On Constructing Locally Computable Extractors and Cryptosystems in the Bounded Storage Model*

Salil P. Vadhan**

Harvard University, Cambridge, MA, salil@eecs.harvard.edu

Abstract. We consider the problem of constructing randomness extractors that are *locally computable*; that is, read only a small number of bits from their input. As recently shown by Lu (*CRYPTO '02*), locally computable extractors directly yield secure private-key cryptosystems in Maurer's bounded storage model (*J. Cryptology*, 1992).
We suggest a general "sample-then-extract" approach to constructing locally computable extractors. Plugging in known sampler and extractor constructions, we obtain locally computable extractors, and hence cryptosystems in the bounded storage model, whose parameters improve upon previous constructions and come quite close to the lower bounds.
The correctness of this approach follows from a fundamental lemma of Nisan and Zuckerman (*J. Computer and System Sciences,* 1996), which states that sampling bits from a weak random source roughly preserves the min-entropy rate. We also present a refinement of this lemma, showing that the min-entropy rate is preserved up to an arbitrarily small additive loss, whereas the original lemma loses a logarithmic factor.

1 Introduction

Maurer's *bounded storage model* for private-key cryptography [2] has been the subject of much recent activity. In this model, one assumes that there is public, high-rate source of randomness and that all parties have limited storage so that they cannot record all the randomness from the source. Remarkably, this quite plausible model makes it possible to construct private-key cryptosystems that are information-theoretically secure and require no unproven complexity assumptions (in contrast to most of modern cryptography). Intuitively, a shared secret key can be used by legitimate parties to randomly select bits from the random source about which the adversary has little information (due to the bound on its storage). With some further processing, the legitimate parties can convert these unpredictable bits into ones which the adversary cannot distinguish from truly random (in an information-theoretic sense), and hence they can safely be used for cryptographic purposes, e.g. as a one-time pad for encryption.

* A preliminary version of this paper has appeared on the Cryptology e-print archive [1] and a full version will appear in the Journal of Cryptology.
** Supported by NSF grants CCR-0205423 and CCR-0133096.

A sequence of works [2–8] has given increasingly secure and efficient protocols in this model. In particular, the works of Aumann, Ding, and Rabin [5, 6] showed that protocols in this model have the novel property of "everlasting security" — the security is preserved even if the key is reused an exponential number of times and is subsequently revealed to the adversary.

Recently, Lu [8] showed that work in this model can be cast nicely in the framework of *randomness extractors*. Extractors, introduced by Nisan and Zuckerman [9], are procedures for extracting almost-uniform bits from sources of biased and correlated bits. These powerful tools have been the subject of intense study, and have found many applications to a wide variety of topics in the theory of computation. (See the surveys [10, 11].) One of the first applications, in the original paper of Nisan and Zuckerman, was to construct pseudorandom generators for space-bounded computation. Thus, they seem a natural tool to use in the bounded storage model, and indeed Lu [8] showed that any extractor yields secure private-key cryptosystems in the bounded storage model. However, the efficiency considerations of the bounded storage model require a nonstandard property from extractors — namely that they are *locally computable*;[1] that is, they can be computed by reading only a few bits from the random source. Lu constructed locally computable extractors by first constructing locally computable error-correcting codes, and then plugging them into the specific extractor construction of Trevisan [13].

In this paper, we suggest a general "sample-then-extract" approach to constructing locally computable extractors: use essentially any randomness-efficient "sampler" to select bits from the source and then apply essentially any extractor to the selected bits. Plugging in known sampler and extractor constructions, we obtain locally computable extractors, and hence cryptosystems in the bounded storage model, whose parameters improve upon previous constructions and come quite close to the lower bounds.

The correctness of this approach follows directly from a fundamental lemma of Nisan and Zuckerman [9]. Roughly speaking, the lemma states that a random of sample of bits from a string of high min-entropy[2] also has high min-entropy. We also present a refinement of this lemma, showing that the min-entropy rate is preserved up to an arbitrarily small additive loss, whereas the original lemma loses a logarithmic factor. This improvement is not necessary for the sample-then-extract approach to work, but increases its efficiency. Together with some of our techniques for constructing samplers, it has also played a role in the recent explicit construction of extractors that are "optimal up to constant factors" [14].

In retrospect, several previous cryptosystems in the bounded storage model, such as [3] and [5], can be viewed as special cases of the sample-then-extract

[1] This terminology was suggested by Yan Zong Ding and we prefer it to the terminology "on-line extractors," which was used (with different meanings) in [12, 8]. The issue of "local computation" versus "on-line computation" is discussed in more detail in Section 3.

[2] Like Shannon entropy, the min-entropy of a probability distribution X is a measure of the number of bits of "randomness" in X. A formal definition is given in Section 3.

approach, with particular choices for the extractor and sampler. By abstracting the properties needed from the underlying tools, we are able to use state-of-the-art extractors and samplers, and thereby obtain our improvements.

2 Preliminaries

Except where otherwise noted, we refer to random variables taking values in discrete sets. We generally use capital letters for random variables and lowercase letters for specific values, as in $\Pr[X = x]$. If S is a set, then $x \stackrel{\mathrm{R}}{\leftarrow} S$ indicates that x is selected uniformly from S. For a random variable A and an event E, we write $A|_E$ to mean A conditioned on E.

The *statistical difference* (or variation distance) between two random variables X, Y taking values in a universe $\mathcal{U}$ is defined to be

$$\varDelta(X,Y) \stackrel{\text{def}}{=} \max_{S \subset \mathcal{U}} \Big| \Pr[X \in S] - \Pr[Y \in S] \Big| = \frac{1}{2} \sum_{x \in \mathcal{U}} \Big| \Pr[X = x] - \Pr[Y = x] \Big| .$$

We say X and Y are *ε-close* if $\varDelta(X,Y) \leq \varepsilon$.

The *min-entropy* of X is $\mathrm{H}_\infty(X) \stackrel{\text{def}}{=} \min_x \log(1/\Pr[X = x])$. (All logarithms in this paper are base 2.) Intuitively, min-entropy measures randomness in the "worst case," whereas standard (Shannon) entropy measures the randomness in X "on average." X is called a *k-source* if $\mathrm{H}_\infty(X) \geq k$, i.e. for all x, $\Pr[X = x] \leq 2^{-k}$.

3 The Bounded Storage Model

The Random Source. The original model of Maurer [2] envisioned the random source as a high-rate stream of perfectly random bits being broadcast from some natural or artificial source of randomness. However, since it may difficult to obtain perfectly random bits from a physical source, particularly at a high rate, we feel it is important to investigate the minimal conditions on the random source under which this type of cryptography can be performed. As noted in [8, 7], the existing constructions still work even if we only assume that the source has sufficient "entropy". Below we formalize this observation, taking particular note of the kind of independence that is needed when the cryptosystem is used many times.

We model the random source as a sequence of random variables $X_1, X_2, \ldots$, each distributed over $\{0,1\}^n$, where X_t is the state of the source at time period t. To model a random source which is a high-rate "stream" of bits, the X_t's can be thought of as a partition of the stream into contiguous substrings of length n. However, one may also consider random sources that are not a stream, but rather a (natural or artificial) "oracle" of length n, which changes over time and can be probed at positions of one's choice. In both cases, n should be thought of as very large, greater than the storage capacity of the adversary (and the legitimate parties).

To obtain the original model of a perfectly random stream, each the X_t's can be taken to be uniform on $\{0,1\}^t$ and independent of each other. Here we wish to allow biases and correlations in the source, only assuming that each X_t has sufficient randomness, as measured by min-entropy (as advocated in [15, 16]). That is, we will require each X_t to be an αn-source for some $\alpha > 0$. Using a worst-case measure like min-entropy rather than Shannon entropy is important because we want security to hold with high probability and not just "on average". (The results will also apply for random sources that are statistically close to having high min-entropy, such as those of high Renyi entropy.)

For our cryptosystems, we will actually need to require that the random source has high min-entropy conditioned on the future.

Definition 1. *A sequence of random variables $X_1, X_2, \ldots$, each distributed over $\{0,1\}^n$ is a* reverse block source *of* entropy rate α *if for every $t \in \mathbb{N}$ and every $x_{t+1}, x_{t+2}, \ldots$, the random variable $X_t|_{X_{t+1}=x_{t+1}, X_{t+2}=x_{t+2}, \ldots}$ is an αn-source.*

As the terminology suggests, this is the same as the Chor-Goldreich [15] notion of a *block source*, but "backwards" in time. Intuitively, it means that X_t possesses αn bits of randomness that will be "forgotten" at the next time step. This is somewhat less natural than the standard notion of a (forward) block source, but it still may be a reasonable model for some physical sources of randomness that are not perfectly random.[3] Below we will see why some condition of this form (high entropy *conditioned on the future*) is necessary for the cryptography. In the special case $\alpha = 1$, Definition 1 is equivalent to requiring that X_t's are uniform and independent, so in this case the issue of reversal is moot.

Cryptosystems. Here, as in previous works, we focus on the task of using a shared key to extract pseudorandom bits from the source. These pseudorandom bits can then be used for private communication or message authentication. A *pseudorandom extraction scheme in the bounded storage model* is a function $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ (typically with $d, m \ll n$). Such a scheme is to be used as follows. Two parties share a key $K \in \{0,1\}^d$. At time t, they apply $\mathrm{Ext}(\cdot, K)$ to the random source X_t to obtain m pseudorandom bits, given by $Y_t = \mathrm{Ext}(X_t, K)$. At time $t+1$ (or later), Y_t will be pseudorandom to the adversary (if the scheme is secure), and hence can be used by the legitimate parties as a shared random string for any purpose (e.g. as a one-time pad for encryption). The pseudorandomness of Y_t will rely on the fact that, at time $t+1$ and later, X_t is no longer accessible to the adversary. More generally, we need X_t to be unpredictable from future states of the random source, as captured by our notion of a reverse block source. Note that even if Y_t will only be used exactly at time $t+1$, we still need X_t to have high min-entropy given the entire future, because the adversary can store Y_t.

We now formally define security for a pseudorandom extraction scheme. Let βn be the bound on the storage of the adversary A, and denote by $S_t \in$

[3] The consideration of such sources raises interesting philosophical questions: does the universe keep a perfect record of the past? If not, then reverse block sources seem plausible.

$\{0,1\}^{\beta n}$ the state of the adversary at time t. For a sequence of random variables $Z_1, Z_2, \ldots$, we will use the shorthand $Z_{[a,b]} = (Z_a, Z_{a+1}, \ldots, Z_b)$, $Z_{[a,\infty)} = (Z_a, Z_{a+1}, \ldots)$. Following the usual paradigm for pseudorandomness, we consider the adversary's ability to distinguish two experiments — the "real" one, in which the extraction scheme is used, and an "ideal" one, in which truly random bits are used. Let A be an arbitrary function representing the way the adversary updates its storage and attempts to distinguish the two experiments at the end.

Real Experiment: Let $X_1, X_2, \ldots$ be the random source, and let $K \xleftarrow{\text{R}} \{0,1\}^d$. For all t, let $Y_t = \text{Ext}(X_t, K)$ be the extracted bits. Let $S_0 = 0^{\beta n}$, and for $t = 1, \ldots, T$, let $S_t = A(Y_{[1,t-1]}, S_{t-1}, X_t)$. Output $A(Y_{[1,T]}, S_T, X_{[T+1,\infty)}, K) \in \{0,1\}$.

Ideal Experiment: Let $X_1, X_2, \ldots$ be the random source, and let $K \xleftarrow{\text{R}} \{0,1\}^d$. For all t, let $Y_t \xleftarrow{\text{R}} \{0,1\}^m$. Let $S_0 = 0^{\beta n}$, and for $t = 1, \ldots, T$, let $S_t = A(Y_{[1,t-1]}, S_{t-1}, X_t)$. Output $A(Y_{[1,T]}, S_T, X_{[T+1,\infty)}, K) \in \{0,1\}$.

Note that at each time step we give the adversary access to all the past Y_i's "for free" (i.e. with no cost in the storage bound), and in the last time step, we give the adversary the adversary access to all future X_i's and the key K. The benefits of doing this are explained below.

Definition 2. *We call* Ext : $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ ε-secure *for storage rate β and entropy rate α if for every reverse block source (X_t) of entropy rate α, every adversary A with storage bound βn, and every $T > 0$, A distinguishes between the real and ideal experiments with advantage at most $T \cdot \varepsilon$.*

Remarks:

- In the real experiment, we give the extracted strings Y_t explicitly to the adversary, as is typical in definitions of pseudorandomness. However, when they are used in a cryptographic application (e.g. as one-time pads), they of course will not be given explicitly to the adversary. The string Y_{t-1} extracted at time $t-1$ is not given to the adversary (i.e. is not used in the application) until time t. As mentioned above, this is crucial for security.
- The definition would be interesting even if $Y_1, \ldots, Y_{t-2}$ were not given to the adversary at time t, (i.e. $S_t = A(Y_{t-1}, S_{t-1}, X_t)$), and if K and $X_{T+2}, X_{T+3}, \ldots$ were not given to the adversary at the end. Giving all the previous Y_i's implies that it is "safe" to use Y_i at any time period after i (rather than exactly at time $i+1$). Giving the adversary all subsequent X_i's at the end is to guarantee that the security does not deteriorate if the adversary waits and watches the source for some future time periods. Giving the adversary the key at the end means that even if the secret key is compromised, earlier transactions remain secure. This is the remarkable property of "everlasting security" noticed and highlighted by [5].
- We require that the security degrade only linearly with the number of times the same key is reused.

- No constraint is put on the computational power of the adversary except for the storage bound of βn (as captured by $S_t \in \{0,1\}^{\beta n}$). This means that the distributions of $(Y_{[1,T-1]}, S_T, X_{[T+1,\infty)}, K)$ in the real and ideal experiments are actually close in a statistical sense — they must have statistical difference at most $T \cdot \varepsilon$.
- The definition is impossible to meet unless $\alpha > \beta$: If $\alpha \leq \beta$, we can take each X_t to have its first αn bits uniform and the rest fixed to zero. Then an adversary with βn storage can entirely record X_t, and thus can compute Y_t once K is revealed. (Even if K is not revealed, in this example the bounded-storage model still clearly provides no advantage over the standard private-key setting, and hence is subject to the usual limitations on information-theoretic security [17].)

As usual, the above definition implies that to design a cryptosystem (e.g. private-key encryption or message authentication) one need only prove its security in the ideal experiment, where the two parties effectively share an infinite sequence of random strings $Y_1, Y_2, \ldots$. Security in the bounded storage model immediately follows if these random Y_i's are then replaced with ones produced by a secure pseudorandom extraction scheme.

Efficiency Considerations. In addition to security, it is important for the extraction scheme to be efficient. In the usual spirit of cryptography, we would like the honest parties to need much smaller resources than the adversary is allowed. In this case, that means we would like the computation of Ext to require much less *space* than the adversary's storage bound of βn. Note that the honest parties will have to store the entire extracted key $Y_t \in \{0,1\}^m$ during time t (when it is not yet safe to use), so reducing their space to m is the best we can hope for (and since we envision $m \ll n$, this is still very useful). However, since n is typically envisioned to be huge, it is preferable to reduce not just the space for Ext to much less than n, but also the time spent.[4] Thus, we adopt as our efficiency measure the *number of bits read from the source.* Of course, once these bits are read, it is important that the actual computation of Ext is efficient with respect to both time and space. In our constructions (and all previous constructions), Ext can be computed in polynomial time and polylogarithmic work space (indeed even in NC). Thus the total storage required by the legitimate parties is dominated by the number of bits read from the source.

The following (proven in the full version) shows that the number of bits read from the source must be linear in m, and must grow when the difference between the entropy rate and storage rate goes to zero.

Proposition 3. *If* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is an* ε*-secure pseudorandom extraction scheme for storage rate* β *and entropy rate* α*, then* $\mathrm{Ext}(\cdot, K)$ *depends on at least* $(1 - \varepsilon - 2^{-m}) \cdot (1/(\alpha - \beta)) \cdot m$ *bits of its input (on average, taken over* K*).*

[4] If having Ext computable in small space with one pass through the random source is considered sufficiently efficient, then the work of Bar-Yossef et al. [12] is also applicable here. See Section 4.

reference	key length	# bits read	restrictions
[3]	$O(\log n)$	$O(m/\varepsilon^2)$	interactive
[4]	$O(\log n \cdot \log(1/\varepsilon))$	$O(m \cdot \log(1/\varepsilon))$	$\alpha = 1$,$\beta < 1/m$
[5]	$O(m \cdot \log n \cdot \log(1/\varepsilon))$	$O(m \cdot \log(1/\varepsilon))$	
[6]	$O(\log n \cdot \log(1/\varepsilon))$	$O(m \cdot \log(1/\varepsilon))$	$\alpha = 1$, $\beta < 1/\log m$
[7]	$O(\log n \cdot \log(1/\varepsilon))$	$O(m \cdot \log(1/\varepsilon))$	
[8]	$O(m \cdot (\log n + \log(1/\varepsilon)))$	$O(m \cdot \log(1/\varepsilon))$	
[8]	$O((\log^2(n/\varepsilon)/\log n))$	$O(m \cdot \log(1/\varepsilon))$	$m \leq n^{1-\Omega(1)}$
here	$O(\log n + \log(1/\varepsilon))$	$O(m + \log(1/\varepsilon))$	$\varepsilon > \exp(-n/2^{O(\log^* n)})$

Fig. 1. Comparison of pseudorandom extraction schemes in the bounded storage model. Parameters are for ε-secure schemes Ext : $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, for constant storage rate β and entropy rate α, where $\alpha > \beta$. We only list the parameters for the case that the number of bits read from the source is $o(n)$, as n is assumed to be huge and infeasible.

Another common complexity measure in cryptography is the key length, which should be minimized. Figure 1 describes the performance of previous schemes and our new constructions with respect to these two complexity measures.[5] With respect to both measures, our constructions are within a constant factor of optimal. In fact, for the number of bits read, this constant factor can be made arbitrarily close to 1.

We now touch upon a couple of additional efficiency considerations. First, if the random source is indeed a high-rate stream (as opposed to an "oracle source"), it is important that the positions to be read from the source can be computed offline (from just the key) and sorted so that they can be quickly read in order as the stream goes by. This is the case for our scheme and previous ones.

Second, one can hope to reduce the space of the legitimate parties to exactly $m+d$ (i.e., the length of the extracted string plus the key). That is, even though the schemes read more than m bits from the source, the actual computation of the m-bit extracted string can be done "in place" as the bits from the source are read. This property holds for most of the previous constructions, as each bit of the extracted string is a parity of $O(\log(1/\varepsilon))$ bits of the source. Our construction does not seem to have this property in general (though specific instantiations may); each bit of the output can be a function of the entire $O(m + \log(1/\varepsilon))$ bits read from the source. Still the space used by our scheme is $O(m+\log(1/\varepsilon))$, only a constant factor larger than optimal.

[5] Most of the previous schemes were explicitly analyzed only for the case of a perfectly random source, i.e. $\alpha = 1$, but the proofs actually also work for weak random sources provided $\alpha > \beta$ (except where otherwise noted) [8]. Also note that the schemes with key length greater than m do not follow trivially from the one-time pad, because the same key can be used many times.

4 Locally Computable Extractors

In this section, we define extractors and locally computable extractors, and recall Lu's result [8] about their applicability to the bounded storage model. Then we discuss averaging samplers and describe how using them to sample bits from a random source preserves the min-entropy rate, via a lemma of Nisan and Zuckerman [9] which we refine. Finally, we give our general sample-then-extract construction which combines any extractor and any averaging sampler to yield a locally computable extractor.

An extractor is a procedure for extracting almost-uniform bits from any random source of sufficient min-entropy. This is not possible to do deterministically, but it is possible using a short *seed* of truly random bits, as captured in the following definition of Nisan and Zuckerman.

Definition 4 ([9]). $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* strong[6] (k, ε)-extractor *if for every k-source X, the distribution $U_d \circ \mathrm{Ext}(X, U_d)$ is ε-close to $U_d \times U_m$.*

The goal in constructing extractors is to minimize the seed length d and maximize the output length m. We will be precise about the parameters in later sections, but, for reference, an "optimal" extractor has a seed length of $d = O(\log n + \log(1/\varepsilon))$ and an output length of $m = k - O(\log(1/\varepsilon))$, i.e. almost all of the min-entropy is extracted using a seed of logarithmic length.

Recently, Lu proved that any extractor yields secure cryptosystems in the bounded storage model:

Theorem 5 (implicit in [8]). *If* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a strong $(\delta n - \log(1/\varepsilon), \varepsilon)$-extractor, then for any $\beta > 0$,* Ext *is an 2ε-secure pseudorandom extraction scheme for storage rate β and entropy rate $\beta + \delta$.*

However, as noted by Lu, for a satisfactory solution to cryptography in the bounded storage model, the extractor should only read only a few bits from the source.

Definition 6. $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is* t-locally computable *(or* t-local*) if for every $r \in \{0,1\}^d$, $\mathrm{Ext}(x, r)$ depends on only t bits of x, where the bit locations are determined by r.*

Thus, in addition to the usual goals of minimizing d and maximizing m, we also wish to minimize t. Bar-Yossef, Reingold, Shaltiel, and Trevisan [12] studied a related notion of *on-line extractors*, which are required to be computable in small space in one pass. They show that space approximately m is necessary and sufficient to evaluate extractors with output length m. Since the small-space requirement is weaker than being locally computable, their lower bound

[6] A standard (i.e. non-strong) extractor requires only that $\mathrm{Ext}(X, U_d)$ is ε-close to uniform.

also applies here.[7] But a stronger lower bound for locally computable extractors can be obtained by combining Proposition 3 and Theorem 5 with $\beta = 0$, or by mimicking the proof of Proposition 3 directly for t-local extractors to obtain the following slightly better bound:

Proposition 7. *If* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a t-local strong $(\delta n, \varepsilon)$-extractor, then* $t \geq (1 - \varepsilon - 2^{-m}) \cdot (1/\delta) \cdot m$.

Lu [8] observed that the encryption schemes of Aumann, Ding, and Rabin [4–6] can be viewed as locally computable extractors, albeit with long seeds. He constructed locally computable extractors with shorter seeds based on Trevisan's extractor [13]. The construction of Dziembowski and Maurer [7] is also a locally computable extractor. The parameters of these constructions can be deduced from Figure 1.

Our construction of locally computable extractors is based on a fundamental lemma of Nisan and Zuckerman [9], which says that if one samples a random subset of bits from a weak random source, the min-entropy rate of the source is (nearly) preserved. More precisely, if $X \in \{0,1\}^n$ is a δn-source and $X_S \in \{0,1\}^t$ is the projection of X onto a *random* set $S \subset [n]$ of t positions, then, with high probability, X_S is ε-close to a $\delta' t$-source, for some δ' depending on δ. Thus, to obtain a locally computable extractor, we can simply apply a (standard) extractor to X_S, and thereby output roughly $\delta' t$ almost-uniform bits. That is, part of the seed of the locally computable extractor will be used to select S, and the remainder as the seed for applying the extractor to X_S.

However, choosing a completely random set S of positions is expensive in the seed length, requiring approximately $|S| \cdot \log n$ random bits. (This gives a result analogous to [5], because $|S| \geq m$.) To save on randomness, Nisan and Zuckerman [9] showed that S could be sampled in a randomness-efficient manner, using k-wise independence and/or random walks on expander graphs. More generally, their proof only requires that w.h.p. S has large intersection with any subset of $[n]$ of a certain density (cf., [18]). In order to achieve improved performance, we will impose a slightly stronger requirement on the sampling method: for any $[0,1]$-valued function, w.h.p. its average on S should approximate its average on $[n]$. Such sampling procedures are known as *averaging* (or *oblivious*) *samplers*, and have been studied extensively [19–22]. Our definition differs slightly from the standard definition, to allow us to obtain some savings in parameters (discussed later).

Definition 8. *A function* $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ *is a* (μ, θ, γ) averaging sampler *if for every function* $f : [n] \to [0,1]$ *with average value* $\frac{1}{n}\sum_i f(i) \geq \mu$, *it holds that*

$$\Pr_{(i_1,\ldots,i_t) \stackrel{R}{\leftarrow} \mathrm{Samp}(U_r)} \left[\frac{1}{t} \sum_{j=1}^{t} f(i_j) < \mu - \theta \right] \leq \gamma.$$

[7] This is because their space lower bounds apply also to a nonuniform branching program model of computation, where the space is always at most the number of bits read from the input.

Samp *has* distinct samples *if for every* $x \in \{0,1\}^r$*, the samples produced by* $\mathrm{Samp}(x)$ *are all distinct.*

That is, for any function f whose average value is at least μ, with high probability (i.e., at least $1-\gamma$) the sampler selects a sample of positions on which the average value of f is not much smaller than μ. The goal in constructing averaging samplers is usually to simultaneously minimize the randomness r and sample complexity t. We will be precise about the parameters in later sections, but, for reference, an "optimal" averaging sampler uses only $t = O(\log(1/\gamma))$ samples and $r = O(\log n + \log(1/\gamma))$ random bits (for constant μ, θ).

In contrast to most applications of samplers, we will not necessarily be interested in minimizing the sample complexity. Ideally, we prefer samplers where the number of distinct samples can be chosen anywhere in the interval $[t_0, n]$, where t_0 is the minimum possible sample complexity. (Note that without the requirement of distinct samples, the number of samples can be trivially increased by repeating each sample several times.) Another atypical aspect of our definition is that we make the parameter μ explicit. Averaging samplers are usually required to give an approximation within additive error θ regardless of the average value of f, but being explicit about μ will allow us to obtain some savings in the parameters.

Using averaging samplers (rather than just samplers that intersect large sets) together with an idea from [23] allows us to obtain a slight improvement to the Nisan–Zuckerman lemma. Specifically, Nisan and Zuckerman show that sampling bits from a source of min-entropy rate δ yields a source of min-entropy rate $\Omega(\delta/\log(1/\delta))$; our method can yield min-entropy rate $\delta - \tau$ for any desired τ.

For a string $x \in \{0,1\}^n$ and a sequence $s = (i_1, \ldots, i_t) \in [n]^t$, define $x_s \in \{0,1\}^t$ to be the string $x_{i_1} x_{i_2} \cdots x_{i_t}$. Recall that for a pair of jointly distributed random variables (A, B), we write $B|_{A=a}$ for B conditioned on the event $A = a$.

Lemma 9 (refining [9]). *Suppose* $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ *is an* (μ, θ, γ) *averaging sampler with distinct samples for* $\mu = (\delta - 2\tau)/\log(1/\tau)$ *and* $\theta = \tau/\log(1/\tau)$*. Then for every* δn*-source* X *on* $\{0,1\}^n$*, the random variable* $(U_r, X_{\mathrm{Samp}(U_r)})$ *is* $(\gamma + 2^{-\Omega(\tau n)})$*-close to* (A, B) *where* A *is uniform on* $\{0,1\}^r$ *and for every* $a \in \{0,1\}^r$*,*[8] *the random variable* $B|_{A=a}$ *is* $(\delta - 3\tau)t$*-source.*

The above lemma is where we use the fact that the sampler has distinct samples. Clearly, sampling the same bits of X many times cannot increase the min-entropy of the output, whereas the above lemma guarantees that the min-entropy grows linearly with t, the number of samples.

An alternative method to extract a shorter string from a weak random source while preserving the min-entropy rate up to a constant factor was given by Reingold, Shaltiel, and Wigderson [18], as a subroutine in their improved extractor construction. However, the string produced by their method consists of bits of

[8] Intuitively, the reason we can guarantee that B has high min-entropy conditioned on *every* value of A, is that the "bad" values of A are absorbed in the $\gamma + 2^{-\Omega(\tau n)}$ statistical difference.

an encoding of the source in an error-correcting code rather than bits of the source itself, and hence is not good for constructing locally computable extractors (which was not their goal). As pointed out to us by Chi-Jen Lu and Omer Reingold, Lemma 9 eliminates the need for error-correcting codes in [18].

The proof of Lemma 9 is deferred to the full version.Given the lemma, it follows that combining an averaging sampler and an extractor yields a locally computable extractor.

Theorem 10 (sample-then-extract). *Suppose that* $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ *is an* (μ, θ, γ) *averaging sampler with distinct samples for* $\mu = (\delta - 2\tau)/\log(1/\tau)$ *and* $\theta = \tau/\log(1/\tau)$*. and* $\mathrm{Ext} : \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ *is a strong* $((\delta - 3\tau)t, \varepsilon)$ *extractor. Define* $\mathrm{Ext}' : \{0,1\}^n \times \{0,1\}^{r+d} \to \{0,1\}^m$ *by*

$$\mathrm{Ext}'(x, (y_1, y_2)) = \mathrm{Ext}(x_{\mathrm{Samp}(y_1)}, y_2).$$

Then Ext' *is a* t*-local strong* $(\delta n, \varepsilon + \gamma + 2^{-\Omega(\tau n)})$ *extractor.*

Proof. For every (y_1, y_2), $\mathrm{Ext}'(x, (y_1, y_2))$ only reads the t bits of x selected by $\mathrm{Samp}(y_1)$, so Ext' is indeed t-local. We now argue that it is a $(\delta n, \varepsilon + \gamma + 2^{-\Omega(\tau n)})$ extractor. Let X be any δn-source. We need to prove that the random variable $Z = (U_r, U_d, \mathrm{Ext}'(X, (U_r, U_d))) = (U_r, U_d, \mathrm{Ext}(X_{\mathrm{Samp}(U_r)}, U_d))$ is close to uniform. By Lemma 9, $(U_r, X_{\mathrm{Samp}(U_r)})$ is $(\gamma + 2^{-\Omega(\tau n)})$-close to (A, B) where A is uniform on $\{0,1\}^r$ and $B|_{A=a}$ is a $(\delta - 3\tau)t$-source for every a. This implies that Z is $(\gamma + 2^{-\Omega(\tau n)})$-close to $(A, U_d, \mathrm{Ext}(B, U_d))$. Since Ext is a strong $((\delta - 3\tau)t, \varepsilon)$ extractor, $(U_d, \mathrm{Ext}(B|_{A=a}, U_d))$ is ε-close to $U_d \times U_m$ for all a. This implies that $(A, U_d, \mathrm{Ext}(B, U_d))$ is ε-close to $A \times U_d \times U_m = U_r \times U_d \times U_m$. By the triangle inequality, Z is $(\varepsilon + \gamma + 2^{-\Omega(\tau n)})$-close to $U_r \times U_d \times U_m$. □

For intuition about the parameters, consider the case when $\delta > 0$ is an arbitrary constant, $\tau = \delta/6$, and $\gamma = \varepsilon$. Then using "optimal" averaging samplers and extractors will give a locally computable extractor with seed length $r + d = O(\log n + \log(1/\varepsilon))$ and output length $m = \Omega(\delta t) - O(\log(1/\varepsilon))$. This matches, up to constant factors, the seed length of an optimal extractor (local or not) and the optimal relationship between the output length and the number of bits read from the source.

We stress that the above refinement to the Nisan–Zuckerman lemma is not necessary to achieve these parameters (or those in Figure 1). Those parameters can be obtained by applying the sample-then-extract method with the original lemma and sampler of Nisan and Zuckerman [9] together with the extractor of Zuckerman [21]. The advantage provided by the refined lemma lies in the hidden constant in the number of bits read from the source. Specifically, by taking $\tau \to 0$, the ratio between m and t approaches δ, which is essentially optimal by Proposition 7.

5 Non-Explicit Constructions

In this section, we describe the locally computable extractors obtained by using truly optimal extractors and samplers in Theorem 10. This does not give efficient constructions of locally computable extractors, because optimal extractors and samplers are only known by nonconstructive applications of the Probabilistic Method. However, it shows what Theorem 10 will yield as one discovers constructions which approach the optimal bounds. In fact, the explicit constructions known are already quite close, and (as we will see in Section 6) match the optimal bounds within constant factors for the range of parameters most relevant to the bounded storage model.

5.1 The Extractor

The Probabilistic Method yields extractors with the following expressions for the seed length d and output length m, both of which are tight up to additive constants [24].

Lemma 11 (nonconstructive extractors (cf., [21, 24])). *For every n, $k \leq n$, $\varepsilon > 0$, there exists a strong (k, ε)-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with $d = \log(n-k) + 2\log(1/\varepsilon) + O(1)$, $m = k - 2\log(1/\varepsilon) - O(1)$.*

5.2 The Sampler

Similarly, the following lemma states the averaging samplers implied by the Probabilistic Method. There are matching lower bounds for both the randomness complexity and the sample complexity [20] (except for the dependence on μ, which was not considered there). The proof of the lemma (given in the full version) follows the argument implicit in [21], with the modifications that it makes the dependence on μ explicit and guarantees distinct samples.

Lemma 12 (nonconstructive samplers). *For every $n \in \mathbb{N}$, $1/2 > \mu > \theta > 0$, $\gamma > 0$, there is a (μ, θ, γ) averaging sampler* $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ *that uses*

- *t distinct samples for any $t \in [t_0, n]$, where $t_0 = O\left(\frac{\mu}{\theta^2} \cdot \log \frac{1}{\gamma}\right)$.*
- *$r = \log(n/t) + \log(1/\gamma) + 2\log(\mu/\theta) + \mathrm{loglog}(1/\mu) + O(1)$ random bits.*

5.3 The Local Extractor

Plugging the above two lemmas into Theorem 10, we obtain the following.

Theorem 13 (nonconstructive local extractors). *For every $n \in \mathbb{N}$, $\delta > 0$, $\varepsilon > 0$, and $m \leq \delta n/2 - 2\log(1/\varepsilon) - O(1)$, there is a t-local strong $(\delta n, \varepsilon)$ extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with*

- *$d = \log n + 3\log(1/\varepsilon) + \mathrm{loglog}(1/\delta) + O(1)$.*
- *$t = O\left(\frac{m + \log(1/\delta) \cdot \log(1/\varepsilon)}{\delta}\right)$.*

In fact, the hidden constant for the m/δ term in the expression for t can be made arbitrarily close to the optimal value of 1 (by paying a price in the other hidden constants). We defer the exact expressions to the full version.

6 Explicit Constructions

In the previous section, we showed that very good locally computable extractors exist, but for applications we need *explicit* constructions. For an extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ or a sampler $\mathrm{Samp} : \{0,1\}^r \to [n]^t$, explicit means that it is computable in polynomial time and polylogarithmic work-space with respect to its input+output lengths (i.e., $n+d+m$ for an extractor and $r+t\log n$ for a sampler). For a t-local extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, we give it oracle access to its first input and view the input length as $\log n + t + d$ ($\log n$ to specify the length of the oracle, and t as the number of bits actually read).

There are many explicit constructions of averaging samplers and extractors in the literature and thus a variety of local extractors can be obtained by plugging these into Theorem 10. We do not attempt to describe all possible combinations here, but rather describe one that seems particularly relevant to cryptography in the bounded storage model. We recall the following features of this application:

- The local extractor should work for sources of min-entropy $(\alpha - \beta)n - \log(1/\varepsilon)$, which is $\Omega(n)$ for most natural settings of the parameters. (Recall that α is the entropy rate of the random source and β is the storage rate of the adversary.) That is, we can concentrate on constant min-entropy rate.
- Optimizing the number of bits read from the source seems to be at least as important as the seed length of the extractor.
- The error ε of the extractor will typically be very small, as this corresponds to the "security" of the scheme.
- We are not concerned with extracting all of the entropy from the source, since we anyhow will only be reading a small fraction of the source.

6.1 The Extractor

With the above criteria in mind, the most natural extractor to use (in Theorem 10) is Zuckerman's extractor for constant entropy rate [21]:

Lemma 14 ([21]). *For every constant $\delta > 0$, every $n \in \mathbb{N}$, and every $\varepsilon > \exp(-n/2^{O(\log^* n)})$, there is an explicit strong $(\delta n, \varepsilon)$-extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $d = O(\log n + \log(1/\varepsilon))$ and $m = \delta n/2$.*

6.2 The Sampler

For the averaging sampler, the well-known sampler based on random walks on expander graphs provides good parameters for this application. Indeed, its randomness and sample complexities are both optimal to within a constant factor when μ and θ are constant (and the minimal sample complexity is used). However, we cannot apply it directly because it does not guarantee distinct samples, and we do not necessarily want to minimize the number of samples. Nisan and Zuckerman [9] presented some methods for getting around these difficulties, but their analysis does not directly apply here since we impose a stronger requirement on the sampler. (As mentioned earlier, we could use their sampler with their version of Lemma 9, at the price of worse constant factors in the number of bits read from the source.) Thus we introduce some new techniques to deal with these issues.

The following gives a modification of the expander sampler which guarantees distinct samples.

Lemma 15 (modified expander sampler). *For every* $0 < \theta < \mu < 1$, $\gamma > 0$, *and* $n \in \mathbb{N}$, *there is an explicit* (μ, θ, γ) *averaging sampler* $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ *that uses*

- t *distinct samples for any* $t \in [t_0, n]$, *where* $t_0 = O\left(\frac{1}{\theta^2} \cdot \log(1/\gamma)\right)$, *and*
- $r = \log n + O(t \cdot \log(1/\theta))$ *random bits.*

The main idea in the proof is to show that a "short" random walk on a "good" expander is unlikely to have "many" repeats. Specifically, in a random walk of length t on an n-vertex expander of normalized second eigenvalue λ, the expected fraction of repeated vertices is at most $t/n + O(\lambda)$. We prove this by the "trace method," which expresses repeat probabilities in terms of the trace of powers of the adjacency matrix. Given this bound on expected repeats, we obtain our sampler by taking several random walks on the expander (dependently, using another expander!) until we obtain a walk with a small fraction of repeats, from which we can safely discard the repeats without substantially affecting the sampler's estimate. Details are given in the full version.

A drawback of the expander sampler is that the randomness increases with the number of samples, whereas in the optimal sampler of Lemma 12 the randomness actually decreases with the number of samples. To fix this, we use the following lemma, which shows that the number of (distinct) samples can be increased at no cost.

Lemma 16. *Suppose there is an explicit* (μ, θ, γ) *averaging sampler* $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ *with distinct samples. Then for every* $m \in \mathbb{N}$, *there is an explicit* (μ, θ, γ) *averaging sampler* $\mathrm{Samp}' : \{0,1\}^r \to [m \cdot n]^{m \cdot t}$ *with distinct samples.*

In fact, there is a gain as the sample complexity increases, because the randomness complexity depends only on the original value of n, rather than $n' = m \cdot n$. This simple observation about samplers (employed in conjunction with Lemma 9) has played a role in the recent construction of extractors that are "optimal up to constant factors" [14].

To apply this lemma to construct a sampler with given values of n, t, μ, θ, and γ, it is best to start with a sampler $\mathrm{Samp}_0 : \{0,1\}^{r_0} \to [n_0]^{t_0}$ using the minimal sample complexity $t_0 = t_0(\theta, \mu, \gamma) < t$ and domain size $n_0 = n \cdot (t_0/t)$. For example, for constant μ and θ, the sampler of Lemma 15 will give $t_0 = O(\log(1/\gamma))$ and $r_0 = \log n_0 + O(\log(1/\gamma)) = \log(n/t) + O(\log(1/\gamma))$. Then setting $m = t/t_0$, Lemma 16 gives a sampler for domain size $n_0 \cdot m = n$, using $t_0 \cdot m = t$ distinct samples, and r_0 random bits. This is how we obtain our final sampler, stated in the next lemma.

Lemma 17. *For every $n \in \mathbb{N}$, $1 > \mu > \theta > 0$, $\gamma > 0$, there is a (μ, θ, γ) averaging sampler* $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ *that uses*

- t *distinct samples for any* $t \in [t_0, n]$, *where* $t_0 = O\left(\frac{1}{\theta^2} \cdot \log \frac{1}{\gamma}\right)$, *and*
- $r = \log(n/t) + \log(1/\gamma) \cdot \mathrm{poly}(1/\theta)$ *random bits.*

Unfortunately, when t/t_0 and $n \cdot (t_0/t)$ are not integers, some care is needed to deal with the rounding issues in the argument given above. The tedious details are given in the full version.

6.3 The Local Extractor

Analogously to Theorem 13, we plug Lemmas 14 and 17 into Theorem 10 to obtain:

Theorem 18 (explicit local extractors). *For every constant $\delta > 0$, $n \in \mathbb{N}$, $\varepsilon > \exp(-n/2^{O(\log^* n)})$, $m \leq \delta n/4$, there is an explicit t-local strong $(\delta n, \varepsilon)$ extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with*

- $d = O(\log n + \log(1/\varepsilon))$.
- $t = O(m + \log(1/\varepsilon))$.

The above construction does generalize to the case of subconstant δ. The expression for t is actually $t = O(m/\delta + \log(1/\varepsilon)/\delta^2)$, which is not too bad compared with non-explicit construction of Theorem 13. In fact, as in Theorem 13, the hidden constant in the m/δ term can be made arbitrarily close to the optimal value of 1. The seed length d is $d = O((\log n + \log(1/\varepsilon)) \cdot \mathrm{poly}(1/\delta))$, but here the multiplicative dependence on $\mathrm{poly}(1/\delta)$ is much worse than the additive dependence on $\mathrm{loglog}(1/\delta)$ in Theorem 13. This is due to both underlying components — the extractor (Lemma 14) and the averaging sampler (Lemma 17). The dependence on δ in the extractor component can be made logarithmic by using one of the many known explicit extractors for subconstant min-entropy rate. For the averaging sampler, too, there are constructions whose randomness complexity is within a constant factor of optimal [21]. However, these constructions have a sample complexity that is only polynomial in the optimal bound, resulting in a t-local extractor with $t \geq \mathrm{poly}(\log(1/\gamma), 1/\delta)$. It is an interesting open problem, posed in [22], to construct averaging samplers whose sample and randomness complexities are both within a constant factor of optimal. (Without the "averaging" constraint, there are samplers which achieve this [25, 26, 22].)

6.4 Previous Constructions.

Some previous constructions of cryptosystems in the bounded storage model can be understood using our approach, namely Theorem 10 together with Theorem 5 (of [8]). For example, the cryptosystem of Cachin and Maurer [3] amounts to using pairwise independence for both the averaging sampler and the extractor. (The fact that pairwise independence yields a sampler follows from Chebychev's

Inequality [27], and that it yields an extractor is the Leftover Hash Lemma of [28].) Actually, in the description in [3], the seed for the extractor is chosen at the time of encryption and sent in an additional interactive step. But it follows from this analysis that it actually can be incorporated in the secret key, so interaction is not necessary.

Our approach also yields an alternative proof of security for the ADR cryptosystem [4, 5]. Consider the sampler which simply chooses a random t-subset of $[n]$ for $t = O(\log(1/\varepsilon))$ and the extractor $\mathrm{Ext} : \{0,1\}^t \times \{0,1\}^t \to \{0,1\}$ defined by $\mathrm{Ext}(x, r) = x \cdot r \bmod 2$. The correctness of the sampler follows from Chernoff-type bounds, and the correctness of the extractor from the Leftover Hash Lemma [28]. Combining these via Theorem 10 yields a locally computable extractor which simply outputs the parity of a random subset of $O(\log(1/\varepsilon))$ bits from the source. This is essentially the same as the ADR cryptosystem, except that the size of the subset is chosen according to a binomial distribution rather than fixed. However, the security of the original ADR cryptosystem follows, because subsets of size exactly $t/2$ are a nonnegligible fraction ($\Omega(1/\sqrt{t})$) of the binomial distribution. To extract m bits, one can apply this extractor m times with independent seeds, as done in [5].

Acknowledgments

Noga Alon, Omer Reingold, and Amnon Ta-Shma also thought of similar approaches to this problem, and I am grateful to them for providing me with several helpful suggestions. I am also grateful to Yan Zong Ding and Chi-Jen Lu for pointing out errors in an earlier version of this paper. I thank Oded Goldreich for his encouragement and many helpful comments on the presentation. I thank Yan Zong Ding, Dick Lipton, and Michael Rabin for a number of illuminating discussions about the bounded storage model. Finally, I thank the anonymous CRYPTO and J. Cryptology reviewers for several helpful suggestions.

References

1. Vadhan, S.P.: On constructing locally computable extractors and cryptosystems in the bounded storage model. Cryptology ePrint Archive, 2002/162 (2002)
2. Maurer, U.: Conditionally-perfect secrecy and a provably-secure randomized cipher. J. Cryptology **5** (1992) 53–66
3. Cachin, C., Maurer, U.: Unconditional security against memory-bounded adversaries. In: CRYPTO '97. Spring LNCS 1294 (1997) 292–306
4. Aumann, Y., Rabin, M.O.: Information theoretically secure communication in the limited storage space model. In: CRYPTO '99. Springer LNCS 1666 (1999) 65–79
5. Aumann, Y., Ding, Y.Z., Rabin, M.O.: Everlasting security in the bounded storage model. IEEE Trans. Information Theory **48** (2002) 1668–1680
6. Ding, Y.Z., Rabin, M.O.: Hyper-encryption and everlasting security (extended abstract). In: 19th STACS. Springer LNCS 2285 (2002) 1–26
7. Dziembowski, S., Maurer, U.: Tight security proofs for the bounded-storage model. In: 34th STOC. (2002) 341–350 See also preliminary journal version, entitled "Optimal Randomizer Efficiency in the Bounded-Storage Model," Dec. 2002.

8. Lu, C.J.: Hyper-encryption against space-bounded adversaries from on-line strong extractors. In: CRYPTO '02. Springer LNCS 2442 (2002) 257–271
9. Nisan, N., Zuckerman, D.: Randomness is linear in space. J. Computer & System Sci. **52** (1996) 43–52
10. Nisan, N., Ta-Shma, A.: Extracting randomness: A survey and new constructions. J. Computer & System Sci. **58** (1999) 148–173
11. Shaltiel, R.: Recent developments in explicit constructions of extractors. Bull. EATCS **77** (2002) 67–95
12. Bar-Yossef, Z., Reingold, O., Shaltiel, R., Trevisan, L.: Streaming computation of combinatorial objects. In: 17th CCC. (2002) 165–174
13. Trevisan, L.: Extractors and pseudorandom generators. JACM **48** (2001) 860–879
14. Lu, C.J., Reingold, O., Vadhan, S., Wigderson, A.: Extractors: Optimal up to constant factors. In: 35th STOC. (2003)
15. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM J. Computing **17** (1988) 230–261
16. Zuckerman, D.: Simulating BPP using a general weak random source. Algorithmica **16** (1996) 367–391
17. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal **28** (1949) 656–715
18. Reingold, O., Shaltiel, R., Wigderson, A.: Extracting randomness via repeated condensing. In: 41st FOCS. (2000)
19. Bellare, M., Rompel, J.: Randomness-efficient oblivious sampling. In: 35th FOCS. (1994) 276–287
20. Canetti, R., Even, G., Goldreich, O.: Lower bounds for sampling algorithms for estimating the average. Information Processing Letters **53** (1995) 17–25
21. Zuckerman, D.: Randomness-optimal oblivious sampling. Random Struct. & Alg. **11** (1997) 345–367
22. Goldreich, O.: A sample of samplers: A computational perspective on sampling. Technical Report TR97-020, ECCC (1997)
23. Ta-Shma, A.: Almost optimal dispersers. Combinatorica **22** (2002) 123–145
24. Radhakrishnan, J., Ta-Shma, A.: Bounds for dispersers, extractors, and depth-two superconcentrators. SIAM J. Discrete Math. **13** (2000) 2–24 (electronic)
25. Bellare, M., Goldreich, O., Goldwasser, S.: Randomness in interactive proofs. Computational Complexity **3** (1993) 319–354
26. Goldreich, O., Wigderson, A.: Tiny families of functions with random properties: A quality-size trade-off for hashing. Random Struct. & Alg. **11** (1997) 315–343
27. Chor, B., Goldreich, O.: On the power of two-point based sampling. J. Complexity **5** (1989) 96–106
28. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28** (1999) 1364–1396