z, y)|_{y=1} = \frac{1}{2^{a+1}}\frac{(2-z)^2 z^a}{(1-z)^2} = \frac{1}{2^{a+1}}(4z^a - 4z^{a+1} + z^{a+2})\sum_{i=0}^{\infty}(i+1)z^i$$

is $\frac{n-a+3}{2^{a+1}}$.

– Coefficient of z^n in

$$\frac{\partial^2}{\partial y^2} P_a(z, y)|_{y=1} = -\frac{1}{2^{2a+1}} \frac{(2-z)^3 z^{2a}}{(1-z)^3}$$

is $\frac{1}{4^{a+1}}(4a^2 - 4an - 18a + n^2 + 9n + 14)$.

– Coefficient of z^n in

$$\frac{\partial}{\partial y} P_{\bar{a}}(z, y)|_{y=1} = \frac{1}{2^a} \frac{(2-z)z^a}{(1-z)^2}$$

is $\frac{n-a+2}{2^a}$.

– Coefficient of z^n in

$$\frac{\partial^2}{\partial y^2} P_{\bar{a}}(z, y)|_{y=1} = \frac{1}{2^{2a-1}} \frac{(2-z)z^{2a}}{(1-z)^3}$$

is $\frac{4a^2 - 4an - 10a + n^2 + 5n + 4}{2^{2a}}$.

The following theorem follows immediately.

Theorem 12 *Let n be a fixed positive integer. Expected values and variances of the random variables X, X_a and $X_{\bar{a}}$ defined on Ω_n are as follows:*

$$E(X) = \frac{n+1}{2}, \qquad Var(X) = \frac{n-1}{4},$$

$$E(X_a) = \frac{n+3-a}{2^{a+1}}, \qquad Var(X_a) = \frac{(2^{a+1} - 2a + 3)n + 3(a^2 - 4a + 2) + 2^{a+1}(3-a)}{4^{a+1}},$$

$$E(X_{\bar{a}}) = \frac{n+2-a}{2^a}, \qquad Var(X_{\bar{a}}) = \frac{(2^a - 2a + 1)n + (3a - 2^a) + (a-2)}{4^a}.$$

7 R-2 composition tests

R-2 composition tests count the number of pre-specified runs in a string. The input of the test is a collection of binary strings with equal length n. We apply the test function to determine the number of occurrences of the restricted runs in each string, and call them as observed values. Afterwards, we apply χ^2 test and produce p-value using the bin probability tables (as described in [17]). We give the probabilities for $n \in \{128, 256, 1024, 4096\}$. It should be noted that the 8 test statistics defined in this paper are not necessarily independent.

7.1 Walkthrough

Tables 10, 11, 12, and 13 present the number of bins, bin values and the probabilities corresponding to each bins, for $n = 128, 256, 1024$ and 4096 respectively. As an example, to test the randomness of a collection of N binary strings of length $n = 128$, the first row of Table 10, that is the line labeled as "Total Runs" suggests the use of 8 bins, and gives the expected values of the number of total runs to be between 0 and 57 as $0.106982 \times N$, to be between 58 and 60 as $0.131978 \times N$, ... , to be between 72 and 128 as $0.106982 \times N$.

The procedure to test a collection of N binary strings of length n, using the R-2 Composition Tests family can be summarized as follows:

1. Depending on n, use the appropriate bin probability table given in the Appendix (Tables 10, 11, 12, or 13) to determine the corresponding number of bins for each of the test functions:
   - *number of all runs*,
   - *number of runs of length 1*,
   - *number of runs of length 2*,
   - *number of runs of length 3*,
   - *number of runs of length 4*,
   - *number of runs of length 5*,
   - *number of runs of length at least 5*,
   - *length of the longest run.*
2. Apply χ^2 Goodness of Fit Test, that is evaluate

$$\chi^2 = \sum_{i=1}^{B} \frac{(O_i - N \cdot p_i)^2}{N \cdot p_i} \quad \text{and} \quad p\text{-value} = \texttt{igamc}\left(\frac{B-1}{2}, \frac{\chi^2}{2}\right)$$

   where p_i's are obtained from bin probability Tables 10, 11, 12, and 13.
3. Print the p-value.

Pseudocode of the test is given at Algorithm 2 in the Appendix. The source code is available online at http://users.metu.edu.tr/muhid/stats.html.

Example 6 We apply R-2 Composition Test with $a = 2$ as an example to AES algorithm [18]. First we fix the key as 0. Then, by encrypting the plaintexts corresponding to the numbers from 0 to 100,000 we generate 100,000 many 128-bit ciphertexts. The observed number of runs of length $a = 2$ in each block are given in Table 3. Using these bin values, we compute the χ^2 and p-values as follows.

$$\chi^2 = \sum_{i=1}^{B} \frac{(O_i - N \cdot p_i)^2}{N \cdot p_i} = 2.2682 \quad \text{and} \quad p\text{-value} = \texttt{igamc}\left(\frac{8-1}{2}, \frac{2.2682}{2}\right) = 0.9435.$$

The complete list of p-values for this case is given in the second column of the following Table 4. The second row in that column shows the p-value for the number of all runs. Next 5 rows of the same column, labeled with numbers "1" to "5" show the p-value for the number of all runs of length a equal to 1 till 5. The row labeled as "≥ 5" is for the p-value of all

Table 3 Expected and observed values for AES, with $n = 128$, $a = 2$, $N = 100,000$

	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6	Bin 7	Bin 8
	0,...,11	12,13	14	15	16	17,18	19,20	21,...,128
Expected	10517.1	14153.2	9444.8	10377.2	10602.3	19119.4	13467.7	12318.3
Observed	10541	14089	9456	10282	10634	19089	13575	12334

Table 4 p-values for a, $N = 100,000$

a	AES with $n = 128$	SHA-3 with $n = 256$	π, with $n = 1024$	$\sqrt{2}$ with $n = 4096$
Total runs	0.731571	0.207460	0.978280	0.312366
1	0.832913	0.219225	0.435394	0.447400
2	0.943515	0.140130	0.446319	0.817061
3	0.118005	0.239282	0.171816	0.234689
4	0.868903	0.661484	0.659976	0.388651
5	0.849717	0.183703	0.949642	0.321176
≥ 5	0.995659	0.797747	0.405766	0.028659
Longest run	0.022376	0.234036	0.462707	0.124030

runs of length a greater than or equal to 5. Finally, the last row shows the p-value for the longest run test.

8 Application

This section reveals the results obtained from the application of the R-2 Composition Tests to various collections of strings in order to show the sensitivity of the tests. For this purpose, we generate pseudorandom and non-random data sets. The details are as follows.

First, we applied the tests to the data set, generated by using the AES algorithm in the same manner explained in the Example 6 above. For each of the 8 different tests, the p-values obtained are presented in the first column of the Table 5. As a second application, we considered the collection of outputs of SHA-3 algorithm, for the case $n = 256$. Starting with initial message 0 and recursive application of the algorithm, $S_i = H(S_{i-1})$, we collected the hash values and evaluated this collection with the 8 different tests introduced. The p-values obtained are presented in the first column of the Table 6. Then, we applied the R-2 Composition Tests to the binary expansions of π and $\sqrt{2}$. In the case of π, we produced 100,000 many 1024 bit blocks from its binary expansion and applied the tests for the case $n = 1024$. Similarly, we applied the test for a collection obtained from the binary expansion of $\sqrt{2}$, taking $n = 4096$. The p-values obtained are given in the first columns of the Tables 7 and 8.

Table 5 128 bits

	AES (128 bit)	$p_1 = 0.505$	$p_1 = 0.525$
Total runs	0.731571	0.249616	0
1	0.832913	0.321419	0
2	0.944	0.910527	0
3	0.118005	0.010618	0
4	0.868903	0.220267	0
5	0.849717	0.985643	0.00014
≥ 5	0.995659	0.42772	0
Longest run	0.022376	0.006053	0

Table 6 256 bits

	SHA-3 (256 bit)	$p_1 = 0.505$	$p_1 = 0.525$
Total runs	0.207460	0.116373	0
1	0.219225	0.325781	0
2	0.140130	0.910347	0
3	0.239282	0.567008	0
4	0.661484	0.718012	0
5	0.183703	0.380512	0
≥ 5	0.797747	0.233176	0
Longest run	0.234036	0.023001	0

Second, we applied the tests to the collection of biased binary strings. By setting each bit a_i as 1 when a randomly generated integer between 0 and 999 is greater than 505, and to 0 otherwise, we produced 100,000 weight-biased binary strings of the lengths 128, 256, 1024, and 4096 with bias 1% ($p_1 = Pr(a_i = 1) = 0.505$). We produced another collection with bias 5% ($p_1 = Pr(a_i = 1) = 0.525$) using the same idea. The results are presented in the second and third columns of the Tables 5, 6, 7, and 8. It is observed that the bias $p_1 = 0.505$ is detected only by the longest runs test for $n = 128$, $n = 256$ and $n = 1024$; and is detected by five of these tests for $n = 4096$. The bias $p_1 = 0.525$ is detected by all the tests.

After getting results for the weight-biased data set, we test the sensitivity of the R-2 Composition Tests on the run-biased binary strings. Since there is no canonical method to introduce a bias to the number of runs, we developed two methods for this purpose. First, we introduced a bias to the number of runs by flipping the last bit of the first run of length 3 in each sequence of the Example 6. We name the collection obtained by this method as *DataSet-1*. Notice that, this manipulation does not change the total number of runs, whereas it changes the number of runs of length 3. As seen in the Table 9, this bias is not detected by tests in the NIST test suite, namely the runs test and the test for the longest run of ones when default parameters are used. On the other hand, all the tests related with the number of runs of a specified length which are defined in this paper detected this bias. The manipulation is detected by sstringRun test of TestU01. The p-values are given in the Table 9.

The main advantage of the R-2 composition test is that it can evaluate a collection of relatively short strings directly as a collection, while the other tests in the literature can

Table 7 1024 bits

	π (1024 bit)	$p_1 = 0.505$	$p_1 = 0.525$
Total runs	0.978280	0.153411	0
1	0.435394	0.436316	0
2	0.44632	0.001407	0
3	0.171816	0.101505	0
4	0.659976	0.758987	0
5	0.949642	0.117841	0
≥ 5	0.405766	0.031281	0
Longest run	0.462707	0.002161	0

Table 8 4096 bits

	$\sqrt{2}$ (4096 bit)	$p_1 = 0.505$	$p_1 = 0.525$
Total runs	0.312366	0	0
1	0.447400	0.04765	0
2	0.817061	0	0
3	0.234689	0.000636	0
4	0.38865	0.967204	0
5	0.321176	0.599904	0
≥ 5	0.028659	0	0
Longest run	0.124029	0	0

evaluate such a collection only after concatenating them. However, after concatenating the collection, the defects in the individual strings may not be recognizable by the randomness tests. This can be a major problem in testing key generators in cryptographic applications. Moreover, depending on the order they are concatenated may cause different randomness properties. In order to illustrate this scenario we produced a non-random data set called *DataSet-2* using Algorithm 1 below. First, we take a random collection generated using the AES algorithm in Example 6. Then, we take a subset of this collection and modify each string in this subset so that each string starts with 1 and ends with 0. Note that, this collection can be flagged as non random easily as the number of total runs deviates from the expected number of total runs. In order make this defect in the individual strings not recognizable after concatenation, we decrease the number of totals runs by a second operation: if the sequence starts with 101, replace it by 111 and if it ends with 010, replace it by 000. We determine the subset of the collection to be modified as every sixth sequence in the collection. In short, for each of the every 6^{th} sequence $\{s_i\}_{i=0}^{127}$, we replace the first three bits of the sequence as follows

$$000 \to 100, \quad 001 \to 111, \quad 010 \to 110, \quad 011 \to 111$$

Table 9 Results of this work, NIST and TestU01 on manipulated data

	DataSet-1	DataSet-2
Total runs	0.311464	0
Longest run	0.094459	0.267156
Runs-1	0	0.000895
Runs-2	0	0.476132
Runs-3	0	0.007232
Runs-4	0	0.796350
Runs-5	0	0.768755
Runs>5	0	0.658832
NIST-Longest run	0.54962	0.552450
NIST-Runs	0.861091	0.000008
TestU01 sstringRun1	Failed	Passed
TestU01 sstringRun2	Passed	Passed

and do not make any change otherwise, the last three bits of the sequence as follows

$$001 \to 000, \quad 011 \to 000, \quad 101 \to 100, \quad 111 \to 110$$

and do not make any change otherwise. Algorithm 1 operates on 100,000 strings to generate *DataSet-2*.

R-2 Composition Tests detected the manipulation, that is the data set gets relatively low p-values from the runs of length 1 and the total number of runs tests. This manipulation is also detected by the NIST Runs test, but not detected by the test for the longest run of ones. However it is not detected by the run tests in Test U01. The p-values are given in the Table 9.

These experiments show that the new tests can detect some defects that other run tests do not.

Algorithm 1 Non-Random *DataSet-2* Generation

for all sequences $s_i = s_0 s_1 ... s_{127}$ in the set $i = 1, 2, \ldots, 100000$ **do**
  if $i \ mod \ 6 = 0$ **then**
    if $s_0 = 0$ **then**
      $s_0 \leftarrow 1$
      if $s_1 = 0$ and $s_2 = 1$ **then**
        $s_1 \leftarrow 1$
      end if
    end if
    if $s_{127} = 1$ **then**
      $s_{127} \leftarrow 0$
      if $s_{126} = 1$ and $s_{125} = 0$ **then**
        $s_{126} \leftarrow 0$
      end if
    end if
  end if
end for

9 Conclusion

In this work, we define a family of randomness tests in the spirit of Golomb's second randomness postulate. We give recursive formulas that are feasible to compute the exact probabilities of different type of runs, namely number of all runs in a binary segment of length n, that of all runs of length a, all runs of length at least a, and the number of all binary segments whose longest run has length at most L. Moreover, we find the exact distributions for all those run types and define new randomness tests. Afterwards, we apply the family of randomness tests to various collections of strings.

Throughout this work, we naturally consider binary strings where zeros and ones are produced with equal probabilities. As a future work, the same computations can be generalized for strings which are produced with unequal probabilities, which can be used for estimating the entropy of the source.

Appendix

Algorithm 2 R-2 Composition Test

for all sequences in the set **do**
  $k \leftarrow 1$
  $longestRun \leftarrow 0$
  $counter_j \leftarrow 0,\ j = 1, \ldots, 5$ and $counter_{k\geq 5} \leftarrow 0$
  $counter_{total} \leftarrow 0$
  for i=1 to n **do**
    if $\sigma_i = \sigma_{i-1}$ **then**
      $k \leftarrow k + 1$
    else
      if $k \leq 5$ **then**
        $counter_k \leftarrow counter_k + 1$
      else
        $counter_{k\geq 5} \leftarrow counter_{k\geq 5} + 1$
      end if
      $counter_{total} \leftarrow counter_{total} + 1$
      if $k \geq longestRun$ **then**
        $longestRun \leftarrow k$
      end if
      $k \leftarrow 1$
    end if
  end for
  Repeat the steps for the final run:
  if $k \leq 5$ **then**
    $counter_k \leftarrow counter_k + 1$
  else
    $counter_{k\geq 5} \leftarrow counter_{k\geq 5} + 1$
  end if
  $counter_{total} \leftarrow counter_{total} + 1$
  if $k \geq longestRun$ **then**
    $longestRun \leftarrow k$
  end if
  Increment the corresponding bin values for all counters and longestRun
end for

Table 10 Bin probabilities for $p_a(n, r)$ $(n = 128)$

Total runs	0,...,57	58,59,60	61,62	63,64	65,66	67,68	69,70,71	72,...,128
	0.106983	0.131978	0.122433	0.138606	0.138606	0.122433	0.131978	0.106983
$a = 1$	0,...,25	26,27,28	29,30	31,32	33,34	35,36	37,38,39	40,...,128
	0.133375	0.137522	0.116540	0.125052	0.121512	0.107511	0.122134	0.136354
$a = 2$	0,...,11	12,13	14	15	16	17,18	19,20	21,...,128
	0.105171	0.141532	0.094448	0.103772	0.106022	0.191194	0.134677	0.123183
$a = 3$	0,...,5	6	7	8	9	10	11	12,...,128
	0.163209	0.124208	0.150293	0.154855	0.137851	0.107208	0.073475	0.088902
$a = 4$	0,1	2	3	4	5	6	7	8,...,128
	0.076203	0.141979	0.205089	0.212690	0.168575	0.106134	0.054467	0.034863
$a = 5$	0	1	2	3	4	5,...,128		
	0.124340	0.274874	0.286809	0.187762	0.0864723	0.039742		
$a \geq 5$	0,1,2	3	4	5	6	7,...,128		
	0.205616	0.217211	0.230423	0.177464	0.102688	0.0665982		
Longest Run	0,...,5	6	7	8	9,...,128			
	0.121542	0.244832	0.248288	0.173365	0.211972			

Table 11 Bin probabilities for $p_a(n, r)$ $(n = 256)$

Total runs	0,...,119	120,...,123	124,125,126	127,128	129,130,131	132,133,134	135,...,138	139,...,256
	0.129807	0.135818	0.135509	0.098866	0.146403	0.127379	0.121057	0,105161
$a = 1$	0,...,54	55,...,58	59,60,61	62,63,64	65,66,67	68,69,70	71,...,74	75,...,256
	0.130924	0.124741	0.120871	0.132339	0.129502	0.113805	0.114540	0.133278
$a = 2$	0,...,26	27,28	29,30	31,32	33,34	35,36	37,38	39,...,256
	0.143346	0.107036	0.136249	0.149555	0.142544	0.118677	0.0867542	0.115838
$a = 3$	0,...,11	12,13	14	15	16	17	18,19	20,...,256
	0.102370	0.146365	0.098738	0.108426	0.110071	0.103673	0.165234	0.165123
$a = 4$	0,...,4	5	6	7	8	9	10	11,...,256
	0.082875	0.088272	0.125326	0.149199	0.151963	0.134453	0.104573	0.163339
$a = 5$	0,1	2	3	4	5	6	7	8,...,256
	0.081571	0.144378	0.202618	0.207016	0.164121	0.105083	0.055843	0.039370
$a \geq 5$	0,...,5	6	7	8	9	10	11	12,...,256
	0.156113	0.129448	0.159794	0.164641	0.143617	0.107217	0.069079	0.070092
Longest run	0,...,6	7	8	9	10	11,...,256		
	0.127564	0.239576	0.243969	0.172820	0.102349	0.113722		

Table 12 Bin probabilities for $p_a(n, r)$ $(n = 1024)$

Total runs	0,...,494	495,...,501	502,...,507	508,...,512	513,...,517	518,...,523	524,...,531	532,...,1024
	0.130173	0.115614	0.131494	0.122719	0.122719	0.131494	0.128394	0.117393
$a = 1$	0,...,235	236,...,244	245,...,250	251,...,256	257,...,262	263,...,268	269,...,277	278,...,1024
	0.119434	0.133837	0.119232	0.131952	0.130520	0.115674	0.128227	0.121124
$a = 2$	0,...,115	116,...,120	121,...,124	125,...,127	128,129,130	131,...,134	135,...,139	140,...,1024
	0.115587	0.121600	0.132223	0.111075	0.111986	0.136132	0.129812	0.141585
$a = 3$	0,...,55	56,57,58	59,60,61	62,63	64,65	66,67,68	69,...,72	73,...,1024
	0.118005	0.106561	0.144537	0.108828	0.109663	0.148893	0.143327	0.120187
$a = 4$	0,...,26	27,28	29,30	31,32	33,34	35,36	37,38	39,...,1024
	0.145406	0.110683	0.140501	0.152801	0.143493	0.117165	0.083691	0.106260
$a = 5$	0,...,11	12,13	14	15	16	17,18	19,20	21,...,1024
	0.114461	0.148465	0.096762	0.104566	0.105133	0.185613	0.127936	0.117064
$a \geq 5$	0,...,26	27,28	29,30	31	32,33	34,35	36,37	38,...,1024
	0.128388	0.112659	0.149263	0.082345	0.163046	0.139882	0.101996	0.122421
Longest run	0,...,8	9	10	11	12	13,...,1024		
	0.132918	0.234780	0.240252	0.172403	0.103271	0.116375		

Table 13 Bin probabilities for $p_a(n, r)$ $(n = 4096)$

Total runs	0,...,2011	2012,...,2026	2027,...,2038	2040,...,2048	2049,...,2058	2059,...,2069	2070,...,2084	2085,...,4096
	0.123759	0.122101	0.131458	0.122682	0.122682	0.121510	0.125544	0.130264
$a = 1$	0,...,983	984,...,1000	1001,...,1013	1014,...,1024	1025,...,1035	1036,...,1048	1049,...,1065	1066,...,4096
	0.125557	0.126626	0.129014	0.121032	0.120425	0.127122	0.123942	0.126281
$a = 2$	0,...,487	488,...,497	498,...,505	506,...,512	513,...,519	520,...,526	527,...,536	537,...,4096
	0.121963	0.123681	0.133246	0.130194	0.128810	0.114429	0.122598	0.125078
$a = 3$	0,...,239	240,...,246	247,...,251	252,...,256	257,...,260	261,...,265	266,...,272	273,...,4096
	0.125813	0.130508	0.123607	0.136688	0.108262	0.121349	0.127063	0.126711
$a = 4$	0,...,116	117,...,121	122,123,124	125,126,127	128,129,130	131,...,134	135,...,139	140,...,4096
	0.134271	0.134767	0.103949	0.113417	0.113822	0.137120	0.128609	0.134044
$a = 5$	0,...,55	56,57,58	59,60,61	62,63	64,65	66,67,68	69,...,72	73,...,4096
	0.129820	0.107084	0.140813	0.104409	0.104704	0.142683	0.140450	0.130037
$a \geq 5$	0,...,116	117,...,121	122,123,124	125,126,127	128,129,130	131,...,134	135,...,139	140,...,4096
	0.116394	0.137084	0.110400	0.122447	0.123151	0.146023	0.130532	0.113970
Longest run	0,...,10	11	12	13	14	15,...,4096		
	0.134605	0.233230	0.239123	0.172301	0.103566	0.117175		

References

1. Golomb, W.S.: Shift Register Sequences. Aegean Park Press, Laguna Hills (1982)
2. Knuth, D.E.: The Art of Computer Programming, 3rd edn., vol. 2, Addison-Wesley Longman Publishing Co., Inc., Seminumerical Algorithms, Boston (1997)
3. L'ecuyer, P., Simard, R.: Testu01: A C library for empirical testing of random number generators. ACM Trans. Math. Softw. **33**(4), 22 (2007)
4. Mood, A.M.: The distribution theory of runs. Ann. Math. Stat. **11.4**, 367–392 (1940)
5. Fu, J.C., Koutras, M.V.: Distribution theory of runs: a Markov chain approach. J. Am. Stat. Assoc. **89.427**, 1050–1058 (1994)
6. Marsaglia, G.: The Marsaglia Random Rumber CDROM Including the Diehard Battery of Tests of Tandomness, http://www.stat.fsu.edu/pub/diehard/ (1995)
7. Bassham III, L.E., Rukhin, A.L., Soto, J., et al.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Tech. Rep Sp 800-22 Rev. 1a, NIST, Gaithersburg (2010)
8. Doğanaksoy, A., Sulak, F., Uğuz, M., Şeker, O., Akcengiz, Z.: New statistical randomness tests based on length of runs. Math. Probl. Eng. **2015**(626408), 14 (2015)
9. van Lint, J.H., Wilson, R.M.: A Course in Combinatorics. Cambridge University Press, New York (1993)
10. Chinn, P., Heubach, S.: Compositions of n with no occurence of k. Congressus Numerantium **164**, 33–51 (2003)
11. Heubach, S., Mansour, T.: Compositions of n with Parts in a Set. Congressus Numerantium **168**, 127–143 (2004)
12. Richmond, B., Knopfmacher, A.: Compositions with distinct parts. Aequationes Math. **49**(1-2), 86–87 (1995)
13. Malandro, M.E.: Integer Compositions with Part Sizes not Exceeding k, Preprint available at: https://arxiv.org/pdf/1108.0337v2.pdf (2012)
14. Munagi, A.O., Sellers, J.A.: Some inplace identities for integer compositions. Quaest. Math. **38**(4), 535–540 (2015)
15. Chinn, P., Heubach, S.: (1,K)-compositions. Congressus Numerantium **164**, 183–194 (2003)
16. Jaklic, G., Vitrih, V., Zagar, E.: Closed form formula for the number of restricted compositions. Bull. Aus. Math. Soc. **81**, 289–297 (2010)
17. Sulak, F., Doğanaksoy, A., Ege, B., Koçak, O.: Evaluation of Randomness Test Results for Short Sequences, Sequences and Their Applications, vol. 6338, pp. 310–319. SETA 2010, Lecture Notes in Computer Science, Berlin (2010)
18. Daeman, J., Rijmen, V.: The Design of Rijndael: AES - the Advanced Encryption Standard. Springer, Berlin (2002)