Tatra Mt. Math. Publ. **25** (2002), 91–107

TATRA MOUNTAINS
Mathematical Publications

EXTRACTION OF RANDOM BITS FOR CRYPTOGRAPHIC PURPOSES

István Vajda

ABSTRACT. Extraction of randomness from random binary sources is considered for cryptographic purposes. A quality measure is suggested for comparison of the performance of extractors. Practical problems are pointed out with the accuracy of entropy estimation, which estimate is a basic input parameter of extractors. Deterministic hashing as an extractor is shown to be effective on average, where the average is taken over all input distributions. Example designs are also presented.

1. Introduction

Assume that we can observe the output of a random binary source. The ultimate aim is to extract as many truly random bits as we can from the observed sequence of bits at the output. When we say that truly random bits are extracted, we mean a sequence of independent, identically distributed, balanced binary random variables; for short a coin tossing sequence. On the one hand, there is a common practice, where cryptographic hashing is hoped to provide a feasible and robust method for extraction without theoretical statements about the performance. On the other hand, computer science has produced a fast growing deep theory of randomized extractors with a multitude of theorems on their quality. however, these results have not been followed by feasibility studies.

A deterministic extractor, EXT is a mapping of an n-bit source block η into an m-bit output block ξ, $\mathrm{EXT}\colon \{0,1\}^n \to \{0,1\}^m$, where $m < n$. A randomized extractor, EXT_r also uses d additional truly random bits at its input: $\mathrm{EXT}_r\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$. Therefore we need an auxiliary input source, which provides these truly random bits for the randomized extractor. (Note the practical disadvantage that we have to assume the possession of truly random bits in advance, while our aim is to produce such.)

2000 Mathematics Subject Classification: 68P25, 94A17.
Keywords: pseudorandom generators, extractors, data encryption, entropy estimation.

The probability distribution (p.d.) of η is denoted by $P = \{p_0, p_1, \ldots, p_{2^n-1}\}$, the p.d. of ξ is $Q = \{q_0, q_1, \ldots, q_{2^m-1}\}$. Two kinds of measures are considered for the evaluation of the quality of extractors. The first is based on the variational distance (statistical distance)[1], the second on Shannon entropy [9]. If our aim is to use the extracted random bits in cryptographic applications (e.g., for generating encryption keys) then we are mainly interested in their Shannon entropy. This quantity should be held above a particular level to prevent exhaustive search of the key space.

Typically, the statistical properties of the source are unknown a priori, therefore we need algorithms which can extract truly random bits from arbitrary sources. The basic result of the theory of extractors states that all min-entropy[2] can be extracted asymptotically by randomized extractors. An introduction to early results can be found in [7], while the basic idea of polynomial time construction is given in [8]. This extractor is universal in the sense that the same extraction algorithm can be used for all distributions with the same min-entropy. In accordance, we call an extraction algorithm non-universal if the algorithm has to be fit to the actual distribution, i.e., all nonzero probabilities have impact on its design.

The every day practice is in contrast to these complex theoretical approaches:

The simplest and widespread practical approach to improve the quality of random bit streams is the usage of the simple XOR operation. The well known and provable effect of such an operation is the improvement in bias of biased i.i.d. binary sources. XOR-ing is an important building block in practical physical random generators driven by noise processes. A generalization of this concept is *linear hashing* [6].

Because of its availability in the PC world, cryptographic hash functions are widely used to compress a random binary block with arbitrary size to an output block with fixed size. Typical usage of these functions is the production of a digest (fingerprint) of a document, e.g., in a digital signature protocol. When the document varies, its digest also varies in an intractable way. The success of this application contributes to the expectation, that - informally speaking—such a deterministic mapping can effectively transfer the variability ('entropy') from the input into the output. The output ('extracted entropy') is accumulated into a so called 'entropy pool', from which applications can obtain 'random bits' on request [2], [10], [11].

[1] The variational distance as a measure for the quality of extractors is the distance of the output p.d. Q from the uniform p.d. U: $\mathrm{dist}(Q,U) = \sum_x \left|Q(x) - 1/2^m\right|$, where the summation is taken over all binary sequences of length m.

[2] A distribution P on $\{0,1\}^n$ is said to have min-entropy $H_{\min}(P) = k$, if $P[\eta = x] \leq 2^{-k}$ for all $x \in \{0,1\}^n$.

Theoretical papers on extractors written by computer scientists use the informal notion of 'weak random source' without an effort for an exact definition. In our view it simply means that such a source is 'far in quality' from random sources based on well behaving physical noise sources, modelled by a stationary source of independent binary random variables.

The structure of the paper is as follows. In Section 2, we consider quality measures for comparison of extractors. Universality and asymptotic optimality of randomized extractors tacitly assumes that we are able to give an accurate estimate on min-entropy. In Section 3, we examine the trade off between accuracy and sample size, as well as its consequences. In Section 4, we point out that deterministic hashing as an extractor can be considered effective in the average sense where the average is taken over all input distributions. Section 5 presents two example designs. The first is a simple construction for a non-universal, randomized extractor. The second is an experiment with a Trevisan-like randomized extractor.

Hereinafter by *entropy* we shall mean "Shannon-entropy".

2. Measuring the quality of extractors

The total entropy is $H_{\text{in}} = H(P) + d$ at input, where $d = 0$ for deterministic extractors. The aim is to maximize the output entropy $H_{\text{out}} = H(Q)$ together with the maximization of the entropy rate $H_{\text{rate}} = H_{\text{out}}/m$ at the output. Informally speaking, we would like to 'pump' as much entropy from the input to the output as we can while attaining as high rate of entropy at the output as we can. Note that the first part of this task alone is easy: an identity transformation 'extracts' all input randomness. Because of the added truly random bits (added randomness), this measure needs further refinement in the case of randomized extractors. The net extracted entropy is $H'_{\text{out}} = H_{\text{out}} - d = H(Q) - d$. Consequently, when we compare extractors applied to the same source S and input dimension n, the pair

$$[H'_{\text{out}},\ H_{\text{rate}}]$$

could be used as a measure of quality. Note that this definition of quality overlooks the "cost" of added entropy: if an extractor provides the same quality as a second one, then the extractor consuming less added entropy should be considered better. In this paper we will only use the pair $[H'_{\text{out}},\ H_{\text{rate}}]$ as quality measure. Extractors can be partially ordered by this measure of quality. Figure 1 shows the corresponding coordinate plane, where the points (bullets) represent different qualities. It means that an extractor with quality $[H'_{\text{out}}\text{\_}1,\ H'_{\text{rate}}\text{\_}1]$ is better than another one with quality $[H\prime_{\text{out}}\text{\_}2,\ H'_{\text{rate}}\text{\_}2]$ if $H'_{\text{out}}\text{\_}1 > H'_{\text{out}}\text{\_}2$ and $H'_{\text{rate}}\text{\_}1 > H'_{\text{rate}}\text{\_}2$. For instance, in Fig. 1 an extractor corresponding to point (6) is better than the one with parameters represented by point (5). However, extractors (6) and (7) are not comparable by this measure. Some resolution of this problem is the following: the ultimate quality of the output is given by the output rate of entropy H_{rate}, which—informally speaking—shows how close are the output bits to truly random selection. The quantity H_{out} first of all indicates the amount of extracted random bits, i.e., how well the extractor squeezes all the entropy from the input.

A few interesting points can be identified in this plane. Quality $\left[H(P), 1\right]$ (point (1)) is the overall optimum for given dimension parameters and source. In terms of source coding this quality corresponds to the case of ideal compression. Point (8) represents the trivial case of identity transformation with quality parameter $\left[H(P), H(P)/n\right]$. The quality point of $\left[H_{\min}(P), 1\right]$ (point (2)), corresponds to the asymptotic quality reachable by randomized extractors. Recall that min-entropy is a lower bound to Shannon-entropy, where equality holds if and only if P is uniform. For instance for p.d. $P = \{1/2, 1/8, 1/8, 1/8, 1/8\}$ we obtain $H(P) = 11/6$ and $H_{\min}(P) = 1$. Unfortunately the difference between these two entropy measures can (in principle) be arbitrarily large. Therefore if the input distribution is far from uniform, then extraction of all min entropy results in a much smaller number of random bits than the amount of randomness measured by the entropy.

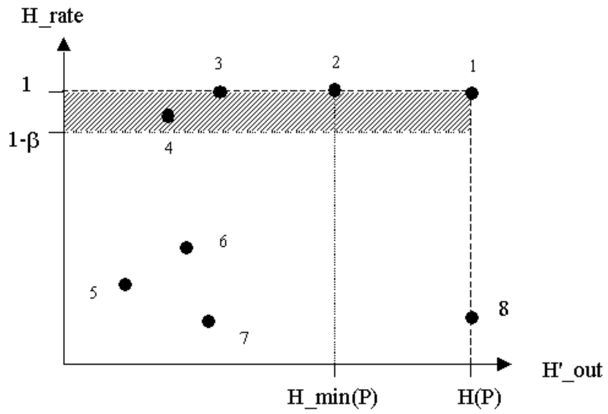

FIGURE 1. A coordinate plane for comparison of different extractors.

Points on the upper borderline of Fig. 1. can be achieved for finite n for some special kind of sources. For specific classes of sources (i.i.d. sources, finite state Markov sources) there are known feasible deterministic algorithms for extraction of truly random bits [5]. Probably the simplest construction for an extractor was given by von Neumann: consider an i.i.d. binary source and divide its output into consecutive pairs of bits. Bit 1 is generated when pair 10 occurs, and bit 0 is produced for pair 01. The idea is to find symmetries and choose equally-probable events. In this way the construction can be generalized for input blocks longer than two bits. The von Neumann's rule produces truly random output with quality $\left[np(1-p), 1\right]$, where p is the probability of 0 at the output of the source.

For finite dimensions in general we have to be satisfied with an output rate within a tolerance region. This region is illustrated by the shadowed area in Fig. 1. For instance, if the key size is k bits and we assume that a log-size of the effective key space k' is already secure for a given environment, then $1-\beta = k'/k$ is an appropriate selection.

3. Extraction and estimation of entropy

All extraction algorithms—both universal and non-universal— assume good estimate of some entropy measure, the min-entropy or the Shannon entropy. This assumes preprocessing, i.e., an estimation procedure has to take place before the step of extraction. The success of this estimation depends on the statistical behavior of the source as well as the availability of a sufficient number of samples. If the samples are dependent and the source is general, then the assessment of the estimation accuracy is an intractable task. Therefore we have to be able to cut the n bit blocks from the output stream of the source in such a way that their independence is ensured. Failing at this level leads to distorted estimates, which makes seeking for perfect extraction pointless, and finely tuned, sophisticated algorithms might prove not better than robust ones.

Assume that we can observe L independent and identically distributed (i.i.d.) samples with p.d. P. Let $P' = \{p_{L,0}, p_{L,1}, \ldots, p_{L,N-1}\}$, let $N = 2^n$ denote the corresponding frequency distribution, where L is the sample size. The min-entropy is determined by the maximum of the probability distribution P, while the entropy is affected by all nonzero probabilities. Therefore, it might seem that it is easier to prepare accurate inputs for those extractors which need the min-entropy value. However, this guess is false:

Fact 1: It is not an advantage that an extractor algorithm needs an estimate *only* for the min-entropy in the sense of accuracy or the needed number of samples. A proof of this fact is shown in the Appendix.

In the rest of this section we give a confidence interval for min-entropy estimates, which provides an estimate for the number of needed samples as well as practical bounds on the feasible dimensions. From these results interesting consequences can be derived.

Proposition 1.

$$P\Big(\big|\max_j p_{L,j} - \max_j p_j\big| > \varepsilon\Big) \le 2\exp\big(-L\varepsilon^2/2\big) \tag{1}$$

for arbitrary $\varepsilon > 0$.

Proof. By Sheffe's theorem [3]

$$\max_A \left|\sum_{j\in A} p_{L,j} - \sum_{j\in A} p_j\right| = \frac{1}{2}\sum_j |p_{L,j} - p_j|$$

from here it follows that

$$P^*(L,\varepsilon) = P(\max_j p_{L,j} - \max_j p_j > \varepsilon)$$
$$\le P\left(\frac{1}{2}\sum_j |p_{L,j} - p_j| > \varepsilon\right) \simeq \exp\big(-Lg(2\varepsilon)\big),$$

here we applied the result

$$\lim_{n\to\infty} \frac{1}{L}\log P\left(\sum_j |p_{L,j} - p_j| > \varepsilon\right) = -g(\varepsilon),$$

where

$$g(\varepsilon) = \min_{0<p<1-\varepsilon/2} D(p, p+\varepsilon/2), \qquad D(\alpha,\beta) = \alpha\log\frac{\alpha}{\beta} + (1-\alpha)\log\tfrac{1-\alpha}{1-\beta}$$

([4]). Because $g(\varepsilon) \ge \varepsilon^2/2$ ([4]), the following simple exponential upper bound is obtained: $P^*(L,\varepsilon) \le \exp(-L\varepsilon^2)$. By the same technique, a similar upper bound can be obtained for probability

$$P\Big(\max_j p_j - \max_j p_{L,j} > \varepsilon\Big),$$

consequently

$$P(L,\varepsilon) = P\Big(\big|\max_j p_j - \max_j p_{L,j}\big| > \varepsilon\Big) \ge 2\exp\big(-L\varepsilon^2/2\big).$$

According to (1) we can give a confidence interval, $\big[\max_j p_{L,j} - \varepsilon, \max_j p_{L,j} + \varepsilon\big]$, on the maximum of p.d. P, which contains the maximum with probability less than the exponent given in the R. H. S. of inequality (1). □

Table 1. shows an illustration on the needed number of samples according to bound (1). It can be seen that even for rough estimates the needed sample size is in the range of a few thousands. For instance, if we collect 1 byte from

confidence	accuracy (ε)	sample size (L)
0.9	0.1	600
0.99	0.1	1060
0.995	0.1	1200
0.9	0.05	2396
0.99	0.05	4238
0.005	0.05	4793

TABLE 1. Sample size vs. accuracy of estimate.

an elementary user interaction, then for the extraction of a 16 byte key, about sixteen thousand interactions are needed to have enough number of samples. The aim of perfect extraction based on accurate entropy estimates practically contradicts the available accuracy of input parameters. Therefore robust, simpler designs may lead to the same performance.

In every day practice for random generation in PC's, several system variables are collected from the computer and from its input channels (keyboard, mouse, network). There variables are represented as binary strings. Concatenation of these strings results in a binary r.v. η of length n. Therefore assume that the input r.v. η is composed as $\eta = \left[\eta^{(1)}, \eta^{(2)}, \dots, \eta^{(r)}\right]$ which means that variable η as a random binary block of length n is a concatenation of independent components $\eta^{(i)}$, $i = 1, \dots, r$. We get

$$H(\eta) = \sum_i H\left(\eta^{(i)}\right). \tag{2}$$

Fortunately, a similar property is true also for the min-entropy, i.e.,

$$H_{\min}(\eta) = \sum_i H_{\min}\left(\eta^{(i)}\right). \tag{3}$$

Formulae (2) and (3) can be considered as rules for the *accumulation of entropy*. In this way, we have a better chance to give accurate entropy estimates, because the dimensions are reduced substantially.

We have to distinguish between a designed and the real quality of extraction. The designed quality should be based on a conservative estimate of entropy, i.e., a proper lower bound on the real value is desirable. If the real entropy values are larger than the estimated values, then the real quality will be at least as good as the designed one.

4. Extraction using deterministic hashing

Practically, the entropy of the input random variable can practically never be extracted perfectly by deterministic mappings, because for a non-invertible deterministic mapping F the inequality $H\big(F(\eta)\big) < H(\eta)$ holds ([9]), i.e., in our notation $H_{\mathrm{out}} < H_{\mathrm{in}}$. A further negative fact is the following:

Fact 2: No deterministic mapping can be universal, i.e., applicable for all possible distributions. Even worse: it is easy to construct an input p.d. to any mapping, for which it performs the worst, i.e., produces zero entropy at the output.

Proof. Because $m < n$, there is an m-tuple into which more n-tuples are mapped. Let the input p.d. P be concentrated on the set of those n-tuples.

With this fact in mind, we might try to fit an optimal mapping to a given p.d. P: let a mapping $F\colon \{0,1\}^n \to \{0,1\}^m$, $m < n$ be called P-optimal for input p.d. P, if for this mapping the variational distance between the output p.d. Q and the uniform p.d. U is the minimum possible. The problem is finite, and this optimum exists. In principle, it would be a straightforward solution if we had an effective deterministic algorithm which, for any input (n, m, P), would output a mapping F (and a corresponding p.d. Q), for which $\operatorname{dist}(U, Q)$ is minimal. Unfortunately, this problem is hard and no effective solution is expected. To see this, consider the smallest case of $m = 2$. This latter task can be reduced to the classical knapsack problem: assume, a genie helps us by revealing the optimal output distribution $(q, 1-q)$, so we can assume that the optimal sum of weights (atoms) in the two knapsacks. Our task is to give an algorithm which sorts the weights for these two knapsacks to reach this a priori known sums. This task is known to be NP complete, in general we can do no better than trying all possible selections. Unfortunately, this task is not feasible even for parameters as small as $n = 8$, i.e., when byte size input is considered. Therefore we obtain:

Fact 3: Finding the P-optimal non-universal deterministic mapping is not feasible.

It seems a plausible requirement that an extractor $\mathrm{EXT}\colon \{0,1\}^n \to \{0,1\}^m$, $m < n$ should transform truly random input into truly random output. This means, that if the mapping happens to get input from a perfect random source, then it shouldn't waste more entropy than what is unavoidable by its output dimensions. This requirement of uniformity is met if and only if the preimage partitioning of the EXT mapping is uniform. Linear affine hash functions guarantee uniformity, however, cryptographic hashing algorithms do not.

"The mother of all extractors"—as it is called in [7]—is a technique for randomized extraction, when a hash function is selected randomly from a family $H = \big\{h\colon \{0,1\}^n \to \{0,1\}^m\big\}$ of hash functions. Random selection of a hash mapping to a fixed input p.d. or a random selection of an input p.d. to a fixed mapping are essentially dual approaches. Proposition 2 below supports the observation, that on average, a high proportion of the entropy can be extracted by deterministic hashing, where averaging is taken over all, randomly selected input distributions. In this sense the effectiveness of deterministic hashing means an average performance.

Let us imagine the following model for a virtual, global hash operation: around the globe there is a huge number of computers using the same hash function (say SHA) for random generation. These generators are fed by random inputs from different sources. This virtual global extractor can be modeled by a single hash function driven by randomly selected input distributions. The nature 'tosses a coin' and selects the actual source, i.e., all kinds of extractors implicitly or explicitly assume the presence of truly random bits in their model.

Let K denote the number of nonzero probabilities (atoms) of an input distribution P. Assume that the input variable has a distribution randomly selected from such kind of distributions. The performance of extraction is affected by the relationship between parameters $k = \log_2(K)$ and m. In Proposition 2 extremal cases are treated, when $k < m/2$ and $k > m$. □

PROPOSITION 2.

i) If $k < m/2$ then the input entropy is essentially fully extracted.

ii) Assume that the input p.d. for atoms has constant probability $1/K$. If $k > m$ and m increases, the average rate of output entropy tends to 1.

Proof.

i) The proof follows from the birthday paradox. Though we can extract the entropy almost entirely, the entropy rate at the output is low. In fact, it is less than $1/2$.

ii) Consider the case, when m is fixed and k grows, i.e., the output dimension of the hash function is fixed and an increasingly longer input stream is fed into it. If the input p.d. is selected at random, the same will be true for the output p.d. (under the natural assumption that there is at least one input value which is transformed into any given output value). Therefore we consider a random selection of output p.d. $Q = \{q_0, q_1, \dots, q_M\}$ for a given M.

Straightforward combinatorial counting can be applied for all selections of parameters k and m (i.e., there are $K = 2^k$ constant atoms in the p.d. P). The corresponding formulas are given in the Appendix. Table 2 shows evaluation figures. The rate of the average output entropy as well as the average proportion of extracted entropy are calculated, defined by $E(H)/m$ and $E(H)/k$ respectively. The parameter k was set to 6 bits (i.e., the constant nonzero probabilities equal to 1/64), while m varies from 1 to 12 bits. As m increases up to the double of k, the extract rate approaches 100 % from below, in accordance with i). It also seems plausible that the extract rate drops essentially linearly with m/k as m decreases and $m < k$. What is not so plausible or not obvious is the pleasantly high extract rate, when m varies in the region $k, k+1, \dots, 2k$. Similar behavior was experienced also for other selections of the parameter values.

For an analytical proof on the asymptotic behavior of $E(H_{\text{out}})$ consider the following continuous approximation. A random selection of one output p.d. is equivalent to a random selection of $M-1$ different points from the open interval $(0,1)$. The average output entropy $E(H_{\text{out}})$ can be given by

$$E(H_{\text{out}}) = -M \int_0^1 x \log_2(x) p(x)\, dx\,,$$

where $p(x) = (M-1)(1-x)^{M-2}$, and simplifying this into a more explicit form we obtain

$$E(H_{\text{out}}) = \frac{1}{\ln(2)} \left[\sum_{i=1}^{M-1} \left(\frac{1}{i}\right) - \frac{M-1}{M} \right].$$

From here it follows that the rate of average output entropy defined by $E(H_{\text{out}})/m$ converges to 1 ($M = 2^m$). The convergence is fast. □

M	$E(H)/m(\%)$	$E(H)/k(\%)$	M	$E(H)/m(\%)$	$E(H)/k(\%)$
1	70.98	11.83	7	74.81	87.28
2	76.40	25.93	8	69.65	92.86
3	79.87	39.93	9	64.12	96.19
4	81.46	54.31	10	58.81	98.02
5	81.14	67.61	11	53.99	98.99
6	78.84	78.85	12	49.74	99.49

TABLE 2. Extraction of entropy.

For the general case of unequal atoms we have the following conjecture: The above results can be generalized by dropping the assumption of constant atoms, if the source is information stable. A well known classical result of information theory claims, that probabilities of typical output blocks coming from an information stable source are asymptotically equal [9]. Up to this asymptotic equality we can reduce the general case to the case of constant atoms.

The above considerations had nothing to do with the usual cryptographic properties of hash functions like collision intractability and one way-ness. These cryptographic properties essentially ensure that one cannot enlist even a few elements from the same preimage partition in a feasible way. The preimage partition can be non-uniform. However, such non-uniformity is not vulnerable to statistical attacks by practical resource limits because the dimensions here are huge (standard hash size means, say, 160 bits).

We have to distinguish two tasks: the design of an extractor which produces strong random output from weak input and the design of a complete PRNG for a cryptographic device. In this paper we are thinking about the former design.

The latter design is more complex, because it has to provide strong random output, as well as it has to protect this fine output from an intelligent attacker. In other words a good extractor can be the soul of such a PRNG, but it also needs a protective shield. Consider the PRNG as a finite state machine, with the usual input, output and state variables. The three main types of attacks are the following (the classification of attacks follows [2]): *direct cryptanalytic attack*, when we assume that the attacker accesses the output of the PRNG. The attacker is considered successful if he can distinguish between the output of the PRNG and real random streams. In case of *input-based attacks*, the attacker is able to know or manipulate the inputs of the PRNG. The attacker is considered successful, if he is able to control or predict the generated output. The one way property of cryptographic hash functions prevents purposeful manipulation of extracted values via manipulation of inputs. A *state compromise extension attack* occurs when we assume that occasional compromise of the state of the PRNG may happen or the attacker somehow learns of the state. The attacker is considered successful if he is able to extend this knowledge of state backward or forward in time. A typical starting point of such an attack is when a PRNG is started in an insecure state due to insufficient starting entropy.

5. Examples

5.1 A simple randomized extractor

Below we present a simple algorithm for a non-universal, finite, randomized extractor, which provides perfect extraction of all input entropies. This extractor achieves the optimum in Fig. 1. The cost appears on the side of the amount of added random bits. Though the construction is straightforward, it gives a good insight into the effect of randomization and the extraction mechanisms. The approach can best be presented via a numerical instance. The generalization should be obvious, therefore it is omitted for the sake of saving space.

Assume that the size of the input space is $N = 2^n = 16$, and there are $K = 6$ different input values with nonzero probabilities. Consider the following input distribution: $P = \{1/4, 1/4, 1/8, 1/8, 1/8, 1/8, 0, \ldots, 0\}$, i.e., $P(6 \ldots 15) = 0$.

The first step is encoding: let the dimension of the output space be equal to $m = 3$ (i.e., $M = 2^m = 8$), where m is the smallest positive integer larger or equal to the base_2 logarithm of the reciprocal value of the smallest probability $p_{\min}$. Then P is multiplied by M, and we obtain $Z = 8 \cdot P = \{2, 2, 1, 1, 1, 1, 0, \ldots, 0\}$. The output space, which is the space of binary 3-tuples here, is partitioned into $K = 6$ subsets with sizes corresponding to components in Z:

$\{000, 001\}, \{010, 011\}, \{100\}, \{101\}, \{110\}, \{111\}$.

The extraction operation is done in two steps. The inputs are mapped to these subsets:

$0 \to \{000, 001\}$, $1 \to \{010, 011\}$, $2 \to \{100\}$, $3 \to \{101\}$, $4 \to \{110\}$, $5 \to \{111\}$, $6 \ldots 15 \to$ arbitrarily.

The next step is the randomization: if the input is 0 or 1, we flip a coin and select an output 3-tuple; for instance if the input is 0 and a head occurs we select 000 otherwise we select 001. Obviously the output distribution will be uniform. The number of d added random bits depends on the actual output of the source, so it becomes a random variable. The entropy balance at the output is: $H_{\text{out}} = H(P) + E(d)$, here 3 bits = 2.5 bits + 0.5 bits, i.e., 0.5 bits are the average amount of random bits introduced during the randomization step. Note, that all input entropy is extracted in a finite number of steps. Let

$$d_i = \log_2(p_i/p_{\min}),$$

$i = 0, \ldots, N-1$. The average amount of added random bits can be given by the following formula:

$$E(d) = \sum_{i=0}^{N-1} p_i d_i = \sum_{i=0}^{N-1} p_i \log p_i - \sum_{i=0}^{N-1} p_i \log p_{\min} = \log_2(1/p_{\min}) - H(X).$$

Because $\log_2(1/p_{\max}) \le H(X) \le \log_2(1/p_{\min})$, if the distribution is closer to uniform, i.e., $p_{\max}/p_{\min}$ is closer to 1, then the percentage of added entropy is smaller.

In the above example we consider probabilities which were integer powers of 2. For the general case we need some refinement of the above procedure:

Assume that the ith subset has integer size $w = p_i/p_{\min}$, such that $2^{j-1} < w < 2^j$. The problem is that using j random bits we can select uniformly one element only from the entire range $\{0, \ldots, 2^j - 1\}$ and not from the range $\{0, \ldots w-1\}$. One possible solution to this problem can be the following: repeat the selection until an element is selected from the range $\{0, \ldots w-1\}$. This means that the number of additional random bits is a random variable. The number of random bits follows a geometric distribution with parameter $q = (1 - w/2^j)$, consequently the average number of needed random bits is j/q and its variance is equal to $j^2(1-q)q$.

The following proposition provides a refined formula for the amount of average added entropy:

Proposition 3. *The average of added entropy for the proposed extractor:*

$$E(d) = \sum_{i=0}^{N-1} p_i \lceil d_i \rceil 2^{\lceil d_i \rceil - d_i}, \tag{4}$$

where $\lceil d \rceil$ *denotes the ceiling of real value* d.

P r o o f. Straightforward. □

In the worst case, formula (4) shows an approximate twofold increase in the average amount of additional random bits, compared to the case, when all probabilities are powers of 2.

We would like to keep the added entropy above an acceptable level. The added entropy sensitively depends on $p_{\min}$. We can force an acceptable minimum level for nonzero probability, for instance, by distorting the distribution by simple preprocessing: the smallest atoms are unified into one atom, with their probabilities summed. This way we shrink the input space in size, somewhat.

Considering the statistical distance between the output p.d. and the corresponding uniform p.d., it is not hard to see that the equality $\operatorname{dist}(\mathrm{EXT}_r(P, U_d), U_m) = \operatorname{dist}(P, P')$ stands for the quality of the above extraction algorithm, i.e., the error in the estimation of the input distribution directly affects the non-uniformity at the output of the extractor.

5.2 An experiment

Luca T r e v i s a n gave the first polynomial time randomized construction for asymptotic extraction of all min-entropy [8]. His extractor works in two consecutive processing steps, which are based on two techniques: one from algebraic error control, the other from combinatorial block designs.

The output of the source as a message is mapped into a codeword c from a binary error correcting code $C(n', k', d')$. Code C has a prescribed distance profile, where the codewords typically have large Hammning distance and Hamming weight (half of the word length). This step is essentially an adjustment in the bias (bit imbalance) of the source. The m output bits are selected from the coordinates of codeword c, where the selection is controlled by selector variables derived from the additional random bits. The strength of the statistical dependence between these selector variables is determined via combinatorial block design: a block corresponds to a selector variable, where the elements of the block come from an appropriate subset of the set of additional random bits and the dependence is controlled through designed intersection of the pairs of blocks. This controlled dependence helps save the amount of additional random bits.

We carried out a simulation experiment. Our aim was to examine the effectiveness of these principles in the case of small finite dimensions and using simple, fast tools.

Selection of code C: Code C is a bit-balanced error control code. Rate 1/2 Reed-Solomon (RS) code $C1$ is concatenated with a bit balanced binary code $C2$. The code parameters are the following:

$C1(N,K,D)$ over $GF(q)$, $q=2^k$, $D=N-K+1$, $K=N/2$, $N=q-1$ ($N<q-1$ for shortened code),

$C2(n,k,d)$ over $GF(2)$: $n=2^k$, $w=n/2$ constant weight code generated from all cyclic shifts of an m-sequence of length $n=2^k-1$ with 0 bit appended.

For instance concatenating the outer code $C1(15,8,8)$ over $GF(16)$ with the inner code $C2(16,4,8)$ over $GF(2)$ produces the concatenated code $C(240,32,64)$ over $GF(2)$ with constant weight $w=120$.

Generation of bit-selector random variables: d real random bits are expanded linearly into a binary sequence of length 2^d-1, by using these d bits as the initial values for a linear feedback shift register (LFSR) generator producing an m-sequence. The set T of selector variables is produced by taking $|T|=m$ subsequences each of length $v=\lceil \log_2 n' \rceil$ bits from this m-sequence. The set T has useful properties: any element of T has a uniform distribution over the set $\{1,2,\ldots,n'\}$. Furthermore for any two different elements t_1 and t_2 of T we obtain $H(t_1|t_2)=H(t_2|t_1)=d-v$, $d\geq v$, where $H(\cdot|\cdot)$ denotes the conditional entropy. The first property ensures that the m output bits are selected uniformly from all coordinates of the codeword. The second property controls the dependence of bit selector variables.

According to simulation experiments, typical quality figures $H'_{\text{out}}/H_{\text{in}}\simeq 0.9$, $H_{\text{rate}}=0.82$ can be achieved using this simple extractor.

Using extractors, considered in this section as a building block in a cryptographic PRNG, additional protective measures are needed against attacks (mentioned in Section 4.). For instance, the linear codes used in the second construction do not hide the output from an attacker having enough knowledge about the input. As a straightforward preventive measure one could apply an invertible one way transformation at the input to foil input-based attacks. Furthermore one should design a safe secondary entropy pool for storage of starting (restarting) entropy to cope with state compromise attack.

6. Appendix

6.1 Proof of Fact 1

Below we show an input distribution for which the estimation of entropy is easier than the estimation of min-entropy. Let $p_{\max,L}$ and $p_{\max}$ denote the maximum of the frequency distribution and the real distribution, respectively. Let δ_i and δ be defined implicitly by $p_{L,i}=(1+\delta_i)p_i$, and $p_{\max,L}=(1+\delta')p_{\max}$, $i=1,2,\ldots,N$. We obtain $\sum_{i=1}^{N}\delta_i=0$, furthermore $E(\delta_i)=0$ and $\mathrm{Var}(\delta_i)=\frac{1}{L}\frac{1-p_i}{p_i}$,

$i = 0, 1, \ldots, N-1$. The relative error of estimation is

$$\mathrm{Err}_{\mathrm{min}} = \log_2(1+\delta')/H_{\mathrm{min}}$$

for the min-entropy and

$$\mathrm{Err}_{\mathrm{Sh}} = \left[\sum_{i=1}^{N} \delta_i (p_i \log_2 p_i) + \sum_{i=1}^{N} p_i (1+\delta_i) \log_2 \delta_i \right] \Big/ H$$

for the Shannon-entropy. Consider constant input probabilities. In this case $H = H_{\mathrm{min}} = k \leq n$. Now we have to compare random variables $\varsigma = \frac{1}{K} \log_2(1+\delta')$ and $\gamma = \frac{1}{K}\left[\frac{1}{K}\sum_{i=1}^{K}(1+\delta_i)\log_2(1+\delta_i)\right]$, $K = 2^k$. Random variables $\delta_0, \ldots, \delta_{N-1}$ will have the same distribution. Obviously $\delta' = \max_{1 \leq i \leq K} \delta_i$. For a sufficiently large number of samples, when $\ln(1+\delta') \simeq \delta'$ and $\ln(1+\delta_i) \simeq \delta_i$, $i = 1, \ldots, N$, we obtain $\varsigma \simeq \frac{1}{\ln(2)K} \max_{1 \leq i \leq K} \delta_i$, and for practical large values of K we get $\gamma \simeq \frac{1}{\ln(2)K} \mathrm{Var}(\delta_i) \simeq \frac{1}{\ln(2)} \frac{1}{L}$. $\mathrm{Err}_{\mathrm{Sh}}$ is essentially a small constant. $\mathrm{Err}_{\mathrm{min}}$ is a random variable, which at chance can be larger than $\mathrm{Err}_{\mathrm{Sh}}$, as

$$\mathrm{Var}(\delta_i) < \mathrm{Var}(\delta'). \tag{5}$$

K	4	8	16	32	64	128
$\mathrm{diff}^{(1)}$	1.36	1.77	2.36	3.28	4.32	5.56
$\mathrm{diff}^{(2)}$	0.66	0.56	0.51	0.49	0.46	0.43

TABLE 3. Comparison of estimation errors.

For illustration, Table 3 shows figures obtained from a simulation experiment. The second row shows the average of $\mathrm{diff}^{(1)} = 100 * (\mathrm{Err}_{\mathrm{Sh}} - \mathrm{Err}_{\mathrm{min}})$. The percentage difference in estimation accuracy grows steadily with K, i.e., the min entropy estimate is getting worse and worse relative to the estimate of the Shannon entropy as the number of nonzero inputs (here constant) probabilities increases. This is in good agreement with (5). For a counter-experiment, we repeated the experiment for distribution where one probability has value 0.5 and the further $K-1$ probabilities are all equal. The corresponding results are shown in the third row of Table 3, which meets the intuitive feeling that in this latter p.d. the estimation of the min-entropy should be easier.

6.2 Combinatorial formula for $E(H_{\mathrm{out}})$

Integer $K > 0$ is decomposed into a sum of M nonnegative integer components $K = \nu_1 + \nu_2 + \cdots + \nu_M$ in all the possible ways. We select randomly one of these decompositions, and consider probability distribution: $q_i = \frac{\nu_i}{K}$, $i = 1, \ldots, M$. For the case of constant atoms, the average of the output entropy can be given by the following formula:

$$E(H_{\mathrm{out}}) = E\left(-\sum_{i=1}^{M} q_i \log_2 q_i\right) = -\sum_{i=1}^{M} E(q_i \log_2 q_i)\,,$$

where $p_{j,M}^{(K)} = P(\nu_i = j)$, furthermore

$$= -\sum_{i=1}^{M}\left[\sum_{j=1}^{K}\Big(\frac{j}{K}\log_2\frac{j}{K}\Big)p_{j,M}^{(K)}\right] = -M\sum_{i=0}^{K}\Big(\frac{j}{K}\log_2\frac{j}{K}\Big)p_{j,M}^{(K)}\,,$$

where

$$p_{j,M}^{(K)} = \frac{\binom{K+M-j-2}{M-2}}{\binom{K+M-1}{M-1}} = \frac{1}{K+1}\prod_{i=1}^{M-2}\frac{1+\frac{K-j}{M-1-i}}{1+\frac{K}{M-i}}\,,$$

$j = 0, 1, \ldots, K$.

REFERENCES

[1] VEMBU, S.—VERDU, S.: *Generating random bits from an arbitrary source: Fundamental Limits*, IEEE Trans. Inform. Theory **41** (1995), 1322–1332.

[2] KELSEY, J.—SCHNEIER, B.—FERGUSON, N.: *Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator*, Technical Report, `http://www.counterpane.com`.

[3] SCHEFF, H.: *A useful convergence theorem for probability distributions*, Ann. of Math. Statist. **18** (1947), 434–458.

[4] BEIRLANT, J.—DEVROYE–GYÖRFI, L.—VAJDA, I.: *Large deviations of divergence measures on partitions*, J. Statist. Plann. Inference **93** (2001), 1–16.

[5] BLUM, M.: *Independent unbiased coin flips from a correlated biased source: a finite Markov chain*, Combinatorica **6** (1986), 9-7-106.

[6] SANTHA, M.—VAZIRANI, U.: *Generating quasi-random sequences from semi-random sources*, J. Comput. System Sci. **33** (1986), 75–87.

[7] NISAN, N.: *Extracting Randomness: How and Why. A survey*, Proc. of the 11th IEEE Conference on Computational Complexity, May 1996.

[8] TREVISAN, L.: *Construction of near optimal extractors using pseudo-random generators*, unpublished manuscript, `http://theory.lcs.mit.edu luca`, August 1998.

[9] COVER, T.—THOMAS, J.: *Elements of Information Theory* (John Wiley, ed.), Wiley Series in Telecommunications, New York, 1991.

[10] EGD: an Entropy Gathering Daemon v0.8, Brian Warner.

[11] *Linux standard /dev/random random number generator.*

Received December 19, 2001

Department of Telecommunication
Budapest University of Technology
and Economics
HUNGARY
E-mail: vajda@hit.bme.hu