Numerical Analysis of Elias's and Peres's Deterministic Extractors

Amonrat Prasitsupparote
Graduate School of Environment and Information Sciences
Yokohama National University, Japan

Norio Konno
Department of Applied Mathematics
Faculty of Engineering
Yokohama National University, Japan

Junji Shikata
Graduate School of Environment and Information Sciences/
Institute of Advanced Sciences
Yokohama National University, Japan

***Abstract*—Generating truly random numbers is important in cryptography. Von Neumann proposed the simple procedure for extracting truly random bits from a sequence of independent, identically distributed random biased bits about half century ago. The improved algorithms of the von Neumann's extractor were later proposed by Elias and Peres. Peres showed that his extractor achieved the optimal rate if the length of input and the number of iterations tend to infinity. In contrast, Elias showed that his extractor achieved the optimal rate if the block size tends to infinity. Although these two algorithms are fundamental and important, their performance including the rates for reasonably finite input sequences is not analyzed in details. In this paper, we show numerical performance of Peres's extractor and Elias's one in terms of practical aspects. Our experimental results show that Peres's extractor is much better than Elias's one under the same input size and the almost same running time.**

***keywords*—True random number generation, von Neumann's extractor, Peres's extractor, Elias's extractor**

I. Introduction

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 the weak random numbers for generating random primes in embedded devices. This tells us that, in RSA-key generation random numbers are important, and RSA will be broken if there is not enough randomness to generate RSA keys. Therefore, the random number generation is important to generate cryptographically secure keys.

There are two basic types of generating random numbers: the true random number generator (TRNG) and the pseudo-random number generator (PRNG). TRNG is a deterministic algorithm which takes as input a biased random sequence generated by utilizing physical phenomena and transforms it into an unbiased random sequence. The PRNG is an algorithm that takes as input a biased random sequence and a seed being short and truly random and outputs a long pseudorandom sequence which is computationally indistinguishable from a truly random sequence. In securely constructing cryptographic protocols having *computational security* (e.g., public-key cryptography), PRNGs are usually used to generate secret-keys in the protocols, since PRNGs can output long random sequences. However, TRNGs are also important, since they are used for generating secret-keys in securely constructing cryptographic protocols having *information-theoretic security* or used even for generating seeds of PRNGs.

Related Work. A deterministic extractor is a deterministic algorithm which takes a non-uniformly random sequence as input and outputs a uniformly random sequence. The previous works[3], [4], [5] usually consider the case that input sequences are given by the Bernoulli source $\mathsf{Bern}(p)$, where $\mathsf{Bern}(p)$ outputs $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ which are i.i.d. (identically and independently distributed) according to $\Pr(x_i = 1) = p$ and $\Pr(x_i = 0) = q = 1 - p$ for some unknown $p \in (0,1)$. In this paper, we also consider that inputs of extractors are binary sequences output by $\mathsf{Bern}(p)$. The most famous extractor is the von Neumann's extractor [3]. He demonstrated a simple procedure for extracting independent *unbiased* bits from a sequence of i.i.d. and *biased* bits.

An extension of the von Neumann's extractor was proposed by Elias [4] in 1971. The basic idea behind the Elias's method is to utilize a block coding technique to improve redundancy of the von Neumann's extractor. Elias's method 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. To improve time and space complexity of the Elias's method, Ryabko and Matchikina [6] proposed a fast method of implementing the Elias's method which we call *RM method* in this paper. The RM method utilizes the enumerative encoding technique from [7] and Schönhage–Strassen algorithm [8] for fast integer multiplication in order to compute assignment of output sequences. By using the RM method, time complexity and space complexity of Elias's extractor are improved. In 1992, Peres [5] proposed a procedure for extracting independent unbiased bits from a sequence of i.i.d. and biased bits which is an improved one from the von Neumann's extractor. The basic idea in Peres's extractor is to reuse the bit sequence which is discarded in the von Neumann's extractor by iterating similar procedures in the von Neumann's extractor.

In 2013, Pae [9] compared rates of Peres's extractor and Elias's extractor for probability $p = \frac{1}{3}$. Pae concludes Elias's extractor requires more computational costs than Peres's extractor, but there is a trade off between computational costs and rates. However, Pae did not consider any $p \in (0,1)$ except for $p = \frac{1}{3}$, and one of purposes in this paper is to show results for any $p \in (0,1)$.

In 2016, Chattopadhyay and Zuckerman [10] proposed a general two-source extractor in which each source has a poly-logarithmic min-entropy. They combined two weak random sequences into a single sequence by using a K-Ramsey graph and a resilient function. Their extractor only outputs one bit and achieves negligible error.

978-1-5090-4780-2/17/$31.00

Furthermore, most researchers are interested in implementing a randomness extractor in a real world. For example, Bouda et al. [11] used mobile phones or pocket computers to generate random data close to being truly random in 2009. They took 12 pictures per second then used their function for 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, thus it would be difficult to use in a practical application.

Later in 2011, Voris et al. [12] investigated the use of accelerators on the RFID tags as a source. They implemented a two-stage extractor on the RFID tags. And, it can produce 128 bits random output in 1.5 seconds and passed the NIST statistical tests. However, they stored a Toeplitz matrix on the RFID tags when performing matrix multiplication operations, thus this process needs a large amount of memory on the RFID tags, though the RFID tags have limited memory.

Our Contribution. Recently, most researchers are interested in two-source or fast seeded extractors (i.e., not deterministic ones). On the other hand, since the deterministic extractors by von Neumann, Elias, and Peres are fundamental and simple, a few researchers are interested in such extractors, especially from viewpoints of practical use. For measuring the performance of a deterministic extractor, most researchers focus on the rate or redundancy. For example, the rate of von Neumann's extractor is pq that is far from $h(p)$, where $h(p)$ is the binary entropy function. The rate of Elias's extractor converges to $h(p)$ as the block size tends to infinity. In addition, Elias considered to take long block-size, thus the good rate is achieved when the block size is equal to the input length. Elias's extractor looks suitable in terms of the rate while it requires the exponential time and the exponential memory size with respect to the block size. In contrast, Peres considered the rate in his extractor and showed that it achieved the optimal rate if the length of input and the number of iterations tend to infinity. However, we cannot know an actual rate for finite input sequences in a real world. Peres's extractor requires small space complexity and time complexity, thus it looks suitable in terms of practical use. From the observation above, it is not easy to conclude which one is the more suitable extractor for practical use in general. Therefore, we show the comparison of both extractors under the same environments, that is, under the same time complexity and a finite input sequence with any biased probability. Firstly, we explain our implementation of both extractors in Section III. After that, we investigate time complexity of both extractors with input sequence of length $n = 100, 200, \ldots, 1000$. As a result, we show that Peres's extractor with iterations $\nu = 6$ and Elias's extractor with block size $N = 6$ have almost same running time. Also, we compare the redundancy of both extractors under the almost same running time, and we show that Peres's extractor is much better than Elias's one under the almost same running time.

II. Preliminaries

A deterministic extractor is a deterministic algorithm which takes non-uniformly random sequences (i.e., biased sequences) as input and outputs uniformly random sequences (i.e., unbiased ones). For simplicity, the previous works [3], [4], [5] usually consider the case that input sequences are given by the Bernoulli source $\mathsf{Bern}(p)$ where $\mathsf{Bern}(p)$ outputs an i.i.d. sequence $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ according to $\Pr(x_i = 1) = p$ and $\Pr(x_i = 0) = q = 1 - p$ for some unknown $p \in (0,1)$. In this paper, we also consider that inputs of extractors are binary sequences output by $\mathsf{Bern}(p)$. Suppose that a deterministic extractor takes $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ as input and outputs $(y_1, y_2, \ldots, y_\ell) \in \{0,1\}^\ell$. Then, we denote the average bit-length of outputs by $\bar{\ell}(n)$ which is a function of n, and define the rate function by $r(p) := \lim_{n\to\infty} \bar{\ell}(n)/n$.

In addition, we define the redundancy function by $f(p) := h(p) - r(p)$, where $h(p)$ is the binary entropy function defined by $h(p) = -p\log p - (1-p)\log(1-p)$. We note that the above definition of redundancy functions is meaningful, since $h(p)$ is shown to be the information bound in [4], [5]. In this paper, we define non-asymptotic functions $r(p,n) := \bar{\ell}(n)/n$, and $f(p,n) := h(p) - r(p,n)$. Furthermore, we define the maximum redundancy by $\Gamma := \sup_{p\in(0,1)} f(p)$ and $\Gamma(n) := \sup_{p\in(0,1)} f(p,n)$ for every n.

A. von Neumann's extractor

In 1951, von Neumann [3] proposed the simple procedure for extracting independent *unbiased* bits from a sequence of i.i.d. and *biased* bits, which is called the von Neumann's extractor. This method divides the input sequence $(x_1, x_2, x_3, x_4, \ldots)$ into the pairs[1] $((x_1x_2), (x_3x_4), \ldots)$ and considers a mapping defined by

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

where the symbol $\wedge$ denotes no output bit. Then, the extractor outputs the unbiased sequence obtained by concatenating the outputs of the mapping (1).

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

Redundancy. The von Neumann's extractor is not desirable, since maximum redundancy is far from zero. Actually, the rate function $r^{\mathsf{vN}}(p)$ of the von Neumann's extractor is evaluated by $r^{\mathsf{vN}}(p) = \lim_{n\to\infty} npq/n = pq$, which is $1/4$ at $p = q = 1/2$ and less elsewhere. In addition, the redundancy function $f^{\mathsf{vN}}(p)$ and maximum redundancy Γ^{vN} are evaluated as $f^{\mathsf{vN}}(p) = h(p) - p(1-p), \Gamma^{\mathsf{vN}} = \sup_{p\in(0,1)} f^{\mathsf{vN}}(p) = f^{\mathsf{vN}}(1/2) = 3/4$.

B. Elias's extractor

An extension of the von Neumann's extractor was proposed by Elias [4] in 1971. The basic idea behind the Elias's method is to utilize a block coding technique to improve redundancy of the von Neumann's extractor. Let N be the block size used in Elias's extractor, and it is a natural number $N \in \mathbb{N}$ with $N \geq 2$. For all binary sequences with length N, we first 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, we note that each sequence of S_k is equiprobable, i.e., the probability is $p^k q^{N-k}$.

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

Let $|S_k| = \binom{N}{k} = \alpha_m 2^m + \alpha_{m-1}2^{m-1} + ... + \alpha_0 2^0$ ($\alpha_i \in \{0,1\}$) be the binary expression of the integer $|S_k|$, and we briefly write $|S_k| = (\alpha_m, \alpha_{m-1}, \ldots, \alpha_0)$ for it. For each j ($1 \le j \le m$) 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 $\wedge$.

Example 1: Suppose that the block size is $N = 4$. Then, we partition the set $\{0,1\}^4$ of possible input sequences into the following subsets:

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

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

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

Suppose that an input sequence $x = (1,0,0,1,0,0,1,1)$ is given. Since $N = 4$, the sequence is divided as $x = ((1,0,0,1),(0,0,1,1))$. Then, by the assignment of output sequences above, the output sequence is $y = ((1,1)(0,1)) = (1,1,0,1)$.

Note that Elias's extractor with $N = 2$ is equivalent to the von Neumann's extractor, or equivalently the mapping (1). In this sense, Elias's extractor is an extension of the von Neumann's extractor.

Redundancy. In general, the rate function and redundancy function of the Elias's extractor depend on block size N. For given n-bit input sequence, if we take the maximum block size $N := n$, the rate function and maximum redundancy would be best. For simplicity, we assume that $N = n$ in the following explanation. The rate function $r^{\mathsf{E}}(p, N)$ is evaluated by

$$
r^{\mathsf{E}}(p,N) \approx \frac{1}{N}\sum_{k=0}^{N}\binom{N}{k}p^k(1-p)^{N-k}\log\binom{N}{k} \qquad (2)
$$

Elias [4] showed that the rate function[2] $r^{\mathsf{E}}(p,N)$ 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.

[2]In Elias's paper [4], it is called *efficiency*.

Complexity. A naive implementation of the Elias's extractor requires much space complexity and time complexity to make a table of the assignment of output sequences as illustrated by Example 1. Actually, it requires exponential time and exponential memory size with respect to N. Ryabko and Matchikina [6] proposed a fast method of implementing the Elias's extractor which we call *RM method* in this paper. The RM method utilizes enumerative encoding technique from [7] and Schönhage–Strassen algorithm [8] for fast integer multiplication in order to compute assignment of output sequences instead of making the large table as illustrated by Example 1. The RM method is executed as follows. Suppose that a binary input sequence $x^N = (x_1, x_2, \ldots, x_N)$ contains k ones and $N - k$ zeros. Then, the number $\mathrm{Num}(x^N)$ is defined by

$$
\mathrm{Num}(x^N) = \sum_{t=1}^{N}\binom{x_t N - t}{k - \sum_{i=1}^{t-1} x_i}. \qquad (3)
$$

Then, a binary codeword $code(x^N)$ of x^N, which is assignment of an output sequence of x^N, is computed as follows:

(i) Compute $\mathrm{Num}(x^N)$ in the set S_k, if x^N contains k ones.
(ii) Let $|S_k| = \binom{N}{k} = 2^{j_0} + 2^{j_1} + ... + 2^{j_m}$ for $0 \le j_0 < j_1 < ... < j_m$.
(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 $\mathrm{Num}(x^N)$.

By using the RM method, time complexity and space complexity of Elias's extractor are improved as follows: Time complexity is $O(N \log^3 N \log\log N)$, and space complexity is $O(N \log^2 N)$ (see [6] for details).

C. Peres's extractor

The basic idea in Peres's extractor is to reuse the bit sequence which is discarded in the mapping (1) to improve redundancy of the von Neumann's extractor. 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 $x_{2m_i-1} \ne x_{2m_i}$ with $m_i \le n/2$. In Peres's extractor, Ψ_ν ($\nu \ge 2$) is defined inductively as follows: For an even n,

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

where $*$ is concatenation; $u_j = x_{2j-1} \oplus x_{2j}$ for $1 \le j \le n/2$; $v_s = x_{2i_s - 1}$ and $i_1 < i_2 < \cdots < i_{\frac{n}{2}-\ell}$ are all the indices satisfying $x_{2i_s-1} = x_{2i_s}$ with $i_s \le n/2$. 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, in Peres's extractor, the number of iterations ν is at most $\lfloor \log n \rfloor$, since Ψ_ν for every $\nu \ge 2$ 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, and Peres's extractor with a large ν (say, $\nu = \lfloor \log n \rfloor$) is considered to be an elegantly improved version from von Neumann's one by utilizing a recursion mechanism.

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

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

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

Redundancy. The rate function $r_\nu^{\mathsf{P}}(p)$ of the procedure Ψ_ν can be computed inductively by the equation

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

for $\nu \ge 2$, and $r_1^{\mathsf{P}}(p) = pq$. Note that $r_1^{\mathsf{P}}(p)$ is the rate of the von Neumann's extractor. It is shown in [5] that $r_\nu^{\mathsf{P}}(p) \le r_{\nu+1}^{\mathsf{P}}(p)$ for all $\nu \in \mathbb{N}$ and for all $p \in (0,1)$, 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: The redundancy function $f_\nu^{\mathsf{P}}(p) = h(p) - r_\nu^{\mathsf{P}}(p)$ satisfies

$$f_\nu^{\mathsf{P}}(p) = \frac{1}{2} f_{\nu-1}^{\mathsf{P}}(p^2+q^2) + \frac{1}{2}(p^2+q^2) f_{\nu-1}^{\mathsf{P}}\left(\frac{p^2}{p^2+q^2}\right), \quad (7)$$

for $\nu \ge 2$ 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 $f_\nu^{\mathsf{P}}(p) \ge f_{\nu+1}^{\mathsf{P}}(p)$ for all $\nu \in \mathbb{N}$ and for all $p \in (0,1)$, and $\lim_{\nu\to\infty} f_\nu^{\mathsf{P}}(p) = 0$ uniformly in $p \in (0,1)$.

III. Numerical Analysis by Implementation

To evaluate the performance of Peres's extractor and Elias's one with RM method, we use Java language version 1.8 with Intel 3.70 GHz RAM 4 GB. We consider three questions:

(i) Do theoretical and experimental redundancy of the Peres's extractor and Elias's one with RM method show the same results?
(ii) What is the actual difference of running time required in Peres's extractor and Elias's one with RM method?
(iii) Under the almost same running time, which extractor is better in terms of redundancy?

We perform analysis to answer the questions as follows. To answer (i), we analyze redundancy of Peres's extractor in Section III-A and Elias's extractor with RM method in Section III-B. We use a pseudorandom number generation program **rand()** in MATLAB [13] to generate biased input sequences by controlling the probability. The reasonability of using **rand()** is explained as follows:

- We can control the probability p for each input sequence in our experiments.
- If theoretical and experimental redundancy results are almost the same, we can rely on **rand()** and can use it to generate biased input sequences.

Section III-A uses the probability $p = 0.001, 0.002, \ldots, 0.999$ and the number of iterations satisfies $\nu \le \lfloor \log 180 \rfloor = 6$. And, Section III-B uses probability $p = 0.1, 0.2, \ldots, 0.9$ and block size $N = 10, 20, 30, 60, 90, 180$. In addition, we use Schönhage–Strassen algorithm of fast multiplication for computing $\binom{N}{k}$. We consider the following:

1) Schönhage–Strassen multiplication algorithm requires $O(N^{1+\epsilon})$ which is asymptotically better than $O(N^2)$ (i.e., the normal multiplication). However, the advantage of Schönhage–Strassen method over the normal multiplication seems to appear when N is large enough. In addition, there may be the case that Schönhage–Strassen multiplication algorithm is not supported in some software, and in this case users need to implement it by themselves.
2) We want to avoid multiplication operations and use only addition operations, since it is simple and makes the basic operations lighter, so that it can be used in various applications and environments.

We use the recursive formula $\binom{N}{k} = \binom{N-1}{k-1} + \binom{N-1}{k}$ for $10 \le N \le 180$ in order to compute $\binom{N}{k}$ only by additions. For our experiment in Sections III-A and III-B, we generate 180-bit input sequences 100 times for each probability p. Then, we calculate the average on the redundancy function $f_\nu^{\mathsf{P}}(p)$ and $f^{\mathsf{E}}(p, N)$ for each probability p.

To answer (ii), we analyze running time of both extractors in Section III-C. This experiment does not use the probability p as a parameter, thus we change the random number generator to RANDOM.ORG [14]. It produces sequences close to being true random with unknown probability p by using randomness of atmospheric noises. And, it provides 131,072 random bits in each time, thus we take random sequences with bit-length $n = 100, 200, \ldots, 1000$ 100 times for each n, then we calculate the average on the running time.

To answer (iii), we compare the redundancy of both extractors under the almost same running time in Section III-D. We can clarify which is better in practice in terms of redundancy under the almost same time complexity.

A. Analysis of redundancy of Peres's extractor

We show the redundancy of Peres's extractor from theoretical aspects in Fig. 1. We calculated $f_\nu^{\mathsf{P}}(p)$ by using (7) with $\nu = 1, 2, \ldots, 6$ and $p = 0.1, 0.2, \ldots, 0.9$, then depicted the graphs of $f_\nu^{\mathsf{P}}(p)$, where x-axis means probability p and y-axis means redundancy. It can be seen that, the redundancy becomes smaller as the number of iterations becomes bigger, for all $p \in (0,1)$.

In Fig. 2, we show experimental redundancy of Peres's extractor with 180-bit inputs. The results are almost the same as theoretical ones in Fig. 1.

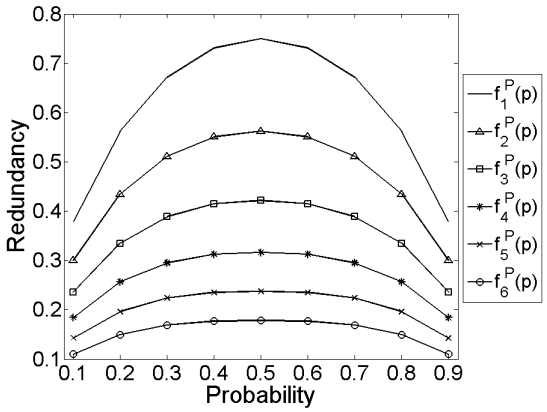

Fig. 1: Theoretical estimate of Peres's extractor.

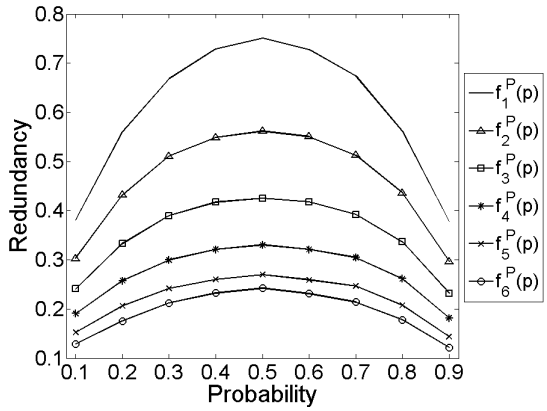

Fig. 2: Experimental estimate of 180-bit Peres's extractor.

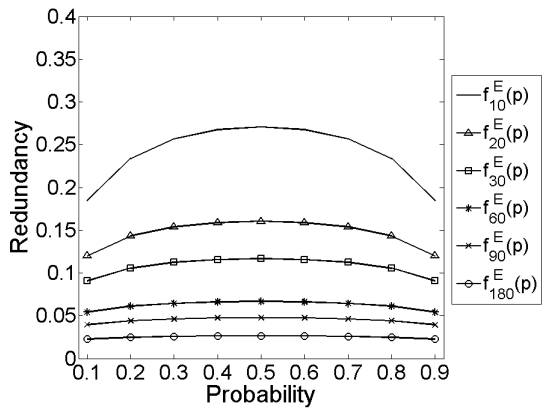

Fig. 3: Theoretical estimate of Elias's extractor with RM method.

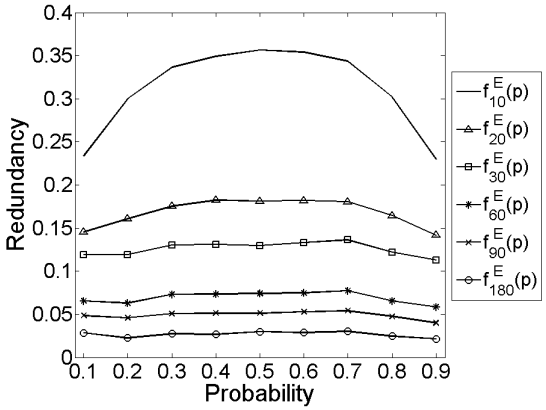

Fig. 4: Experimental estimate of 180-bit Elias's extractor with RM method.

B. Analysis of redundancy of Elias's extractor with RM method

In Fig. 3, we calculated $f^{\mathsf{E}}(p, N) = h(p) - r^E(p, N)$ by using (2) with $p = 0.1, 0.2, \ldots, 0.9$ and $N = 10, 20, 30, 60, 90, 180$. It can be easily seen that redundancy becomes smaller as block size becomes larger. Although there is slight difference between theoretical estimate in Fig. 3 and experimental estimate in Fig. 4, the experimental redundancy in our implementation is similar to the theoretical redundancy.

C. Analysis of running time of both extractors

Fig. 5 shows running time of Peres's extractor with $\nu = 1, 2, \ldots, 6$. As the number of iterations becomes larger, the running time is required more. In addition, the running time increases almost linearly but slope depends on ν, as supported by theoretical estimate of time complexity $O(\nu n)$. Furthermore, the running time of Peres's extractor with all parameters in our experiment is at most 0.32 milliseconds, which implies that the Peres's extractor is quite practical.

Fig. 6 shows running time of Elias's extractor with RM method with $N = 2, 4, 6, 8, 10, 12, 16, 20$. As the block size becomes larger, the running time is required more. The running time increases almost linearly but slope depends on N, as time complexity can be theoretically evaluated as $(n/N) \cdot O(N^2 \log N) = O(nN \log N)$. Furthermore, the running time of Elias's extractor with RM method with all parameters in our experiment is at most 8.1 milliseconds.

By comparing the running time of both extractors, we can conclude that the Peres's extractor is faster than the Elias's one with RM method for the same bit-length of inputs.

D. Comparison of redundancy under the almost same running time

By Fig. 5 and 6, the running time of Peres's extractor with $\nu = 6$ is almost the same as Elias's one with RM method with block size $N = 6$. Hence, we compare $f_6^{\mathsf{P}}(p)$ and $f^{\mathsf{E}}(p, 6)$ in terms of practical aspects, and our result is shown in Fig. 7. It is shown that Peres's extractor is much better than Elias's one with RM method in terms of redundancy.

Furthermore, we depicted the graphs of $f^{\mathsf{E}}(p, N)$ with $N = 10, 20$ in addition to $f^{\mathsf{E}}(p, 6)$ in Fig. 7. This result shows that: even if $f^{\mathsf{E}}(p, 10)$ is allowed to use, $f_6^{\mathsf{P}}(p)$ is better than $f^{\mathsf{E}}(p, 10)$; if $f^{\mathsf{E}}(p, 20)$ is allowed to use, $f_6^{\mathsf{P}}(p)$ is almost the same as $f^{\mathsf{E}}(p, 20)$, though running time of Elias's extractor with RM method with block size $N = 10, 20$ is quite larger than that of Peres's extractor with $\nu = 6$.

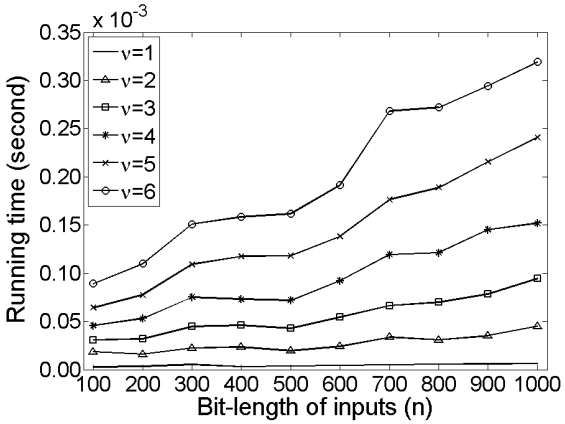

Fig. 5: Running time of Peres's extractor.

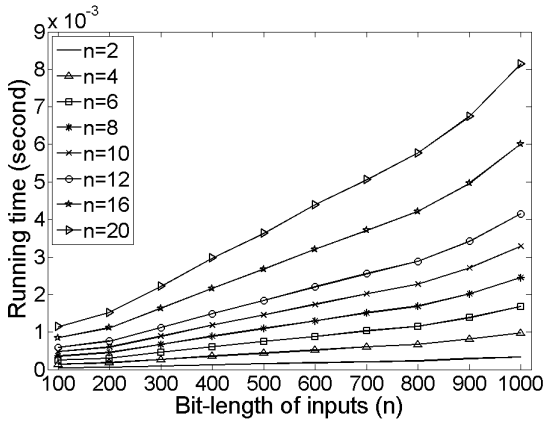

Fig. 6: Running time of Elias's extractor with RM method.

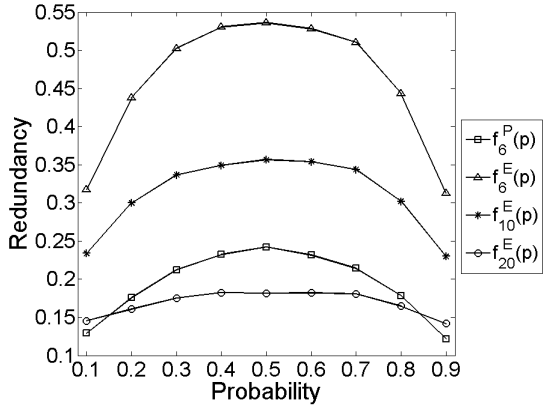

Fig. 7: Comparison of redundancy of Peres's extractor and Elias's one with RM method for 180-bit inputs.

IV. Conclusion

One may think that the rate of Elias's extractor is better than Peres's one, while Peres's extractor is superior to Elias's one in practical use in terms of the time complexity and the space complexity. In this paper, we evaluated numerical performance of Peres's extractor and Elias's one with RM method in term of practical aspects. For analysis of the redundancy, we used a pseudo-random number from **rand()** in MATLAB to generate 180-bit biased input sequences by controlling the probability p, then calculated the redundancy of both extractors in term of theoretical and experimental aspects (see Fig. 1-4). Our result shows that the theoretical and experimental redundancy are almost the same in both extractors. For analysis of the running time, our result shows that the redundancy of Peres's extractor was much better than Elias's one with RM method under the almost same running time. Consequently, Peres's extractor will be better to use in applications such as cryptography.

Acknowledgment

This work was in part supported by JSPS KAKENHI Grant Number 15H02710, and in part conducted under the auspices of the MEXT Program for Promoting the Reform of National Universities.

References

[1] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your Ps and Qs: Detection of widespread weak keys in network devices," in *Proceedings of the 21st USENIX Security Symposium*, Aug. 2012.

[2] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter, "Public Keys," in *Advances in Cryptology – CRYPTO 2012*, ser. Lecture Notes in Computer Science, R. Safavi-Naini and R. Canetti, Eds. Springer Berlin Heidelberg, 2012, no. 7417, pp. 626–642.

[3] J. v. Neumann, "Various Techniques Used in Connection with Random Digits, Notes by G E Forsythe," *National Bureau of Standards Applied Math Series*, vol. 12, pp. 36–38, 1951.

[4] P. Elias, "The Efficient Construction of an Unbiased Random Sequence," *Ann. Math. Statist.*, vol. 43, no. 3, pp. 865–870, Jun. 1972.

[5] Y. Peres, "Iterating Von Neumann's Procedure for Extracting Random Bits," *Ann. Statist.*, vol. 20, no. 1, pp. 590–597, Mar. 1992.

[6] B. Ryabko and E. Matchikina, "Fast and efficient construction of an unbiased random sequence," *IEEE Transactions on Information Theory*, vol. 46, no. 3, pp. 1090–1093, May 2000.

[7] T. Cover, "Enumerative source encoding," *IEEE Transactions on Information Theory*, vol. 19, no. 1, pp. 73–77, Jan. 1973.

[8] A. Schönhage and V. Strassen, "Schnelle multiplikation großer zahlen," *Computing*, vol. 7, no. 3, pp. 281–292, 1971.

[9] S.-i. Pae, "Exact output rate of Peres's algorithm for random number generation," *Information Processing Letters*, vol. 113, no. 5–6, pp. 160–164, Mar. 2013.

[10] E. Chattopadhyay and D. Zuckerman, "Explicit Two-source Extractors and Resilient Functions," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2016. New York, NY, USA: ACM, 2016, pp. 670–683.

[11] J. Bouda, J. Krhovjak, V. Matyas, and P. Svenda, "Towards True Random Number Generation in Mobile Environments," in *Identity and Privacy in the Internet Age*, ser. Lecture Notes in Computer Science, A. Jøsang, T. Maseng, and S. J. Knapskog, Eds. Springer Berlin Heidelberg, Oct. 2009, no. 5838, pp. 179–189, dOI: 10.1007/978-3-642-04766-4_13.

[12] J. Voris, N. Saxena, and T. Halevi, "Accelerometers and Randomness: Perfect Together," in *Proceedings of the Fourth ACM Conference on Wireless Network Security*, ser. WiSec '11. New York, NY, USA: ACM, 2011, pp. 115–126.

[13] "Uniformly distributed random numbers - MATLAB rand." [Online]. Available: http://www.mathworks.com/help/matlab/ref/rand.html

[14] "RANDOM.ORG - Byte Generator." [Online]. Available: http://www.random.org/bytes/