Article

Numerical and Non-asymptotic Analysis of Elias's and Peres's Extractors with Finite Input Sequence

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

[1] Graduate School of Environment and Information Sciences, Yokohama National University, 240-8501, Japan
[2] Department of Applied Mathematics, Faculty of Engineering, Yokohama National University, 240-8501, Japan
* Correspondence: amonrat-prasitsupparote-zp@ynu.jp
† This paper is an extended version of our paper published in 51^{st} Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, March 2017.

Abstract: Many cryptographic systems require random numbers, and weak random numbers lead 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 probability that each bit of input sequences occurs. Peres's extractor achieves the optimal rate $h(p)$ if the length of input and the number of iterations tend to infinity. The previous researches related to both extractors did not mention 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 one in terms of the maximum redundancy (or the rates) if we do not pay attention to 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. For doing it, 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 one for given finite input sequences under the almost same 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

It is undeniable that random numbers play important roles in cryptography, for example, key generation, nonces, one-time pads, etc. The quality of random numbers directly determines the strength of cryptographic systems. A low quality of random numbers lead to that an adversary can break a system. It can be seen that in 2012, Heninger et al. [1] and Lenstra et al. [2] explored RSA keys in TLS and SSH servers on the Internet. Their experiment showed that a weak random number for generating a random prime in embedded devices led to the result that an adversary could break a system. This tells us that a cryptographic system will be broken if insufficient randomness is used to generate keys. Moreover, there is a hacker group which calling itself fail0verflow [3]. They could recover ECDSA private key generated by weak random numbers for PlayStation 3 game console by Sony in Annual Chaos Communication Congress (27C3) in 2010. Furthermore, Microsoft Windows also generated weak random numbers, as shown by Leo Dorrendorf et al. [4] in 2007. The Windows operating system had an unpublished pseudorandom number generator (PRNG) called CryptGenRandom. Their work examined the binary code of Windows 2000 and reconstructed CryptGenRandom. After that, they found several vulnerabilities, which can be used to predict all random values, such as SSL keys.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729

Overall, the random number generation is very important in cryptography to ensure that secret keys are random and unpredictable.

A natural source such as physical phenomena, the stock market, or Bitcoin [5] can produce unpredictable random sequences, though such sequences from the source are not uniformly random (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). And, as applications in cryptography, the output sequence of those extractors can be used as secret keys in information-theoretic cryptography or symmetric key cryptography. In particular, Elias's and Peres's extractors are well known and fundamental and shown to be optimal in terms of the rate (or redundancy), if we suppose input-size tends to infinity (i.e., in 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., in a non-asymptotic viewpoint) by numerical analysis to make it clear which is better for the practical use.

1.1. Related work

There are several works that proposed the methods for extracting uniformly random sequences from non-uniformly random sequences. The most famous one of them is the 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.

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 implemantation of this extractor requires exponential time and exponential memory size with respect to N, where N is block size, to store all 2^N input sequences with their assignment of output sequences. Later in 2000, Ryabko and Matchikina [8] proposed an extension of Elias's extractor that improved time complexity and space complexity by using the enumerative encoding technique from [9] and Schönhage–Strassen algorithm [10] for fast integer multiplication in order to compute 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 [11] proposed a procedure which is an improved one from the 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. For Elias's extractor, it can be observed in the works [7]. However, for Peres's extractor, it is not explicitly known. As a related work for Peres's extractor, Pae [12] 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 one, and concluded that the rate of Peres's extractor increased much slower than that of Elias's one by the 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 [12].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at Entropy 2018, 20, 729; doi:10.3390/e20100729

There are several works for constructing extractors using multiple sources (i.e., not a single source). Bourgain [13] 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 bit-length of output of the sources. Raz [14] proposed 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 [15] constructed a 3-source extractor, where one source having min-entropy δn, the second source having min-entropy $O(\log n)$ and the third source having min-entropy $O(\log \log n)$. In 2016, Chattopadhyay and Zuckerman [16] 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 high complexity than Peres's extractor or Elias's extractor.

Furthermore, many researchers are interested in implementing a randomness extractor in a real world. In particular, in 2009, Bouda et al. [17] used mobile phones or pocket computers to generate random data that is close to truly random ones. They took 12 pictures per second then used their function to get random 4 bits in each picture, and then applied Carter-Wegman universal$_2$ hash functions. Their output passed 15 of 16 items in NIST statistical tests at the confidence level $\alpha = 0.01$. However, their proposed model was not a simultaneous system, and hence it would be difficult to use in practical applications. Halprin and Naor [18] 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 then extracted with a pairwise independent hash function that it can generate 128 bits 2^{64}-close to random in less than two minutes. For using human as a random generator, there are several impact on the entropy of sources such as the skill of player, interesting and entertain player, the number of rounds in game, etc. Later in 2011, Voris et al. [19] 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 random 128 bits in 1.5 seconds and passed the NIST statistical tests. However, they stored a Toeplitz matrix on the RFID tags and performed matrix multiplications, though the RFID tags have limited computational resources in general.

1.2. Our contribution

In this paper, we revisit the extractors by von Neumann, Elias, and Peres, since they are very fundamental and only require a single source. In the studies for those extractors, it is usual to asymptotically analyze the rate or redundancy of the extractors in the literatures, where the rate is the average bit-length of outputs per bit of input (see Section 2 for detals). 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 with high rate, when it take a long block-size equal to the input length. However, it has 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 the practical use in general. As a related work, there is only one work by Pae [12] which showed comparison of both extractors as mentioned in Section 1.1, but it does not completely answer the question, since it analyzed 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 anwer the question: which is more suitable in the practical use in applications in a real world. For doing it, we evaluate numerical performance of Peres's extractor and the Elias's one with the RM method in terms of practical aspects including achievable rates (or

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at Entropy 2018, 20, 729; doi:10.3390/e20100729

redundancy) and running time with finite input sequences. Specifically, the contribution of the paper is as follows:

- 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 one in general, if we focus only on redundancy (or rates) and we do not pay attention to time complexity or space complexity.
- 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, frameworks for computation. Therefore, it can be applied in various cryptographic applications and platforms without any restrictions. 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 time complexity of both extractors with respect to 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 one under the almost same running time. As a result, Peres's extractor would be more suitable for generating uniformly random sequences for the practical use in applications.

2. Preliminaries

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 [11] in 1992. The prior work [6,7,11] considered Bernoulli source $\mathsf{Bern}(p)$ from which input sequences were generated, namely $\mathsf{Bern}(p)$ outputs i.i.d. $(x_1, x_2, \dots, x_n) \in \{0,1\}^n$ 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 $(x_1, x_2, \dots, x_n) \in \{0,1\}^n$ as input and outputs $(y_1, y_2, \dots, y_\ell) \in \{0,1\}^\ell$, and its average bit-length of output is denoted by $\bar{\ell}(n)$ which is a function of n, and define 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 $f^{\mathsf{A}}(p) := h(p) - r^{\mathsf{A}}(p)$, where $h(\cdot)$ is the binary entropy function defined by $h(p) = -p \log p - (1-p)\log(1-p)$, 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 [7,11]. 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

The von Neumann's extractor was a simple algorithm for extracting independent unbiased bits from biased bits. This algorithm divides the input sequences $(x_1, x_2, x_3, x_4, \dots, x_n)$ into the pairs[1] $((x_1x_2), (x_3x_4), \dots)$ 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 the symbol $\wedge$ means no output was generated. After that, it concatenates all resulting outputs of (1). For the help of understanding, we give an example as follows.

[1] If n is odd, we discard the last bit.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729

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 pairs with the mapping (1). Finally, the extractor outputs* $(y_1, y_2) = (1, 0)$.

Complexity: The 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: The von Neumann extractor is not desirable, since the maximum redundancy is far from zero. Actually, the rate function $r^{\mathrm{vN}}(p)$ of the von Neumann extractor is evaluated by $r^{\mathrm{vN}}(p) = \lim_{n\to\infty} np(1-p)/n = p(1-p)$, 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 are evaluated as follows: $f^{\mathrm{vN}}(p, n) = f^{\mathrm{vN}}(p) = h(p) - p(1-p)$, $\Gamma^{\mathrm{vN}}(n) = \Gamma^{\mathrm{vN}} = 3/4$.

2.2. Elias's extractor

Elias [7] improved the 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 ($k = 0, 1, 2, \ldots, N$), where S_k consists of all the $\binom{N}{k}$ sequences of length N which have k ones and $N-k$ zeros. Here, each sequence of S_k is equiprobable (i.e., the probability is $p^k q^{N-k}$).

Define $\binom{N}{k} = \alpha_m 2^m + \alpha_{m-1} 2^{m-1} + \ldots + \alpha_0 2^0, m_k = \lfloor \log_2 \binom{N}{k} \rfloor$. Let $|S_k| = (\alpha_{m_k}, \alpha_{m_k-1}, \ldots, \alpha_0)$ is the binary expansion of the integer $\binom{N}{k}, \alpha_{m_k} = 1, \alpha_j \in \{0, 1\}, m_k > j \geq 0$. 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 $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$ 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* $N = 4$*, which 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:*

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at Entropy 2018, 20, 729; doi:10.3390/e20100729

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 to 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. For reducing time and space complexity of Elias's extractor, Ryabko and Matchikina [8] proposed a method that is extended from Elias's extractor, which we call the *RM method* in this paper. The RM method utilizes enumerative encoding technique from [9] and Schönhage–Strassen algorithm [10] for fast integer multiplication in order to compute assignment of output sequences without making the large table. The procedure of RM method is described as follows.

Firstly, suppose a binary input sequence $x^N = (x_1, x_2, \ldots, x_N)$ contains k ones and $N-k$ zeros. Let $Num(x^N)$ be a number which corresponds to x^N when we lexicographical order set S_k. If x^N has k ones, then the number $\mathrm{Num}(x^N)$ is defined by

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

Then, we calculate a binary codeword $code(x^N)$ of x^N, which is 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 $0 \le j_0 < j_1 < \ldots < j_m$.
(iii) If $j_0 = 0$ and $\mathrm{Num}(x^N) = 0$, then $code(x^N) = \wedge$.
(iv) If $0 \le \mathrm{Num}(x^N) < 2^{j_0}$, then $code(x^N)$ is defined to be the j_0 low-order binary string of $\mathrm{Num}(x^N)$.
(v) If $\sum_{s=0}^{t} 2^{j_s} \le \mathrm{Num}(x^N) < \sum_{s=0}^{t} 2^{j_s} + 2^{j_{t+1}}$ for some $0 \le t \le m$, 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 $Num(x^N)$ follow the above conditions.*

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

Afterwards, the RM method computes $code(1,0,0,1) = (1,1)$ and $code(0,0,1,1) = (0)$. Finally, 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 [8] for details).

Redundancy: Generally, the rate function and redundancy function of Elias's extractor depend on block size N. For given n-bit input sequence, if we take the block size equal to the length of input

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729

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 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 the 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 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) from 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 the von Neumann's extractor by Ψ_1. For an n-bit sequence $(x_1, x_2, \ldots, x_n)$, we describe the 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 utilize the case of an even n above.

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 the equation (4) is at most $n/2$. Obviously, Peres's extractor with $\nu = 1$ is the same as the von Neumann's extractor. In addition, Peres's extractor with a large ν is considered to be an elegantly improved version from von Neumann's one 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 satisfy $\nu \leq \lfloor \log 8 \rfloor = 3$. 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 the 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 time complexity of the 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, 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 $O(1)$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729

Table 1. Comparison of extractors.

	Redundancy $\Gamma(n)$	Time complexity	Space complexity
von Neumann extractor	$3/4$	$O(n)$	$O(1)$
Elias extractor (with maximum block-size)	$O(1/n)$ (by [7])	$O(n\log^3 n \log\log n)$ (by [8])	$O(n\log^2 n)$ (by [8])
Peres extractor (with maximum iterations)	$o(1)$ (by [11])	$O(n\log n)$ (by [11])	$O(1)$ (by [11])

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 the 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 [11] 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 the 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 the von Neumann's, Elias's, and Peres's extractors.

3. Lower Bound on Redundancy of Peres's Extractor

Although it is shown that $\Gamma^{\mathsf{P}}(n) = o(1)$ in the 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 the Peres's extractor with ν iterations is not concave in $p \in (0,1)$ if $\nu \geq 5$. More generally, for the 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* $\nu \geq 5$.

Peer-reviewed version available at Entropy 2018, 20, 729; doi:10.3390/e20100729

Here, we assume that the following proposition holds true. It does not seem to be easy to provide a proof, however, it seems to be true from our experimental results that are provided in Appendix B.

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

The following theorem shows a lower bound on $\Gamma^{\mathsf{P}}(n)$ that are 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 $1 \leq \nu \leq \log n$, we define $a_\nu := r_\nu(1/2)$. Then, by the equation (6) we have

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

By solving the equation above, we have

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

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

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

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

Therefore, we have

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

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 one, since time complexity and space complexity of Peres's extractor are better than those of Elias's one from 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 one by numerical analysis with various parameters.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at Entropy 2018, 20, 729; doi:10.3390/e20100729

4. Implementation and Numerical Analysis

In this section, we describe our experimental results of Peres's extractor and Elias's one 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 would also be performed on a general PC and do not require any special resources, libraries, frameworks for computation. Actually, we can use other languages instead of Java language, however, Java language can evaluate it on every platform without any support software. Thereby, we used Java language for implementation. For comparing Peres's extractor and Elias's one 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 experimental redundancy of Elias's extractor with the RM method better than 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 almost same running time?

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

To answer the questions (1) and (2), we evaluate theoretical and experimental redundancy of Peres's extractor and Elias's one by using a pseudorandom number generation program **rand()** in MATLAB [20] to get biased input sequences with 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 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. Actually, we implemented various bit-length of input sequences such as $n = 80, 100, \ldots, 200$ bit-length, and obtained almost the same results with the case of 180-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 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. For computing $\binom{N}{k}$ in Elias's extractor with the RM method, we consider the following:

- Schönhage–Strassen multiplication algorithm requires $O(N^{1+\epsilon})$ which is asymptotically faster than the normal multiplication requiring $O(N^2)$;
- For avoiding 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. For computing 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 to generate input sequences 100 times and calculate the average of experimental redundancy. Actually, we repeated 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. 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 RM method in Section 4.2. Then, we calculate the average on the redundancy function $f^{\mathsf{P}}_{\nu}(p)$ 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 for extracting 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 [21] for generating input sequences. This random number generator can produce a sequence that is very close

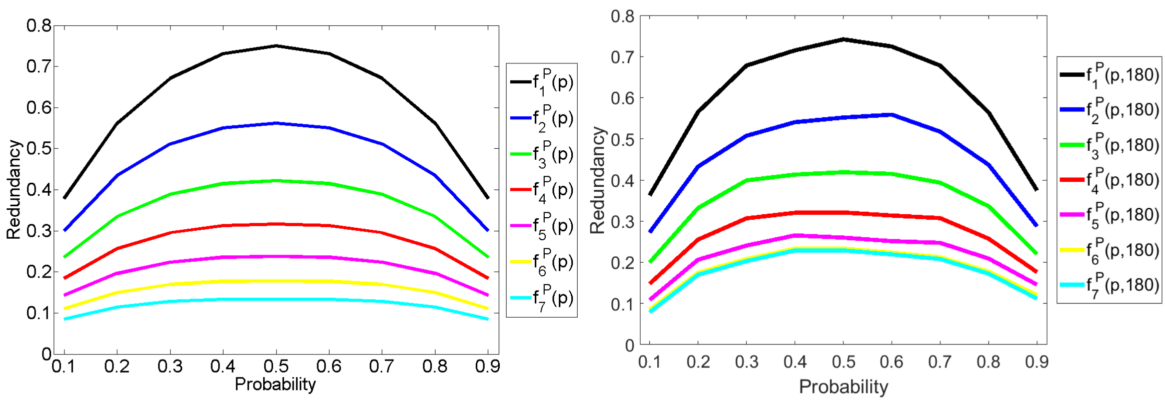
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729

to a true random number with unknown probability p by using 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 bit-length of input sequences. For reliability of our experiment, we repeated to extract unbiased random sequences 100 times for each n, and then calculated the average on their 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 almost same running time (see Section 4.4).

4.1. Analysis of redundancy of Peres's extractor

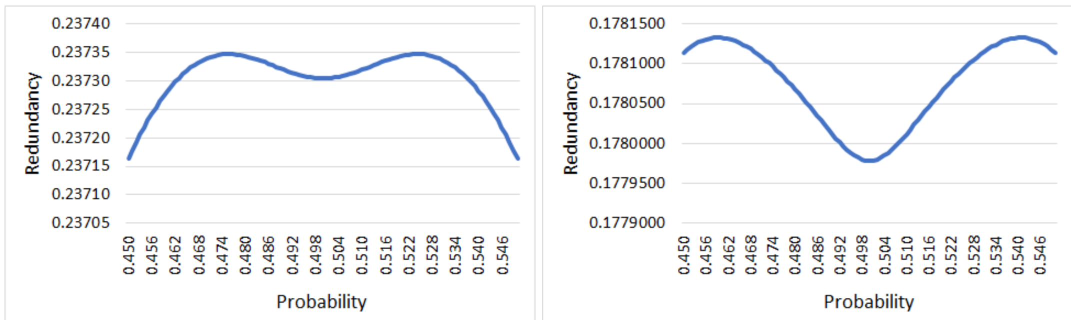


(a) Asymptotic and theoretical estimate of redundancy by equation (7).

(b) Non-asymptotic and experimental estimate of redundancy with 180-bit input sequences.

Figure 1. Redundancy of Peres's extractor.

In Fig. 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 x-axis means probability p and y-axis means redundancy. It can be easily seen that the redundancy becomes smaller as the number of iterations become 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 Fig. 1b. As a result, the theoretical redundancy in Fig. 1a is almost the same as the experimental redundancy in Fig. 1b.



(a) Graph of $f_5^{\mathrm{P}}(p)$

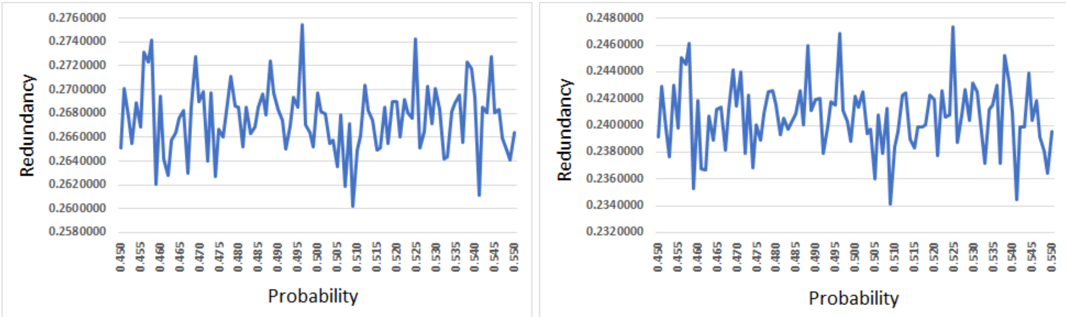
(b) Graph of $f_6^{\mathrm{P}}(p)$

Figure 2. Asymptotic and theoretical estimate of redundancy of Peres's extractor with $\nu = 5, 6$ and $0.450 \leq p \leq 0.550$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729

In Fig. 2, we depiced the graphs of theoretical redundancy $f_\nu^{\mathrm{P}}(p)$ with $\nu = 5, 6$ arround $p = 1/2$, namely, $0.450 \le p \le 0.550$. Both graphs support Proposition 1 in 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 $p \approx 0.541$.

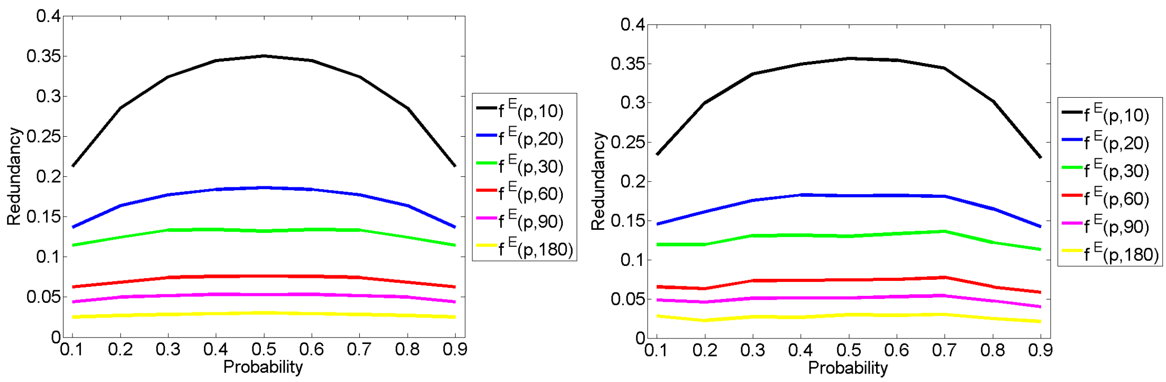


(a) Graph of $f_5^{\mathrm{P}}(p, 180)$ **(b)** Graph of $f_6^{\mathrm{P}}(p, 180)$

Figure 3. Non-asymptotic and experimental estimates on redundancy of Peres's extractor for 180-bit input sequences with $\nu = 5, 6$ and $0.450 \le p \le 0.550$.

In Fig. 3, we show experimental redundancy with probability $0.450 \le p \le 0.550$ at x-axis as in Fig. 2. It can be seen that $f_\nu^{\mathrm{P}}(p)$ $(\nu = 5, 6)$ would not be concave but there is much fluctuation, although $f_\nu^{\mathrm{P}}(p)$ $(\nu = 5, 6)$ in Fig. 1b look to be concave.

4.2. Analysis of redundancy of Elias's extractor with the RM method



(a) Asymptotic and theoretical estimate of redundancy by equation (3) and $f^{\mathrm{E}}(p, n) := h(p) - r^{\mathrm{E}}(p, n)$. **(b)** Non-asymptotic and experimental estimate of redundancy with 180-bit input sequences.

Figure 4. Redundancy of Elias's extractor with RM method.

In Fig. 4a, we show the redundancy of Elias's extractor with the RM method from theoretical aspects, that is, we calculated the theoretical redundancy $f^{\mathrm{E}}(p, N) = h(p) - r^{\mathrm{E}}(p, N)$ 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 Fig. 4a and experimental redundancy in Fig. 4b, we can say that most of them have similarity.

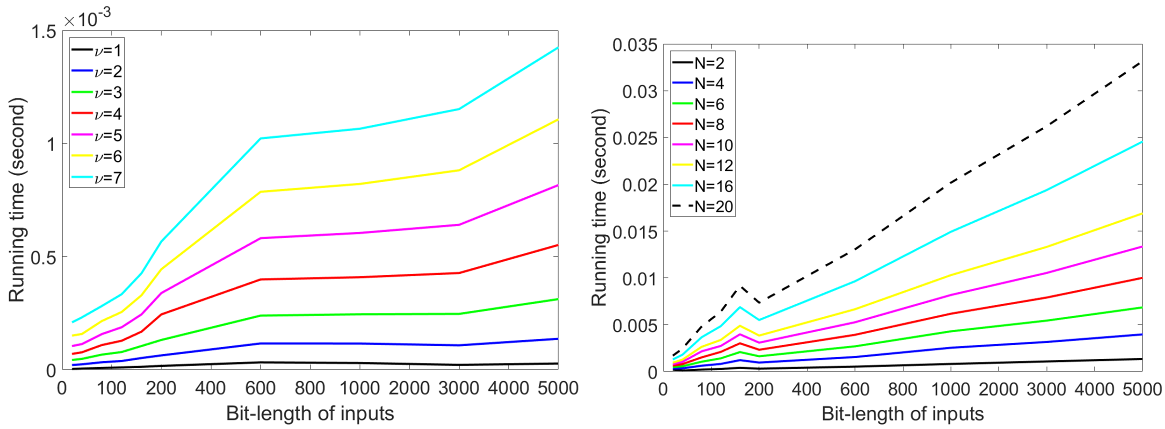
As a result, the redundancy of Elias's extractor with large block size is better than that of Peres's extractor, which is an answer to the second question of ours. Moreover, we can observe that the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729

theoretical redundancy is almost the same as the experimental redundancy in both extractors, which is an answer to the 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 time complexity of both extractors



(a) Peres's extractor. **(b)** Elias's extractor with RM method.

Figure 5. Running time.

This section will answer the third question. In Fig. 5a, we show 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 x-axis means bit-length of input sequences and y-axis means running time in the second unit. It is clearly seen that, if the number of iterations become larger, it leads to the large running time. The running time increases almost linearly but the slope depends on the iterations ν, as supported by theoretical estimate of time complexity $O(\nu n)$. Additionally, the running time of iterations $\nu = 7$ and bit-length of input sequences $n = 5000$ is the largest running time (1.425 milliseconds), which means that it can be used in practice in a real world.

In Fig. 5b, we show running time of Elias's extractor with RM method with block size $N = 2, 4, 6, 8, 10, 12, 16, 20$. It can be seen that, if the block size becomes larger, it leads to the large running time. The running time increases linearly, but the slope depends on the block size N, as supported by 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 $n = 5000$ is the largest running time (33.155 milliseconds), which is much larger than that of Peres's extractor.

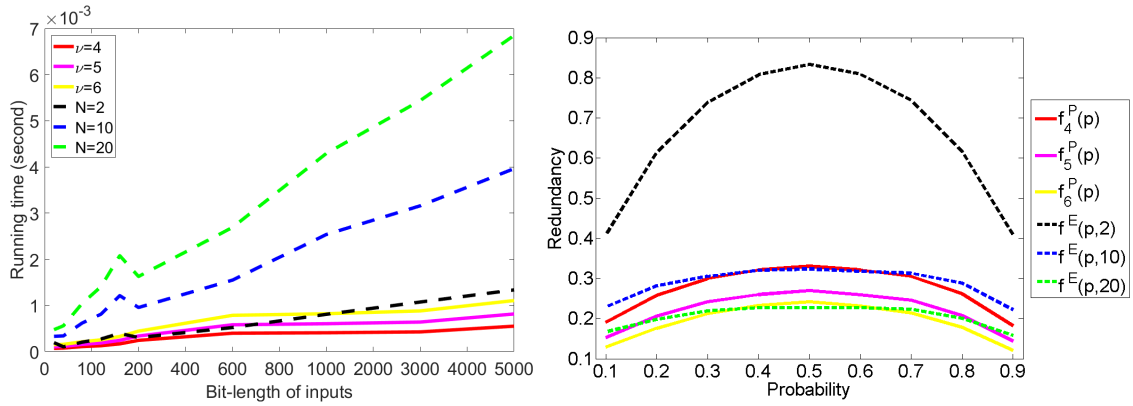
By comparing the running time of both extractors, the running time of Peres's extractor is better than that of Elias's extractor with the RM method at the same bit-length of input sequences. In case of long bit-length of input sequences, the difference between running time of both extractors can be seen 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 comparison of redundancy (or rate) under the almost same running time in the next section.

4.4. Comparison under the almost same running time

By 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; but, the time complexity of Peres's extractor is

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at Entropy 2018, 20, 729; doi:10.3390/e20100729



(a) Comparison of running time.

(b) Comparison of redundancy for 180-bit inputs.

Figure 6. Comparison of Peres's and Elias's extractors.

better than that of Elias's extractor with the RM method. Therefore, we will answer the fourth question by comparing running time in Fig. 6a and redundancy under the almost same running time in Fig. 6b.

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

5. Conclusion

Evidently, Elias's extractor achieved the optimal rate if the block size tends to infinity. On the other hand, Peres's extractor achieved the optimal rate if the length of input and the number of iterations tend to infinity. Note that we used an improved version of Elias's extractor from Ryabko and Matchikina [8]. For finite input sequences, it is not easy to decide which extractor is more appropriate to use in applications (e.g., cryptography) in practice.

In this paper, we evaluated numerical performance of Peres's extractor and Elias's one 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 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 numerical performance of Peres's extractor and Elias's one with the RM method for finite input sequences. Our implementation evaluated it on a general PC and did not require any special resources, libraries, frameworks for computation, which means that it can be easily utilized for the practical use in various applications. As a result, we showed that Peres's

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

Peer-reviewed version available at Entropy 2018, 20, 729; doi:10.3390/e20100729

extractor is faster than the Elias's one at the same bit-length of input sequences. Moreover, Peres's extractor is also much better than Elias's one with the RM method under the almost same running time and the same bit-length of input sequences. Consequently, Peres's extractor will be better in practical use to produce uniformly random sequences, and more appropriate to use in applications such as cryptography.

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

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

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

And, by setting $p = 1/2$ in (A8), for $\nu \geq 2$, we have

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

where the first equality follows from (A8), and the second equality (A9) follows from (A1) and (A6). Then, by solving the equation (A9) ($\nu \geq 2$) and $\frac{d^2 f_1^{\mathsf{P}}(1/2)}{dp^2} = 2 - 4/\ln 2$, we get

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

From the equation (A10), 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 Fig. B1, 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 $1 \leq \nu \leq \lfloor \log n \rfloor$. The x-axis means the probability $p = 01., 0.2, ..., 0.9$ and y-axis means the difference values defined by $f_\nu^{\mathsf{P}}(p,n) - f_\nu^{\mathsf{P}}(p)$.

Proposition 2 states that $f_{\lfloor \log n \rfloor}^{\mathsf{P}}(p,n) - f_{\lfloor \log n \rfloor}^{\mathsf{P}}(p) \geq 0$ 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.

References

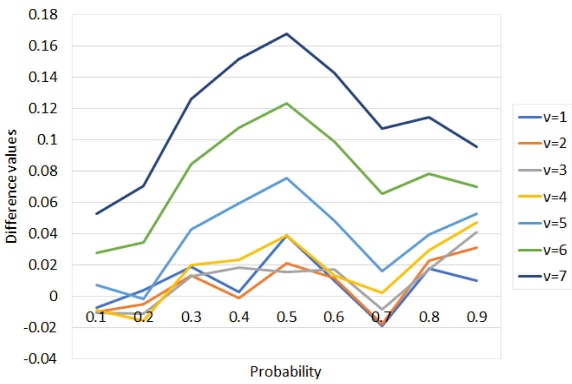
1. Heninger, N.; Durumeric, Z.; Wustrow, E.; Halderman, J.A. Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. Proceedings of the 21st USENIX Security Symposium, 2012.
2. Lenstra, A.K.; Hughes, J.P.; Augier, M.; Bos, J.W.; Kleinjung, T.; Wachter, C. Public Keys. In *Advances in Cryptology – CRYPTO 2012*; Safavi-Naini, R.; Canetti, R., Eds.; Number 7417 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012; pp. 626–642.
3. Bendel, M. Hackers Describe PS3 Security As Epic Fail, Gain Unrestricted Access - Exophase.com.
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*, 10:1–10:32. doi:10.1145/1609956.1609966.
5. Bonneau, J.; Clark, J.; Goldfeder, S. On Bitcoin as a public randomness source. *IACR Cryptology ePrint Archive* **2015**, *2015*, 1015.
6. Neumann, J.v. Various Techniques Used in Connection with Random Digits, Notes by G E Forsythe. *National Bureau of Standards Applied Math Series* **1951**, *12*, 36–38.
7. Elias, P. The Efficient Construction of an Unbiased Random Sequence. *Ann. Math. Statist.* **1972**, *43*, 865–870. doi:10.1214/aoms/1177692552.
8. Ryabko, B.; Matchikina, E. Fast and efficient construction of an unbiased random sequence. *IEEE Transactions on Information Theory* **2000**, *46*, 1090–1093. doi:10.1109/18.841190.
9. Cover, T. Enumerative source encoding. *IEEE Transactions on Information Theory* **1973**, *19*, 73–77. doi:10.1109/TIT.1973.1054929.
10. Schönhage, A.; Strassen, V. Schnelle Multiplikation großer Zahlen. *Computing* **1971**, *7*, 281–292. doi:10.1007/BF02242355.
11. Peres, Y. Iterating Von Neumann's Procedure for Extracting Random Bits. *Ann. Statist.* **1992**, *20*, 590–597. doi:10.1214/aos/1176348543.
12. Pae, S.i. Exact output rate of Peres's algorithm for random number generation. *Information Processing Letters* **2013**, *113*, 160–164. doi:10.1016/j.ipl.2012.12.012.

Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729

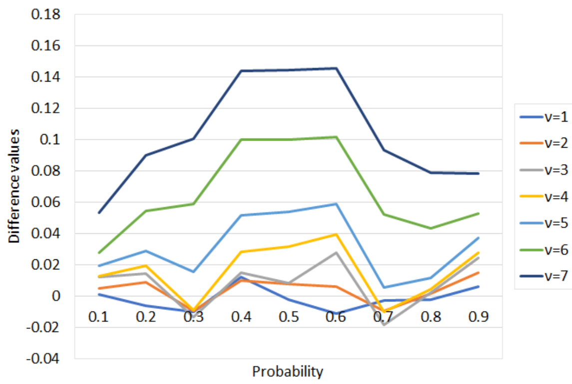
13. Bourgain, J. More on the sum-product phenomenon in prime fields and its applications. *International Journal of Number Theory* **2005**, *01*, 1–32. doi:10.1142/S1793042105000108.
14. Raz, R. Extractors with Weak Random Seeds. Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing; ACM: New York, NY, USA, 2005; STOC '05, pp. 11–20. doi:10.1145/1060590.1060593.
15. Cohen, G. Local Correlation Breakers and Applications to Three-Source Extractors and Mergers. 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, 2015, pp. 845–862. doi:10.1109/FOCS.2015.57.
16. Chattopadhyay, E.; Zuckerman, D. Explicit Two-source Extractors and Resilient Functions. Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing; ACM: New York, NY, USA, 2016; STOC 2016, pp. 670–683. doi:10.1145/2897518.2897528.
17. 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, 2009; pp. 179–189.
18. Halprin, R.; Naor, M. Games for Extracting Randomness. Proceedings of the 5th Symposium on Usable Privacy and Security; ACM: New York, NY, USA, 2009; SOUPS '09, pp. 12:1–12:12. doi:10.1145/1572532.1572548.
19. Voris, J.; Saxena, N.; Halevi, T. Accelerometers and Randomness: Perfect Together. Proceedings of the Fourth ACM Conference on Wireless Network Security; ACM: New York, NY, USA, 2011; WiSec '11, pp. 115–126. doi:10.1145/1998412.1998433.
20. The MathWorks, I. Uniformly distributed random numbers - MATLAB rand.
21. RANDOM.ORG. RANDOM.ORG - Byte Generator.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0038.v1

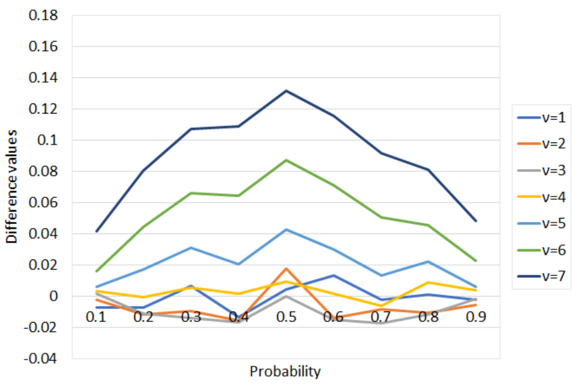
Peer-reviewed version available at *Entropy* **2018**, *20*, 729; doi:10.3390/e20100729



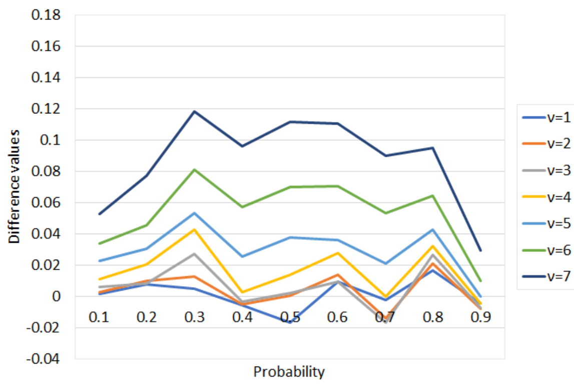
(a) 80-bit input sequences.



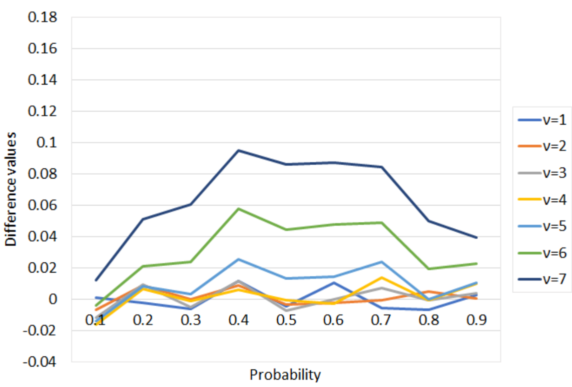
(b) 100-bit input sequences.



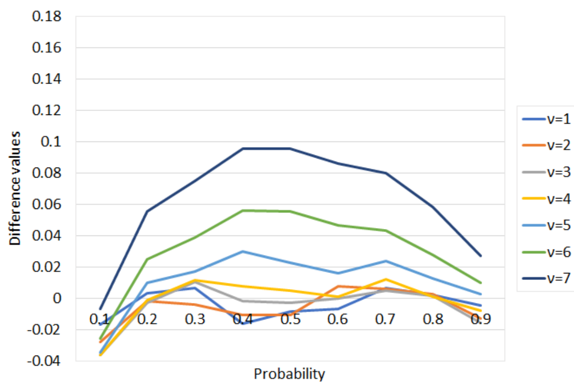
(c) 120-bit input sequences.



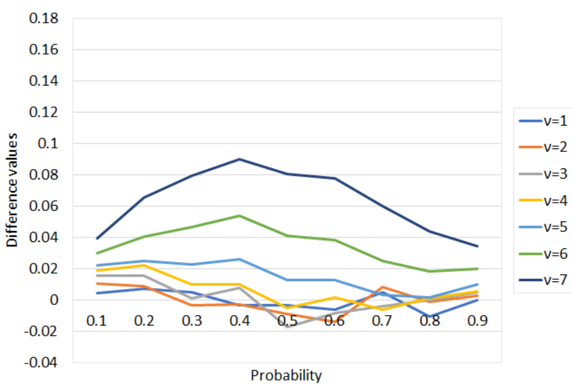
(d) 140-bit input sequences.



(e) 160-bit input sequences.



(f) 180-bit input sequences.



(g) 200-bit input sequences.

Figure B1. Difference values $f_{\nu}^{\mathsf{P}}(p,n) - f_{\nu}^{\mathsf{P}}(p)$ with $n = 80, 100, ..., 200$ and $1 \leq \nu \leq \lfloor \log n \rfloor$.