Entropy **2015**, *17*, 7118-7132; doi:10.3390/e17107118

OPEN ACCESS

entropy

ISSN 1099-4300

www.mdpi.com/journal/entropy

Article

A Novel Method for Increasing the Entropy of a Sequence of Independent, Discrete Random Variables

Mieczyslaw Jessa

Faculty of Electronics and Telecommunications, Poznan University of Technology, ul. Polanka 3, 61-131 Poznan, Poland; E-Mail: mjessa@et.put.poznan.pl; Tel.: +48-61-665-3854

Academic Editor: Raúl Alcaraz Martínez

Received: 23 April 2015 / Accepted: 12 October 2015 / Published: 22 October 2015

Abstract: In this paper, we propose a novel method for increasing the entropy of a sequence of independent, discrete random variables with arbitrary distributions. The method uses an auxiliary table and a novel theorem that concerns the entropy of a sequence in which the elements are a bitwise exclusive-or sum of independent discrete random variables.

Keywords: entropy; sequences; random variables; distribution; combining exclusive-or

PACS Codes: 05.10.±a

1. Introduction

Sequences of random variables play an important role in many fields of science. If we have a sequence of k random variables, then a natural and important question is as follows: what is the entropy of the sequence, and how does it grow with k? Examples include dynamical systems, cryptography, simulations, and statistics. Usually, we require that the value of the entropy be maximal or at least close to the maximum.

The literature devoted to sequences includes many books, journals and conference papers. The authors consider various aspects of sequences and their applications [1] but most of these papers concentrate on the mathematical description of sequences, e.g., as a ring of integers, the correlation within a sequence elements, measures of sequence complexity, whether a sequence can be modeled as a sequence of independent, identically distributed random numbers, and so on. The problem of maximizing the entropy of a sequence of independent, discrete random variables is considered mainly by scientists involved in the theory and practice of random numbers. Because high entropy of a random sequence is a necessary condition of its use in cryptography, several general methods that increase the sequence entropy have been proposed [2–11]. These methods include both very simple correctors and complicated hash functions or ciphers. Examples of simple correctors are combining exclusive-or two or more neighbor bits produced by a random source, feeding a linear feedback shift register (LFSR) with a sequence with small entropy, or using the von Neumann corrector. Examples of complicated correctors are hash functions SHA-1, SHA-2, decryption-encryption standard (DES), advanced encryption standard (AES), and so on. Solutions using extractor algorithms and resilient functions that "filter out" any deterministic bits from a raw sequence with a deterministic function also exist [5–10]. A good review of post-processing methods is given in [6].

In this paper, we define a bitwise exclusive-or sum of discrete random variables and prove formally that under certain assumptions, an infinite bitwise exclusive-or sum of independent discrete random variables with values encoded by l bits has a uniform distribution. We propose to use this property, the symbols that have already been produced, and an auxiliary table to produce a sequence that has almost maximal entropy using a single random source. The differences between existing algorithms and the algorithm XOR-B proposed in this paper are such that the uniform distribution has been proven formally and the algorithm does not need auxiliary deterministic circuit. Known algorithms are based on heuristic arguments and empirical experiments or introduce deterministic bits such as methods using LFSRs. The exception is the von Neumann corrector. Unfortunately, the von Neumann corrector produces sequences with variable and difficult to predict bit rates, which significantly limits its applications. Consequently, the most frequently used post-processing without an external auxiliary deterministic circuit and significant bit reduction uses a hash function or a cipher implemented in a field programmable gate array (FPGA) or in an application specific integrated circuit (ASIC) [11].

The organization of this paper is as follows. Section 2 contains the basic definitions. The proof of the proposed theorem and the algorithms for producing a sequence with almost maximal entropy with the use of a sequence with smaller entropy and an auxiliary table are contained in Section 3. The same section contains an example that illustrates a practical use of the proposed theorem. The paper ends with the conclusions in Section 4.

2. Bitwise Exclusive-or Sum of Random Variables

The mathematical environment for the method and theorem proposed in this paper is the probability space and operations on its elements. The probability space is a measurable space that is defined by a triple (Ω, F, P), where [12–14]:

(1) Ω is any non-empty set called the sample space.

(2) F is a σ-algebra or σ-field, *i.e.*, a collection of subsets of Ω that satisfies four postulates:

i. $\Omega \in F$,

ii. for any subset $A \subseteq \Omega$, if $A \in F$, then $A^C \in F$,

iii. for any countable collection of subsets $\{A_i \subseteq \Omega\}$, $i = 1,2,\ldots$, if $A_i \subseteq \Omega$, then $\left(\bigcup_i A_i\right) \in F$,

and,

iv. for any countable collection of subsets $\{A_i \subseteq \Omega\}$, $i = 1,2,...$, if $A_i \subseteq \Omega$, then $\left(\bigcap_i A_i\right) \in F$.

(3) P is a countably additive measure, *i.e.*, a mapping from F to the closed interval $\left[0,1\right]$ such that $P(\varnothing) = 0$ and $P(\Omega) = 1$.

The sample space Ω is the set of all possible random outcomes of some experiment. A random variable assigns a numerical value to each of those outcomes [15]:

Definition 1. Given a probability space (Ω, F, P), a random variable is a function *X* from Ω to the real numbers *R* such that

$$\{\omega \in \Omega;\quad X(\omega) \le x\} \in F \tag{1}$$

Formula (1) states that the function *X* must be measurable. For a continuous probability space, the sample space Ω is uncountable, and the random variable *X* can assign an uncountable number of values. For the discrete probability space, the sample space Ω is finite or countable [12–15].

Definition 2. We say that *X* is a discrete random variable if the function *X* assumes only a finite or countable number of values.

In this paper, we assume that a random source S_X produces the sequences of symbols x_k, $k = 1,2,...$. The symbols are represented unequivocally by *l*-bit integer words 0, 1, …, *p* − 1 from the finite set $A = \{0,1,...,p-1\}$, where $p = 2^l$. The process of emitting symbols can be modeled as a stochastic process

$$S_X = \{X(k)\},\ k = 1,2,... \tag{2}$$

where $X(k)$ are discrete random variables with values from the closed interval $[0, 2^l - 1]$.

Definition 3. The entropy of a stochastic process $S_X = \{X(k)\}$ is defined by [16]

$$H_A(S_X) = \lim_{n\to\infty} \frac{1}{k} H\left(X(1), X(2), ..., X(k)\right) \tag{3}$$

when the limit exists.

Formula (2) determines the average rate of the asymptotic growth of the entropy $H_A(S_X)$ with increasing *k* if the limit exists [16]. The entropy $H_A(S_X)$ is also termed the entropy rate. The value of $H_A(S_X)$ is in bits per time-step (bit/time-step). If $\{X(k)\}$ is a sequence of independent and identically distributed (*i.i.d.*) random variables, then [16]

$$H_A(S_X) = H\left(X(1)\right) = H\left(X(2)\right) = ... = H\left(X(k)\right) \tag{4}$$

If $\{X(k)\}$ is a sequence of independent but not identically distributed random variables, then [16]

$$H\{(X(k)\} = \sum_{j=1}^{k} H(X(j)) \tag{5}$$

Following Thomas and Cover, it worth noting that we can choose sequences of distributions on $X(0), X(1),...$ such that the limit $H_A(S_X) = \frac{1}{k}\sum_{j=1}^{k} H\left(X(j)\right)$ does not exist [16].

The greatest value of $H_A(S_X)$ is equal to l bits per time-step. This value is obtained when $\{X(k)\}$ is a sequence of independent and uniformly distributed random variables. Thus, the key point for the greatest entropy is the independence and uniformity of the distributions on $X(1), X(2),...$.

Because the values of $X(k)$ are encoded by l bits, the random variable $X(k)$ can be written unequivocally as a sequence of l binary random variables $X_i(k)$, $i = 0,1,...,l-1$, *i.e.*, $X(k) = \left\{X_i(k)\right\}$.

Let S_Y be another random source that produces sequences of symbols y_k, $k = 1,2,...$. As previously noted, the symbols are represented unequivocally by l-bit integer words 0, 1, …, p − 1 from the finite set $A = \{0,1,...,p-1\}$, where $p = 2^l$, and the process of emitting symbols can be modeled as a stochastic process

$$S_Y = \{Y(k)\},\ k = 1,2,... \tag{6}$$

where $Y(k)$ are random variables with values from the closed interval $[0, 2^l - 1]$. Each random variable $Y(k)$ can be written unequivocally as a sequence of l binary random variables $Y_i(k)$, $i = 0,1,...,l-1$, *i.e.*, $Y(k) = \left\{Y_i(k)\right\}$.

Definition 4. A bitwise exclusive-or sum $Z(k)$ of independent random variables $X(k)$ and $Y(k)$ is an operation on sequences $\left\{X_i(k)\right\}$ and $\left\{Y_i(k)\right\}$ of equal length of binary random variables that combines $X_i(k)$ and $Y_i(k)$, $i = 0,1,...,l-1$ using the exclusive-or operation, *i.e.*,

$$Z(k) = X(k) \oplus Y(k) = \left\{X_i(k)\right\} \oplus \left\{Y_i(k)\right\} = \left\{X_i(k) \oplus Y_i(k)\right\} \tag{7}$$

If $X_i(k)$ and $Y_i(k)$ are binary random variables, then $Z_i(k)$ is also a binary random variable. Consequently, $Z(k) = \left\{Z_i(k)\right\} = \left\{X_i(k) \oplus Y_i(k)\right\}$ are random variables with values in the closed interval $[0, 2^l - 1]$. The probability that $X(k)$ assumes a given value x from $[0, 2^l - 1]$ is equal to $P\left(X(k) = x\right) = p_x$. Similarly, the probability that $Y(k)$ assumes a given value y from $[0, 2^l - 1]$ is equal to $P\left(Y(k) = y\right) = p_y$. For the random variable $Z(k)$, $P\left(Z(k) = z\right) = p_z$, where $z \in [0, 2^l - 1]$.

3. Increasing the Entropy of a Sequence of Discrete Random Variables with an Auxiliary Table

As noted in the previous section, the process of emitting symbols from $A = \{0,1,...,p-1\}$ can be modeled as a stochastic process $S_X = \{X(k)\}$, $k = 1,2,...$. The maximal entropy of $S_X = \{X(k)\}$ is obtained when $X(k)$ are independent and uniformly distributed random variables. In this section, we assume that $X(k)$ are independent but not uniformly distributed. They can also have different distributions for different k. We search for a computationally efficient algorithm for processing a sequence of symbols x_k emitted by S_X that provides at the output a sequence that can be modeled as a sequence of independent and uniformly distributed random variables. The proposed algorithm uses an auxiliary table T with L cells and has the following form:

Algorithm XOR-A

```
Initialization:
Fill all of the cells of T with the L subsequent l-bit words produced by $S_X$;
Computations:
for $n := 1$ to $N$ do
    Produce $l-bit\ word\ x_{L+n}$
    Produce $m-bit\ random\ number\ s_n \in \left[0, 2^m - 1\right]$ with uniform distribution,
    $\hat{s}_n = s_n + 1$
    $j := n \bmod L, \quad L \geq R \cdot 2^m$
    $z_n := x_{L+n} \oplus T[(j + \hat{s}_n) \bmod L] \oplus \ldots \oplus T[(j + R \cdot \hat{s}_n) \bmod L]$
    $T[j] := x_{L+n}$
end;
```

The subsequent *l*-bit words x_k are written cyclically into table *T* and addressed from 0 to $L-1$. During each step, we choose *R* + 1 *l*-bit words $t_0, t_1, \ldots, t_R$, where t_0 is the current x_k and $t_1, \ldots, t_R$ are read from *T*. The addresses of $t_1, \ldots, t_R$ depend on the random number $s_n \in [0, 2^m - 1]$ being produced by an auxiliary random source with a uniform distribution. The elements of the *R* + 1 vectors $t_0, t_1, \ldots, t_R$ are next summed modulo 2, which forms a single output z_n with values from $[0, 2^l - 1]$. To avoid multiple summing of the same number, the size of *T* satisfies the condition $L \geq R \cdot 2^m$, where *R* is an integer. For the same reason, we use $\hat{s}_n = s_n + 1$ instead of s_n. Producing an *N*-element sequence $\{z_n\}$ requires the emission of $(N + L)$ *l*-bit words x_k.

Algorithm XOR-A uses the property that a sequence formed from a sequence of independent random numbers is also random. This property has been used by MacLaren and Marsaglia [17] to permute sequences from the linear congruential pseudorandom generator with an auxiliary and simpler, pseudorandom number generator. They noticed that the shuffled sequence might have better statistical properties and a longer period. This method is known in the literature as Algorithm M [18]. Another heuristic approach used in algorithm XOR-A is the observation that combining XOR random words can also provide sequences with better statistical properties, in particular with higher entropy rates [19]. No exact theory explaining why combined generators can significantly improve the properties of sequences has been published yet. We know only the heuristic arguments. For example, Deng and George, and later Deng *et al.*, provided arguments based on probability densities [20,21]. Some heuristic arguments come from Marsaglia [22]. One of the intuitive arguments for why combining improves the statistical properties of sequences and why it is better than shuffling is that a combined generator can produce new numbers, whereas shuffling only permutes existing elements.

An important factor in combined generators is the choice of mathematical operation that combines independent digital words. Because combining should not significantly increase the computation effort, only simple operations are considered: addition, addition modulo *p*, where *p* is an integer, and bitwise exclusive-or. The latter has a mixing property, commonly used, e.g., in ciphers. Examples include the decryption-encryption standard, the advanced encryption standard or the secure hash algorithms SHA-1 or SHA-2, *etc.*, [11,23,24]. The XOR operation applied to sequences of bits produced by independent source generators also reduces the correlation between adjacent bits [3,6]. To use this property, we can divide a sequence into, e.g., l-bit disjoint blocks and compute the XOR function for each block. As the result, a sequence with a lower correlation between adjacent bits is obtained, but this method reduces the bit rate l times. Combining source streams according to Algorithm XOR-A decreases the output speed only slightly because of the fast operations used (this problem will be discussed at the end of Section 3). The distance between the elements that form the i-th, $i = 1, 2, ..., R$, source stream changes randomly because s_n is random. Let us emphasize, that this distance cannot be constant because the obtained source sequences are shifted versions of the same sequence. It cannot also be deterministic because elements of the source streams may repeat periodically, producing periodic patterns in the output sequence. This limits the use of algorithm XOR-A to sequences with a length not greater than the period of the auxiliary generator when the source of numbers $\{s_n\}$ is deterministic. However, if $\{s_n\}$ is produced by an entropy source, then this limitation does not exist.

Algorithm XOR-A provides a sequence $\{z_n\}$ of l-bit words. Because a bitwise exclusive-or sum of random variables is also a random variable, each z_n is a value of a certain discrete random variable $Z(n)$, $n = 1,2,...,N$. A set of l-bit words $t_0, t_1, ..., t_R$ is formed randomly and independently of the previous sets. Consequently, $Z(n)$ and $Z(n+j)$ are also independent for any $j \neq 0$. The open problem is the distribution of variables $Z(n)$ when $X(k)$ can have arbitrary, the same or different distributions.

Theorem 1.

If

(1) Random variables $X(k)$, $k = 1,2,...$ are independent,

(2) An auxiliary source provides the random numbers s_n with a uniform distribution

(3) $\underset{x \in [0, 2^{l-1}]}{\forall} \; p_x \neq 0$,

(4) $\underset{x \in [0, 2^{l-1}]}{\forall} \; p_x \neq 1$,

then the random variable $Z(n) = X(k) \oplus X(k + \hat{s}_n) \oplus ... \oplus X(k + R \cdot \hat{s}_n)$ has a uniform distribution in the interval $[0, 2^l - 1]$ for $R \to \infty$.

Proof. Independently of k and the value of $\hat{s}_n$, the random variable $Z(n)$, $n = 1,2,...$ is a bitwise exclusive-or sum of $R+1$ independent random variables with theoretically arbitrary, the same or different distributions. Consequently, proving the distribution of $Z(n) = X(k) \oplus X(k + \hat{s}_n) \oplus ... \oplus X(k + R \cdot \hat{s}_n)$, $k = 1,2,...$ can be reduced to proving the distribution of the random variable

$$Z = U(0) \oplus U(1) \oplus ... \oplus U(R) \tag{8}$$

where $U(0), U(1), ..., U(R)$ are independent with arbitrary, the same or different distributions. Because the values of $U(r)$, $r = 0,1,...,R$ are encoded by *l* bits, the random variable $U(r)$ can also be written unequivocally as a sequence of *l* binary random variables $U_i(k)$, $i = 0,1,...,l-1$, *i.e.*, $U(r) = \left\{ \underset{i}{U}(r) \right\}$. Similarly, the values of *Z* are also encoded by *l* bits, and Z can be written unequivocally as a sequence of *l* binary random variables Z_i, $i = 0,1,...,l-1$, *i.e.*, $Z = \{Z_i\}$. It is also that

$$Z = U(0) \oplus U(1) \oplus ... \oplus U(R) = \{Z_i\} = \{U_i(0) \oplus U_i(1) \oplus ... \oplus U_i(R)\} \tag{9}$$

To prove that *Z* has a uniform distribution, we first show that $P(Z_i = 0) = P(Z_i = 1) = 1/2$ for $R \to \infty$, where *P* denotes the probability and $i = 0,1,...,l-1$. The key to this proof is a calculation trick and the methodology introduced by R. B. Davies in a private paper [3]. The link to this page can also be found at the National Institute of Standards and Technology (NIST) webpage [25]. If U_i takes the values of 0 and 1, then $f(U_i) = 1 - 2U_i$ takes the values of 1 and −1. Consequently [3],

$$f(Z_i) = f\left(U_i(0) \oplus U_i(1) \oplus ... \oplus U_i(R)\right) = f\left(U_i(0)\right) \cdot f\left(U_i(1)\right) \cdot ... \cdot f\left(U_i(R)\right) \tag{10}$$

If $f(Z_i) = 1 - 2Z_i$, then $Z_i = \left(1 - f(Z_i)\right)/2$. Assuming that $E(Z_i)$ is the expected value (mean) of the random variable Z_i, we obtain the following:

$$E(Z_i) = E\left[\frac{1 - f(Z_i)}{2}\right] = \frac{1}{2} - \frac{1}{2}E\left[f(Z_i)\right] = \frac{1}{2} - \frac{1}{2}E\left[f\left(U_i(0)\right) \cdot f\left(U_i(1)\right) \cdot ... \cdot f\left(U_i(R)\right)\right] \tag{11}$$

If $U_i(0), U_i(1), ..., U_i(R)$ are independent, then

$$E(Z_i) = \frac{1}{2} - \frac{1}{2}\left\{E\left[f\left(U_i(0)\right)\right] \cdot E\left[f\left(U_i(1)\right)\right] \cdot ... \cdot E\left[f\left(U_i(R)\right)\right]\right\} \tag{12}$$

Because

$$E\left[f\left(U_i(k)\right)\right] = 1 - 2E\left(U_i(k)\right) \tag{13}$$

we obtain

$$E(Z_i) = \frac{1}{2} - \frac{1}{2}\left\{\left[1 - 2E\left(U_i(0)\right)\right] \cdot \left[1 - 2E\left(U_i(1)\right)\right] \cdot ... \cdot \left[1 - 2E\left(X_i(R)\right)\right]\right\} \tag{14}$$

or

$$E(Z_i) = \frac{1}{2} - \frac{1}{2}\left\{2\Delta_i(0) \cdot 2\Delta_i(1) \cdot ... \cdot 2\Delta_i(R)\right\} = \frac{1}{2} - 2^R \prod_{r=0}^{R} \Delta_i(r) \tag{15}$$

where

$$\Delta_i(r) = \frac{1}{2} - E\left(X_i(r)\right) \tag{16}$$

Considering the sign of Δ_i,

$$E(Z_i) = \begin{cases} \frac{1}{2} - 2^R \prod_{r=0}^{R} \left|\Delta_i(r)\right| & \text{for } \prod_{r=0}^{R} \Delta_i(r) \geq 0 \\ \frac{1}{2} + 2^R \prod_{r=0}^{R} \left|\Delta_i(r)\right| & \text{for } \prod_{r=0}^{R} \Delta_i(r) < 0 \end{cases} \tag{17}$$

Based on the assumption, the random variables can assume any value from the interval $[0, 2^l - 1]$ with a nonzero probability, and no value from $[0, 2^l - 1]$ is "certain," *i.e.*, it cannot be assumed with a probability equal to unity. Consequently, any subsequence of an *l*-bit sequence also appears with a nonzero and smaller than unity probability, and the variables $U_i(r)$, $r = 0,1,...,R$, $i = 0,1,...,l-1$ assume both zero and one for each *i*. In this case, the expected value $E(U_i(r))$ is always nonzero and is smaller than unity, *i.e.*,

$$\underset{i=0,1,...,l-1,\, r=0,1,...,R}{\forall} \quad 0 < 2\left|\Delta_i(r)\right| < 1 \tag{18}$$

Because the product of the numbers with values in the interval (0,1) decreases to zero as the quantity of multiplied numbers increases, we obtain

$$\lim_{R\to\infty} E(Z_i) = \lim_{R\to\infty} \begin{cases} \frac{1}{2} - 2^R \prod_{r=0}^{R} \left|\Delta_i(r)\right| & for \quad \prod_{r=0}^{R} \Delta_i(r) \geq 0 \\ \frac{1}{2} + 2^R \prod_{r=0}^{R} \left|\Delta_i(r)\right| & for \quad \prod_{r=0}^{R} \Delta_i(r) < 0 \end{cases} = \frac{1}{2} \tag{19}$$

Variable Z_i takes only two values, zero or one. It leads to the following equality

$$P(Z_i = 0) = P(Z_i = 1) = 1/2 \tag{20}$$

Because the random variables $U(r)$, $r = 0,1,...,R$ assume all values from $[0, 2^l - 1]$ with a nonzero and smaller than unity probability and $P(Z_i = 0) = P(Z_i = 1) = 1/2$ for all $i = 0,1,...,l-1$, the variable $Z = \{Z_i\}$ has a uniform distribution in $[0, 2^l - 1]$ for *R* approaching infinity. This completes the proof.

Let us emphasize that the assumption that $U(r)$, $r = 0,1,...,R$ assumes values from $[0, 2^l - 1]$ with a nonzero and smaller than unity probability is necessary to draw the conclusion that equal probabilities of assuming zeros and ones by Z_i results in a uniform distribution of $Z = \{Z_i\}$. This assumption eliminates all of the specific situations, *i.e.*, combining variables with fixed (probability equal unity) values or the lack of some numbers (zero probability). As an example, let us consider the case of $l = 2$. If $U(r)$ assumes, e.g., only two values, 3 and 0, with a probability of 1/2 (the numbers 1 and 2 are assumed to have zero probability), then the result of the combination is the numbers 3 or 0, independent of *R*. The number 3 is obtained if we combine an odd number of number 3s, and the number 0 is obtained for an even number of number 3s, although $P(Z_0 = 0) = P(Z_0 = 1) = 1/2$ and $P(Z_1 = 0) = P(Z_1 = 1) = 1/2$.

Words $t_1, t_2, ..., t_R$ are chosen from the words that were already produced by the random source S_X. Theorem 1 shows that perfect uniformity (the entropy equal to *l* bits/time-step) can only be obtained in the limit $R \to \infty$, *i.e.*, for table *T* with $L \to \infty$ cells. In a practical system, *L* is finite, and consequently, the entropy can only be close to the theoretical limit. Assuming a fixed *R*, which results from the assumed acceptable value of the entropy $H_A(S_Z)$ of sequence $\{Z(n)\}$, the smallest *L* is equal to $2R$ ($m = 1$). A small *m* enables some of the words $t_1, t_2, ..., t_R$ chosen in step *n* to repeat for the next several iterations with a large probability. A greater value for *m* reduces this probability but increases the size of table *T*. A smallest acceptable *m* should ensure that the smallest distance, *i.e.*, $\hat{s}_n = 1$, repeats statistically no more frequently than every R iterations when we use x_{n+L} as t_0. This condition can be written as

$$P(\hat{s}_n = 1) \leq \frac{1}{R} \tag{21}$$

where P is the probability. Because $\{\hat{s}_n\}$ has a uniform distribution, Condition (21) takes the following form:

$$\frac{1}{2^m} \leq \frac{1}{R} \tag{22}$$

or

$$m \geq \log_2 R \tag{23}$$

Because m must be an integer, inequality (23) takes the form

$$m \geq \lceil \log_2 R \rceil \tag{24}$$

where $\lceil \alpha \rceil$ is the smallest integer number that is greater than α, $\alpha \geq 0$. A given distance $\hat{s}_n$ repeats statistically every R iterations for $m = \log_2 R$ or rarely, when $m > \log_2 R$.

The basic weakness of algorithm XOR-A is the necessity of using an additional source of randomness that provides random numbers s_n with a uniform distribution. Because the proof of the proposed theorem indicates that such numbers are available at the output of algorithm XOR-A, they can be used to produce the numbers s_n. We exploit here the fact that any subsequence of digits of a random number can be used to form another random number. Therefore, if z_n is random, then any sequence formed from bits of z_n is also random [19]. Assuming that

$$\hat{s}_n := 1 + \mathrm{trunc}_m(z_n) \tag{25}$$

where the truncation operation leaves the m higher-order bits of number z_n, the algorithm XOR-A can be modified in the following way:

Algorithm XOR-B

```
Initialization:

Choose the size $L$ of an auxiliary Table $T$;
Fill all of the cells of $T$ with the $L$ subsequent $l$-bit words produced by $S_X$;
    $\hat{s}_1 := 1$

Computations:

for $n := 1$ to $N$ do
    Produce $l - bit\ word\ x_{L+n}$
    $j := n \bmod L, \quad L \geq R \cdot 2^m$
    $z_n := x_{L+n} \oplus T[(j + \hat{s}_n) \bmod L] \oplus \ldots \oplus T[(j + R \cdot \hat{s}_n) \bmod L]$
    $T[j] := x_{L+n}$
    $\hat{s}_{n+1} := 1 + \mathrm{trunc}_m(z_n), \quad m \leq l$
end;
```

A limitation for algorithm XOR-B is the number R of words $t_1, t_2, \ldots, t_R$ that are read from T. Parameter R must satisfy the condition

$$R \le 2^l \tag{26}$$

One of the factors determining the utility of a proposed algorithm is its computational complexity. The framework for the complexity of computations is contained in a classical paper by M. O. Rabin [26]. In algorithm XOR-B the following operations on l-bit words can be extracted during the production a single l-bit word z_n:

i $A \bmod B$—division of two l-bit numbers with a rest,

ii $A \cdot B$—multiplication of two l-bit numbers,

iii $A + B$—addition of two l-bit numbers,

where operation (i) can be reduced to multiplication by an inverse number. Assuming that the elementary operation is addition, the complexity of computations of (i) and (ii) can be equal to $O(l^2)$, $O(l^{1.58})$ or $O(l \log l / \log \log l)$, where O is one of the family of Bachmann-Landau notations known as "Big O" [27,28]. The concrete value depends on the assumed multiplication algorithm. The complexity of the computation of (iii) is $O(l)$ or $O(\log l)$. As previously, the details depend on the addition algorithm assumed. Algorithm XOR-B also uses logic operation XOR on l-bit words. The computational complexity of this operation can be reduced to l multiplications of one-bit numbers with addition in the Galois field GF(2). The complexity of this operation is $O(l)$. The dominating operation in algorithm XOR-B is:

$$z_n := x_{L+n} \oplus T[(j + \hat{s}_n) \bmod L] \oplus \ldots \oplus T[(j + R \cdot \hat{s}_n) \bmod L] \tag{27}$$

The assignment from Equation (27) can be factored into R operations XOR of the form

$$A = A \oplus T[(B + (C \cdot D)) \bmod E] \tag{28}$$

The complexity of Equation (28) is equal to:

$$\begin{aligned} f(l) = {} & l^2(\mathit{multiplications}) + l(\mathit{addition}) + l^2(\mathit{division\ with\ rest}) \\ & + C(\text{constant time of reading elements from table T}) + l(l\text{-bit XOR}) \end{aligned} \tag{29}$$

Omitting a constant C, Equation (29) is reduced to

$$f(l) = 2l(l+1) \tag{30}$$

Because the dominating operation is multiplication, the computational complexity of algorithm XOR-B can be assessed as $O(l^2)$ for a simple multiplication of l-bit numbers, $O(l^{1.58})$ for the multiplication of l-numbers with Karatsuba's algorithm and $O(l \log l / \log \log l)$ for the Schönhage–Strassen algorithm. When algorithm XOR-B is implemented in a hardware supporting multiplication of l-bit numbers, the time of multiplication can be the same as for the elementary operation. Consequently, the complexity of Equation (28) can be reduced to $O(l)$. It is the same as the complexity of hash algorithms with the same word length ($l = 32$ for SHA-1 and SHA-256/224 and $l = 64$ for SHA-512/384).

The total computational effort depends on both l and parameter R. Parameter R does not depend on l and is assumed to be fixed value for algorithm XOR-B. Computational effort determines a total time of computations and is usually measured in clock cycles necessary to execute an algorithm. In our case, it is production of a single number z_n. It requires: R operations XOR defined by Equation (28), one load of l-bit word, one computation of $j = n \bmod L$, one truncation, and one substitution $T[j] := x$. The detailed comparison of XOR-B with SHA-1 and SHA-2 is reliable when the same software and hardware platform is used for all algorithms. We limit our considerations to general comment, because it is not a subject of this paper. For very large R, greater than 80 rounds for SHA-1, SHA-512/384 and 64 rounds for SHA-256/224, the algorithm XOR-B can theoretically be slower. However, the high value of R indicates that some numbers x_n occur with very small probability. There is no proof that SHA-1 and SHA-2 always provide uniform distribution at the output for biased raw sequences [29].

Example

Theorem 1 indicates that the simplest method for increasing the entropy of a sequence is to group its elements into blocks with $R+1$ elements and compute the bitwise exclusive-or sum of all of the elements for each block. This method is numerically inefficient because to obtain an N-element sequence with a greater entropy, a random source S_X must produce $N(R+1)$ symbols x_k. Using an auxiliary table and the symbols already produced by source S_X, the production of the N-element sequence with theoretically the same entropy requires emitting $N+L$ symbols, where $L = R \cdot 2^m$, $m = \lceil \log_2 R \rceil$. For example, if $N = 1000$ and $R = 10$, then the source S_X must produce 11,000 symbols for the first method and 1160 symbols when algorithm XOR-B is used.

To compare different methods and assess the uniformity of the distribution of numbers obtained using the proposed theorem, a pseudorandom number generator with a Gaussian distribution of generated numbers was used. The Gaussian distribution occurs in many physical phenomena and plays a key role in the Shannon model of communication. As a source of independent numbers, we used a pseudorandom number generator from the *Mathematica* package [30]. The instruction $\mathit{NormalDistribution}[\mu, \sigma]$ provides numbers with a Gaussian distribution with an average μ and standard deviation σ. Because the obtained numbers are both positive and negative, we first compute their absolute values. Next, we consider numbers from the interval encoded by $l = l_{\text{int}} + l_{frac}$ bits, where l_{int} is the number of bits that encode the integer part and l_{frac} is the number of bits that encode the fractional part. If a generated number does not belong to the interval $[2^{-l_{frac}}, 2^{l_{\text{int}}} - 1]$ (generated numbers can, theoretically, assume values from minus to plus infinity), then we repeat the generation to obtain a number from this interval. The output is the bitwise exclusive-or sum of the $R+1$ numbers $t_0, t_1, \ldots, t_R$, for which the values belong to $[2^{-l_{frac}}, 2^{l_{\text{int}}} - 1]$. The distance between the addresses of $t_0, t_1, \ldots, t_R$ changes randomly for each iteration. This additional randomness comes from the higher-order bits of the output number z_n, excluding the first iteration, when it is fixed and equal to unity. In the experiment, l_{frac} is fixed and equal to 16 bits. The value of l_{int} is changed to obey the three exemplary ranges of the Gaussian distribution: ± 3, ± 5, and ± 7. Consequently, $l_{\text{int}} = 2$ for the range ± 3, and $l_{\text{int}} = 3$ for the ranges ± 5 and ± 7. It was also assumed that $\mu = 1$ and $\sigma = 2$. The size of table L is equal to $L = R \cdot 2^m$, where $m = \lceil \log_2 R \rceil$.

To assess the uniformity of the obtained distributions, a chi-square test was used. This test measures the agreement between the empirical distribution and the theoretical distribution for a given degree of freedom (number of categories) and significance level. The value of the statistic χ^2 is computed using the formula [18,19]

$$\chi^2 = \sum_i^H \frac{\left(N_i - N \cdot P_i\right)^2}{N \cdot P_i} \tag{31}$$

where H is the number of categories, N_i is the number of samples from the i-th category, P_i is the probability that each sample falls into category i, and N is the total number of samples. The critical value χ_c of the chi-square test for 50 categories and the significance level $\beta = 0.01$, chosen in the numerical experiment, is equal to 74.92 (we do not assess the parameters of the distribution). The results of the tests for $N = 10^5$, ranges ± 3, ± 5, ± 7, and the two methods of producing uniformly distributed random numbers are summarized in Tables 1–3.

The results shown in Table 1 indicate that we require only four numbers from the interval ±3 to form uniformly distributed numbers from the numbers with Gaussian distributions with $\mu = 1$ and $\sigma = 2$ for both methods. When the probability of producing certain numbers from $[2^{-l_{frac}}, 2^{l_{\text{int}}} - 1]$ decreases, the parameter R increases.

Table 1. The values of the statistic χ^2 for the range ± 3.

Grouping Subsequent Elements into $R+1$ Element Blocks						
R	1	2	3	4	5	6
χ^2	**226.41**	**76.42**	55.28	39.59	53.14	51.69
Algorithm XOR-B						
R	1 (m = 0, L = 1)	2 (m = 1, L = 4)	3 (m = 2, L = 12)	4 (m = 2, L = 16)	5 (m = 3, L = 40)	6 (m = 3, L = 48)
χ^2	**219.81**	**97.13**	37.19	37.52	45.13	34.85

Table 2. The values of the statistic χ^2 for the range ± 5.

Grouping Subsequent Elements into $R+1$ Element Blocks						
R	4	5	6	7	8	9
χ^2	**109.98**	**85.61**	65.79	21.93	33.99	48.16
Algorithm XOR-B						
R	4 (m = 2, L = 16)	5 (m = 3, L = 49)	6 (m = 3, L = 48)	7 (m = 3, L = 56)	8 (m = 3, L = 128)	9 (m = 4, L = 144)
χ^2	**173.56**	**86.24**	54.96	41.76	38.86	51.23

Table 3. The values of the statistic χ^2 for the range ± 7.

Grouping Subsequent Elements into $R+1$ Element Blocks						
R	5	10	15	20	25	30
χ^2	**4400.60**	**383.53**	**80.57**	50.69	51.81	48.97
Algorithm XOR-B						
R	5 (m = 3, L = 40)	10 (m = 4, L = 160)	15 (m = 4, L = 240)	20 (m = 5, L = 640)	25 (m = 5, L = 800)	30 (m = 5, L = 960)
χ^2	**3589.55**	**362.47**	**75.39**	64.20	50.29	53.58

Considering the numbers from the interval ± 5, we must sum the bitwise exclusive-or for at least seven numbers to pass the chi-square test. Exploiting the numbers from the tail of the Gaussian distribution, we need 21 or more numbers from the interval ± 7 for both methods. The use of algorithm XOR-B requires significantly fewer numbers x_k to produce an N element output sequence than grouping subsequent x_k into $R+1$ element blocks.

4. Conclusions

In this paper, a novel method for increasing the entropy of a sequence of discrete random variables has been proposed. The elements of this sequence are a bitwise exclusive-or sum of independent random variables that model the output of a random source. The proposed algorithms use an auxiliary table and a novel theorem proved in this paper. The theorem and the algorithms provide an efficient method for converting a sequence of independent random variables with a practically arbitrary distribution into a sequence of independent and uniformly distributed random variables, *i.e.*, with entropy close to maximal. The algorithm XOR-B does not need an additional circuit and requires only $1+L/N$ input words per one output word. Grouping the subsequent words into $R+1$ element blocks requires $R+1$ input words per one output symbol.

The areas for applications include dynamical systems, especially chaotic systems as a source of random numbers, chaos-based cryptography, traditional cryptography, statistics, simulation experiments, and many others fields in which random sequences with entropy close to the maximal value are needed.

Acknowledgments

The author wishes to thank Jakub Nikonowicz for the valuable discussion on the computational complexity of the proposed algorithms. The presented work was founded by the Polish Ministry of Science and Higher Education within the status activity task “08/83/DSPB/4707” in 2014.

Conflicts of Interest

The author declares no conflict of interest.

References

1. Ding, C.; Helleseth, T.; Niederreiter, H. *Sequences and Their Applications: Proceedings of SETA'98*; Springer; London, UK, 1999.
2. Von Neumann, J. Various techniques for use in connection with random digits. In *Von Neumann's Collected Works, Vol. 5*; Pergamon Press: Oxford, UK, 1962; pp. 768–770.
3. Davies, R. Exclusive OR (XOR) and Hardware Random Number Generators, 2002. Available online: http://www.robertnz.net (accessed on 12 October 2015).
4. Tkacik, T.E. A hardware random number generator. In *Cryptographic Hardware and Embedded Systems (CHES)*; Kaliski, B.S., Jr., Koç, Ç.K., Paar, C., Eds.; Springer: London, UK, 2002; pp. 450–453.

5. Sunar, B.; Martin, W.J.; Stinson, D.R. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Trans. Comput.* **2007**, *56*, 109–119.
6. Sunar, B. True random number generators for cryptography. In *Cryptographic Engineering*; Koç, Ç.K., Ed.; Springer: New York, NY, USA, 2009; pp. 55–73.
7. Golić, J.D. New methods for digital generation and postprocessing of random data. *IEEE Trans. Comput.* **2006**, *55*, 1217–1229.
8. Shaltiel, R. How to get more mileage from randomness extractors. *Random Struct. Algorithms* **2008**, *33*, 157–187.
9. Lacharme, P. Post processing functions for a biased physical random number generator. In *Fast Software Encryption (FSE'08)*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5086, pp. 334–342.
10. Lacharme, P. Analysis and construction of correctors. *IEEE Trans. Inf. Theory* **2009**, *55*, 4742–4748.
11. Via Technologies, Inc., VIA PadLock Security Engine—Technology Brief, 2005. Available online: http://www.via.com.tw/en/downloads/whitepapers/initiatives/padlock/VIAPadLockSecurityEngine.pdf (accessed on 11 March 2015).
12. Billingsley, P. *Probability and Measure*; Wiley: New York, NY, USA, 1979.
13. Rosenthal, J.F. *A First Look at Rigorous Probability Theory*; World Scientific: Singapore, Singapore, 2006.
14. Roth, S.M. *Introduction to Probability Models*, 7th ed.; Harcourt Academic Press: San Diego, CA, USA, 2000.
15. Gubner, J.A. *Probability and Random Processes for Electrical and Computer Engineers*; Cambridge University Press: Cambridge, UK, 2006.
16. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2006.
17. MacLaren, M.D.; Marsaglia, G. Uniform random number generators. *JACM* **1965**, *12*, 83–89.
18. Knuth, D.E. *The Art of Computer Programming*, 3rd ed.; Addison Wesley: Reading, MA, USA, 1998; Volume 2.
19. Gentle, J.E. *Random Number Generation and Monte Carlo Methods*; Springer: New York, NY, USA, 2003.
20. Deng, L.-Y.; George, E.O. Generation of uniform variates from several nearly uniformly distributed variables. *Commun. Stat.* **1990**, *19*, 145–154.
21. Deng, L.-Y.; Lin, D.K.J.; Wang, J.; Yuan, Y. Statistical justification of combination generators. *Am. Stat.* **1997**, *54*, 145–150.
22. Marsaglia, G. A current view of random number generators. In *Computer Science and Statistics: 16th Symposium on the Interface*; Billard, L., Ed., Elsevier: North-Holland, The Netherlands, 1985; pp. 3–10.
23. Menezes, A.J.; van Oorschot, P.C.; Vanstone, S.C. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1997.
24. Azad, S.; al-Sakib Khan, P. *Practical Cryptography, Algorithms and Implementations Using C++*; CRC Press: Boca Raton, FL, USA, 2015.

25. National Institute of Standards and Technology. Available online: http://csrc.nist.gov/groups/ST/toolkit/rng/references.html (accessed on 12 October 2015).
26. Rabin, M.O. Complexity of Computations. *Commun. ACM* **1977**, *20*, 625–633.
27. Knuth, D.E. *The Art of Computer Programming*, 3rd ed.; Addison Wesley: Reading, MA, USA, 1998; Volume 1.
28. Arora, S.; Barak, B. *Computational Complexity. A Modern Approach*, 3rd ed.; Cambridge University Press: Cambridge, UK, 2010.
29. Wayne, M.A.; Kwiat, P.G. Low-bias high speed quantum random number generator via shaped optical pulses. *Opt. Express* **2010**, *18*, 9351–9357.
30. Ruskeepää, H. *Mathematica Navigator*, 3rd ed.; Academic Press: Amsterdam, The Netherlands, 2006.