entropy

MDPI

Article

Numerical and Non-Asymptotic Analysis of Elias's and Peres's Extractors with Finite Input Sequences †

Amonrat Prasitsupparote [1,*], Norio Konno [2] and Junji Shikata [1]

1 Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama 240-8501, Japan; shikata@ynu.ac.jp
2 Department of Applied Mathematics, Faculty of Engineering, Yokohama National University, Yokohama 240-8501, Japan; konno-norio-bt@ynu.ac.jp
* Correspondence: amonrat-prasitsupparote-zp@ynu.jp

† This paper is an extended version of our paper published in 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, 22–24 March 2017.

Received: 29 July 2018; Accepted: 19 September 2018; Published: 23 September 2018

Abstract: Many cryptographic systems require random numbers, and the use of weak random numbers leads to insecure systems. In the modern world, there are several techniques for generating random numbers, of which the most fundamental and important methods are deterministic extractors proposed by von Neumann, Elias, and Peres. Elias's extractor achieves the optimal rate (i.e., information-theoretic upper bound) $h(p)$ if the block size tends to infinity, where $h(\cdot)$ is the binary entropy function and p is the probability that each bit of input sequences occurs. Peres's extractor achieves the optimal rate $h(p)$ if the length of the input and the number of iterations tend to infinity. Previous research related to both extractors has made no reference to practical aspects including running time and memory size with finite input sequences. In this paper, based on some heuristics, we derive a lower bound on the maximum redundancy of Peres's extractor, and we show that Elias's extractor is better than Peres's extractor in terms of the maximum redundancy (or the rates) if we do not pay attention to the time complexity or space complexity. In addition, we perform numerical and non-asymptotic analysis of both extractors with a finite input sequence with any biased probability under the same environments. To do so, we implemented both extractors on a general PC and simple environments. Our empirical results show that Peres's extractor is much better than Elias's extractor for given finite input sequences under a very similar running time. As a consequence, Peres's extractor would be more suitable to generate uniformly random sequences in practice in applications such as cryptographic systems.

Keywords: true random number generation; von Neumann's extractor; Peres's extractor; Elias's extractor

1. Introduction

Many cryptographic systems require random numbers, and the use of weak random numbers leads to insecure systems. In fact, many past security problems were due to the use of weak random numbers [1–4]. This tells us that random number generation is very important in cryptography, in particular to ensure that secret keys are random and unpredictable. In the modern world, there are several techniques for generating random numbers. A natural source such as physical phenomena, the stock market, or Bitcoin [5] can produce unpredictable random sequences, although such sequences are not uniformly random at the source (i.e., biased). However, there is a solution to solve this problem, namely, to use deterministic extractors. A deterministic extractor is a function which takes a non-uniformly random sequence as input and outputs a uniformly random sequence.

The deterministic extractors have been studied in mathematics, information theory, and cryptography. In information theory, those extractors can also be treated for the intrinsic randomness problem (i.e., the problem of generating truly random numbers). Furthermore, as applications in cryptography, the output sequence of those extractors can be used as secret keys in information-theoretic cryptography or symmetric key cryptography. The extractors by von Neumann [6], Elias [7], and Peres [8] are fundamental and important ones. In particular, Elias's and Peres's extractors are interesting, since they can achieve the optimal rate (or redundancy), if we suppose that input size tends to infinity (i.e., from an asymptotic viewpoint). However, it is not easy to conclude which one is better, since those are constructed by completely different approaches. The main purpose of this paper is to investigate those with finite inputs (i.e., from a non-asymptotic viewpoint) by numerical analysis to make it clear which is better for practical use.

1.1. Related Work

Several works have proposed methods for extracting uniform random sequences from non-uniform random sequences. The most famous among them is von Neumann's extractor [6] proposed in 1951. He demonstrated a simple procedure for extracting independent unbiased bits from a sequence of independent, identically distributed (i.i.d.) and biased bits. The technique by von Neumann guarantees that the output sequences are independent and uniform if the input sequence is independent and constantly biased, while this cannot be guaranteed if the bias is not constant (e.g., see [9]).

An improved algorithm of von Neumann's extractor was proposed by Elias [7] in 1971. Elias's extractor utilizes a block coding technique to improve the rate (or redundancy) of von Neumann's extractor; however, the straightforward implementation of this extractor requires exponential time and exponential memory size with respect to N, where N is the block size, to store all 2^N input sequences with their assignment of output sequences. In 2000, Ryabko and Matchikina [10] proposed an extension of Elias's extractor that improved time complexity and space complexity by using the enumerative encoding technique from [11] and the Schönhage–Strassen algorithm [12] for fast integer multiplication in order to compute the assignment of output sequences. In this paper, we call this improved method the RM method.

Peres's extractor is another extended algorithm of von Neumann's extractor. In 1992, Peres [8] proposed a procedure which improved upon von Neumann's extractor. The basic idea of Peres's extractor is to reuse the discarded bits in von Neumann's extractor by iterating similar procedures in von Neumann's extractor.

The extractors by von Neumann, Elias, and Peres are the most fundamental and important ones using a single source. In particular, Elias's and Peres's extractors are interesting, since they can achieve the optimal rate (i.e., information-theoretic upper bound) $h(p)$ if input size tends to infinity (i.e., in an asymptotic case), where each bit of input sequences from a single source occurs with probability $p \in (0, 1)$ and $h(\cdot)$ is the binary entropy function. In this paper, we are interested in the non-asymptotic case, namely, the achievable rate for finite input-sizes. The rate of Elias's extractor for finite input-sizes can be observed in the work [7], but the rate of Peres's extractor for finite input-sizes is not explicitly known. As a work related to Peres's extractor, Pae [13] reported a recursion formula to compute the rate for finite input-sizes, but it is difficult to give the rate function with finite input-sizes since the recursion formula is complicated. Pae also computed the rate by the recursion formula in the case $p = 1/3$, compared the rates of Peres's extractor and Elias's extractor, and concluded that the rate of Peres's extractor increased much slower than that of Elias's extractor via numerical analysis. However, it is not explicitly known which extractor is better to use in practice, if we take into account the running time, implementation cost, and memory size required in the extractors, as mentioned in [13].

There are several works for constructing extractors using multiple sources (i.e., not a single source). Bourgain [14] provided a 2-source extractor under the condition that the two sources are independent and each source has min-entropy $0.499n$, where n is the bit-length of the output of the sources. Raz [15] proposed an improvement in terms of total min-entropy, and constructed 2-source extractors with the condition that one source has min-entropy more than $n/2$ and the other source requires min-entropy $O(\log n)$. In 2015, Cohen [16] constructed a 3-source extractor, where one source has min-entropy δn, the second source has min-entropy $O(\log n)$ and the third source has min-entropy $O(\log \log n)$. In 2016, Chattopadhyay and Zuckerman [17] proposed a general 2-source extractor, where each source has a polylogarithmic min-entropy. They combined two weak random sequences into a single sequence by using K-Ramsey graphs and resilient functions. Their extractor has only one-bit output and achieves negligible error and higher complexity than Peres's extractor or Elias's extractor.

Furthermore, there are various reports about extracting random bits in the real world. In particular, in 2009, Bouda et al. [18] used mobile phones or pocket computers to generate random data that is close to truly random data. They took 12 pictures per second then used their function to obtain four random bits in each picture, and then applied Carter–Wegman universal$_2$ hash functions. Halprin and Naor [19] presented the idea of using human game-play as a randomness source in 2009. They constructed the Hide and Seek game that produced approximately 17 bits of raw data per click, and then generated with a pairwise independent hash function a 128-bit string which is 2^{64}-close to the random one in less than two minutes. In 2011, Voris et al. [20] investigated the use of accelerators on the RFID tags as a source. They implemented a two-stage extractor on the RFID tags. It can produce 128 random bits in 1.5 s by storing a Toeplitz matrix on the RFID tags and performing matrix multiplications.

1.2. Our Contribution

In this paper, we revisit the extractors by von Neumann, Elias, and Peres, since they are fundamental and only require a single source. In the studies on those extractors, it is normal to asymptotically analyze the rate or redundancy of the extractors in the literature, where the rate is the average bit-length of outputs per bit of input (see Section 2 for details). Specifically, the rate of von Neumann's extractor is $p(1-p)$ that is far from the optimal rate (i.e., information-theoretic upper bound) $h(p)$. Meanwhile, the rate of Elias's extractor converges to $h(p)$ if the block size tends to infinity. Specifically, Elias's extractor outputs a uniformly random sequence at a high rate, when it takes a long block size equal to the input length. However, it has a trade-off between the rates and computational resources such as time complexity and memory size. On the other hand, Peres's extractor achieves the optimal rate $h(p)$ if the length of input and the number of iterations tend to infinity, and it requires smaller time complexity and memory size. However, it would be hard to explicitly derive the exact rate for finite input sequences. Thus, it is not easy to conclude which is a more suitable extractor for practical use in general. Among related work, only one, by Pae [13], compared both extractors as mentioned in Section 1.1, but it does not completely answer the question, since it analyzed the performance of both extractors only for restricted parameters, in particular, the case where each bit of input sequences occurs with probability $p = 1/3$ and did not consider the running time. In this paper, we will perform non-asymptotic analysis for the wide range of parameters for Elias's and Peres's extractors, to answer the following question: which is more suitable for practical use in real-world applications? To do this, we evaluate the numerical performance of Peres's extractor and Elias's extractor with the RM method in terms of practical aspects including achievable rates (or redundancy) and running time with finite input sequences. Specifically, the contribution of this paper is as follows:

(i) Based on some heuristics, we derive a lower bound on the maximum redundancy of Peres's extractor in Section 3. This result shows that the maximum redundancy of Elias's extractor is superior to Peres's extractor in general, if we focus only on redundancy (or rates) and we do not pay attention to the time complexity or space complexity.

(ii) By numerical analysis, we design our experiments by comparing both extractors with finite input sequences of which each bit occurs with any biased probability $p \in (0,1)$ under the same environments in terms of practical aspects. Both extractors are implemented on a general PC and do not require any special resources, libraries, or frameworks for computation.

Our implementation and results will be explained in Section 4. We calibrate our implementation by comparing the theoretical and experimental redundancy of both extractors. Afterwards, we analyze the time complexity of both extractors with respect to the bit-length of input sequences from 100 to 5000. We compare the redundancy of both extractors, and our implementation shows that Peres's extractor is much better than Elias's extractor under a very similar running time. As a result, Peres's extractor would be more suitable for generating uniformly random sequences for practical use in applications.

The primary version of this paper appeared in CISS2017 [21], and this paper is an extended and full version of it. The difference between the primary version [21] and this paper is as follows: This paper contains the above result (i) in Section 3, and reports more detailed implementation results for (ii) in Section 4. In particular, we implemented and confirmed the results (ii) at a larger scale (see Section 4 for details) in addition to obtaining new figures in Section 4.1.

2. Preliminaries

Throughout this paper, we assume that $\log(\cdot) := \log_2(\cdot)$ and $\ln(\cdot) := \log_e(\cdot)$, and we define $0 \log 0 := 0$. $h(\cdot)$ is the binary entropy function defined by $h(p) = -p \log p - (1-p)\log(1-p)$ for $p \in [0,1]$. Let $\binom{n}{k}$ be a binomial coefficient defined by $\binom{n}{k} := \frac{n(n-1)(n-2)\cdots(n-k+1)}{k(k-1)(k-2)\cdots 1}$ for nonnegative integers n and k, and $\binom{n}{0} := 1$ for any $n \geq 0$ (see [22] for an extension of the traditional definition of binomial coefficients). Note that $\binom{n}{k} > 0$ if $k \leq n$, and $\binom{n}{k} = 0$ if $k > n$.

The first deterministic extractor was constructed by von Neumann [6] in 1951, and later improved ones were proposed by Elias [7] in 1971, and by Peres [8] in 1992. The prior work [6–8] considered Bernoulli source $\mathsf{Bern}(p)$ from which input sequences were generated, namely $\mathsf{Bern}(p)$ outputs i.i.d. $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ according to $\Pr(x_i = 1) = p$ and $\Pr(x_i = 0) = q = 1 - p$ for some unknown $p \in (0,1)$.

A deterministic extractor A takes $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ as input and outputs $(y_1, y_2, \ldots, y_\ell) \in \{0,1\}^\ell$, and its average bit-length of output is denoted by $\bar{\ell}(n)$ which is a function of n, and defines its rate function by $r^{\mathsf{A}}(p) := \lim_{n\to\infty} \bar{\ell}(n)/n$. Additionally, for a deterministic extractor A, we define the redundancy function by $f^{\mathsf{A}}(p) := h(p) - r^{\mathsf{A}}(p)$, and the maximum redundancy by $\Gamma := \sup_{p\in(0,1)} f^{\mathsf{A}}(p)$. Note that the above definition of redundancy functions is meaningful, since $h(p)$ is shown to be the information-theoretic upper bound of the extractors in [7,8]. Furthermore, in this paper we define a non-asymptotic rate function $r^{\mathsf{A}}(p,n) := \bar{\ell}(n)/n$, a non-asymptotic redundancy function $f^{\mathsf{A}}(p,n) := h(p) - r^{\mathsf{A}}(p,n)$, and the non-asymptotic maximum redundancy $\Gamma(n) := \sup_{p\in(0,1)} f^{\mathsf{A}}(p,n)$, which will be used in our non-asymptotic analysis.

2.1. Von Neumann's Extractor

Von Neumann's extractor was a simple algorithm for extracting independent unbiased bits from biased bits. This algorithm divides the input sequences $(x_1, x_2, x_3, x_4, \ldots, x_n)$ into the pairs (If n is odd, we discard the last bit.) $((x_1x_2), (x_3x_4), \ldots)$ and maps each pair with a mapping as follows:

$$00 \mapsto \wedge, \quad 01 \mapsto 0, \quad 10 \mapsto 1, \quad 11 \mapsto \wedge, \tag{1}$$

where $\wedge$ means no output was generated. After that, it concatenates all resulting outputs of (1). To facilitate understanding, we give an example as follows.

Example 1. *Suppose that an input sequence is* $(x_1, x_2, x_3, \ldots, x_8) = (1,0,0,1,0,0,1,1)$. *Firstly, divide it into the pairs as* $((1,0),(0,1),(0,0),(1,1))$. *Next, map each pair with the mapping* (1). *Finally, the extractor outputs* $(y_1, y_2) = (1,0)$.

Complexity: Von Neumann's extractor is efficient in the sense that both time complexity and space complexity are small such that time complexity is evaluated as $O(n)$, and space complexity is evaluated as $O(1)$.

Redundancy: Von Neumann's extractor is not desirable, since the maximum redundancy is far from zero. In fact, the rate function $r^{\mathsf{vN}}(p)$ of von Neumann's extractor is evaluated by $r^{\mathsf{vN}}(p) = \lim_{n\to\infty} np(1-p)/n = p(1-p)$, which is $1/4$ at $p = 1/2$ and less elsewhere. In addition, the (non-asymptotic) rate functions, (non-asymptotic) redundancy functions, and the (non-asymptotic) maximum redundancy is evaluated as follows: $f^{\mathsf{vN}}(p,n) = f^{\mathsf{vN}}(p) = h(p) - p(1-p)$, $\Gamma^{\mathsf{vN}}(n) = \Gamma^{\mathsf{vN}} = 3/4$.

2.2. Elias's Extractor

Elias [7] improved von Neumann's extractor by using a block coding technique in 1971. Let $N \in \mathbb{N}$ $(N \geq 2)$ be the block size in Elias's extractor. For all binary sequences with bit-length N, partition them into $N+1$ sets S_k $(k = 0,1,2,\dots,N)$, where S_k consists of all the $\binom{N}{k}$ sequences of length N which have k ones and $N-k$ zeros. Here, we note that each sequence of S_k is equiprobable, i.e., the probability $p^k q^{N-k}$.

We consider binary representation of the nonnegative integer $|S_k| = \binom{N}{k}$ as follows: $\binom{N}{k} = \alpha_m 2^m + \alpha_{m-1}2^{m-1} + ... + \alpha_0 2^0$, where $m = \lfloor \log \binom{N}{k} \rfloor$, $\alpha_j \in \{0,1\}$, and $\alpha_m = 1$. In this case, we briefly write $|S_k| = \binom{N}{k} = (\alpha_m, \alpha_{m-1}, \dots, \alpha_0)$. For each j $(1 \leq j \leq m)$ such that $\alpha_j = 1$, we assign 2^j distinct output sequences of length j to 2^j distinct sequences of S_k which have not already been assigned. If $\alpha_0 = 1$, one sequence of S_k is assigned to $\wedge$. In particular, since $|S_0| = |S_N| = 1$, two sequences $(0,0,\dots,0)$ and $(1,1,\dots,1)$ are assigned to $\wedge$. For instance, we show a procedure of Elias's extractor in Example 2.

Example 2. *Suppose that the given input sequence $x = (1,0,0,1,0,0,1,1)$ with block size $N = 4$, is the same as in Example 1. Firstly, we partition the set $\{0,1\}^4$ of possible input sequences into the following subsets:*

$$\begin{aligned}
S_0 &= \{(0,0,0,0)\},\\
S_1 &= \{(1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1)\},\\
S_2 &= \{(0,0,1,1),(0,1,0,1),(0,1,1,0),(1,1,0,0),(1,0,1,0),(1,0,0,1)\},\\
S_3 &= \{(1,1,1,0),(1,0,1,1),(1,1,0,1),(0,1,1,1)\},\\
S_4 &= \{(1,1,1,1)\}.
\end{aligned}$$

Then, we have $|S_0| = |S_4| = 1 = (1)$, $|S_1| = |S_3| = 4 = (1,0,0)$, $|S_2| = 6 = (1,1,0)$. We consider the following assignment of output sequences:

$$\begin{aligned}
&(0,0,0,0) \mapsto \wedge, && (1,1,1,1) \mapsto \wedge,\\
&(1,0,0,0) \mapsto (0,0), && (1,1,1,0) \mapsto (0,0),\\
&(0,1,0,0) \mapsto (0,1), && (1,0,1,1) \mapsto (1,0),\\
&(0,0,1,0) \mapsto (1,0), && (1,1,0,1) \mapsto (1,1),\\
&(0,0,0,1) \mapsto (1,1), && (0,1,1,1) \mapsto (0,1),\\
&(0,0,1,1) \mapsto (0,1), && (1,0,1,0) \mapsto (1,0),\\
&(0,1,1,0) \mapsto (0,0), && (1,0,0,1) \mapsto (1,1),\\
&(0,1,0,1) \mapsto (0), && (1,1,0,0) \mapsto (1).
\end{aligned}$$

Suppose that an input sequence $x = (1,0,0,1,0,0,1,1)$ is given. Since the block size $N = 4$, the sequence is divided as $x = ((1,0,0,1),(0,0,1,1))$. By the above assignment of output sequences, the output sequence is $y = ((1,1)(0,1)) = (1,1,0,1)$. Furthermore, there are several ways to assign

m_k bits to binary output sequences with the same probability that affect the output sequence y. Thus, the output sequence of 10010011 will not be 1101, if we use another assignment. Note that Elias's extractor with block size $N = 2$ is equivalent to von Neumann's extractor, or equivalently the mapping (1). In this sense, Elias's extractor is an extension of von Neumann's extractor.

Complexity: It can be seen that the straightforward implementation of Elias's extractor requires much space and time complexity to make a table of the assignment of output sequences, as illustrated by Example 2. Specifically, it requires exponential time and exponential memory size with respect to N to store all 2^N binary sequences with their assignment of output sequences. To reduce the time and space complexity of Elias's extractor, Ryabko and Matchikina [10] proposed a method that is extended from Elias's extractor, which we call the RM method in this paper. The RM method utilizes the enumerative encoding technique from [11] and the Schönhage–Strassen algorithm [12] for fast integer multiplication in order to compute the assignment of output sequences without making the table large. The procedure of the RM method is described as follows.

Firstly, suppose that a binary input sequence $x^N = (x_1, x_2, \ldots, x_N)$ contains k ones and $N - k$ zeros. Let $Num(x^N) \geq 0$ be a number which is defined by x^N depending on the lexicographical order set S_k. Namely, if x^N has k ones, then the number $\mathrm{Num}(x^N) \geq 0$ is defined by

$$\mathrm{Num}(x^N) = \sum_{t=1}^{N} \binom{x_t N - t}{k - \sum_{i=1}^{t-1} x_i}, \tag{2}$$

where the summation is taken over all $1 \leq t \leq N$ such that $x_t = 1$, and $\mathrm{Num}(0^N) := 0$. Then, we calculate a binary codeword $code(x^N)$ of x^N, which is the assignment of an output sequence of x^N as follows:

(i) Compute $\mathrm{Num}(x^N)$ in the set S_k, if x^N contains k ones.
(ii) Let $|S_k| = \binom{N}{k} = 2^{j_0} + 2^{j_1} + \ldots + 2^{j_m}$ for $0 \leq j_0 < j_1 < \ldots < j_m$.
(iii) If $j_0 = 0$ and $\mathrm{Num}(x^N) = 0$, then $code(x^N) = \wedge$.
(iv) If $0 \leq \mathrm{Num}(x^N) < 2^{j_0}$, then $code(x^N)$ is defined to be the j_0 low-order binary string of $\mathrm{Num}(x^N)$.
(v) If $\sum_{s=0}^{t} 2^{j_s} \leq \mathrm{Num}(x^N) < \sum_{s=0}^{t} 2^{j_s} + 2^{j_{t+1}}$ for some $0 \leq t \leq m$, then $code(x^N)$ is defined to be the suffix consisting of the j_{t+1} binary string of $\mathrm{Num}(x^N)$.

Example 3. *Suppose that the block size $N = 4$, and the given input sequence is $x = (1,0,0,1,0,0,1,1)$, which is the same as all previous examples. After that, the sequence is divided as $x = ((1,0,0,1),(0,0,1,1))$. Next, compute $\mathrm{Num}(x^N)$ following Equation (2):*

$$\begin{aligned} \mathrm{Num}((1,0,0,1)) &= \binom{4-1}{2} + \binom{4-4}{2-1} = 3, \\ \mathrm{Num}((0,0,1,1)) &= \binom{4-3}{2} + \binom{4-4}{2-1} = 0. \end{aligned}$$

Then, the RM method computes $code(1,0,0,1) = (1,1)$ and $code(0,0,1,1) = (0)$. Finally, it outputs $y = (1,1,0)$ by concatenating $code(1,0,0,1)$ and $code(0,0,1,1)$.

The time and space complexity of Elias's extractor with the RM method are $O(N \log^3 N \log\log N)$ and $O(N \log^2 N)$, respectively (see [10] for details).

Redundancy: Generally, the rate function and redundancy function of Elias's extractor depend on block size N. For a given n-bit input sequence, if we take the block size equal to the length of input sequence $N := n$, the rate function (or redundancy) achieve the best value. For simplicity, we assume that $N = n$ in the following explanation. Then, the rate function $r^{\mathsf{E}}(p,n)$ is evaluated by

$$r^{\mathsf{E}}(p,n) \approx \frac{1}{n}\sum_{k=0}^{n}\binom{n}{k}p^k(1-p)^{n-k}\log\binom{n}{k}. \tag{3}$$

Elias's extractor takes i.i.d. with non-uniform distribution as the input, and it will output i.i.d. with uniform distribution such that its rate is given by Equation (3). Elias [7] showed that the rate function $r^{\mathsf{E}}(p,n)$ of Elias's extractor converges to $h(p)$ as $n \to \infty$, or equivalently, the redundancy function $f^{\mathsf{E}}(p,n) := h(p) - r^{\mathsf{E}}(p,n)$ converges to zero as $n \to \infty$. More precisely, it was shown that $f^{\mathsf{E}}(p,n) = O(1/n)$ for any fixed p. Therefore, for a given n-bit input sequence, if we set the maximum block size to be the input size, the non-asymptotic maximum redundancy $\Gamma^{\mathsf{E}}(n)$ converges to zero not slower than $1/n$.

2.3. Peres's Extractor

Peres's extractor is another method that improved the rates (or redundancy) of von Neumann's extractor. The basic idea behind Peres's extractor is to reuse the discarded bits in the mapping (1). In the following, we denote von Neumann's extractor by Ψ_1. For an n-bit sequence $(x_1, x_2, \ldots, x_n)$, we describe von Neumann's extractor by $\Psi_1(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_\ell)$, where $y_i = x_{2m_i-1}$ and $m_1 < m_2 < \cdots < m_\ell$ are all the indices satisfying $x_{2m_i-1} \neq x_{2m_i}$ with $m_i \leq n/2$. In Peres's extractor, Ψ_ν ($\nu \geq 2$) is defined inductively as follows: For an even n,

$$\Psi_\nu(x_1, x_2, \ldots, x_n) = \Psi_1(x_1, x_2, \ldots, x_n) * \Psi_{\nu-1}(u_1, u_2, \ldots, u_{\frac{n}{2}}) * \Psi_{\nu-1}(v_1, v_2, \ldots, v_{\frac{n}{2}-\ell}), \tag{4}$$

where $*$ is concatenation; $u_j = x_{2j-1} \oplus x_{2j}$ for $1 \leq j \leq n/2$; $v_s = x_{2i_s-1}$ and $i_1 < i_2 < \cdots < i_{\frac{n}{2}-\ell}$ are all the indices satisfying $x_{2i_s-1} = x_{2i_s}$ with $i_s \leq n/2$. For an odd input size n, $\Psi_\nu(x_1, x_2, \ldots, x_n) := \Psi_\nu(x_1, x_2, \ldots, x_{n-1})$, i.e., the last bit is discarded and the above case of an even n is utilized.

Note that the number of iterations ν is at most $\lfloor \log n \rfloor$, since Ψ_ν for every $\nu \geq 2$ is defined by $\Psi_{\nu-1}$ having an input sequence whose bit-length is at most $n/2$, i.e., the bit-length of both $(u_1, u_2, \ldots, u_{\frac{n}{2}})$ and $(v_1, v_2, \ldots, v_{\frac{n}{2}-\ell})$ in Equation (4) is at most $n/2$. Obviously, Peres's extractor with $\nu = 1$ is the same as von Neumann's extractor. In addition, Peres's extractor with a large ν is considered to be an elegantly improved version of von Neumann's extractor by utilizing a recursion mechanism.

Example 4. *Suppose that an input sequence is given as $x = (1,0,0,1,0,0,1,1)$, which is the same as all previous examples. The number of iterations satisfies $\nu \leq \lfloor \log 8 \rfloor = 3$. Then, Peres's extractor is executed as follows:*

$$\begin{aligned}
\Psi_1(x) &= (1,0),\\
\Psi_2(x) &= \Psi_1(x) * \Psi_1(1,1,0,0) * \Psi_1(0,1) = (1,0,0),\\
\Psi_3(x) &= \Psi_1(x) * \Psi_2(1,1,0,0) * \Psi_2(0,1)\\
&= \Psi_1(x) * (\Psi_1(1,1,0,0) * \Psi_1(0,0) * \Psi_1(1,0)) * (\Psi_1(0,1) * \Psi_1(1))\\
&= (1,0,1,0).
\end{aligned}$$

Complexity: We denote the time complexity of Ψ_ν by $T_\nu(n)$. By Equation (4), we have

$$T_\nu(n) = T_1(n) + n/2 + T_{\nu-1}(n/2) + T_{\nu-1}(n/2 - \ell), \tag{5}$$

and $T_1(n) = O(n)$ (see Section 2.1 for the time complexity of von Neumann's extractor). From the condition (5), we obtain $T_\nu(n) = O(\nu n)$ for Ψ_ν with $1 \leq \nu \leq \lfloor \log n \rfloor$. In particular, the time complexity of Peres's extractor with the maximum iterations $\nu = \lfloor \log n \rfloor$ is evaluated as $T_\nu(n) = O(n \log n)$ and the space complexity is $O(1)$.

Redundancy: The rate function $r_\nu^{\mathsf{P}}(p)$ of Peres's extractor can be computed inductively by the equation

$$r_\nu^{\mathsf{P}}(p) \quad = pq + \tfrac{1}{2} r_{\nu-1}^{\mathsf{P}}(p^2+q^2) + \tfrac{1}{2}(p^2+q^2) r_{\nu-1}^{\mathsf{P}}\left(\tfrac{p^2}{p^2+q^2}\right) \tag{6}$$

for $\nu \geq 2$, and $r_1^{\mathsf{P}}(p) = pq$. Note that $r_1^{\mathsf{P}}(p)$ is the rate of von Neumann's extractor. Peres's extractor takes i.i.d. with non-uniform distribution as input, and it will output i.i.d. with uniform distribution such that its rate is given by Equation (6) if $n \to \infty$. It is shown in [8] that $r_\nu^{\mathsf{P}}(p) \leq r_{\nu+1}^{\mathsf{P}}(p)$ for all $\nu \in \mathbb{N}$, $p \in (0,1)$, and $\lim_{\nu \to \infty} r_\nu^{\mathsf{P}}(p) = h(p)$ uniformly in $p \in (0,1)$.

In other words, the above result is described in terms of redundancy as follows:

$$\begin{aligned} f_\nu^{\mathsf{P}}(p) &= h(p) - r_\nu^{\mathsf{P}}(p) \\ &= \frac{1}{2} f_{\nu-1}^{\mathsf{P}}(p^2+q^2) + \frac{1}{2}(p^2+q^2) f_{\nu-1}^{\mathsf{P}}\left(\frac{p^2}{p^2+q^2}\right) \end{aligned} \tag{7}$$

for $\nu \geq 2$ and $f_1^{\mathsf{P}}(p) = h(p) - p(1-p)$, where the above Equation (7) follows from Equation (6). Furthermore, it holds that $f_\nu^{\mathsf{P}}(p) \geq f_{\nu+1}^{\mathsf{P}}(p)$ for all $\nu \in \mathbb{N}$, $p \in (0,1)$, and $\lim_{\nu \to \infty} f_\nu^{\mathsf{P}}(p) = 0$ uniformly in $p \in (0,1)$. Suppose that we take the maximum $\nu = \lfloor \log n \rfloor$ and $n \to \infty$, and then, we have $\Gamma^{\mathsf{P}}(n) = o(1)$.

In Table 1, we summarize the redundancy, time complexity and space complexity (memory size) for von Neumann's, Elias's, and Peres's extractors.

Table 1. Comparison of extractors.

	Redundancy $\Gamma(n)$	**Time Complexity**	**Space Complexity**
von Neumann's extractor	$3/4$	$O(n)$	$O(1)$
Elias's extractor (with maximum block-size)	$O(1/n)$ (by [7])	$O(n \log^3 n \log\log n)$ (by [10])	$O(n \log^2 n)$ (by [10])
Peres's extractor (with maximum iterations)	$O(1)$ (by [8])	$O(n \log n)$ (by [8])	$O(1)$ (by [8])

3. Lower Bound on Redundancy of Peres's Extractor

Although it is shown that $\Gamma^{\mathsf{P}}(n) = o(1)$ in Peres's extractor (i.e., $\Gamma^{\mathsf{P}}(n)$ converges to zero as $n \to \infty$), it is not known whether $\Gamma^{\mathsf{P}}(n)$ converges to zero rapidly or slowly. To investigate it, we analyze the non-asymptotic redundancy function $f_\nu^{\mathsf{P}}(p,n)$ and non-asymptotic maximum redundancy $\Gamma^{\mathsf{P}}(n)$. In particular, we derive a lower bound on $\Gamma^{\mathsf{P}}(n)$ based on some heuristics.

Let $f_\nu^{\mathsf{P}}(p) = h(p) - r_\nu^{\mathsf{P}}(p)$ be the redundancy function for Peres's extractor with ν iterations. Then, we first show that $f_\nu^{\mathsf{P}}(p)$ is not concave in $p \in (0,1)$ for $\nu \geq 5$ as follows. The proof is given in Appendix A.

Proposition 1. *The redundancy function $f_\nu^{\mathsf{P}}(p)$ in Peres's extractor with ν iterations is not concave in $p \in (0,1)$ if $\nu \geq 5$. More generally, for Peres's extractor with ν iterations, the redundancy function $f_\nu^{\mathsf{P}}(p)$ satisfies*

$$\frac{d^2 f_\nu^{\mathsf{P}}(\frac{1}{2})}{dp^2} = 8 - \frac{4}{\ln 2} - 6\left(\frac{3}{4}\right)^{\nu-1}. \tag{8}$$

In particular, $\frac{d^2 f_\nu^{\mathsf{P}}}{dp^2}\left(\frac{1}{2}\right) < 0$ for $1 \leq \nu \leq 4$ and $\frac{d^2 f_\nu^{\mathsf{P}}}{dp^2}\left(\frac{1}{2}\right) > 0$ for $\nu \geq 5$.

Here, we assume that the following proposition, Proposition 2, holds true. Although it is not easy to provide proof, it seems to be true from our experimental results that are provided in Appendix B. In Figure A1 in Appendix B, we depict the difference values $f_\nu^{\mathsf{P}}(p,n) - f_\nu^{\mathsf{P}}(p)$ with input bit-length $n = 80, 100, ..., 200$ and iterations $1 \le \nu \le \lfloor \log n \rfloor$. We note that Proposition 2 states that $f_{\lfloor \log n \rfloor}^{\mathsf{P}}(p,n) - f_{\lfloor \log n \rfloor}^{\mathsf{P}}(p) \ge 0$ for $p \in (0,1)$, and we can observe that it holds true for input bit-length $n = 80, 100, ..., 200$ by our experimental results given in Figure A1.

Proposition 2 (Heuristics). *Suppose $\nu = \lfloor \log n \rfloor$. Then, we have $f_\nu^{\mathsf{P}}(p,n) \ge f_\nu^{\mathsf{P}}(p)$, or equivalently $r_\nu^{\mathsf{P}}(p,n) \le r_\nu^{\mathsf{P}}(p)$, for a sufficiently large n and any $p \in (0,1)$.*

The following theorem shows a lower bound on $\Gamma^{\mathsf{P}}(n)$ that is derived based on Proposition 2.

Theorem 1. *Suppose that Proposition 2 holds true. Then, in Peres's extractor with the maximum iterations $\nu = \lfloor \log n \rfloor$, we have $\Gamma^{\mathsf{P}}(n) > 1/n^{2-\log 3}$. In particular, $\Gamma^{\mathsf{P}}(n) = \omega(1/n)$.*

Proof. Let n be a large natural number. For a natural number $\nu \in \mathbb{N}$ with $1 \le \nu \le \log n$, we define $a_\nu := r_\nu(1/2)$. Then, by Equation (6) we have

$$a_1 = \frac{1}{4}, \qquad a_\nu = \frac{1}{4} + \frac{3}{4} a_{\nu-1} \text{ for } \nu \ge 2.$$

By solving the above equation, we have

$$a_\nu = 1 - \left(\frac{3}{4}\right)^\nu \text{ for } \nu \ge 1. \tag{9}$$

Thus, for $\nu = \lfloor \log n \rfloor$, we obtain

$$\begin{aligned} f_\nu^{\mathsf{P}}(1/2, n) &\ge f_\nu^{\mathsf{P}}(1/2) \quad (10)\\ &= (3/4)^\nu \quad (11)\\ &\ge (3/4)^{\log n}\\ &= \frac{1}{n^{2-\log 3}}, \end{aligned}$$

where the inequality (10) follows from Proposition 2, and the equality (11) follows from (9).

Therefore, we have

$$\begin{aligned} \Gamma^{\mathsf{P}}(n) &= \sup_{p \in (0,1)} f_{\lfloor \log n \rfloor}^{\mathsf{P}}(p,n)\\ &> f_{\lfloor \log n \rfloor}^{\mathsf{P}}(\frac{1}{2}, n) \quad (12)\\ &\ge \frac{1}{n^{2-\log 3}}, \end{aligned}$$

where the inequality (12) follows from Proposition 1. □

Theorem 1 shows that the non-asymptotic maximum redundancy $\Gamma^{\mathsf{P}}(n)$ does converge to zero slower than $1/n$. This means that Peres's extractor is worse than Elias's extractor in terms of the maximum redundancy, since $\Gamma^{\mathsf{E}}(n) = O(1/n)$ if block size is set to be n. However, this result does not always mean that Peres's extractor is worse than Elias's extractor, since the time complexity and space complexity of Peres's extractor are better than those of Elias's extractor, as shown in Table 1. In this sense, it is not easy to conclude which extractor is superior. In the next section, from a viewpoint of practicality including running time, we compare both extractors and show that Peres's extractor is better than Elias's extractor by numerical analysis with various parameters.

4. Implementation and Numerical Analysis

In this section, we describe our experimental results of Peres's extractor and Elias's extractor with the RM method. We used Java language version 1.8 to implement both extractors and evaluated the performance on a desktop PC with Intel Core i3 3.70 GHz and 4 GB of RAM. Our experiments could also be performed on a general PC and do not require any special resources, libraries, or frameworks for computation. In fact, we can use other languages instead of Java language, but Java language can evaluate it on every platform without any support software. Thereby, we used Java language for implementation. To compare Peres's extractor and Elias's extractor with the RM method with finite input sequences in terms of non-asymptotic viewpoints, we consider the following four questions.

(1) Is theoretical redundancy the same as experimental redundancy in both extractors?
(2) Is the experimental redundancy of Elias's extractor with the RM method better than the experimental redundancy of Peres's extractor?
(3) What is the exact running time of both extractors?
(4) Which extractor achieves better redundancy (or rate) under the very similar running time?

To answer the above questions, we design our experiments as follows.

To answer the Questions (1) and (2), we evaluate the theoretical and experimental redundancy of Peres's extractor and Elias's extractor by using a pseudorandom number generation program **rand()** in MATLAB [23] to obtain biased input sequences while controlling the probability (See Sections 4.1 and 4.2). This experiment used **rand()** to generate input sequences because we can control the probability p for each input sequence. Therefore, we vary the probability $p = 0.1, 0.2, \ldots, 0.9$. We show the results for a finite input sequence with 180 bits that would be used in various cryptographic algorithms. In fact, we implemented various bit-lengths of input sequences such as $n = 80, 100, \ldots, 200$ bit-length, and obtained very similar results to the case of 180-bit length (In the primary version [21], we implemented only the case of 180-bit length, and in this paper we further investigated $n = 80, 100, \ldots, 200$ bit-length.). Hence, we will describe only the input length with 180 bits, and we omit the cases of other bit-length in this paper. In addition, to investigate the efficiency of Elias's extractor, the input size should be divided by a reasonable block size. Therefore, the 180 bit-length is also suitable, because it can be divided by many simple block-sizes 10, 20, 30, 60, 90, 180. To compute $\binom{N}{k}$ in Elias's extractor with the RM method, we consider the following:

- The Schönhage–Strassen multiplication algorithm requires $O(N^{1+\epsilon})$ which is asymptotically faster than the normal multiplication requiring $O(N^2)$;
- To avoid multiplication, we use only the addition operation because it is simple and makes the basic operation lighter so that it can be used in various applications and environments.

Additionally, we use the recursive formula $\binom{N}{k} = \binom{N-1}{k-1} + \binom{N-1}{k}$ for $10 \leq N \leq 180$ in order to compute $\binom{N}{k}$ only by additions and also by dynamic programming. To compute experimental redundancy with finite input sequences, we use 180 bit-length of inputs and generate 100 times for each probability p. The **rand()** will produce different sequences in every time under the same probability, thus we repeat the process to generate input sequences 100 times and calculate the average experimental redundancy. In fact, we repeated the process to generate input sequences 100, 1000, and 2000 times, but all the results on the average of experimental redundancy are almost the same, and hence, we focus on generating input sequences 100 times only (In the primary version [21], we repeated the process to generate input sequences only 100 times, and in this paper we conducted further investigations when repeating the process 1000 and 2000 times.). Next, we note that the number of iterations satisfies $\nu \leq \lfloor \log 180 \rfloor = 7$ for Peres's extractor in Section 4.1, and we take the block size $N = 10, 20, 30, 60, 90, 180$ for Elias's extractor with the RM method in Section 4.2. Then, we calculate the average on the redundancy function $f_\nu^{\mathsf{P}}(p)$ of Peres's extractor by using (7) and the redundancy function $f^{\mathsf{E}}(p, N) = h(p) - r^{E}(p, N)$ of Elias's extractor with the RM method by using (3) for each probability p.

To answer the Question (3), we investigate running time to extract uniformly random sequences for both extractors (See Section 4.3). Time complexity depends on the length of input sequences, and thus the probability is not a parameter in this investigation. Thereby, this experiment changes the random number generator for input sequences to RANDOM.ORG [24] to generate input sequences. This random number generator can produce a sequence that is very close to a true random number with unknown probability p by using the randomness of atmospheric noises. In addition, it can produce 131,072 random bits in each time. This experiment takes $n = 100, 200, 400, 600, 800, 1000, 2000, 3000, 4000, 5000$ as the bit-length of input sequences (In the primary version [21], we took only $n = 100, 200, 400, 600, 800, 1000$, and we further investigated longer bit-length $2000, 3000, 4000, 5000$ in this paper.). For reliability of our experiment, we repeated the process to extract unbiased random sequences 100 times for each n, and then calculated their average running time.

By analyzing all the results of the experiments above, we can answer the Question (4): we can compare the redundancy of both extractors under the very similar running time (see Section 4.4).

4.1. Analysis of the Redundancy of Peres's Extractor

In Figure 1a, we show the redundancy of Peres's extractor from theoretical aspects, that is, we calculated the redundancy $f_\nu^{\mathrm{P}}(p)$ of Peres's extractor by using (7) with the iterations $\nu = 1, 2, \ldots, 7$ and the probability $p = 0.1, 0.2, \ldots, 0.9$. We depicted the graphs of redundancy $f_\nu^{\mathrm{P}}(p)$, where the x-axis means probability p and the y-axis means redundancy. It can be easily seen that the redundancy becomes smaller as the number of iterations becomes bigger, for all $p \in (0, 1)$. Furthermore, we showed the experimental redundancy of Peres's extractor with 180 bit-length of input sequences in Figure 1b. As a result, the theoretical redundancy in Figure 1a is almost the same as the experimental redundancy in Figure 1b.

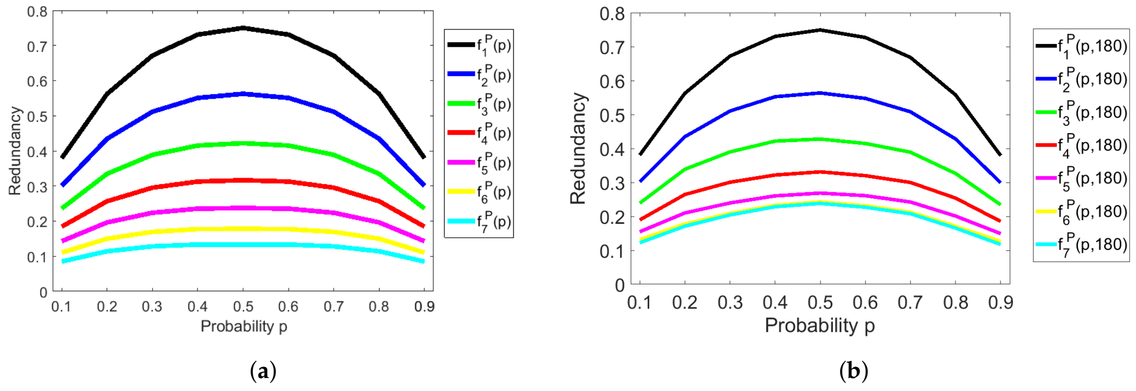
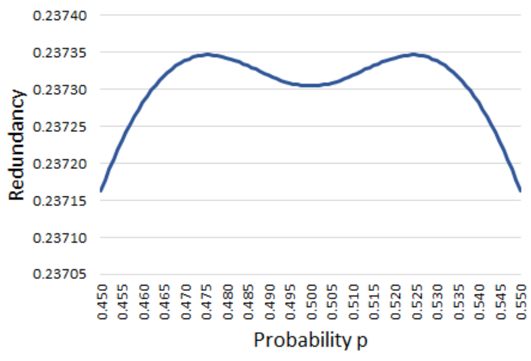


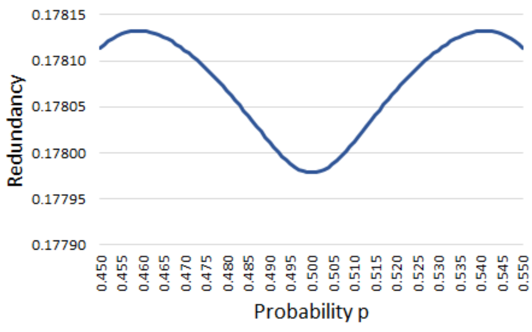
Figure 1. Redundancy of Peres's extractor. (**a**) Asymptotic and theoretical estimate of redundancy by Equation (7); (**b**)Non-asymptotic and experimental estimate of redundancy with 180-bit input sequences.

Figure 2 depicts the graphs of theoretical redundancy $f_\nu^{\mathrm{P}}(p)$ with $\nu = 5, 6$ around $p = 1/2$, namely, $0.450 \le p \le 0.550$. Both graphs support Proposition 1 from a geometric viewpoint. In addition, our experiment shows that $f_5^{\mathrm{P}}(p)$ would approximately take the maximum 0.2373467 at $p \approx 0.476$ and $p \approx 0.524$, and $f_6^{\mathrm{P}}(p)$ would approximately take the maximum 0.1781326 at $p \approx 0.459$ and $p \approx 0.541$.

Figure 3 is provided to observe the difference or similarity between $f_\nu^{\mathrm{P}}(p)$ and $f_\nu^{\mathrm{P}}(p, n)$ $(\nu \ge 5)$ for a large fixed n. Figure 3 shows experimental redundancy with probability $0.450 \le p \le 0.550$ at the x-axis as in Figure 2. When we observe $f_\nu^{\mathrm{P}}(p)$ and $f_\nu^{\mathrm{P}}(p, n)$ by a rough scale, those graphs are very similar as shown in Figure 1; however, when we observe $f_\nu^{\mathrm{P}}(p)$ and $f_\nu^{\mathrm{P}}(p, n)$ with $n = 180$ and $\nu = 5, 6$ by a fine scale, we can see the difference between those graphs. Actually, the graphical forms of $f_5^{\mathrm{P}}(p, 180)$ and $f_6^{\mathrm{P}}(p, 180)$ are quite different from $f_5^{\mathrm{P}}(p)$ and $f_6^{\mathrm{P}}(p)$, respectively, as shown in Figures 2 and 3, although there is the fluctuation in Figure 3 depending on our experiments. This implies that we need to analyze a non-asymptotic function $f_\nu^{\mathrm{P}}(p, n)$ much more from a theoretical aspect in the future, and this analysis is also important to validate the assumption given in Proposition 2.

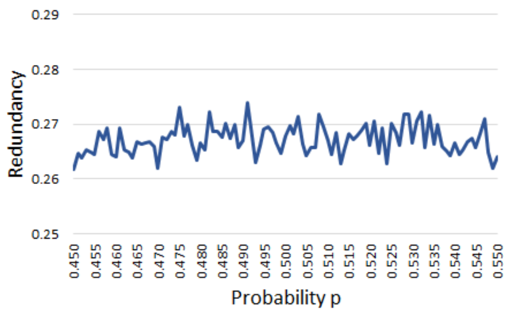


(**a**)

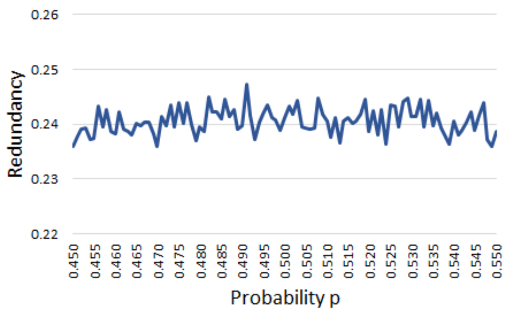


(**b**)

Figure 2. Asymptotic and theoretical estimate of the redundancy of Peres's extractor with $\nu = 5, 6$ and $0.450 \le p \le 0.550$. (**a**) Graph of $f^{\mathrm{P}}_5(p)$; (**b**) Graph of $f^{\mathrm{P}}_6(p)$.



(**a**)



(**b**)

Figure 3. Non-asymptotic and experimental estimates on the redundancy of Peres's extractor for 180-bit input sequences with $\nu = 5, 6$ and $0.450 \le p \le 0.550$. (**a**) Graph of $f^{\mathrm{P}}_5(p, 180)$; (**b**) Graph of $f^{\mathrm{P}}_6(p, 180)$.

4.2. Analysis of the Redundancy of Elias's Extractor with the RM Method

In Figure 4a, we show the redundancy of Elias's extractor with the RM method from theoretical aspects, that is, we calculated the theoretical redundancy $f^{\mathrm{E}}(p, N) = h(p) - r^E(p, N)$ of Elias's extractor with the RM method by using (3) with probability $p = 0.1, 0.2, \ldots, 0.9$ and the block size $N = 10, 20, 30, 60, 90, 180$. It can be seen that the redundancy becomes smaller as block size becomes larger, for all $p \in (0, 1)$. In spite of the fact that there is a slight difference between theoretical redundancy in Figure 4b and experimental redundancy in Figure 4a, we can say that most of them are similar.

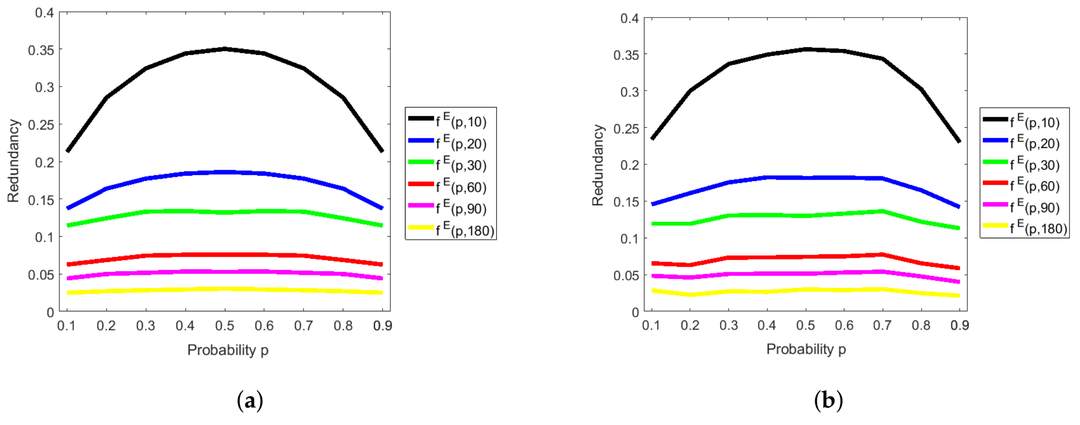


Figure 4. Redundancy of Elias's extractor with the RM method. (**a**) Asymptotic and theoretical estimate of redundancy by Equation (3) and $f^{\mathsf{E}}(p,n) := h(p) - r^{\mathsf{E}}(p,n)$; (**b**) Non-asymptotic and experimental estimate of redundancy with 180-bit input sequences.

As a result, the redundancy of Elias's extractor with large block size is better than that of Peres's extractor, which is an answer to our second question. Moreover, we can observe that the theoretical redundancy is almost the same as the experimental redundancy in both extractors, which is an answer to our first question. Therefore, we can rely on our implementation, and we will use this implementation for analyzing the running time in the next section.

4.3. Analysis of the Time Complexity of Both Extractors

This section will answer the third question. In Figure 5a, we show the running time of Peres's extractor with iterations $\nu = 1, 2, \ldots, 7$ and bit-length of input sequences n = 100, 200, 400, 600, 800, 1000, 2000, 3000, 4000, 5000. We depicted the graphs of the running time, where the x-axis is the bit-length of input sequences and the y-axis is the running time in the second unit. It is clearly seen that an increase in the number of iterations leads to a large running time. The running time increases almost linearly but the slope depends on the iterations ν, as supported by a theoretical estimate of time complexity $O(\nu n)$. Additionally, the running time of iterations $\nu = 7$ and the bit-length of input sequences $n = 5000$ leads to the largest running time (1.425 milliseconds), which means that it can be used in real-world applications.

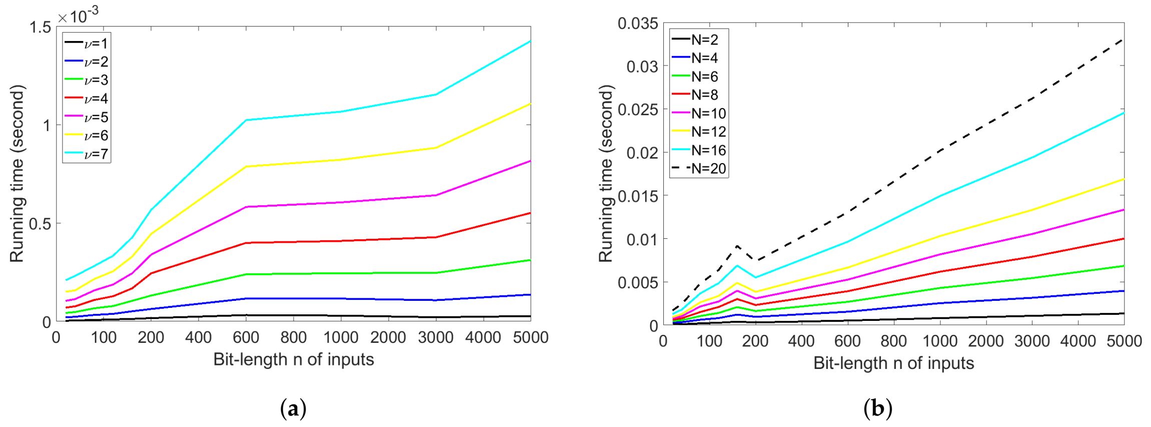


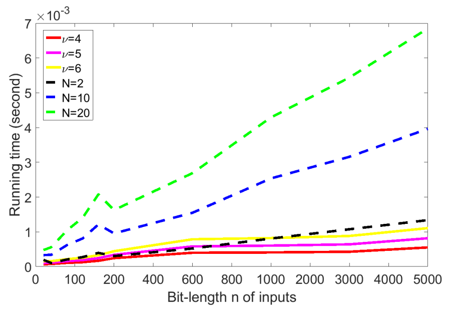
Figure 5. Running time. (**a**) Peres's extractor; (**b**) Elias's extractor with the RM method.

In Figure 5a, we show the running time of Elias's extractor with the RM method with block size $N = 2, 4, 6, 8, 10, 12, 16, 20$. It can be seen that an increase in the block size leads to a large running time. The running time increases linearly, but the slope depends on the block size N, as supported by a theoretical estimate of time complexity $O(N \log^3 N \log\log N)$. In addition, the running time with block size $N = 20$ and bit-length of input sequences $n = 5000$ leads to the largest running time (33.155 milliseconds), which is much larger than that of Peres's extractor.

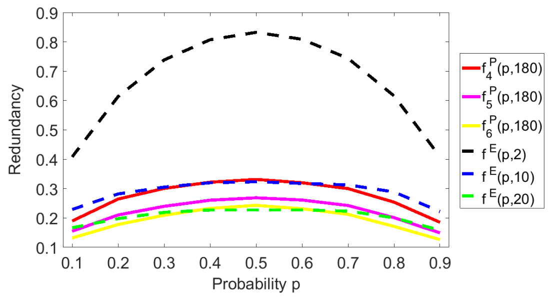
By comparing the running time of both extractors, the running time of Peres's extractor is better than that of Elias's extractor with the RM method at the same bit-length of input sequences. In the case of a long bit-length of input sequences, the difference between running time of both extractors can be seen more clearly. Therefore, we can conclude that Peres's extractor is faster than Elias's extractor with the RM method at the same bit-length of input sequences. On the other hand, according to the results in Sections 4.1 and 4.2, we have seen that the redundancy of Elias's extractor with the RM method is better than that of Peres's extractor. Thus, we analyze the comparison of redundancy (or rate) under the very similar running time in the next section.

4.4. Comparison under the Very Similar Running Time

In all previous experiments, we have observed that the redundancy of Elias's extractor with the RM method is better than that of Peres's extractor; however, the time complexity of Peres's extractor is better than that of Elias's extractor with the RM method. Therefore, we will answer the fourth question by comparing the running time in Figure 6a and redundancy under the very similar running time in Figure 6b.



(**a**)



(**b**)

Figure 6. Comparison of Peres's and Elias's extractors. (**a**) Comparison of running time; (**b**) Comparison of redundancy for 180-bit inputs.

In Figure 6a, we show the comparison of the running time of Peres's extractor with iterations $\nu = 4, 5, 6$ and the running time of Elias's extractor with the RM method with block size $N = 2, 10, 20$. The running time of Peres's extractor with iterations $\nu = 6$ (the yellow line) is almost the same as the running time of Elias's extractor with the RM method having block size $N = 2$ (the black dashed line). Thereby, we can compare the experimental redundancy of Peres's extractor and that of Elias's extractor with the RM method under the very similar running time, that is, $f_6^{\mathsf{P}}(p, 180)$ and $f^{\mathsf{E}}(p, 2)$ in Figure 6b. It is clearly seen that $f_6^{\mathsf{P}}(p, 180)$ (the yellow line) is much better than $f^{\mathsf{E}}(p, 2)$ (the black dashed line), and $f_6^{\mathsf{P}}(p, 180)$ is close to $f^{\mathsf{E}}(p, 20)$ (the green dashed line). However, the running time of Elias's extractor with the RM method with block size $N = 20$ is much larger than the running time of Peres's extractor with iterations $\nu = 6$, as seen in Figure 6a. In addition, we can observe that the redundancy $f_4^{\mathsf{P}}(p, 180)$ of Peres's extractor with iterations $\nu = 4$ (the red line) is close to the redundancy $f^{\mathsf{E}}(p, 10)$ of Elias's extractor with the RM method with block size $N = 10$ (the blue dashed line), but the running time of Elias's extractor with the RM method with block size $N = 10$ is approximately 16 times larger than that of Peres's extractor with iterations $\nu = 4$, as seen in Figure 6a (i.e., the blue dashed line and the red line). As a result, we can conclude that Peres's extractor achieves a better rate (or redundancy) than Elias's extractor with the RM method under the very similar running time.

5. Conclusions

It is known that Elias's extractor achieves the optimal rate if the block size tends to infinity. We considered an improved version of Elias's extractor from Ryabko and Matchikina [10] to reduce both the time complexity and space complexity. Peres's extractor achieves the optimal rate if the length of the input and the number of iterations tend to infinity. These are the results of asymptotic analysis, but it is important and interesting to non-asymptotically analyze and compare both extractors for finite input sequences, since the resulting information will be useful in applications (e.g., cryptography) in practice.

In this paper, we evaluated the numerical performance of Peres's extractor and Elias's extractor with the RM method in terms of practical aspects. Firstly, we derived a lower bound on the maximum redundancy of Peres's extractor based on some heuristics, and we showed that the maximum redundancy of Elias's extractor (with the RM method) was superior to that of Peres's extractor in general, if we do not pay attention to the time complexity or space complexity. We also found that $f_\nu^{\mathsf{P}}(p)$ is not concave in $p \in (0,1)$ for every $\nu \geq 5$. Afterwards, we evaluated the numerical performance of Peres's extractor and Elias's extractor with the RM method for finite input sequences. Our implementation evaluated it on a general PC and did not require any special resources, libraries, or frameworks for computation. Our empirical results showed that Peres's extractor is much better than Elias's extractor for given finite input sequences under a very similar running time. As a consequence, Peres's extractor would be more suitable to generate uniformly random sequences in practice in applications such as cryptographic systems.

Author Contributions: Conceptualization, A.P. and J.S.; Data curation, A.P. and J.S.; Formal analysis, A.P., N.K., and J.S.; Investigation, A.P., N.K., and J.S.; Methodology, A.P., N.K., and J.S.; Resources, A.P.; Software, A.P.; Supervision, J.S.; Validation, A.P. and J.S.; Visualization, A.P.; Writing—Original Draft, A.P., N.K., and J.S.; Writing—Review & Editing, A.P., N.K., and J.S.

Funding: This work was supported by JSPS KAKENHI under Grant Numbers JP18H03238 and JP17H01752.

Acknowledgments: The authors would like to thank anonymous referees for valuable and helpful comments.

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

Appendix A. Proof of Proposition 1

First, we note that, for $\nu \geq 1$,

$$f_\nu^{\mathsf{P}}(1/2) = h(1/2) - r_\nu^{\mathsf{P}}(1/2) = \left(\frac{3}{4}\right)^\nu, \tag{A1}$$

where the last equality follows from (9).

For $p \in (0,1)$, we define $\tilde{p} := p^2 + (1-p)^2$ and $\hat{p} := p^2/\tilde{p}$. Then, it holds that

$$\frac{d\tilde{p}}{dp} = 2(2p-1), \quad \frac{d\hat{p}}{dp} = \frac{2p(1-p)}{\tilde{p}^2}. \tag{A2}$$

Next, for the first-order derivative of $f_\nu^{\mathsf{P}}(p)$, we have

$$\frac{df_1^{\mathsf{P}}(p)}{dp} = \frac{1}{\ln 2}\ln\frac{1-p}{p} + 2p - 1, \tag{A3}$$

$$\frac{df_\nu^{\mathsf{P}}(p)}{dp} = (2p-1)\left(f_{\nu-1}^{\mathsf{P}}(\hat{p}) + \frac{df_{\nu-1}^{\mathsf{P}}(\tilde{p})}{dp}\right) + \frac{p(1-p)}{\tilde{p}}\frac{df_{\nu-1}^{\mathsf{P}}(\hat{p})}{dp} \quad \text{for } \nu \geq 2. \tag{A4}$$

Then, by setting $p = 1/2$ in (A4), for $\nu \geq 2$, we have

$$\begin{aligned} \frac{df_{\nu}^{\mathsf{P}}(1/2)}{dp} &= \frac{1}{2}\frac{df_{\nu-1}^{\mathsf{P}}(1/2)}{dp} && \text{(A5)} \\ &= \left(\frac{1}{2}\right)^{\nu-1}\frac{df_{1}^{\mathsf{P}}(1/2)}{dp} \\ &= 0, && \text{(A6)} \end{aligned}$$

where (A5) follows from (A4), and (A6) follows from (A3).

Moreover, for the second-order derivative of $f_{\nu}^{\mathsf{P}}(p)$, we obtain

$$\frac{d^2 f_1^{\mathsf{P}}(p)}{dp^2} = -\frac{1}{\ln 2}\frac{1}{p(1-p)} + 2, \qquad \text{(A7)}$$

$$\begin{aligned} \frac{d^2 f_{\nu}^{\mathsf{P}}(p)}{dp^2} &= 2f_{\nu-1}^{\mathsf{P}}(\hat{p}) + 2\frac{df_{\nu-1}^{\mathsf{P}}(\tilde{p})}{dp} + \frac{1-2p}{\tilde{p}}\frac{df_{\nu-1}^{\mathsf{P}}(\hat{p})}{dp} \\ &+2(2p-1)^2\frac{d^2 f_{\nu-1}^{\mathsf{P}}(\tilde{p})}{dp^2} + && \text{(A8)} \\ &\frac{2p^2(1-p)^2}{\tilde{p}^3}\frac{d^2 f_{\nu-1}^{\mathsf{P}}(\hat{p})}{dp^2} \quad \text{for } \nu \geq 2. \end{aligned}$$

Furthermore, by setting $p = 1/2$ in (A9), for $\nu \geq 2$, we have

$$\begin{aligned} \frac{d^2 f_{\nu}^{\mathsf{P}}(1/2)}{dp^2} &= 2f_{\nu-1}^{\mathsf{P}}(1/2) + 2\frac{df_{\nu-1}^{\mathsf{P}}(1/2)}{dp} + \frac{d^2 f_{\nu-1}^{\mathsf{P}}(1/2)}{dp^2} \\ &= 2\left(\tfrac{3}{4}\right)^{\nu-1} + \frac{d^2 f_{\nu-1}^{\mathsf{P}}(1/2)}{dp^2}, \end{aligned} \qquad \text{(A9)}$$

where the first equality follows from (A9), and the second equality (A9) follows from (A1) and (A6). Then, by solving Equation (A9) ($\nu \geq 2$) and $\frac{d^2 f_1^{\mathsf{P}}(1/2)}{dp^2} = 2 - 4/\ln 2$, we obtain

$$\begin{aligned} \frac{d^2 f_{\nu}^{\mathsf{P}}(1/2)}{dp^2} &= \frac{d^2 f_1^{\mathsf{P}}(1/2)}{dp^2} + 2\sum_{k=1}^{\nu-1}\left(\frac{3}{4}\right)^k \\ &= 2 - \frac{4}{\ln 2} + 6\left\{1 - \left(\frac{3}{4}\right)^{\nu-1}\right\} && \text{(A10)} \\ &= 8 - \frac{4}{\ln 2} - 6\left(\frac{3}{4}\right)^{\nu-1}. \end{aligned}$$

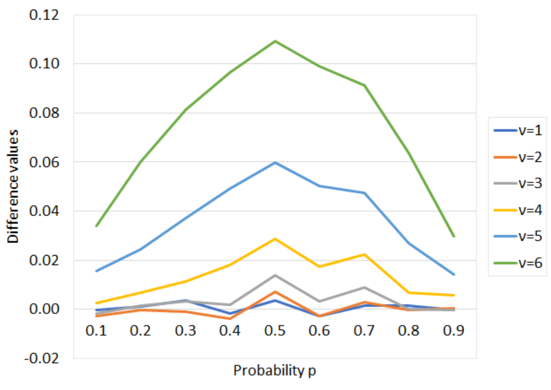
From Equation (A11), it follows that

$$\begin{aligned} \frac{d^2 f_{\nu}^{\mathsf{P}}(1/2)}{dp^2} &< 0 \quad \text{for } 1 \leq \nu \leq 4, \\ \frac{d^2 f_{\nu}^{\mathsf{P}}(1/2)}{dp^2} &> 0 \quad \text{for } \nu \geq 5. \end{aligned}$$

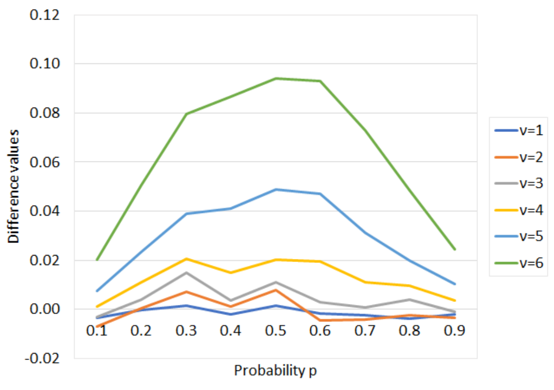
Appendix B. Experimental Results for Proposition 2

In this appendix, we show experimental results for Proposition 2, which support that Proposition 2 holds true. In Figure A1, we depict the difference values $f_{\nu}^{\mathsf{P}}(p,n) - f_{\nu}^{\mathsf{P}}(p)$ with input bit-length $n = 80, 100, \ldots, 200$ and iterations $1 \leq \nu \leq \lfloor \log n \rfloor$. The x-axis is the probability $p = 0.1, 0.2, \ldots, 0.9$ and the y-axis is the difference values defined by $f_{\nu}^{\mathsf{P}}(p,n) - f_{\nu}^{\mathsf{P}}(p)$.

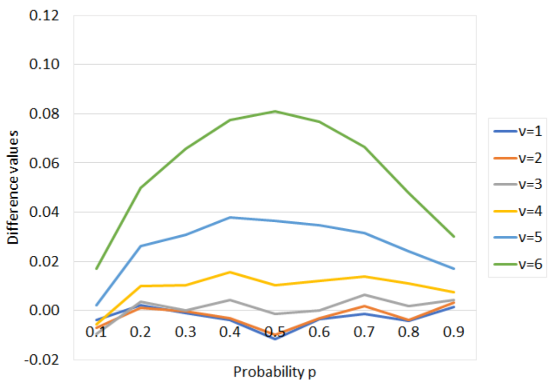
Proposition 2 states that $f^{\mathsf{P}}_{\lfloor \log n \rfloor}(p,n) - f^{\mathsf{P}}_{\lfloor \log n \rfloor}(p) \geq 0$ for $p \in (0,1)$, and we can observe that it holds true for input bit-length $n = 80, 100, \ldots, 200$ by our experimental results given in Figure A1.



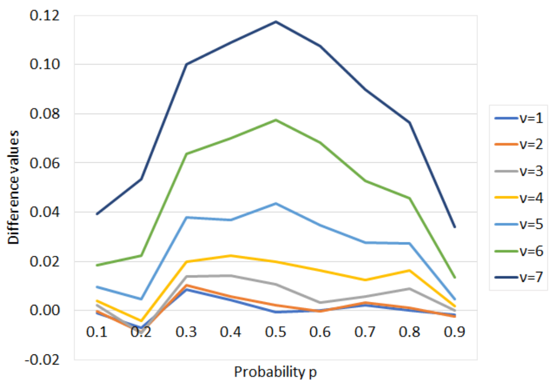
(**a**) 80-bit input sequences.



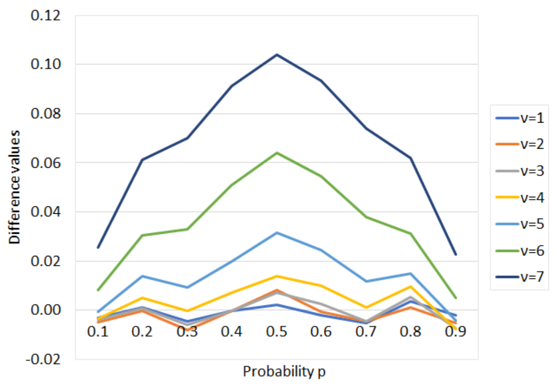
(**b**) 100-bit input sequences.



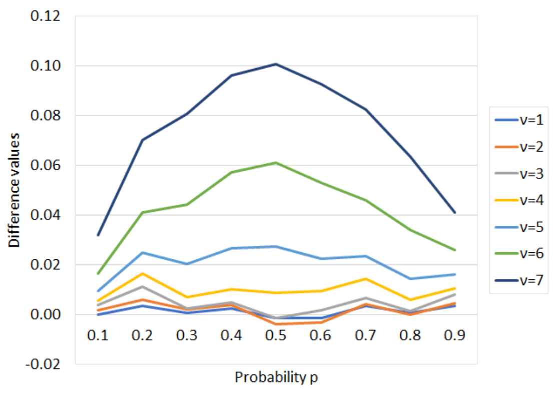
(**c**) 120-bit input sequences.



(**d**) 140-bit input sequences.

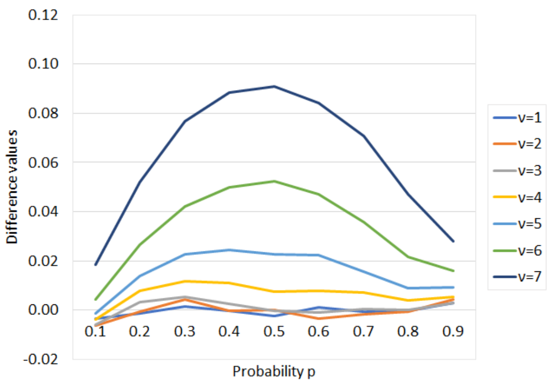


(**e**) 160-bit input sequences.



(**f**) 180-bit input sequences.

Figure A1. *Cont.*



(**g**) 200-bit input sequences.

Figure A1. Difference values $f_v^{\mathsf{P}}(p,n) - f_v^{\mathsf{P}}(p)$ with $n = 80, 100, ..., 200$ and $1 \le v \le \lfloor \log n \rfloor$.

References

1. Heninger, N.; Durumeric, Z.; Wustrow, E.; Halderman, J.A. Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In Proceedings of the 21st USENIX Security Symposium, Bellevue, WA, USA, 8–10 August 2012.
2. Lenstra, A.K.; Hughes, J.P.; Augier, M.; Bos, J.W.; Kleinjung, T.; Wachter, C. Public Keys. In *Advances in Cryptology—ECRYPTO 2012*; Safavi-Naini, R., Canetti, R., Eds.; Number 7417 in Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; pp. 626–642.
3. Bendel, M. Hackers Describe PS3 Security As Epic Fail, Gain Unrestricted Access. Available online: Exophase.com (accessed on 20 September 2018).
4. Dorrendorf, L.; Gutterman, Z.; Pinkas, B. Cryptanalysis of the Random Number Generator of the Windows Operating System. *ACM Trans. Inf. Syst. Secur.* **2009**, *13*, 10. [CrossRef]
5. Bonneau, J.; Clark, J.; Goldfeder, S. On Bitcoin as a public randomness source. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 1015.
6. Von Neumann, J. Various Techniques Used in Connection with Random Digits. *J. Res. Nat. Bur. Stand. Appl. Math. Ser.* **1951**, *12*, 36–38.
7. Elias, P. The Efficient Construction of an Unbiased Random Sequence. *Ann. Math. Stat.* **1972**, *43*, 865–870. [CrossRef]
8. Peres, Y. Iterating Von Neumann's Procedure for Extracting Random Bits. *Ann. Stat.* **1992**, *20*, 590–597. [CrossRef]
9. Abbott, A.A.; Calude, C.S. Von Neumann Normalisation and Symptoms of Randomness: An Application to Sequences of Quantum Random Bits. In *Unconventional Computation*; Calude, C.S., Kari, J., Petre, I., Rozenberg, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 40–51.
10. Ryabko, B.; Matchikina, E. Fast and efficient construction of an unbiased random sequence. *IEEE Trans. Inf. Theory* **2000**, *46*, 1090–1093. [CrossRef]
11. Cover, T. Enumerative source encoding. *IEEE Trans. Inf. Theory* **1973**, *19*, 73–77. [CrossRef]
12. Schönhage, A.; Strassen, V. Schnelle Multiplikation großer Zahlen. *Computing* **1971**, *7*, 281–292. (In German) [CrossRef]
13. Pae, S.I. Exact output rate of Peres's algorithm for random number generation. *Inf. Process. Lett.* **2013**, *113*, 160–164. [CrossRef]
14. Bourgain, J. More on the sum-product phenomenon in prime fields and its applications. *Int. J. Number Theory* **2005**, *1*, 1–32. [CrossRef]
15. Raz, R. Extractors with Weak Random Seeds. In Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, Hunt Valley, MD, USA, 22–24 May 2005; ACM: New York, NY, USA, 2005; pp. 11–20.
16. Cohen, G. Local Correlation Breakers and Applications to Three-Source Extractors and Mergers. In Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, Berkeley, CA, USA, 17–20 October 2015; pp. 845–862.

17. Chattopadhyay, E.; Zuckerman, D. Explicit Two-Source Extractors and Resilient Functions. In Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, Cambridge, MA, USA, 18–21 June 2016; ACM: New York, NY, USA, 2016; pp. 670–683.
18. Bouda, J.; Krhovjak, J.; Matyas, V.; Svenda, P. Towards True Random Number Generation in Mobile Environments. In *Identity and Privacy in the Internet Age*; Jøsang, A., Maseng, T., Knapskog, S.J., Eds.; Number 5838 in Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; pp. 179–189.
19. Halprin, R.; Naor, M. Games for Extracting Randomness. In Proceedings of the 5th Symposium on Usable Privacy and Security, Mountain View, CA, USA, 15–17 July 2009; ACM: New York, NY, USA, 2009; p. 12.
20. Voris, J.; Saxena, N.; Halevi, T. Accelerometers and Randomness: Perfect Together. In Proceedings of the Fourth ACM Conference on Wireless Network Security, Hamburg, Germany, 14–17 June 2011; ACM: New York, NY, USA, 2011; pp. 115–126.
21. Prasitsupparote, A.; Konno, N.; Shikata, J. Numerical Analysis of Elias's and Peres's Deterministic Extractors. In Proceedings of the 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, 22–24 March 2017.
22. Graham, R.L.; Knuth, D.E.; Patashnik, O. *Concrete Mathematics*, 2nd ed.; Addison-Wesley: Boston, MA, USA, 1994; pp. 153–256.
23. The MathWorks, Inc. Uniformly Distributed Random Numbers—MATLAB Rand. Available online: Mathworks.com/help/matlab/ref/rand.html (accessed on 20 September 2018).
24. RANDOM.ORG. RANDOM.ORG—Random Byte Generator. Available online: Random.org/bytes (accessed on 20 September 2018).