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

Salil P. Vadhan[†]
Division of Engineering & Applied Sciences
Harvard University
Cambridge, MA 02138
salil@eecs.harvard.edu
http://eecs.harvard.edu/~salil/

September 24, 2003

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 (*this issue*), 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: use essentially any randomness-efficient sampler to select bits from the input and then apply 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. We also provide lower bounds showing that the parameters we achieve are nearly optimal.

The correctness of the sample-then-extract 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.

Keywords: extractors, bounded storage model, everlasting security, space-bounded adversaries, unconditional security, averaging samplers, expander graphs

[*] Preliminary versions of this paper have appeared on the Cryptology e-print archive [Vad02] and in CRYPTO 03 [Vad03].

[†] Supported by NSF grants CCR-0205423 and CCR-0133096 and a Sloan Research Fellowship.

1 Introduction

Maurer's *bounded storage model* for private-key cryptography [Mau92] 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 of the randomness coming 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 sequence of works [Mau92, CM97, AR99, ADR02, DR02, DM, Lu] has given increasingly secure and efficient protocols in this model. In particular, the works of Aumann, Ding, and Rabin [ADR02, DR02] showed that protocols in this model have the novel property of "everlasting security" — the security is preserved even if, after the protocol is used, the key is revealed to the adversary and the adversary's storage becomes unbounded.

Recently, Lu [Lu] showed that work in this model can be cast nicely in the framework of *randomness extractors*. Extractors, introduced by Nisan and Zuckerman [NZ96], 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 [NT99, Sha02].) 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 [Lu] 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 [Tre01].

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. We also provide lower bounds that show that the parameters we achieve are nearly optimal.

The correctness of the sample-then-extract approach follows directly from a fundamental lemma of Nisan and Zuckerman [NZ96]. 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

[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 [BRST02, Lu].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.

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" [LRVW03].

In retrospect, several previous cryptosystems in the bounded storage model, such as [CM97] and [ADR02], can be viewed as special cases of the sample-then-extract 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.

Organization

Section 2 contains some general definitions and notation. In Section 3, we present Maurer's bounded storage model [Mau92], generalized to sources that need not be perfectly random, along with the relevant definitions of security. In Section 4, we identify the two main efficiency measures we consider, and state the performance of both previous and our constructions with respect to these measures. Section 5 recalls the notion of randomness extractors [NZ96] and Lu's connection between locally computable extractors and the bounded-storage model [Lu]. In Section 6, we present our sample-then-extract approach for constructing locally computable extractors and our refinement to the Nisan–Zuckerman lemma. Section 7 contains the results obtained by plugging optimal (but nonconstructive) samplers and extractors into the sample-then-extract approach. In Section 8, we instead plug in known explicit constructions to obtain an explicit and efficient cryptosystem. Finally, in Section 9, we prove lower bounds showing that the parameters we achieve are nearly optimal.

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 lower-case letters for specific values, as in $\Pr[X = x]$. For a random variable X, we write $x \overset{\mathrm{R}}{\leftarrow} X$ to indicate that x is selected according to X. If S is a set, then $x \overset{\mathrm{R}}{\leftarrow} S$ 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. We write U_n to denote a random variable distributed uniformly on $\{0,1\}^n$.

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

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

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

The *min-entropy* of a random variable X is defined to be $\mathrm{H}_\infty(X) \overset{\text{def}}{=} \min_x \log(1/\Pr[X = x])$. (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 $\mathrm{H}_\infty(X) \ge k$, i.e. for all x, $\Pr[X = x] \le 2^{-k}$.

3 The Bounded Storage Model

The Random Source. A central component of the Maurer's bounded storage model [Mau92] is a public, high-rate source of randomness. In the original model, the random source was envisioned as a 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 [Lu, DM],the existing constructions still work even if we only assume that the source has sufficient min-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 disjoint, contiguous substrings of the stream. However, our formulation also allows for the possibility that the random source is 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).

In the original model of a perfectly random stream, the X_t's are uniform on $\{0,1\}^n$ and independent of each other. However, we will allow biases and correlations in the source, only assuming that each X_t has sufficient randomness, as measured by min-entropy (the measure advocated in [CG88, Zuc96]). 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.)

We will also not insist that the X_t's are independent, only that they have high min-entropy conditioned on the future.

Definition 3.1 *A sequence of random variables $X_1, X_2, \ldots$, each distributed over $\{0,1\}^n$ is a* reverse block source *of* min-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 [CG88] 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 imperfect physical sources of randomness.[3] Below we will see why some condition of this form (high entropy *conditioned on the future*) is necessary for cryptography in the bounded storage model. In the special case $\alpha = 1$, Definition 3.1 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 encryption (as a

[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.

stream cipher) or message authentication. A *bounded storage model (BSM) pseudorandom generator* is a function $\mathrm{PRG} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ (typically with $d, m \ll n$). Such a scheme is to be used as follows. Two parties share a seed $K \in \{0,1\}^d$. At time t, they apply $\mathrm{PRG}(\cdot, K)$ to the random source X_t to obtain m pseudorandom bits, given by $Y_t = \mathrm{PRG}(X_t, K)$. At time $t+1$ (or later), Y_t will be pseudorandom to the adversary (if the scheme satisfies the definition of security below), 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 BSM pseudorandom generator. Let βn be the bound on the storage of the adversary A, and denote by $S_t \in \{0,1\}^{\beta n}$ 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 pseudorandom generator 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, let $K \xleftarrow{\mathrm{R}} \{0,1\}^d$ be the key, and let the adversary's initial state be $S_0 = 0^{\beta n}$.
- For $t = 1, \ldots, T$, let $Y_t = \mathrm{PRG}(X_t, K)$ be the pseudorandom bits, and let $S_t = A(Y_{[1,t-1]}, S_{t-1}, X_t)$ be the adversary's new state.
- 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, let $K \xleftarrow{\mathrm{R}} \{0,1\}^d$ be the key, and let the adversary's initial state be $S_0 = 0^{\beta n}$.
- For $t = 1, \ldots, T$, let $Y_t \xleftarrow{\mathrm{R}} \{0,1\}^m$ be truly random bits, and let $S_t = A(Y_{[1,t-1]}, S_{t-1}, X_t)$ be the adversary's new state.
- 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 3.2 *We call* $\mathrm{PRG} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *an* ε-secure BSM pseudorandom generator *for storage rate* β *and min-entropy rate* α *if for every reverse block source* (X_t) *of min-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 $T \cdot \varepsilon$. That is,*

$$\left| \Pr_{\text{real}}[A(Y_{[1,T]}, S_T, X_{[T+1,\infty)}, K) = 1] - \Pr_{\text{ideal}}[A(Y_{[1,T]}, S_T, X_{[T+1,\infty)}, K) = 1] \right| \leq T \cdot \varepsilon$$

Remarks:

- In the real experiment, we give the pseudorandom 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. Note that the string Y_{t-1} extracted at time $t-1$ is not given to the adversary (i.e. is not used in an application) until time t. As mentioned above, this is crucial for Y_{t-1} to be pseudorandom (as required by Definition 3.2).

- 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 $(Y_{[1,T]}, S_T, X_{[T+1,\infty)}, K)$ 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 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 analogous to the "forward security" property studied in the computational setting (e.g., as applied to pseudorandom generators in [BY03]). However, unlike the computational setting, here the security is preserved even if the adversary subsequently gains more resources (i.e., storage). That is, the pseudorandomness of $Y_{[1,T]}$ only relies on the adversary's storage being limited *until time* T (along with $X_{[1,T]}$ being inaccessible after time T, except via S_T and $X_{[T+1,\infty)}$). This nice property was dubbed "everlasting security" in [ADR02].

- We require that the security degrade only linearly with the number of times the same key is reused.

- 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 [Sha49].[4]

- Definition 3.2 should not be confused with the notion of pseudorandom generators for space-bounded computation, e.g. as studied in [Nis92, NZ96]. In the latter notion, there is no

[4] Shannon's impossibility result is for encryption schemes defined in the standard, noninteractive model. It was generalized to interactive protocols in [Mau93].

public random source and one typically allows both the seed length and space usage of the pseudorandom generator to be greater than the distinguisher's space bound. (Here we want to set the storage bound to be infeasibly large, and hence will aim for all the resources used by the BSM pseudorandom generator to be much smaller). Nevertheless, our construction (building on [Lu]) makes heavy use of the techniques from [NZ96].

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 BSM pseudorandom generator.

Even though Definition 3.2 explicitly requires that security be preserved under multiple uses (degrading linearly), it turns out that it suffices to analyze schemes for one time period, as shown by the following lemma, which generalizes the "key reuse" results of [DR02, Lu].

Lemma 3.3 $\mathrm{PRG} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is an* ε*-secure BSM pseudorandom generator for storage rate* β *and min-entropy rate* α *if and only if for every* αn*-source* X *on* $\{0,1\}^n$ *and for every function* $A : \{0,1\}^n \to \{0,1\}^{\beta n}$*, the random variable* $(\mathrm{PRG}(X,K), A(X), K)$ *is* ε*-close to* $(U_m, A(X), K)$*, where* $K \overset{R}{\leftarrow} \{0,1\}^d$*.*

Proof Sketch: We begin with the "only if" direction. Definition 3.2, with $T = 1$, says that for every αn-source X_1 and every A with output length βn, the random variable $(\mathrm{PRG}(X_1, K), A(0^{\beta n}, X_1), K)$ is ε-close to $(U_m, A(0^{\beta n}, X_1), K)$. This is clearly equivalent to the condition in Lemma 3.3.

For the "if" direction, we only outline the proof, as it is essentially the same as that of [Lu] with a minor augmentation to deal with reverse block sources. Let $X_1, X_2, \ldots$ be a reverse block source of min-entropy rate α, and let A be any adversary with storage bound βn. We distinguish random variables in the Real Experiment and the Ideal Experiment by superscripts, e.g. Y_t^{real} vs. Y_t^{ideal}. We need to prove that, for every T, the random variable $Z_T^{\mathrm{real}} = (Y_{[1,T]}^{\mathrm{real}}, S_T^{\mathrm{real}}, X_{[T+1,\infty)}, K)$ has statistical difference at most $T \cdot \varepsilon$ from Z_T^{ideal} defined analogously. We prove this by induction on T. Z_T^{real} is obtained from Z_{T-1}^{real} by applying a function f_T which updates the state $S_T^{\mathrm{real}} = A(Y_{[1,T-1]}^{\mathrm{real}}, S_{T-1}^{\mathrm{real}}, X_T)$, computes $Y_T^{\mathrm{real}} = \mathrm{PRG}(X_T, K)$, and removes X_T from the distribution. Then we have:

$$\begin{aligned}
\Delta(Z_T^{\mathrm{real}}, Z_T^{\mathrm{ideal}}) &= \Delta(f_T(Z_{T-1}^{\mathrm{real}}), Z_T^{\mathrm{ideal}}) \\
&\leq \Delta(f_T(Z_{T-1}^{\mathrm{real}}), f_T(Z_{T-1}^{\mathrm{ideal}})) + \Delta(f_T(Z_{T-1}^{\mathrm{ideal}}), Z_T^{\mathrm{ideal}}) \\
&\leq \Delta(Z_{T-1}^{\mathrm{real}}, Z_{T-1}^{\mathrm{ideal}}) + \Delta(f_T(Z_{T-1}^{\mathrm{ideal}}), Z_T^{\mathrm{ideal}}) \\
&\leq (T-1) \cdot \varepsilon + \Delta(f_T(Z_{T-1}^{\mathrm{ideal}}), Z_T^{\mathrm{ideal}}).
\end{aligned}$$

Thus it suffices to show that the following two random variables are ε-close:

$$\begin{aligned}
f_T(Z_{T-1}^{\mathrm{ideal}}) &= (Y_{[1,T-1]}^{\mathrm{ideal}}, \mathrm{PRG}(X_T, K), A(Y_{[1,T-1]}^{\mathrm{ideal}}, S_{T-1}^{\mathrm{ideal}}, X_T), X_{[T+1,\infty)}, K), \text{ and} \\
Z_T^{\mathrm{ideal}} &= (Y_{[1,T-1]}^{\mathrm{ideal}}, Y_T^{\mathrm{ideal}}, A(Y_{[1,T-1]}^{\mathrm{ideal}}, S_{T-1}^{\mathrm{ideal}}, X_T), X_{[T+1,\infty)}, K).
\end{aligned}$$

The proof of this proceeds as follows. By the definition of a reverse block source, X_T is an αn-source even conditioned on $X_{[T+1,\infty)}$. We can view the adversary's new state $A(Y_{[1,T-1]}^{\mathrm{ideal}}, S_{T-1}^{\mathrm{ideal}}, X_T)$ as a probabilistic function of X_T mapping to $\{0,1\}^{\beta n}$. Thus, by the hypothesis of the lemma,

$(K, \mathrm{PRG}(X_T, K))$ is ε-close to $(K, U_m) \equiv (K, Y_T^{\mathrm{ideal}})$ even conditioned on all other components of $f_T(Z_{T-1}^{\mathrm{ideal}})$. Thus, $f_T(Z_{T-1}^{\mathrm{ideal}})$ and Z_T^{ideal} have statistical difference at most ε, as desired. □

4 Efficiency Criteria and Results

In addition to providing security, it is important for BSM pseudorandom generators 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 PRG 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 PRG to much less than n, but also the time spent.[5] Thus, we adopt as our main 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 PRG is efficient with respect to both time and space. In our constructions (and all previous constructions), PRG can be computed in polynomial time and polylogarithmic work space (indeed even in NC).[6] Thus the total storage required by the legitimate parties is dominated by the number of bits read from the source.

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.[7] With respect to both measures, our constructions give an asymptotic improvement over previous constructions. In Section 9 we give lower bounds showing that our parameters are within a constant factor of optimal. In fact, for the number of bits read from the source, this constant factor can be made arbitrarily close to 1 in the (natural) case that $\varepsilon = 2^{-o(m)}$.

We now touch upon a couple of additional efficiency considerations. First, if the random source is implemented as a high-rate stream, 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 for computing the pseudorandom generator 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 computation does not require any workspace beyond the locations where it writes its output. This property holds for most of the previous constructions, as each bit of the output 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] If having PRG computable in small space with one pass through the random source is considered sufficiently efficient, then the work of Bar-Yossef et al. [BRST02] is also applicable here. See Section 5.

[6] Actually, in Section 7 we present some nonconstructive results showing the existence of BSM pseudorandom generators that read few bits from the source, but are not efficiently computable. In Section 8, we present explicit constructions that come close to the nonconstructive bounds, and these are the results we refer to below in Figure 1.

[7] 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) [Lu]. 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.

reference	key length	# bits read	restrictions
[CM97]	$O(\log n)$	$O(m/\varepsilon^2)$	interactive
[AR99]	$O(\log n \cdot \log(1/\varepsilon))$	$O(m \cdot \log(1/\varepsilon))$	$\alpha = 1$,$\beta < 1/m$
[ADR02]	$O(m \cdot \log n \cdot \log(1/\varepsilon))$	$O(m \cdot \log(1/\varepsilon))$	
[DR02]	$O(\log n \cdot \log(1/\varepsilon))$	$O(m \cdot \log(1/\varepsilon))$	$\alpha = 1$, $\beta < 1/\log m$
[DM]	$O(\log n \cdot \log(1/\varepsilon))$	$O(m \cdot \log(1/\varepsilon))$	
[Lu]	$O(m \cdot (\log n + \log(1/\varepsilon)))$	$O(m \cdot \log(1/\varepsilon))$	
[Lu]	$O((\log^2(n/\varepsilon)/\log n))$	$O(m \cdot \log(1/\varepsilon))$	$m \leq n^{1-\Omega(1)}$
here	$O(\log n + \log(1/\varepsilon))$	$O(m + \log(1/\varepsilon))$	$\varepsilon > \exp(-n/2^{O(\log^* n)})$

Figure 1: Comparison of pseudorandom generators in the bounded storage model. Parameters are for ε-secure schemes $\mathrm{PRG} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, for constant storage rate β and min-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.

The fact that in previous constructions each bit of the pseudorandom generator's output depends on only $O(\log(1/\varepsilon))$ bits of the source has one other potential benefit: it can enable error-correction in case the random source cannot be accessed perfectly [Rab01]: Assume that the pseudorandom generator's output is used as a one-time pad for encryption. Then a bit-error probability of $O(1/\log(1/\varepsilon))$ in accessing the source can be viewed as a constant bit-error probability in the output of the pseudorandom generator, which in turn can be viewed as constant bit-error probability in the plaintext (since the ciphertext is the XOR of the pseudorandom generator's output and the plaintext). A constant bit-error probability in the plaintext can be overcome by encoding messages in a standard, asymptotically good error-correcting code.

5 Locally Computable Extractors

In this section, we define extractors and locally computable extractors, and recall Lu's result [Lu] about their applicability to the bounded storage model.

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 5.1 ([NZ96]) $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* strong[8] (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.

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

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

Theorem 5.2 (implicit in [Lu]) *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 every* $\beta > 0$*,* $\mathrm{PRG} = \mathrm{Ext}$ *is an* 2ε*-secure BSM pseudorandom generator for storage rate* β *and min-entropy rate* $\beta + \delta$*.*

Proof Sketch: We use Lemma 3.3. Let $A : \{0,1\}^n \to \{0,1\}^{\beta n}$ be any function, and let X be any $(\beta+\delta)n$-source. We need to show that $(\mathrm{Ext}(X, U_d), A(X), U_d)$ is 2ε-close to $(U_m, A(X), U_d)$. As in [NZ96, Lu], with probability at least $1 - \varepsilon$ over $s \overset{\mathrm{R}}{\leftarrow} A(X)$, the random variable $X|_{A(X)=s}$ is a $((\beta+\delta)n - \beta n - \log(1/\varepsilon))$-source (because $|s| = \beta n$), i.e. $X|_{A(X)=s}$ is a $(\delta n - \log(1/\varepsilon))$-source. For each such s, the strong extractor property says that $(U_d, \mathrm{Ext}(X|_{A(X)=s}, U_d))$ is ε-close to $U_d \times U_m$. It follows that $(A(X), U_d, \mathrm{Ext}(X, U_d))$ is 2ε-close to $(A(X), U_d, U_m)$. □

However, as noted by Lu, we cannot simply use off-the-shelf constructions of extractors. Recall that our main efficiency measure in the bounded storage model is the number of bits read from the source. In contrast, most works in the extractor literature assume that the source can be accessed in its entirety. Thus we make the following definition:

Definition 5.3 $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *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. Our results will show that reading $t = m/\delta + \Theta(\log(1/\varepsilon)$ bits is necessary and sufficient to construct a t-local $(\delta n, \varepsilon)$-extractor (for constant δ).

Bar-Yossef, Reingold, Shaltiel, and Trevisan [BRST02] studied a related notion of *on-line extractors*, which are required to be computable in small space in one pass, but we feel that locally computable extractors are preferable in the bounded-storage model (where the source is envisioned to be huge).

Lu [Lu] observed that the encryption schemes of Aumann, Ding, and Rabin [AR99, ADR02, DR02] can be viewed as locally computable extractors, albeit with long seeds. He constructed locally computable extractors with shorter seeds based on Trevisan's extractor [Tre01]. The construction of Dziembowski and Maurer [DM] is also a locally computable extractor. The parameters of these constructions can be deduced from Figure 1.

6 Sample-then-Extract

There are many constructions of extractors in the literature that have excellent parameters, but lack the local computability property that we need. Thus we seek a general method for converting non-local extractors into local ones. Our approach is to separate the task of a local extractor into two parts: the "sample" step — deciding which bits to read from the source, and the "extract" step — the actual processing of those bits. In our approach, the extract step will be done by an extractor, but this extractor need not be local (since it only operates on the bits selected in the sample step) and hence can be taken from the existing literature. So the question is how to perform and analyze the sample step.

The sample step is based on a fundamental lemma of Nisan and Zuckerman [NZ96], 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 [ADR02], because $|S| \geq m$.) To save on randomness, Nisan and Zuckerman [NZ96] 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., [RSW00]). 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 [BR94, CEG95, Zuc97, Gol97]. Our definition differs slightly from the standard definition, to allow us to obtain some savings in parameters (discussed later).

Definition 6.1 *A function* 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*

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

Samp *has* distinct samples *if for every* $x \in \{0,1\}^r$, *the samples produced by* 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 usually require that the sample average is within θ of the global average, regardless of the average value of f, but being explicit about μ will allow us to obtain some savings in the parameters.[9]

[9] Averaging samplers also typically consider deviations above the average in addition to deviations below the average as in Definition 6.1. However, if the sampler works for all values of μ, then a bound on deviations below implies a bound on deviations above by considering the function $1 - f$.

Using averaging samplers (rather than just samplers that intersect large sets) together with an idea from [Ta-02] 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 6.2 (refining [NZ96]) *Let $1 \geq \delta \geq 3\tau > 0$. 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 $\theta = \tau/\log(1/\tau)$. 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 (U_r, W) where for every $a \in \{0,1\}^r$,*[10] *the random variable $W|_{U_r=a}$ 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 [RSW00], as a subroutine in their improved extractor construction. However, the string produced by their method consists of bits of an encoding of the source in an error-correcting code rather than bits of the source itself. Since computing even a single bit of the encoding might require reading all of the bits in the source, their method is not suitable 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 6.2 eliminates the need for error-correcting codes in [RSW00].

The proof of Lemma 6.2 is deferred to Section 6.1. Given the lemma, it follows that combining an averaging sampler and an extractor yields a locally computable extractor.

Theorem 6.3 (sample-then-extract) *Let $1 \geq \delta \geq 3\tau > 0$. 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 that $\mathrm{Ext} : \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ 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 6.2, $(U_r, X_{\mathrm{Samp}(U_r)})$ is $(\gamma+2^{-\Omega(\tau n)})$-close to (U_r, W) where $W|_{U_r=a}$ is a $(\delta-3\tau)t$-source for every a. This implies that Z is $(\gamma+2^{-\Omega(\tau n)})$-close to $(U_r, U_d, \mathrm{Ext}(W,U_d))$. Since Ext is a strong $((\delta-3\tau)t,\varepsilon)$ extractor, $(U_d, \mathrm{Ext}(W|_{U_r=a}, U_d))$ is ε-close to $U_d \times U_m$ for all a. This implies that $(U_r, U_d, \mathrm{Ext}(W,U_d))$ is ε-close to $U_r \times U_d \times U_m$. By the triangle inequality, Z is $(\varepsilon+\gamma+2^{-\Omega(\tau n)})$-close to $U_r \times U_d \times U_m$. ■

[10] 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.

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 $m = \Omega(\delta t) - O(\log(1/\varepsilon))$. As we will see in Section 9, both of these bounds are optimal up to constant factors.

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 [NZ96] together with the extractor of Zuckerman [Zuc97]. The advantage provided by the refined lemma lies in the hidden constant in the number of bits read from the source. Specifically, when $\varepsilon = 2^{-o(t)}$ and we take $\tau \to 0$, then m approaches the optimal bound of δt. With the Nisan–Zuckerman lemma, however, the min-entropy guaranteed in the t sampled bits is smaller than $(\delta/\log(1/\delta)) \cdot t$, and thus we can extract at most $m < (\delta/\log(1/\delta)) \cdot t$ almost-uniform bits.

6.1 Proof of Lemma 6.2

In this section, we prove Lemma 6.2. For readers willing to assume the lemma, this section can be skipped. Our proof has the same general structure as the Nisan–Zuckerman proof; the main point of departure will be pointed out below.

Recall our convention that capital letters denote random variables and that lower-case letters denote specific values for them.

Let X be a δn-source X and let $S = \mathrm{Samp}(U_d)$. We will write X_i (resp., x_i) for the i'th bit of X (resp., x). (Note that this different from Section 3 where X_i denotes the i'th block of a reverse block source.)

For every $x \in \mathrm{Supp}(X)$, define $p_i(x) = \Pr[X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}]$. Let $h_i(x) = \log(1/p_i(x))$. Intuitively, $h_i(x)$ is the min-entropy contributed by the i'th bit of x. Note that for any $x \in \mathrm{Supp}(X)$, $\Pr[X = x] = \prod_i p_i(x) = 2^{-\sum_i h_i(x)}$. Since X is a δn-source, $2^{-\sum_i h_i(x)} \leq 2^{-\delta n}$, i.e. the average of $h_i(x)$ is at least δ for every $x \in \mathrm{Supp}(X)$. The intuition for the lemma is that, with high probability, the sampler will select a sequence of positions i on which this average is preserved, and hence the min-entropy rate will be preserved on the sample. One problem is that the sampler is guaranteed to work for $[0,1]$-valued functions, but $h_i(x)$ may be greater than 1. Thus, we truncate large values and argue that we do not lose much by doing this. Specifically, let $h'_i(x) = \min\{h_i(x), \log(1/\tau)\}$.

Here we see the main difference between our proof and the Nisan–Zuckerman proof (which is analogous to one of the improvements made by Ta-Shma [Ta-02] to the disperser construction of [SSZ98]). In the Nisan–Zuckerman proof, they consider the set of positions i where $p_i(x) \leq 1/2$, and argue that there are usually many such positions so the sampler will hit this set many times. This can be seen as counting either 1 or 0 bits of min-entropy per position (define $h'_i(x)$ to be 1 if $h_i(x) \geq 1$ and 0 otherwise). Here, the bits of min-entropy we count per position can be fractional or greater than 1 (up to $\log(1/\tau)$). This allows us to lose less min-entropy, but forces us to use samplers for functions (rather than sets).

We argue now that truncating the h_i's does not cost us too much min-entropy. Call x *well-spread* if $\sum_i h'_i(x) \geq (\delta - 2\tau)n$ (i.e. truncating costs at most 2τ in the min-entropy rate).

Claim 6.4 $\Pr[X$ *is not well-spread*$] \leq 2^{-\Omega(\tau n)}$.

Proof of claim: Consider the random variables $\Delta_1, \ldots, \Delta_n$ defined by $\Delta_i = h_i(X) - h'_i(X)$. We need to show that $\Pr[\sum_i \Delta_i > 2\tau n] \leq 2^{-\Omega(\tau n)}$. We first bound the tails of the individual Δ_i's, conditioned on any prefix. For any $q > 0$ and $x \in \{0,1\}^{i-1}$, we have

$$\begin{aligned}
&\Pr\left[\Delta_i = q | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}\right] \\
&\quad = \Pr\left[h_i(X) = q + \log(1/\tau) | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}\right] \\
&\quad = \Pr\left[p_i(X) = 2^{-q+\log(1/\tau)} | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}\right] \\
&\quad = \begin{cases} 2^{-q+\log(1/\tau)} & \text{if } \exists x_i \in \{0,1\} \text{ s.t. } \Pr\left[X_i = x_i | X_1 = x_1 \cdots X_{i-1} = x_{i-1}\right] = 2^{-q+\log(1/\tau)} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

In particular, the above conditional probability is nonzero for at most one value of q, and thus, for any $q > 0$,

$$\Pr\left[\Delta_i \geq q | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}\right] \leq \tau \cdot 2^{-q}. \tag{1}$$

Since each Δ_j is a function of $X_1, \ldots, X_j$, this implies that for every $q_1, q_2, \ldots, q_{i-1} \geq 0$ and $q > 0$,

$$\Pr\left[\Delta_i \geq q | \Delta_1 = q_1, \ldots, \Delta_{i-1} = q_{i-1}\right] \leq \tau \cdot 2^{-q}. \tag{2}$$

The claim now follows from a Chernoff-type bound proven in Appendix A, which says that for any sequence of nonnegative random variables satisfying the Inequalities (2), $\Pr\left[\sum_i \Delta_i > 2\tau n\right] < 2^{-\Omega(\tau n)}$. □

Let's call a sequence $s = (i_1, \ldots, i_t) \in [n]^t$ *good* for x if

$$\frac{1}{t} \sum_{j=1}^{t} h'_{i_j}(x) \geq \delta - 3\tau,$$

and otherwise call s *bad* for x. Let $b(s) = \Pr\left[s \text{ is bad for } X\right]$. First we argue that the sampler rarely produces bad sequences.

Claim 6.5 $\mathrm{E}[b(S)] \leq \gamma + 2^{-\Omega(\tau n)}$.

Proof of claim: Consider any well-spread element $x \in \mathrm{Supp}(X)$. Consider the function $f : [n] \to [0,1]$ defined by $f(i) = h'_i(x)/\log(1/\tau)$. Because x is well-spread, f has average value at least $\mu = (\delta - 2\tau)/\log(1/\tau)$. A sequence s is good for x iff the average of f on s is at least $(\delta - 3\tau)/\log(1/\tau) = \mu - \theta$. Thus, since S comes from an (μ, θ, γ) averaging sampler, we have $\Pr\left[S \text{ is not good for } x\right] \leq \gamma$ (for any well-spread x). Now, averaging over $x \overset{\mathrm{R}}{\leftarrow} X$, we get:

$$\begin{aligned}
\mathrm{E}[b(S)] &= \Pr\left[S \text{ is not good for } X\right] \\
&\leq \Pr\left[S \text{ is not good for } X | X \text{ well-spread}\right] + \Pr\left[X \text{ not well-spread}\right] \\
&\leq \gamma + 2^{-\Omega(\tau n)}.
\end{aligned}$$

□

Next, we show that good sequences yield high min-entropy on the sampled bits.

Claim 6.6 *For every s, the random variable X_s is $b(s)$-close to a $(\delta - 3\tau)t$-source.*

> **Proof of claim:** Fix $s = (i_1, \ldots, i_t)$. The basic idea is to "fix" the bits of X in positions outside of s, and then argue that whenever s is good for x, x_s has low probability mass. Then averaging over the fixed bits can only increase the min-entropy. However, as in the proof of [NZ96], fixing the bits of X is somewhat delicate because of the correlations between the bits of X.
>
> We can envision X (indeed, any random variable on $\{0,1\}^n$) being generated by a process of the following form. The probability space consists of independent random variables $F_1, \ldots, F_n$, where F_i is distributed (arbitrarily) over functions from $\{0,1\}^{i-1} \to \{0,1\}$, and, given these functions, X is deterministically generated by setting $X_i = F_i(X_1 X_2 \cdots X_{i-1})$.
>
> We will fix F_i for every $i \notin s$. Consider any $\overline{f} = (f_i)_{i \notin s}$, and let $X^{\overline{f}}$ denote X conditioned on $F_i = f_i$ for all $i \notin s$. Then, for any string $z \in \{0,1\}^t$ in the support of $X_s^{\overline{f}}$, there is a unique $x \in \{0,1\}^n$ in the support of $X^{\overline{f}}$ such that $x_s = z$. (Namely the x inductively defined by setting $x_i = z_i$ if $i \in s$ and $x_i = f_i(x_1, \ldots, x_{i-1})$ if $i \notin s$.) We denote this x by $x^{\overline{f}}(z)$. Then
>
> $$\begin{aligned} \Pr\left[X_s^{\overline{f}} = z\right] &= \Pr\left[X^{\overline{f}} = x^{\overline{f}}(z)\right] \\ &= \prod_{i=1}^{n} \Pr\left[X_i^{\overline{f}} = x^{\overline{f}}(z)_i | X_1^{\overline{f}} = x^{\overline{f}}(z)_1, \ldots, X_{i-1}^{\overline{f}} = x^{\overline{f}}(z)_{i-1}\right] \end{aligned}$$
>
> The conditional probability in the i'th factor above is 1 when $i \notin s$ (because then $X_i^{\overline{f}} = f_i(X_1^{\overline{f}} \cdots X_{i-1}^{\overline{f}})$ always) and equals $p_i(x^{\overline{f}}(z))$ otherwise. Thus, if s is good for $x^{\overline{f}}(z)$ (and consists of distinct samples), we have:
>
> $$\Pr\left[X_s^{\overline{f}} = z\right] = \prod_{i \in s} p_i\left(x^{\overline{f}}(z)\right) \leq \prod_{i \in s} 2^{-h_i'(x^{\overline{f}}(z))} \leq 2^{-(\delta - 3\tau)t}.$$
>
> Let $b(s, \overline{f})$ be the probability that s is bad for $X^{\overline{f}}$. Then, by the above, $X_s^{\overline{f}}$ is $b(s, \overline{f})$-close to some $(\delta - 3\tau)t$-source, call it $Z^{\overline{f}}$. Now consider the random variable $\overline{F} = (F_i)_{i \notin s}$. Then $X_s = X_s^{\overline{F}}$ and $Z^{\overline{F}}$ is a convex combination of $(\delta - 3\tau)t$-sources and hence is a $(\delta - 3\tau)t$-source itself. The statistical difference between X_s and $Z^{\overline{F}}$ is at most $\mathrm{E}[b(s, \overline{F})] = b(s)$, as desired. □

Now we deduce Lemma 6.2 from Claims 6.5 and 6.6. By Claim 6.6, X_s is $b(s)$-close to some $(\delta - 3\tau)t$-source Z_s. Consider the random variable $(U_r, Z_{\mathrm{Samp}(U_r)})$ (i.e. first sample $y \xleftarrow{\mathrm{R}} U_r$, then sample $z \xleftarrow{\mathrm{R}} Z_{\mathrm{Samp}(y)}$, and output (y, z)). The statistical difference between $(U_r, Z_{\mathrm{Samp}(U_r)})$ and $(U_r, X_{\mathrm{Samp}(U_r)})$ is exactly the expected statistical difference between X_s and Z_s over $s \xleftarrow{\mathrm{R}} \mathrm{Samp}(U_r) = S$, which is at most $\mathrm{E}[b(S)] \leq \gamma + 2^{-\Omega(\tau n)}$ by Claim 6.5. Thus, $(A, B) = (U_r, Z_{\mathrm{Samp}(U_r)})$ satisfies the requirements of Lemma 6.2.

7 Non-Explicit Constructions

In this section, we describe the locally computable extractors obtained by using truly optimal extractors and samplers in Theorem 6.3. 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 6.3 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 8) match the optimal bounds within constant factors for the range of parameters most relevant to the bounded storage model.

7.1 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 [RT00].

Lemma 7.1 (nonconstructive extractors (cf., [Zuc97, RT00])) *For every n and $\delta, \varepsilon > 0$, there exists a strong $(\delta n, \varepsilon)$-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with $d = \log((1-\delta)n) + 2\log(1/\varepsilon) + O(1)$, $m = \delta n - 2\log(1/\varepsilon) - O(1)$.*

7.2 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 [CEG95] (except for the dependence on μ, which was not considered there). The proof of the lemma follows the argument implicit in [Zuc97], with the modifications that it makes the dependence on μ explicit and guarantees distinct samples.

Lemma 7.2 (nonconstructive samplers) *For every $n \in \mathbb{N}$, $1/2 > \mu > \theta > 0$, $\gamma > 0$, there exists 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.*

Proof: Let $R = 2^r$ so we can associate $[R] = \{0,1\}^r$. For simplicity, in this proof we assume that the quantities γR, μn, and n/t are all integers.[11] Thus we can write $n = t \cdot n_0$ for an integer n_0 and associate $[n] = [t] \times [n_0]$. We will consider a randomly chosen function $\mathrm{Samp}_0 : [R] \to [n_0]^t$ and set $\mathrm{Samp}(x)_j = (j, \mathrm{Samp}_0(x))$. (This guarantees distinct samples.)

Instead of directly proving that Samp is a sampler, we will first prove that it has the following property with high probability.

[11] The integrality assumptions for γR and μn can easily be achieved by instead constructing a (μ', θ', γ') averaging sampler for $\mu' = \lfloor \mu n \rfloor / n$, $\theta' = \theta \cdot \mu'/\mu$, and $\gamma' = \lfloor \gamma R \rfloor / R$, which will then also be a (μ, θ, γ) averaging sampler. To remove the integrality assumption on n/t, the sampler below, $\mathrm{Samp} : [R] \to [n]^t$, should be chosen as a random function mapping each $x \in [R]$ to a uniformly selected sequence of t distinct samples from $[n]$ rather than defining it in terms of Samp_0. Then the random variables $\{X_{x,j}\}$ below are no longer independent, but they do have "negative dependence" so the Chernoff bound still applies (cf., [DS01]).

Claim 7.3 Samp *has the following property with probability at least* $1/2$. *For every subset* $S \subset [R]$ *of size* γR*, and every* boolean $f : [n] \to \{0,1\}$ *with average value* μ*,*

$$\Pr_{x \overset{R}{\leftarrow} S, j \overset{R}{\leftarrow} [t]} [f(\mathrm{Samp}(x)_j) = 1] \geq \mu - \theta$$

Proof of claim: Consider a fixed S and boolean f as in the claim. For every $x \in S$ and $j \in [t]$, define the random variable $X_{x,j} = f(\mathrm{Samp}(x)_j)$ (over the choice of Samp). These are $|S| \cdot t$ independent random variables. Let X be the average of the $X_{x,j}$'s. The expectation of X is the average value of f, which equals μ. We wish to show that $X \geq \mu - \theta$ with very high probability.

By a Chernoff Bound (e.g. [ASE92, Thm A.13]),

$$\Pr_{\mathrm{Samp}} [X < \mu - \theta] \leq \exp\left(-\Omega(|S| \cdot t \cdot \theta^2/\mu)\right) \tag{3}$$

Thus, taking a union bound over f and S, the probability that Samp fails to satisfy the claim is at most

$$\begin{aligned} & \exp\left(-\Omega(|S| \cdot t \cdot \theta^2/\mu)\right) \cdot \binom{R}{\gamma R} \cdot \binom{n}{\mu n} \\ \leq\ & \exp\left(-\Omega(|S| \cdot t \cdot \theta^2/\mu)\right) \cdot \left(\frac{eR}{\gamma R}\right)^{\gamma R} \cdot \left(\frac{en}{\mu n}\right)^{\mu n} \\ \leq\ & \exp\left(-\Omega(\gamma R \cdot t \cdot \theta^2/\mu)\right) \cdot \exp(\log(1/\gamma) \cdot \gamma R) \cdot \exp(\log(1/\mu) \cdot \mu n) \end{aligned}$$

This probability is at most $1/2$ if the following two conditions hold for a sufficiently large constant c:

$$\gamma \cdot R \cdot t \cdot \theta^2/\mu \geq c \cdot \gamma \cdot \log(1/\gamma) \cdot R,$$

and

$$\gamma \cdot R \cdot t \cdot \theta^2/\mu \geq c \cdot \mu \cdot \log(1/\mu) \cdot n.$$

The first condition is $t \geq c \cdot (\mu/\theta^2) \cdot \log(1/\gamma)$, which is guaranteed by the hypothesis of the lemma. The second condition says $R \geq c \cdot (n/t) \cdot (1/\gamma) \cdot (\mu/\theta)^2 \cdot \log(1/\mu)$. Taking logs, we obtain exactly the condition on r in the hypothesis of the lemma. □

We now argue that any Samp satisfying Claim 7.3 is a (μ, θ, γ) averaging sampler. Suppose not. Then there is a (not necessarily boolean) function $f : [n] \to [0,1]$ with average value μ such that

$$\Pr_{x \overset{\mathrm{R}}{\leftarrow} U_r} \left[\frac{1}{t} \sum_{j=1}^{t} f(\mathrm{Samp}(x)_j) < \mu - \theta \right] > \gamma$$

That is, there is a set S of γR seeds $x \in [R]$ such that $\frac{1}{t}\sum_{j=1}^{t} f(\mathrm{Samp}(x)_j) < \mu - \theta$. In particular, we have:

$$\Pr_{x \overset{\mathrm{R}}{\leftarrow} S, j \overset{\mathrm{R}}{\leftarrow} [t]} [f(\mathrm{Samp}(x)_j) = 1] < \mu - \theta \tag{4}$$

This would contradict Claim 7.3, except that f is not necessarily boolean. However, every $[0,1]$-valued function with average value μ is a convex combination of boolean functions with average value μ.[12] Thus, the function f in Inequality (4) can be viewed as a distribution over boolean functions. By averaging, there is a boolean function satisfying Inequality (4), in contradiction to Claim 7.3. ■

7.3 The Local Extractor

Plugging the above two lemmas into Theorem 6.3, we obtain the following local extractors.

Theorem 7.4 (nonconstructive local extractors) *Let $\kappa > 0$ be an arbitrary constant. For every $n \in \mathbb{N}$, $\delta > 0$, $\varepsilon > 0$, and $m \leq \delta n - 2\log(1/\varepsilon) - O(1)$, there exists a t-local strong $(\delta n, \varepsilon)$ extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with*

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

Proof: Set $\tau = \kappa\delta/6$, $\mu = (\delta - 2\tau)/\log(1/\tau)$, and $\theta = \tau/\log(1/\tau)$. By Lemma 7.2, there exists a $(\mu, \theta, \varepsilon/3)$ averaging sampler $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ with distinct samples, using

$$\begin{aligned} r &= \log(n/t) + \log(2/\varepsilon) + 2\log(\mu/\theta) + \mathrm{loglog}(1/\mu) + O(1) \\ &= \log n - \log t + \log(1/\varepsilon) + \mathrm{loglog}(1/\delta) + O(1) \end{aligned}$$

random bits, provided $t \in [t_0, n]$ for $t_0 = \Theta\left((\mu/\theta^2)\cdot\log(2/\varepsilon)\right) = O\left(\log(1/\delta)\cdot\log(1/\varepsilon)/\delta\right)$, which is satisfied by the above setting for t. (If $t \geq n$, then the theorem follows from the existence of standard (non-local) extractors in Lemma 7.1.)

By Lemma 7.1, there is a strong $((1-\kappa/2)\delta t, \varepsilon/3)$ extractor $\mathrm{Ext} : \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ with $d = \log t + 2\log(1/\varepsilon) + O(1)$, provided $m \leq (1-\kappa/2)\delta t - 2\log(1/\varepsilon) - O(1)$. This follows from the above setting for t (for sufficiently small κ).

Noting that $(\delta - 3\tau)t = (1-\kappa/2)\delta t$, we combine these via Theorem 6.3 to obtain a t-local extractor with seed length $r + d = \log n + 3\log(1/\varepsilon) + \mathrm{loglog}(1/\delta) + O(1)$. The error of the extractor is $2\varepsilon/3 + 2^{-\Omega(\kappa\delta n)}$, which we can assume is at most ε (otherwise, we can set $t = n$ and again the result follows from Lemma 7.1). ■

Using Theorem 5.2 (of [Lu]), we obtain the following BSM pseudorandom generators.

Corollary 7.5 (nonconstructive BSM pseudorandom generators) *Let $\delta, \kappa > 0$ be arbitrary constants. Then for every $n \in \mathbb{N}$, $\varepsilon > 0$, and $m \leq \delta n - 2\log(1/\varepsilon) - O(1)$, there exists a BSM pseudorandom generator* $\mathrm{PRG} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *such that*

1. *For every $\beta > 0$,* PRG *is ε-secure for min-entropy rate $\beta + \delta$ and storage rate β.*
2. PRG *has key length $d = \log n + 3\log(1/\varepsilon) + O(1)$.*
3. *For every key K,* $\mathrm{PRG}(\cdot, K)$ *reads at most $t = (1+\kappa)m/\delta + O(\log(1/\varepsilon))$ bits from the source (nonadaptively).*

When $\varepsilon = 2^{-o(m)}$ and κ is set to be close to zero, then $t \approx m/\delta$, which is the optimal relationship (as we will prove in Section 9). The dependence of the hidden constants on κ can be deduced from the proof of Theorem 7.4, and is omitted for sake of readability.

[12] This is equivalent to the fact that every distribution of min-entropy k is a convex combination of flat distributions of min-entropy k. Alternatively, it can be proven directly by calculating the vertices of the parallelepiped consisting of all $[0,1]$-valued functions with average value μ.

8 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 from it).

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 6.3. 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 min-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.

The construction we present here is aimed at optimizing asymptotic performance (given the above considerations), in particular to achieve the parameters in Figure 1 and even to have the number of bits read from the source approach the lower bounds we give in Section 9. However, if this level of optimization is not needed, there are simpler constructions of extractors and samplers that can be used in the sample-then-extract paradigm, and indeed some previous constructions of cryptosystems in the bounded storage model can be obtained in this way. We discuss these previous constructions and other appealing choices for the extractor and sampler in Section 8.4.

8.1 The Extractor

With the above criteria in mind, the most natural extractor to use (in Theorem 6.3) is Zuckerman's extractor for constant min-entropy rate [Zuc97]:

Lemma 8.1 ([Zuc97]) *Let $\delta, \kappa > 0$ be arbitrary constants. For 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 $m = (1-\kappa)\delta n$.*

8.2 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 [NZ96] 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 6.2, at the price of a constant factor in the number of bits read from the source.) Thus we introduce some new techniques to deal with these issues. We also note that our use of expander walks to construct locally computable extractors is similar to Lu's use of expander walks to construct a locally computable list-decodable error-correcting code (equivalently, locally computable extractor with output length 1) [Lu] (cf., Section 8.4).

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

Lemma 8.2 (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.*

Proof: Consider an explicit d-regular expander graph G on n vertices.[13] Let M be the adjacency matrix of G divided by d, i.e. the stochastic matrix which describes the random walk on G. M has an eigenvalue of 1, and we require that all its other eigenvalues have absolute value less than $\lambda = \theta/16$. This is possible to achieve with degree $d = \mathrm{poly}(1/\theta)$ (by taking an appropriate power of any explicit constant-degree expander with bounded second eigenvalue, e.g. [Mar73, GG81]).

The standard expander sampler outputs a random walk $w = (w_1, \ldots, w_t)$ on G of length t started at a random vertex w_1. For any function $f : [n] \to [0,1]$, Gillman's Chernoff bound for random walks on expanders [Gil98] says:

$$\Pr_w\left[\left|\frac{1}{t}\sum_{i=1}^{t} f(w_i) - \mu\right| > \theta/2\right] < \exp\left(-\Omega(\theta^2 t)\right) < \gamma/4,$$

for $t \geq O(\log(1/\gamma)/\theta^2)$.

This sampler uses $r_0 = \log n + t \cdot \log d = \log n + O(t \cdot \log(1/\theta))$ random bits, as desired. However, it does not guarantee distinct samples. We now show that discarding the repeated vertices does not affect the sampler too much. For a walk $(w_1, \ldots, w_t)$, call w_i a *repeat* if there exists a $j < i$ such that $w_j = w_i$, and *new* otherwise. We modify our sampler to use the first $(1 - (\theta/2))t$ new vertices in the walk, and output `fail` if there are not this many new vertices. If failure does not occur, this sampler uses distinct samples and the approximation is only affected by an additive error of at most $\theta/2$. We will describe what to do in case of failures later.

[13] Actually, we do not have explicit constructions of expander graphs for every n. Instead, we use an expander with $n' \in [n, (1+\theta)n]$ vertices and extend f by setting $f(i) = 0$ for all $i \in [n'] \setminus [n]$ (which can be simulated by arbitrarily reassigning samples in $[n'] \setminus [n]$ to values in $[n]$ subject to distinctness).

We now bound the failure probability. For a fixed $j < i$, the probability over a random walk w that $w_i = w_j$ is exactly the trace of M^{i-j} divided by n, which equals the sum of the eigenvalues of M^{i-j}. We use this to bound the probability that w_i is a repeat as follows.

$$\Pr_w[\exists j < i, w_j = w_i] \leq \sum_{k=1}^{i-1} \frac{1}{n}\mathrm{Tr}(M^k) \leq \sum_{k=1}^{i-1} \frac{(1+(n-1)\lambda^k)}{n} \leq \frac{t}{n} + 2\lambda \leq \theta/4,$$

where the last inequality assumes $t \leq \theta n/8$ (otherwise $\log\binom{n}{t} = O(t\cdot\log(n/t)) = O(t\cdot\log(1/\theta)) < r$, so we have enough randomness to sample a uniformly random subset of $[n]$ of size t). Thus, the expected fraction of vertices in a random walk that are repeats is at most $\theta/4$. By Markov's inequality, the probability that there are more than $\theta/2$ fraction of repeats is at most $1/2$.[14]

So our modified sampler outputs `fail` with probability at most $1/2$, and conditioned on non-failure, outputs a sample with error greater than θ with probability at most $(\gamma/4)/(1/2) = \gamma/2$. Thus our task is reduced to sampling from the non-failing subset of our sample space with high probability $(1-\gamma/2)$. This can be done by using a random walk of length $O(\log(1/\gamma))$ on an expander on 2^{r_0} vertices, for a total of $r = r_0 + O(\log(1/\gamma))$ random bits. (When this sampler doesn't succeed in finding a non-failing expander walk, it outputs an arbitrary t-subset, e.g. $(1,\ldots,t)$.) ■

A drawback of the expander sampler is that the randomness increases with the number of samples, whereas in the optimal sampler of Lemma 7.2 the randomness actually decreases with the number of samples. To fix this, we use a simple technique that increases the number of (distinct) samples at no cost. Roughly speaking, we view the domain as a matrix $[m] \times [n]$, use our sampler to select a subset $S \subset [n]$ of the columns, and let their union $S' = [m] \times S$ be the final sample set. Formally, we have the following lemma.

Lemma 8.3 *Suppose there is an explicit (μ, θ, γ) averaging sampler* 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* 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 6.2) has played a role in the recent construction of extractors that are "optimal up to constant factors" [LRVW03].

Proof: We associate $[m \cdot n]$ and $[m \cdot t]$ with $[m] \times [n]$ and $[m] \times [t]$, respectively, and define $\mathrm{Samp}'(x)_{(i,j)} = (i, \mathrm{Samp}(x)_j)$. For any function $f' : [m] \times [n] \to [0,1]$, define $f : [n] \to [0,1]$ by setting $f(z)$ to be the average of $f'(i,z)$ over $i \in [m]$. The average value of f is the same as the average value of f', and for any coin tosses x, if Samp(x) successfully approximates the average of f, then Samp$'(x)$ also succeeds for f'. ■

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 8.2 will

[14] It seems likely that a stronger result may be true, namely that a random walk of length t on an expander with second eigenvalue λ has an $O(t/n + \lambda)$ fraction of repeats with probability $1 - \exp(-c_\lambda t)$, but we do not have a proof of this.

use $t_0 = O(\log(1/\gamma))$ samples and $r_0 = \log n_0 + O(\log(1/\gamma)) = \log(n/t) + O(\log(1/\gamma))$ random bits. Then setting $m = t/t_0$, Lemma 8.3 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 8.4 *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 follow.

Proof: Let $h = h(\theta, \gamma) = O(\log(1/\gamma)/\theta^2)$ be the lower bound on t in Lemma 8.2. Given $n \geq t \geq h$, we can find $t_0 \in [h, 2h]$ and m such that $m \cdot t_0 \leq t \leq m \cdot t_0 + \min\{m, t_0\}$. In particular, this implies that $m \cdot t_0 \in [t - \sqrt{t}, t] \subseteq [(1-\theta)t, t]$. Now, we can find n_0 such that $m \cdot n_0 \geq n \geq m \cdot n_0 - m$, so $m \cdot n_0 \in [n, n+m] \subseteq [n, (1+\theta)n]$. By Lemma 8.2, there is a (μ, θ, γ) sampler $\mathrm{Samp}_0 : \{0,1\}^r \to [n_0]^{t_0}$ with

$$r = \log n_0 + O(t_0 \cdot \log(1/\theta)) = \log n + \log(1/\gamma) \cdot \mathrm{poly}(1/\theta) - \log t$$

(using the bound $n_0 \leq nt_0/t + 1$).

Applying Lemma 8.3, we get a sampler $\mathrm{Samp} : \{0,1\}^r \to [m \cdot n_0]^{m \cdot t_0}$. Since both $m \cdot t_0$ approximates t and $m \cdot n_0$ approximates n within relative errors of θ, we can view this sampler as a $(\mu + \theta, 2\theta, \gamma)$ sampler $\mathrm{Samp}' : \{0,1\}^r \to [n]^t$. (Samples landing in $[m \cdot n_0] \setminus [n]$ can be arbitrarily reassigned to values in $[n]$ subject to distinctness.) ■

8.3 The Local Extractor

Analogously to Theorem 7.4, we plug Lemmas 8.1 and 8.4 into Theorem 6.3 to obtain:

Theorem 8.5 (explicit local extractors) *Let $\delta, \kappa > 0$ be arbitrary constants. For every $n \in \mathbb{N}$, $\varepsilon > \exp(-n/2^{O(\log^* n)})$, and $m \leq (1-\kappa)\delta n$, 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*

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

Proof: Set $\tau = \kappa\delta/9$, $\mu = (\delta - 2\tau)/\log(1/\tau)$, and $\theta = \tau/\log(1/\tau)$. When δ and κ are constant, so are τ, μ, and θ. By Lemma 8.4, there is a $(\mu, \theta, \varepsilon/3)$ averaging sampler $\mathrm{Samp} : \{0,1\}^r \to [n]^t$ with distinct samples, using $r = \log(n/t) + O(\log(1/\varepsilon))$ random bits, provided $t \geq t_0$ for $t_0 = O(\log(1/\varepsilon))$, which is satisfied by the above setting.

By Lemma 8.1, there is a strong $((1-\kappa/3)\delta t, \varepsilon/3)$ extractor $\mathrm{Ext} : \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ with $d = O(\log t + \log(1/\varepsilon)) = O(\log m + \log(1/\varepsilon))$, provided $m \leq (1-\kappa/2)\delta t$, which is satisfied by the above setting. Noting that $(\delta - 3\tau)t = (1-\kappa/3)\delta t$, we combine these via Theorem 6.3 to obtain a t-local extractor with seed length $r + d = \log n + O(\log m + \log(1/\varepsilon))$ and error $2\varepsilon/3 + 2^{-\Omega(n)} < \varepsilon$. ■

Corollary 8.6 (explicit BSM pseudorandom generators) *Let $\delta, \kappa > 0$ be arbitrary constants. For every $n \in \mathbb{N}$, $\varepsilon > \exp(-n/2^{O(\log^* n)})$, and $m \leq (1-\kappa)\delta n$, there is a BSM pseudorandom generator $\mathrm{PRG} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ such that*

1. *For every $\beta > 0$, PRG is ε-secure for min-entropy rate $\beta + \delta$ and storage rate β.*
2. *PRG has key length $d = O(\log n + \log(1/\varepsilon))$.*
3. *For every key K, $\mathrm{PRG}(\cdot, K)$ reads at most $t = (1+\kappa)m/\delta + O(\log(1/\varepsilon))$ bits from the source (nonadaptively).*
4. *PRG is computable in time $\mathrm{poly}(t, d)$ and uses workspace $\mathrm{poly}(\log t, \log d)$ in addition to the t bits read from the source and the key of length d.*

The above construction does generalize to the case of subconstant δ. Keeping κ constant, the expression for t is actually $t = (1+\kappa)m/\delta + O(\log(1/\varepsilon)/\delta^2)$, which is not too bad compared with non-explicit construction of Theorem 7.4 and the lower bound we prove in Section 9. The seed length 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 7.4. This is due to both underlying components — the extractor (Lemma 8.1) and the averaging sampler (Lemma 8.4). 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 [Zuc97].[15] However, these constructions have a sample complexity that is only polynomial in the optimal bound, resulting in a t-local extractor with $t \geq \mathrm{poly}(\log(1/\gamma), 1/\delta)$.[16] It is an interesting open problem, posed in [Gol97], 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 [BGG93, GW97, Gol97].)

8.4 Other Sample-then-Extract Constructions

Here we describe some additional incarnations of the sample-then-extract paradigm. We begin by describing how some previous constructions of cryptosystems in the bounded storage model can be understood using our approach (namely Theorem 6.3 together with Theorem 5.2 (of [Lu])). We then mention some alternative choices for the sampler and extractor that can be used in our approach; their parameters are not quite as good for the bounded-storage model as the ones used in Theorem 8.5 above, but are appealing for their simplicity and directness (which may be important in an implementation).

In the informal discussion below, we focus on the case of constructing a t-local $(\delta n, \varepsilon)$ extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, with $\delta > 0$ constant, and $t, m, \log(1/\varepsilon) = o(n)$.

[15] The averaging samplers of [Zuc97] can be made to have distinct samples: he shows that taking any (not necessarily strong) extractor Ext, defining $\mathrm{Samp}(x)_y = \mathrm{Ext}(x, y)$ yields an averaging sampler whose parameters are related to those of the extractor. Thus, if Ext is a strong extractor, then $\mathrm{Samp}(x)_y = y \circ \mathrm{Ext}(x, y)$ is an averaging sampler with distinct samples.

[16] To obtain this bound rather than $t = \mathrm{poly}(\log(1/\gamma), 1/\delta, \log n)$ as claimed in [Zuc97], the averaging samplers of [RVW00] should be used; these are also obtained by applying Zuckerman's transformation to appropriate extractors.

Previous constructions. The cryptosystem of Cachin and Maurer [CM97] 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 [CG89], and that it yields an extractor is the Leftover Hash Lemma [HILL99, BBR88].) Actually, in the description in [CM97], 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 [AR99, ADR02]. 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 [HILL99, BBR88]. Combining these via Theorem 6.3 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 [ADR02].

The basic construction of Lu [Lu] of a locally computable 1-bit extractor (equivalently, locally computable list-decodable error-correcting code) can be viewed as using an expander walk in place of the random-subset sampler in the above description of the ADR cryptosystem. To obtain many bits, Lu plugs this 1-bit extractor into extractor construction of Trevisan [Tre01, RRV02].

Other Samplers. As mentioned above, choosing a completely random t-subset is an excellent sampler (requiring only $t = O(m + \log(1/\varepsilon))$), except for its poor randomness complexity. In particular, it requires $r = \Theta(t \log n) = \Theta((m + \log(1/\varepsilon)) \cdot \log n)$ random bits in order to construct a locally computable extractor with output length m and error ε. However, the dependence on m can be easily eliminated using Lemma 8.3: let $t_0 = \Theta(\log(1/\varepsilon))$, view the bits of the source as a matrix $[n] = [t/t_0] \times [n_0]$, sample a random t_0-subset $S_0 \subset [n_0]$ of the columns, and let their union $S = [t/t_0] \times S_0$ be the final sample set. This sampling procedure uses only $O(\log n \cdot \log(1/\varepsilon))$ random bits, and is similar in spirit to the access pattern of the [DM] construction (though the latter directly XORs the bits read to obtain the output, whereas we need to apply an additional extractor). We also recall that our sampling method in Section 8.2 is constructed in the same way, except that we use an expander walk to choose the t_0-subset of $[n_0]$ (with some minor complications to ensure distinctness).

Other Extractors. The extractor of Zuckerman [Zuc97], while based on intuitive and appealing techniques, involves a sophisticated recursion. Here we mention some recent constructions of extractors that are more direct, while still achieving excellent parameters (though not as good for the case of constant min-entropy rate).

There are elegant extractor constructions in [TZS01] and [SU01] that have optimal seed length, but extract only a sublinear fraction of the min-entropy. Using either of these in a local extractor would contribute $O(\log m + \log(1/\varepsilon))$ to the seed length, while reading $t = m^{1+\Omega(1)} + \mathrm{polylog}(1/\varepsilon)$ bits from the source. Another direct extractor from [Tre01, RRV02] extracts a constant fraction of the min-entropy, but has seed length polynomial in the optimal. Using it in a local extractor would contribute $O((\log^2 m) \cdot \log(1/\varepsilon))$ to the seed length while reading only $t = O(m + \log(1/\varepsilon))$ bits from the source.

The benefits of these two constructions can be combined to obtain a better local extractor using the following technique (on which most of the early extractor constructions, including [NZ96], were based): run the sampler *twice*, first to sample a "large" block of length $O(m)$ and next to sample a "small" block of length $\mathrm{polylog}(m, 1/\varepsilon)$. As in [NZ96], the small block is likely to have constant min-entropy rate even conditioned on the large one. Using the extractor of [TZS01] or [SU01] mentioned above, we can use a very short seed of length $O(\mathrm{loglog}\ m + \log(1/\varepsilon))$ to extract $O((\log^2 m) \cdot \log(1/\varepsilon))$ bits from the small block, and then use these extracted bits as the seed for [Tre01, RRV02] to extract $\Omega(m)$ bits from the large block. Thus, this combined construction contributes only $O(\log m + \log(1/\varepsilon))$ to the seed length and requires reading only $t = O(m + \mathrm{polylog}(m, 1/\varepsilon))$ bits from the source.

9 Lower Bounds

In this section, we prove lower bounds on the efficiency of cryptosystems in the bounded storage model and of locally computable extractors, addressing both the number of bits read from the source and the key length.

9.1 Number of bits read

First, we show that the number of bits read from the source must be linear in m, and must grow when the difference between the min-entropy rate α and storage rate β goes to zero. For simplicity, the results in this section assume that αn and βn are integers. This assumption can be removed with an insignificant cost in the bounds by using the rounded values $\alpha' = \lceil \alpha n \rceil / n$ and $\beta' = \lfloor \beta n \rfloor / n$.

Theorem 9.1 *Suppose that* $\mathrm{PRG} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is an* ε*-secure BSM pseudorandom generator for storage rate* β *and min-entropy rate* α*, such that for every key* $K \in \{0,1\}^d$*,* $\mathrm{PRG}(\cdot, K)$ *reads at most* t *bits of its input. Then, setting* $\delta = \alpha - \beta$*, we have*

1. $t \geq (1 - \varepsilon - 2^{-m}) \cdot (1/\delta) \cdot m$*, and*
2. $t \geq \min\left\{ \frac{(1-\delta)n}{2}, \frac{\log((1-2^{-m})/\varepsilon)}{\log((1+\delta)/(1-\delta))} \right\}$*. In particular if* $\delta = 1 - \Omega(1)$*, then* $t = \Omega(\min\{n, \log(1/\varepsilon)/\delta\})$*.*

When δ is a constant, our constructions (in Corollaries 7.5 and 8.6) match the above lower bounds to within a constant factor. Moreover, in the natural case that $\omega(1) \leq m \leq o(n)$ and $o(1) \geq \varepsilon \geq 2^{-o(m)}$, the above lower bound says $t \geq (1 - o(1))m/\delta$, which is quite close to the upper bound of $t = (1 + \kappa)m/\delta$ achieved by our constructions.

Proof: For simplicity, we assume that $\mathrm{PRG}(\cdot, K)$ selects the t bits to read *nonadaptively*, and sketch how to extend the result to adaptive access at the end. In the proof, we use the equivalent formulation of security for BSM pseudorandom generators from Lemma 3.3. For a subset $T^{\mathrm{rnd}} \subset [n]$ of size $|T^{\mathrm{rnd}}| = \alpha n$, consider a source X that is uniform on the bits in T^{rnd} and 0 elsewhere. For a subset $T^{\mathrm{adv}} \subset T^{\mathrm{rnd}}$ of size $|T^{\mathrm{adv}}| = \beta n$, consider an adversary $A : \{0,1\}^n \to \{0,1\}^{\beta n}$ that records the bits of the source in T^{adv}. (We will describe how to choose T^{rnd} and T^{adv} later.) For any T^{adv} and T^{rnd}, this source has min-entropy rate α and the adversary has storage rate β. For a key $k \in \{0,1\}^d$, let $T_k^{\mathrm{key}} \subset [n]$ denote the set of positions from the source read by $\mathrm{PRG}(\cdot, k)$. Intuitively, PRG accesses only $|T_k^{\mathrm{key}} \cap (T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}})|$ bits from the source that are unknown to the adversary, so this should be an upper bound on m.

Formally, we consider the distribution $\mathrm{PRG}(X, K)$ conditioned on the key K and the adversary's state $A(X)$. Specifically, let $\varepsilon(k, s)$ be the statistical difference between $\mathrm{PRG}(X, K)|_{K=k, A(X)=s}$ and U_m. The expectation of $\varepsilon(K, A(X))$ is precisely the statistical difference between $(\mathrm{PRG}(X, K), A(X), K)$ and $(U_m, A(X), K)$, which is at most ε by Lemma 3.3. On the other hand, when the key k and state s are fixed, $\mathrm{PRG}(X, K)|_{K=k, A(X)=s}$ is a function of only the positions of X in $T_k^{\mathrm{key}} \cap (T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}})$. Hence it takes on at most $2^{|T_k^{\mathrm{key}} \cap (T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}})|}$ different values, and its statistical difference from uniform satisfies

$$\varepsilon(k, s) \geq 1 - \frac{2^{\min\left\{\left|T_k^{\mathrm{key}} \cap \left(T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}}\right)\right|, m\right\}}}{2^m}. \tag{5}$$

We will complete each part of the proof by choosing T^{adv} and T^{rnd} so that $|T_k^{\mathrm{key}} \cap \left(T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}}\right)|$ is small for many keys k if t is small.

For Part 1, we first deduce from Inequality (5) that

$$\varepsilon(k, s) \geq 1 - \frac{\left|T_k^{\mathrm{key}} \cap \left(T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}}\right)\right|}{m} - \frac{1}{2^m}.$$

Here we use the fact that $2^u/2^v \leq u/v$ for all integers $v \geq u \geq 1$, setting $v = m$ and $u = \min\{|T_k^{\mathrm{key}} \cap (T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}})|, m\}$. (The case $u = 0$ is handled by the $1/2^m$ term.) Taking the expectation of both sides over $k \overset{\mathrm{R}}{\leftarrow} K$ and $s \overset{\mathrm{R}}{\leftarrow} A(X)$, we get

$$\varepsilon \geq \mathop{\mathrm{E}}_{K, A(X)} [\varepsilon(K, A(X))] \geq 1 - \frac{\mathrm{E}_K\left[\left|T_K^{\mathrm{key}} \cap \left(T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}}\right)\right|\right]}{m} - \frac{1}{2^m} \tag{6}$$

Thus, we seek sets T^{rnd} and T^{adv} such that the expectation of $|T_K^{\mathrm{key}} \cap \left(T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}}\right)|$ is small if t is small. We obtain such sets via the Probabilistic Method. Consider a random set T of size δn. Then for every fixed key k, the expectation of $\left|T_k^{\mathrm{key}} \cap T\right|$ equals $\delta \cdot |T_k^{\mathrm{key}}| \leq \delta t$. By averaging, there exists a fixed set T^* such that $\mathrm{E}_K[|T_K^{\mathrm{key}} \cap T^*|] \leq \delta t$. Thus, if we take $T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}} = T^*$, set T^{adv} to be an arbitrary set of βn elements outside T^*, and substitute into Inequality (6), we conclude that

$$\varepsilon \geq 1 - \frac{\delta t}{m} - \frac{1}{2^m}.$$

Solving for t completes the proof of Part 1.

For Part 2, we begin by noting that Inequality (5) implies that if $T_k^{\mathrm{key}} \cap \left(T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}}\right) = \emptyset$, then $\varepsilon(k, s) \geq 1 - 2^{-m}$. Thus,

$$\varepsilon \geq \mathop{\mathrm{E}}_{K, A(X)} [\varepsilon(K, A(X))] \geq \Pr_K \left[T_K^{\mathrm{key}} \cap \left(T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}}\right) = \emptyset\right] \cdot (1 - 2^{-m}) \tag{7}$$

As before, we will use the Probabilistic Method to obtain sets T^{rnd} and T^{adv} such that there is a significant probability that $T^{\mathrm{rnd}} \setminus T^{\mathrm{adv}}$ is disjoint from T_K^{key}. Consider a random set T of size δn. For every fixed key k, since $|T_k^{\mathrm{key}}| \leq t$, we have

$$\Pr_T \left[T_k^{\mathrm{key}} \cap T = \emptyset\right] \geq \frac{\binom{(1-\delta)n}{t}}{\binom{n}{t}} \geq \left(\frac{(1-\delta)n - t}{n - t}\right)^t \geq \left(\frac{1-\delta}{1+\delta}\right)^t,$$

where the first inequality uses the fact that the probability that a random set of size δn intersects a fixed set of size t equals the probability that a random set of size t intersects a fixed set of size δn, and the last inequality assumes $t \le (1-\delta)n/2$ (otherwise, Part 2 holds and we are done). By averaging, there is a fixed T^* of size δn such that

$$\Pr_K \left[T_K^{\text{key}} \cap T^* = \emptyset \right] \ge \left(\frac{1-\delta}{1+\delta} \right)^t .$$

As before, we set $T^{\text{rnd}} \setminus T^{\text{adv}} = T^*$ and substitute into Inequality (7) to obtain

$$\varepsilon \ge \left(\frac{1-\delta}{1+\delta} \right)^t \cdot (1 - 2^{-m}),$$

as desired.

We now sketch how to generalize the results to the case that the locations read by PRG may be chosen adaptively. In this case, the set T_k^{key} is no longer well defined. Thus, instead of studying the quantity $|T_K^{\text{key}} \cap (T^{\text{rnd}} \setminus T^{\text{adv}})|$, we consider the logarithm of the size of the support of the random variable $\text{PRG}(X, K)$ conditioned on K, T^{adv}, T^{rnd}, $X|_{T^{\text{adv}}}$ (so that the randomness is only over $X|_{T^{\text{rnd}} \setminus T^{\text{adv}}}$). For Part 1, we consider the expectation of this log-support-size (over K, T^{adv}, T^{rnd}, and $X|_{T^{\text{adv}}}$, where T^{adv} and T^{rnd} are chosen by the Probabilistic Method as above). It can be shown (by induction on t) that if PRG makes t adaptive queries to X, then the expected log-support-size is at most δt. Then the proof proceeds similarly to the above, using the fact that a distribution of support size at most 2^k has statistical difference at least $1 - 2^{\min\{k,m\}}/2^m$ from U_m. For Part 2, we note that the probability that the log-support-size is 0 at least the probability that $\text{PRG}(X, K)$ doesn't query any points in $T^{\text{rnd}} \setminus T^{\text{adv}}$, and argue that this probability is the same as in the nonadaptive case. ■

Setting $\beta = 0$ in the above theorem, we obtain the same bounds for locally computable extractors.

Corollary 9.2 *Suppose that* $\text{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a t-local strong $(\delta n, \varepsilon)$-extractor. Then*

1. $t \ge (1 - \varepsilon - 2^{-m}) \cdot (1/\delta) \cdot m$, *and*
2. $t \ge \min \left\{ \frac{(1-\delta)n}{2}, \frac{\log((1-2^{-m})/\varepsilon)}{\log((1+\delta)/(1-\delta))} \right\}$. *In particular if $\delta = 1 - \Omega(1)$, then $t = \Omega(\min\{n, \log(1/\varepsilon)/\delta\})$.*

We recall that Bar-Yossef et al. [BRST02] have previously shown lower (and upper) bounds for "on-line" extractors. Specifically, they have shown to evaluate extractors with one pass through the input, space approximately m is both necessary and sufficient. Since the small-space requirement is weaker than being locally computable, their lower bound also implies that t is at least (roughly) m.[17] Part 1 of Corollary 9.2 provides a stronger lower bound of roughly m/δ.

[17] 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.

9.2 Lower Bounds on Key Length

Radhakrishnan and Ta-Shma [RT00] have already given tight lower bounds for the seed length of extractors:

Theorem 9.3 ([RT00]) *There is a constant c such that if* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a $(\delta n, \varepsilon)$-extractor, where $\delta \le 1 - c/n$ and $\varepsilon < 1/2$, then $d \ge \log((1-\delta)n) + 2\log(1/\varepsilon) - O(1)$.*

This already implies that the seed lengths of our locally computable extractors in Theorems 7.4 and 8.5 are optimal to within constant factors (when $\delta \in (0,1)$ is a constant). However, it does not imply optimality for the key length of our BSM pseudorandom generators, since these are potentially weaker objects (particularly for the case of min-entropy rate $\alpha = 1$). Nevertheless we show that the same lower bound holds.

Theorem 9.4 *There is a constant c such that the following holds. Suppose that* $\mathrm{PRG} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is an ε-secure BSM pseudorandom generator for min-entropy rate α and storage rate β, with $\delta \overset{\mathrm{def}}{=} \alpha - \beta \le 1 - c/n$ and $\varepsilon < 1/2$. Then,*

$$d \ge \log((1-\delta)n) + 2\log(1/\varepsilon) - O(1).$$

Proof: Throughout the proof, we use the equivalent formulation of BSM pseudorandom generators from Lemma 3.3. Since the output length m is unconstrained in the statement of the lemma, we must prove the lower bound even for the case of extracting only 1 bit, and hence we set $m = 1$ for the rest of the proof.

A lower bound of $d \ge \log((1-\alpha)n) + 2\log(1/\varepsilon) - O(1)$ already follows from Theorem 9.3, since a BSM pseudorandom generator for min-entropy rate α is also an $(\alpha n, \varepsilon)$-extractor. Thus it suffices to prove $d \ge \log(\beta n) + 2\log(1/\varepsilon) - O(1)$ (since $\max\{\log((1-\alpha)n), \log(\beta n)\} \ge \log(((1-\alpha)n + \beta n)/2) = \log((1-\delta)n) - 1$). Since we must prove this even when $\alpha = 1$, from now on we will only consider the source $X = U_n$. We also

As a warm-up, we begin by sketching a weaker lower bound of $d > \log(\beta n)$, based on the lower bound techniques of [CG88, NZ96]. Suppose that $d \le \log(\beta n)$. This means that there are at most $2^d \le \beta n$ different keys. Thus we can define an adversary $A : \{0,1\}^n \to \{0,1\}^{\beta n}$ that on input x, records the bit $\mathrm{PRG}(x,k)$ for all keys $k \in \{0,1\}^d$. (Recall that $m = 1$.) Then $(\mathrm{PRG}(X,K), A(X), K)$ can be easily distinguished from $(U_1, A(X), K)$: in the former distribution, the first component can be predicted perfectly from the second two components, whereas in the second it can be predicted only with probability 1/2. Hence the two random variables have statistical difference at least 1/2. Similarly, to show that the statistical difference is at least ε, we only need A to store the bit $\mathrm{PRG}(x,k)$ for a 2ε fraction of the keys k. With this observation, we obtain a lower bound of $d \ge \log(\beta n) + \log(1/\varepsilon) - O(1)$.

To get a lower bound with $2\log(1/\varepsilon)$ rather than $\log(1/\varepsilon)$ requires a more delicate argument, based on the ideas of [RT00]. Suppose that $2^d \le (\beta n)/c\varepsilon^2$ (where c is a constant to be set later in the proof. Then we can partition the keys into βn sets $\mathcal{K}_1, \ldots, \mathcal{K}_{\beta n}$ of size at most $1/c\varepsilon^2$ each. Fix a function $P : \{0,1\}^d \to \{0,1\}$ (we will describe how to pick P later). Then our adversary $A : \{0,1\}^n \to \{0,1\}^{\beta n}$, will do the following on input x: for each $i = 1, \ldots, \beta n$, A will store a bit $b_i = \mathrm{maj}_{k \in \mathcal{K}_i}[\mathrm{PRG}(x,k) \oplus P(k)]$, i.e. b_i indicates whether or not $\mathrm{PRG}(x,\cdot)$ (dis)agrees with P on a majority of keys in $\mathcal{K}_i$.

To lower-bound the statistical difference between $(\mathrm{PRG}(X,K),A(X),K)$ and $(U_1,A(X),K)$, we give a method for predicting $\mathrm{PRG}(X,K)$ from $(A(X),K)$ that succeeds with probability significantly greater than 1/2. Given $(A(x),k)$, we let i be such that $k\in\mathcal{K}_i$, read the i'th bit b_i of $A(x)$, and output the prediction $b_i\oplus P(k)$.

This predicts successfully iff $b_i\oplus P(k)=\mathrm{PRG}(x,k)$, i.e.

$$\mathrm{PRG}(x,k)\oplus P(k)=\mathrm{maj}_{k'\in\mathcal{K}_i}\mathrm{PRG}(x,k')\oplus P(k').$$

Thus, the prediction probability over random X and K is precisely $\mathrm{E}_{X,I}\left[\mathrm{MajProb}(X,I)\right]$, where $I\overset{\mathrm{R}}{\leftarrow}\{1,\ldots,\beta n\}$ and MajProb is the "majority probability" defined as follows.

$$\begin{aligned}\mathrm{MajProb}(x,i) &\overset{\mathrm{def}}{=} \Pr_{K\overset{\mathrm{R}}{\leftarrow}K_i}\left[\mathrm{PRG}(x,K)\oplus P(K)=\mathrm{maj}_{k'\in\mathcal{K}_i}\mathrm{PRG}(x,k')\oplus P(k')\right]\\ &= \frac{1}{2}+\left|\frac{\#\{k\in\mathcal{K}_i:\mathrm{PRG}(x,k)=P(k)\}}{|\mathcal{K}_i|}-\frac{1}{2}\right|\end{aligned}$$

Thus our aim is to choose P such that $\#\{k\in\mathcal{K}_i:\mathrm{PRG}(x,k)=P(k)\}/|\mathcal{K}_i|$ deviates from 1/2 by more than ε, on average. We do this by the Probabilistic Method. Consider a random function $P:\{0,1\}^d\to\{0,1\}$. Then for each x and i, the quantity $\#\{k\in\mathcal{K}_i:\mathrm{PRG}(x,k)=P(k)\}$ is the sum of $|\mathcal{K}_i|\le 1/c\varepsilon^2$ independent, unbiased $\{0,1\}$ random variables. By standard bounds on the binomial distribution, the average deviation of this sum from its expectation ($|\mathcal{K}_i|/2$) is $\Omega(\sqrt{|\mathcal{K}_i|})>\varepsilon\cdot|\mathcal{K}_i|$ (when c is a sufficiently large constant). That is $\mathrm{E}_P[\mathrm{MajProb}(x,i)]>1/2+\varepsilon$. By averaging, there exists a fixed P such that $\mathrm{E}_{X,I}[\mathrm{MajProb}(X,I)]>1/2+\varepsilon$. This contradicts the hypothesis that $(\mathrm{PRG}(X,K),A(X),K)$ and $(U_1,A(X),K)$ are ε-close. ■

Acknowledgments

Noga Alon, Omer Reingold, and Amnon Ta-Shma also thought of similar approaches to this problem, and I am grateful to them for sharing their ideas with me. 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

[ASE92] Noga Alon, Joel H. Spencer, and Paul Erdös. *The Probabilistic Method.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., 1992.

[ADR02] Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, June 2002.

[AR99] Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 65–79. Springer-Verlag, 1999, 15–19 August 1999.

[BRST02] Ziv Bar-Yossef, Omer Reingold, Ronen Shaltiel, and Luca Trevisan. Streaming computation of combinatorial objects. In *Proceedings of the Seventeenth Annual IEEE Conference on Computational Complexity*, pages 165–174, Montreal, Canada, 21–24 May 2002. IEEE Computer Society Press.

[BGG93] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993.

[BR94] Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science*, pages 276–287, Santa Fe, New Mexico, 20–22 November 1994. IEEE.

[BY03] Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In M. Joye, editor, *Topics in Cryptology - CT-RSA 03*, volume 2612 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[BBR88] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988. Special issue on cryptography.

[CM97] Christian Cachin and Ueli Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer-Verlag, 1997.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53(1):17–25, 13 January 1995.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, April 1988.

[CG89] Benny Chor and Oded Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5(1):96–106, 1989.

[DR02] Yan Zong Ding and Michael O. Rabin. Hyper-encryption and everlasting security (extended abstract). In *STACS 2002 — 19th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, pages 1–26. Springer-Verlag, 14–16 March 2002.

[DS01] Devdatt Dubhashi and Sandeep Sen. Concentration of measure for randomized algorithms: Techniques and analysis. In S. Rajasekaran et al., editor, *Handbook of Randomized Computing, Vol. I.*, chapter 3, pages 35–100. Kluwer, 2001.

[DM] Stefan Dziembowski and Ueli Maurer. Optimal randomizer efficiency in the bounded-storage model. This issue.

[GG81] Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22(3):407–420, June 1981.

[Gil98] David Gillman. A Chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220 (electronic), 1998.

[Gol97] Oded Goldreich. A sample of samplers: A computational perspective on sampling. Technical Report TR97-020, Electronic Colloquium on Computational Complexity, May 1997.

[GW97] Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures & Algorithms*, 11(4):315–343, 1997.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396 (electronic), 1999.

[Lu] Chi-Jen Lu. Encryption against storage-bounded adversaries from on-line strong extractors. This issue.

[LRVW03] Chi-Jen Lu, Omer Reingold, Salil Vadhan, and Avi Wigderson. Extractors: Optimal up to constant factors. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 602–611, San Diego, CA, June 2003.

[Mar73] G. A. Margulis. Explicit constructions of expanders. *Problemy Peredači Informacii*, 9(4):71–80, 1973.

[Mau92] Ueli Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[Mau93] Ueli M. Maurer. Secret key agreement by public discussion from common information. *IEEE Trans. Inform. Theory*, 39(3):733–742, 1993.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NT99] Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. *Journal of Computer and System Sciences*, 58(1):148–173, 1999.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, February 1996.

[Rab01] Michael O. Rabin. Personal communication, December 2001.

[RT00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discrete Math.*, 13(1):2–24 (electronic), 2000.

[RRV02] Ran Raz, Omer Reingold, and Salil Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *Journal of Computer and System Sciences*, 65(1):97–128, 2002.

[RSW00] Omer Reingold, Ronen Shaltiel, and Avi Wigderson. Extracting randomness via repeated condensing. In *41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, California, 12–14 November 2000.

[RVW00] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings of 41st Annual Symposium on Foundations of Computer Science*, pages 3–13, 2000.

[SSZ98] Michael Saks, Aravind Srinivasan, and Shiyu Zhou. Explicit OR-dispersers with polylogarithmic degree. *Journal of the ACM*, 45(1):123–154, January 1998.

[Sha02] Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the European Association for Theoretical Computer Science*, 77:67–95, June 2002.

[SU01] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *42nd Annual Symposium on Foundations of Computer Science*, pages 648–657. IEEE, 14–17 October 2001.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[Ta-02] Amnon Ta-Shma. Almost optimal dispersers. *Combinatorica*, 22(1):123–145, 2002.

[TZS01] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. Extractors from Reed–Muller codes. In *42nd Annual Symposium on Foundations of Computer Science*, pages 638–647. IEEE, 14–17 October 2001.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, July 2001.

[Vad02] Salil P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. Cryptology ePrint Archive, Report 2002/162, 2002. `http://eprint.iacr.org/`.

[Vad03] Salil P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In Dan Boneh, editor, *Advances in Cryptology—CRYPTO '03*, Lecture Notes in Computer Science. Springer-Verlag, 2003, 17–21 August 2003.

[Zuc96] David Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4/5):367–391, October/November 1996.

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997.

A A Chernoff Bound

Here we prove the Chernoff-type bound needed in the proof of Lemma 6.2:

Lemma A.1 *Let $\tau \in [0,1]$. Suppose that $\Delta_1, \dots, \Delta_n$ are nonnegative random variables such that for every $q_1, q_2, \dots, q_{i-1} \geq 0$ and $q > 0$,*

$$\Pr\left[\Delta_i \geq q | \Delta_1 = q_1, \dots, \Delta_{i-1} = q_{i-1}\right] \leq \tau \cdot 2^{-q}. \tag{8}$$

Then $\Pr\left[\sum_i \Delta_i > 2\tau n\right] < 2^{-\Omega(\tau n)}$.

Unlike the most basic Chernoff-Hoeffding bounds, this lemma does not require that the random variables Δ_i are bounded or independent. Instead, it requires that the tail of Δ_i is exponentially vanishing even conditioned on the previous Δ_j's. We begin by proving the lemma for *independent* random variables whose distribution function is exactly (rather than merely bounded by) an exponential function. Here and below, by a *distribution function* of a real-valued random variable X we mean the function $F(q) = \Pr\left[X \geq q\right]$, which is more convenient for us than the standard definition (which sets $F(q) = \Pr\left[X \leq q\right]$).

Lemma A.2 *Let $\tau \in [0,1]$. Suppose $Z_1, \dots, Z_n$ are independent, nonnegative random variables with probability distribution function given by $\Pr\left[Z_i \geq q\right] = \tau \cdot 2^{-q}$ for all $q > 0$ (and thus $\Pr\left[Z_i = 0\right] = 1 - \lim_{q \to 0} \tau \cdot 2^{-q} = 1 - \tau$). Then* $\Pr\left[\sum_i Z_i > 2\tau n\right] < 2^{-\Omega(\tau n)}$.

Proof: Let $f(q) = \tau \cdot 2^{-q}$ be the distribution function of the Z_i's (for $q > 0$). As usual in the proofs of Chernoff bounds, we consider the expectation of an exponential generating function $2^{tZ} = \prod_{i=1}^{n} 2^{tZ_i}$. For every $t \in (0,1)$, we have:

$$\begin{aligned} \mathrm{E}\left[2^{tZ_i}\right] &= \Pr\left[Z_i = 0\right] \cdot 2^0 + \int_{q=0}^{\infty} \left(-\frac{d}{dq} f(q)\right) \cdot 2^{tq} dq \\ &= (1-\tau) + \int_{q=0}^{\infty} (\tau \ln 2 \cdot 2^{-q}) \cdot 2^{tq}) dq \\ &= (1-\tau) + \left[\frac{-\tau}{1-t} \cdot 2^{-(1-t)q}\right]_{q=0}^{\infty} \\ &= 1 + \frac{\tau t}{1-t}. \\ &\leq e^{\tau t/(1-t)} \end{aligned}$$

Thus,

$$\mathrm{E}[2^{tZ}] = \prod_i \mathrm{E}[2^{tZ_i}] \leq \left(e^{\tau t/(1-t)}\right)^n.$$

By Markov's Inequality,

$$\Pr\left[Z \geq 2\tau n\right] = \Pr\left[2^{tZ} \geq 2^{2\tau t n}\right] \leq \frac{e^{\tau t n/(1-t)}}{2^{2\tau t n}} = \left(\frac{e^{t/(1-t)}}{2^{2t}}\right)^{\tau n},$$

for every $t \in (0,1)$. For $t = 1/4$, $e^{t/(1-t)} < 2^{2t}$, so this probability is indeed $2^{-\Omega(\tau n)}$, as desired. ■

Now we deduce the general case of dependent random variables, using a standard coupling argument.

Lemma A.3 *If A is a real-valued random variable and $F : \mathbb{R} \to [0,1]$ is a distribution function such that for every q,* $\Pr[A \geq q] \leq F(q)$*, then there is a probabilistic function g such that* $\Pr[g(a) \geq a] = 1$ *for all a and $g(A)$ has distribution F.*

Proof Sketch: We define $f(a)$ as follows: let $\alpha = \Pr[A > a]$ and $\beta = \Pr[A \geq a] = \alpha + \Pr[A = a]$. Choose $u = u(a)$ uniformly from the interval $[\alpha, \beta]$. Let $g(a) = \max\{b : F(b) \geq u\}$.

To see that $g(A)$ is distributed identically to B, note that the definition of u is the standard transformation mapping a random variable A to the uniform distribution on $[0,1]$ and that the $\max\{b : F(b) \geq u\}$ is the standard transformation mapping a uniform random variable u to a random variable with a specified distribution function F.

Thus, we only need to argue that $g(a) \geq a$ with probability 1. We have $\gamma \leq \beta = \Pr[A \geq a] \leq g(a)$. Since $g(a)$ is defined to be the largest b such that $F(b) \geq \gamma$, we have $g(a) \geq a$ as desired. □

Proof of Lemma A.1: Given random variables $\Delta_1, \ldots, \Delta_n$ satisfying the hypothesis of Lemma A.1, we will use Lemma A.3 to extend the probability space to include *independent* random variables $Z_1, \ldots, Z_n$ satisfying the hypothesis of Lemma A.2 such that $\Pr[\Delta_i \leq Z_i] = 1$. This will imply $\Pr[\sum_i \Delta_i > 2\tau n] \leq \Pr[\sum_i Z_i > 2\tau n] < 2^{-\Omega(\tau n)}$, as desired.

Suppose $\Delta_1 = q_1, \ldots, \Delta_n = q_n$. We now describe how to generate the Z_i's, conditioned on these values for the Δ_i's. For each prefix $\overline{q} = (q_1, \ldots, q_{i-1})$, we let $g_{\overline{q}}$ be the probabilistic function given by Lemma A.3 that transforms $\Delta_i|_{\Delta_1 = q_1, \ldots, \Delta_{i-1} = q_{i-1}}$ to a random variable with distribution function $F(q) = \tau \cdot 2^{-q}$. We then define $Z_i = g_{\overline{q}}(q_i)$.

Lemma A.3 guarantees that $\Pr[Z_i \geq \Delta_i] = 1$, and that

$$\Pr[Z_i \geq q | \Delta_1 = q_1, \ldots, \Delta_{i-1} = q_{i-1}] = \tau \cdot 2^{-q}.$$

Since this holds for every $(q_1, \ldots, q_{i-1})$, we conclude that Z_i is independent of $(\Delta_1, \ldots, \Delta_{i-1})$ and hence also of $(Z_1, \ldots, Z_{i-1})$, as desired. ■