RANDOMNESS EXTRACTORS AND THEIR APPLICATIONS

Aristeidis Tentes

School of Applied Mathematics and Physics, National Technical University of Athens,
E-mail: aristentes@yahoo.gr

Abstract

Randomness plays a crucial role in the theory of algorithms. Many problems can be dealt much faster if randomness is allowed, while some cryptographic tasks, as communication protocols, are impossible if we do not use randomness. The problem is that these algorithms require perfect randomness, namely a source which provides us a sequence of unbiased and independent bits. However, there is no source known to be easily accessible and perfect. Randomness Extractors are functions, which deal with the problem of making the output of a (not perfect) weak source almost perfect,that is they extract almost truly random bits from an imperfect source. In this survey we will see some basic definitions, an up to date list of the most efficient extractors, some other almost equivalent pseudorandom objects and some applications of extractors.

1 Introduction

Randomized Algorithms is a very well studied area, because randomization not only allows some problems to be solved faster than deterministically, but it also allows some tasks to be performed that are impossible otherwise. Most of the cryptographic tasks rely on the fact that all parties have access to a source of independent random bits. However, the problem is that there is no explicit way of obtaining independent random bits as all current sources of randomness, like electromagnetic noise, which are accessible by computers and used nowadays are highly believed to produce correlated bits. Randomness Extractors are functions, which take as input sequences of bits of weak sources, namely sources which produce random but not necessarily independed bits, and output another sequence of almost unbiased and independed bits. The initial motivation was: given such Randomness Extractors simulate randomized algorithms using weak sources.

The first attempt of obtaining unbiased and independed random bits from a source which produces independed but biased bits, with unknown bias, was made by von Neuman in [37]. The idea was simple: if the source outputs 1 with probability *p*, then we can take every two bits and assign 1 if the outcome is 10 (which happens with probability *p(1-p)*) and assign 0 if the outcome is 01 (which happens with the same probability *(1-p)p*). Intuitively we can see that we cannot obtain as many truly random bits as the output of the source. This is the price we pay for trying to convert the bits of a weak source to truly random. In this survey we will see some constructions, which try to optimize the number of truly random bits and compare them with respect to some other parameters.

Moreover it turned out that Randomness Extractors are almost equivalent to some other Pseudorandom objects like Pseudorandom Generators, Expander Graphs and others. We will see some of these disguises of Randomness Extractors and describe, somewhat informally, what they are. In the end we will see some applications of Randomness Extractors in Complexity Theory and Cryptography.

2 Definitions

First of all we need to characterize the weak sources in some way. The first objective is to have a measure, which is general and is also useful for our purpose. Such a measure is of course Shannon Entropy, however it is not very convenient. Therefore another measure is used, called MinEntropy, defined below:

Definition: Let *X* be a random variable.

- Shanon Entropy: Its Shanon Entropy is $\mathbf{H}(X) = \mathbf{E}[\log(\frac{1}{\Pr[X=x]})]$.
- MinEntropy: Its MinEntropy is $H_\infty(X) = \min_x\{\log\left(\frac{1}{\Pr[X=x]}\right)\}$.
- k-Source: *X* is a k-Source if $H_\infty(X) \geq k$.

To understand the difference between the two measures of entropy note that while the first gives roughly the number of independed bits we can extract on average (having many samples of *X*), the latter gives the smallest number of independed bits we can get from any sample. Suppose we have a source *X* which outputs with probability 0.99 a sequence of *n* 1's and with probability 0.1 a sequence of n truly random bits (i.e. independed and unbiased). Then $H(X) \geq 0.01n$, while $H_\infty(X) < 1$, that is although in average we can get $0.01n$ bits in most cases we cannot get not even one truly random bit. Therefore, the stronger measure of MinEntropy is used. Here we will focus on general sources, that is sources, for which the only assumption is their MinEntropy and nothing else.

Before we formally define Randomness Extractors we have to define what their objective is. As we said their purpose is to output a sequence which is almost uniformly random. This means that the statistical difference of the output and a uniformly random variable of the same length is very small.

Definition (Statistical Difference):

- Let *X, Y* ∈ *S* be two random Variables. Their Statistical Difference is defined to be $\Delta(X,Y) = 1/2\sum_x |Pr(X = x) - Pr(Y = x)|$ or equivalently (can be shown) $\Delta(X,Y) = max_{T\subseteq S}|Pr[X\in T] - Pr[Y \in T]|$.
- Let U_n denote the random variable of *n* uniformly distributed unbiased bits. If $\Delta(X, U_n) \leq \varepsilon$ then we say that *X* is *ε-close* to uniform.

There is one hurdle in constructing Extractors. One can easily show that for every boolan function and for every *k* there is a *k*-source (of a random variable with $n > k$ bits) from which it is impossible to extract even a single random bit. As we want to construct an Extractor, which works for every k-source we have to allow to the Extractors to take as an additional input a uniformly distributed variable, called the seed. Now we are ready to formally define Randomness Extractors.

Definition ((k,ε)-Extractor):

The function Ext:$\{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$ is a (k,ε)-Extractor if for every k-source *X* with *n* bits, the random variable $Ext(X, U_d)$ is *ε-close* to U_m. If in addition $(U_d, Ext(X, U_d))$ is is *ε-close* to U_{m+d} then this Extractor is called (k,ε)- strong Extractor.

3 Constructions

As we can see there are many parameters concerning Extractors. Using the probabilistic method we can prove the following theorem

Theorem: For every $k \leq n$ and ε>0, there exists a (k, ε)-Extractor, with $d = \log(n-k) + 2\log\left(\frac{1}{\varepsilon}\right) + O(1)$ and $m = d + k - 2\log\left(\frac{1}{\varepsilon}\right) - O(1)$.

The parameters we want to optimize are *d* and *m.* Given a k-source of length *n* we want an extractor with a seed length as small as possible and an output as long as possible. It is also proven in [20] that for every $k \leq n$ and ε>0 such an extractor is optimal with respect to seed length and output length. Sometimes the objective is to reduce the error parameter ε. However, as we mentioned, the proof of the above theorem is existential and non-constructive, therefore a lot of work has been done in explicit constructions, namely extractors which can be constructed and are computable in polynomial time. Below, there is a table with some of the most popular extractors. Until now no construction has been proposed, which is optimal in both parameters of seed length and output length. The first construnction is implied in [ILL89], with the use of universal Hash Functions (Leftover Hash Lemma) and an improvement upon this can be found in [SZ99], using almost universal Hash Functions ([NN90,AGHP92]). As we can see the construction of [TSUZ01] may have almost optimal parameters but does not work for any k-source, because there is an upper bound on the MinEntropy. The best Extractor proposed until today is the one of [LRVW03] and [GUV06] which may have the same parameters in seed and output length but the latter has an improvement in case of a very small error *ε*.

Table 1 – Constructions of Randomness Extractors

Publication	MinEntropy (***k***)	Seed length (***d***)	Output length (***m***)
[10]	for all *k*	*O(n)*	*k + d + O(1)*
[27]	for all *k*	*O(k + logn)*	*k + d + O(1)*
[38]	*Ω(n)*	*O(logn)*	*Ω(k)*
[29]	for all *k*	$O(log^2 n/logk)$	$k^{1-\delta}$
[21]	*k>n/2*	$log^{\Theta(1)}(n-k)$	*k + d - O(1)*
[30]	$k \leq 2^{log^{1/2+\delta}n}$	*O(logn)*	*k + d - O(1)*
[32]	*Ω(n)*	*logn + O(loglogn)*	*Ω(k)*
[25]	for all *k*	*(1+a)logn*	$O(k/log^{1/a}n)$
[14]	for all *k*	*O(logn)*	*(1-a)k*
[8]	for all *k*	*O(logn)*	*(1-a)k*
Optimal	**for all *k***	***logn + O(1)***	***k + d + O(1)***

4 Other Pseudorandom objects

There is a list of other pseudorandom objects, which are almost equivalent to Randomness Extractors and consequently almost equivalent to each other. Some of these objects are Expander graphs, Error Correcting Codes, Pseudorandom Generators, Samplers and Hardness Amplifiers, which if seen from the appropriate point of view can also be realized to be different disguises of Randomness Extractors. The significance of Randomness Extractors can be seen by this (almost) equivalence as these objects have been studied separately as well. This connection between these objects has been exploited through the last years by using techniques and

ideas from one object to another.

Expander Graphs: Expander Graphs are graphs which are sparse, namely they have few edges, yet at the same time they are very well connected. The most common measure of well connectedness is the vertex expansion. A Graph is called (k,a)-vertex expander if for every set of vertices S with at most k vertices, the neighborhood of S, that is the number of vertices of the graph connected to a vertex of S, is at least $a|S|$. Ideally these graphs have constant degree d, they are d-regular, $k=\Omega(n)$ where n the number of the vertices of the graph and $a=1+\Omega(1)$. Good Expander graphs have the useful property that a random walk converges very quickly to the stationary distribution, which in this case means that we can use few random bits to take a very close to uniform sample from the set of the vertices. There is a mass of work concerning Expander graphs due to their interesting properties. A good survey is [23].

Error Correcting codes: Error Correcting Codes deal with the problem of sending a message over a channel, which may corrupt the original message. These codes make it possible for the receiver to recover the original message in case it has not been changed a lot. The aspect of Error Correcting codes, which is concerned here is List Decoding where the algorithm of Decoding does not decode the original message with a single message, but it ouputs a list of messages, one of which matches the original message. List Decoding enables to correct more errors than standard Error Correcting Algorithms, while their running time is generally lower. For more see [31].

Pseudorandom Generators: Pseudorandom generators are polynomially running functions, which take as input a small seed, i.e. a short sequence of uniformly random bits, and outputs a much longer sequence of bits which look random to every polynomial computation. This means that there is no polynomial algorithm which can separate the output of the generator from an equally long uniformly random bit. Pseudorandom Generators were firstly studied in [19] and the connection with randomness extractors was exploited in [28].

Samplers: Suppose we have a very large set S. If we want to choose a uniformly random element then we have to use roughly $\log |S|$ unbiased and independed bits. Samplers are functions which take as input a few uniformly random bits and output a list of elements of |S|, which have almost similar properties as if we had chosen these elements uniformly at random. The connection between Samplers and Extractors was introduced in [38].

Hardness Amplifiers: Hardness Amplifiers are methods, which increase the hardness of computing a function. Suppose we have a function $f:A \to B$, with A,B finite, then we say that this function is (δ,s)-hard, if every Boolean circuit of size does not compute correctly f for a fraction bigger than δ of A. A Hardness Amplifier transforms this function to another one with a bigger δ. For the connection between Hardness Amplifiers and List Decoding see [24] and [33].

A survey which shows the connection and the differences between these objects is [36]. However, in this survey these objects are presented by the prism of Error Correcting Codes, which seems to be the most convenient way of viewing the so called unified theory of Pseudorandomness.

5 Applications of Randomness Extractors

Randomness Extractors turned out to have various applications, other than the original motivation of converting weak sources to good ones, some of which are quite surprising. Let us see now some of them:

Derandomization: Derandomization deals with the problem of converting randomized algorithms to non randomized, without making them run too much slower. Take for example any language in BPP. It is trivial that this language also belongs to EXP, as we can take all possible random bits we use in the BPP algorithm (which are at most 2^{n^k}) and decide with the majority. However, under some complexity assumptions we can show that BPP=P (we do not know yet if the equality holds), [2]. There are also many other similar results, for example about the equality of AM and NP, in which Pseudorandom Generators and Randomness Extractors play crucial role. See [26, 12]

Hardness of Approximation: Some problems which are thought to be difficult to be solved exactly are easy to be approximated. However, if we make some reasonable assumptions, such as $P \neq NP$, some problems turned out to be impossible to be efficiently approximated closer than a bound. In [39] there are results concerning the inapproximability of the problems Max Clique and Chromatic Number. In [17] the hardness of approximating the VC dimension is considered. These are some inapproximability results using Randomness Extractors.

Embeddings: An Embedding is a map of projecting elements of a space X to another space Y in such a way that the structure of the mapped X does not differ a lot from that of X. In [11] an explicit embedding is given, which makes use of Randomness Extractors.

Fuzzy Extractors: Fuzzy Extractors deal with the problem of using private keys obtained by biometric data for any cryptographic task. The problem is that biometric data (e.g. fingerprints) are not uniformly distributed, which is very important in Cryptography, however they seem to have large MinEntropy. Moreover, this kind of data is not always exactly the same (for instance a fingerprint may have a scratch), which makes it impossible to be used as private key on its own. Randomness Extractors are very useful in this cryptographic

setting, for more see [4, 3]

Cryptography versus Bounded Storage Adversaries: The Bounded Storage Model was introduced by Maurer in [16]. It assumes that a sender Alice and a receiver Bob have agreed on a short secret key and have access to a public source. Eve, the adversary, has access to this source but can store only a limited number of bits. Alice and Bob can then use their secret key as a seed for an extractor, which extracts the remaining MinEntropy of the source, with respect to Eve's view. The output looks almost universal to Eve and can be used by Alice and Bob as a key as one-time pad for encryption. This extractor must have specific properties, see [15, 35].

Exposure Resilient Cryptography: Most Cryptosystems guarantee no security if part of the key is exposed by the adversary. This area of cryptography deals with the problem of constructing functions which have the property, that even a big fraction of the input is known, the outputs looks almost uniformly random. Randomness Extractors turned out to play a major role, see [5, 13]

6 Conclusions

As we can see, Randomness Extractors are very important in Theory of Computation. It is not only their original objective, which makes them so important, but also the variety of connections they turned out to have with other areas of research. We saw that they are closely connected with other object, which carry a mass of research on their own. In addition Extractors have many applications in other research areas, like Cryptography.

The main open problem is whether there exists an explicit construction with truly optimal parameters. Recently a lot of work has been done in constructing seedless Extractors, for more specific sources, such as affine sources [6], bit-fixing sources [7, 13] and others.

References:

[1] Noga Alon, Oded Goldreich, Johan Hºastad, and Rene Peralta. Simple construction of almost k-wise independent random variables.Random Struct. Algorithms, 3(3):289{304, 1992.
[2] Lazlo Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson.Bpp has subexponential time simulations unless exptime has publishable proofs. Computational Complexity, 3:307{318, 1993.
[3] Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith.Robust fuzzy extractors and authenticated key agreement from close secrets. In CRYPTO, pages 232{250, 2006.
[4] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data.In EUROCRYPT, pages 523{540, 2004.
[5] Yevgeniy Dodis, Amit Sahai, and Adam Smith. On perfect and adaptive security in exposure-resilient cryptography. In EUROCRYPT, pages 301{324, 2001.
[6] Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. In FOCS, pages 407{418, 2005.
[7] Ariel Gabizon, Ran Raz, and Ronen Shaltiel. Deterministic extractors for bit-fixing sources by obtaining an independent seed. SIAMJ. Comput., 36(4):1072{1094, 2006.
[8] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan.Unbalanced expanders and randomness extractors from parvaresh vardy codes. In IEEE Conference on Computational Complexity, pages 96{108, 2007.
[9] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. Simplified derandomization of bpp using a hitting set generator. Electronic Colloquium on Computational Complexity (ECCC), 7(4), 2000.
[10] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudorandom generation from one-way functions (extended abstracts). InSTOC, pages 12{24, 1989.
[11] Piotr Indyk. Uncertainty principles, extractors, and explicit embeddings of l2 into l1. In STOC, pages 615{620, 2007.
[12] Russell Impagliazzo and Avi Wigderson.B = BPP if requires exponential circuits: Derandomizing the xor lemma. In STOC, pages220{229, 1997.
[13] Jesse Kamp and David Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. SIAM J. Comput., 36(5):1231{1247, 2007.
[14] Chi-Jen Lu, Omer Reingold, Salil P. Vadhan, and Avi Wigderson.Extractors: optimal up to constant factors. In STOC, pages 602{611, 2003.
[15] Chi-Jen Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In CRYPTO, pages 257{271, 2002.
[16] Ueli Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. Journal of Cryptology, 5(1):53{66, 1992. Preliminary version
[17] Elchanan Mossel and Christopher Umans. On the complexity of approximating the vc dimension. J. Comput. Syst. Sci., 65(4):660{671, 2002.
[18] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In STOC, pages 213{223, 1990.

[19] Noam Nisan and Avi Wigderson. On rank vs. communication complexity. In FOCS, pages 831{836, 1994.
[20] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. SIAM J.Discrete Math., 13(1):2{24, 2000.
[21] Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In FOCS, pages 3{13, 2000.
[22] Ronen Shaltiel. Recent developments in explicit constructions of extractors. Bulletin of the EATCS, 77:67{95, 2002.
[23] N Linial S Hoory and Avi Wigderson. Expander graphs and their applications. In Bull. of the AMS, 43(4), pages 439{561, 2006.
[24] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the xor lemma. J. Comput. Syst. Sci., 62(2):236{266, 2001.
[25] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In FOCS, pages 648{657, 2001.
[26] Ronen Shaltiel and Christopher Umans. Low-end uniform hardness vs. randomness tradeoffs for am. In STOC, pages 430{439, 2007.
[27] Aravind Srinivasan and David Zuckerman. Computing with very weak random sources. SIAM J. Comput., 28(4):1433{1459, 1999.
[28] Luca Trevisan. Extractors and pseudorandom generators. J. ACM,48(4):860{879, 2001.
[29] Luca Trevisan. Pseudorandomness and combinatorial constructions. Electronic Colloquium on Computational Complexity (ECCC),(013), 2006.
[30] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Lossless condensers, unbalanced expanders, and extractors. In STOC,pages 143{152, 2001.
[31] Amnon Ta-Shma and David Zuckerman. Extractor codes. In STOC,pages 193{199, 2001.
[32] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. Extractors from reed-muller codes. In FOCS, pages 638{647, 2001.
[33] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In IEEE Conference on Computational Complexity, pages 129{138, 2002.
[34] Salil Vadhan. Lecture notes on pseudorandomness, http://eecs.harvard.edu/ salil/cs225, 2007.
[35] Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. J. Cryptology,17(1):43{77, 2004.
[36] Salil P. Vadhan. The unified theory of pseudorandomness: guest column. SIGACT News, 38(3):39{54, 2007.
[37] J. von Newman. Various techniques used with connection with random digits. National Bureau of Standards, Applied Math. Series,12:36{38, 1951.
[38] David Zuckerman. Randomness-optimal oblivious sampling. Random Struct. Algorithms, 11(4):345{367, 1997.
[39] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In STOC, pages 681{690,2006.