Space Pseudorandom Generators by Communication Complexity Lower Bounds*

Anat Ganor[1] and Ran Raz[1,2]

1 **Weizmann Institute of Science, Rehovot, Israel**
{anat.ganor,ran.raz@weizmann.ac.il}@weizmann.ac.il
2 **Institute for Advanced Study, Princeton, New Jersey**

Abstract

In 1989, Babai, Nisan and Szegedy [2] gave a construction of a pseudorandom generator for logspace, based on lower bounds for multiparty communication complexity. The seed length of their pseudorandom generator was $2^{\Theta(\sqrt{\log n})}$, because the best lower bounds for multiparty communication complexity are relatively weak. Subsequently, pseudorandom generators for logspace with seed length $O(\log^2 n)$ were given by [19] and [15].

In this paper, we show how to use the pseudorandom generator construction of [2] to obtain a third construction of a pseudorandom generator with seed length $O(\log^2 n)$, achieving the same parameters as [19] and [15]. We achieve this by concentrating on protocols in a restricted model of multiparty communication complexity that we call the *conservative one-way unicast model* and is based on the conservative one-way model of [8]. We observe that bounds in the conservative one-way unicast model (rather than the standard Number On the Forehead model) are sufficient for the pseudorandom generator construction of [2] to work.

Roughly speaking, in a conservative one-way unicast communication protocol, the players speak in turns, one after the other in a fixed order, and every message is visible only to the next player. Moreover, before the beginning of the protocol, each player only knows the inputs of the players that speak after she does and a certain function of the inputs of the players that speak before she does. We prove a lower bound for the communication complexity of conservative one-way unicast communication protocols that compute a family of functions obtained by compositions of strong extractors. Our final pseudorandom generator construction is related to, but different from the constructions of [19] and [15].

1998 ACM Subject Classification F.1.0 Computation by Abstract Devices – General

Keywords and phrases Communication complexity, Logspace, Pseudorandom generator

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2014.692

1 Introduction

Derandomizing space bounded computations has attracted a lot of attention over the last few decades. The most important problem is to simulate randomized logspace machines with deterministic ones. Savitch [26] result on nondeterministic machines implies that $RL \subseteq L^2$. Subsequently, this problem was studied, for example, by [1], [2], [19], [15] and [22]. Currently, the best derandomization of general logspace machines is due to Saks and Zhou [25], proving that $BPL \subseteq L^{3/2}$.

* This work was supported by an ISF grant, by the I-CORE Program of the Planning and Budgeting Committee and by NSF grant numbers CCF-0832797, DMS-0835373.

17th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'14) / 18th Int'l Workshop on Randomization and Computation (RANDOM'14).
Editors: Klaus Jansen, José Rolim, Nikhil Devanur, and Cristopher Moore; pp. 692–703

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

One way to simulate a randomized, space bounded computation with a deterministic one is using a space pseudorandom generator. Roughly speaking, a space pseudorandom generator converts, efficiently, a short truly random seed into a long string that looks random to machines with limited space. A major open problem in the theory of pseudorandomness is to construct an explicit pseudorandom generator that stretches a seed of length $O(\log n)$ to n bits that cannot be distinguished from uniform by any logspace machine with input length n. Such a generator would imply that $RL = L$. Nisan [19] constructed space pseudorandom generators that convert $O(\log^2 n)$ random bits to $poly(n)$ bits that look random to any logspace machine. Subsequently, [15] showed a different construction with the same parameters. Since [19] and [15], no better seed length was obtained for derandomizing general logspace machines. There were other constructions of space pseudorandom generators for more restricted classes of space bounded computations, such as [23], [5], [6], [18], [14], [3] and [24].

In this paper, we give a new construction of a space pseudorandom generator for general logspace machines, with seed length $O(\log^2 n)$, achieving the same parameters as [19] and [15]. Our pseudorandom generator construction is based on a lower bound for a certain model of multiparty communication complexity, relying on the pseudorandom generator construction of Babai, Nisan and Szegedy [2]. The pseudorandom generator of [2] has seed length $2^{\Theta(\sqrt{\log n})}$. The proof that their construction gives a pseudorandom generator relies on a lower bound for multiparty communication complexity. [2] gave a lower bound for the multiparty communication complexity of protocols in the Number On the Forehead (NOF) model with blackboard communication. In this model, each player knows all inputs except her own input and the communication is done by writing messages on a blackboard (broadcast) so that every player sees all the previous communication. For this model, [2] gave a lower bound of $\Omega(\frac{n}{2^k})$ (where n is the length of each input and k is the number of players). Improving this lower bound is a major open problem.

We observe that the pseudorandom generator construction of [2] can be based on lower bounds for a restricted model of multiparty communication complexity. For this model we are able to obtain improved lower bounds, resulting in a pseudorandom generator with seed length $O(\log^2 n)$.

▶ **Definition 1** (Conservative One-way Unicast Communication Protocol)**.** Let P be a deterministic, multiparty communication protocol for k players $p_1, \ldots, p_k$. For a function $f : \mathcal{B} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_k \to \mathcal{B}$, we say that P is a *conservative one-way unicast communication protocol with respect to* f if for an input $b, a_1, \ldots, a_k \in \mathcal{B} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_k$ the following holds:

1. For every $i \in [k]$, before the beginning of the protocol, the i^{th} player only knows $a_{i+1}, \ldots, a_k$ and the (truth table[1] of the) function $f_i : \mathcal{A}_i \times \cdots \times \mathcal{A}_k \to \mathcal{B}$, defined by:

$$f_i(z_i, \ldots, z_k) = f(b, a_1, \ldots, a_{i-1}, z_i, \ldots, z_k)$$

   for every $z_i, \ldots, z_k \in \mathcal{A}_i \times \cdots \times \mathcal{A}_k$.
2. The players communicate one after the other in the fixed order $p_1, p_2, \ldots, p_k$.
3. For every $1 \leq i < k$, the i^{th} message is visible only to p_{i+1}. The message of the last player is the output of the protocol.

Usually, we will take f to be the function that the players are trying to compute. Note that the i^{th} player doesn't know $b, a_1, \ldots, a_{i-1}$ as in the NOF model, but she does know the relevant in-

[1] The truth table is not counted as part of the length of the input.

formation on $b, a_1, \ldots, a_{i-1}$ that is needed to compute the function $f(b, a_1, \ldots, a_{i-1}, z_i, \ldots, z_k)$ for every $z_i, \ldots, z_k \in \mathcal{A}_i \times \cdots \times \mathcal{A}_k$.

Our definition of conservative one-way unicast communication protocols is based on definitions by Damm, Jukna and Sgall [8]. [8] defined conservative communication protocols as protocols satisfying item (1) in Definition 1, and conservative one-way communication protocols as protocols satisfying items (1),(2) in Definition 1, where the communication is done by writing messages on a blackboard (broadcast) so that every player sees all the previous communication. The motivation of [8] to study the conservative one-way model was different than ours. They studied this model as an interesting communication model in its own right, without relating it to pseudorandom generators for logspace computations.

[8] proved lower bounds for the communication complexity of conservative one-way (blackboard) communication protocols that compute the pointer jumping problem. For $k = O((n/\log n)^{1/3})$, [8] proved a lower bound of $\Omega(n/k^2)$, and for $k \leq \log^* n - \omega(1)$, they proved a lower bound of $n \log^{(k-1)} n(1 - o(1))$ (where k is the number of players and n is the length of each input). The conservative one-way model was further studied by Chakrabarti in [7], where the $\Omega(n/k^2)$ lower bound due to [8] was extended so that it applies for all k.

The unicast setting, where the players communicate by sending messages to each other over private channels, was studied in the context of message-passing models of multiparty communication. These models have been used extensively in distributed computing, for example in [12], [16], [17], and [4]. Message passing models are also used to study privacy and security in multiparty computations.

For conservative communication protocols (satisfying item (1) in Definition 1) it is convenient to consider composed functions as we define next.

▶ **Definition 2** (Composed Functions)**.** For a function $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^m$ and $1 < i \in \mathbb{N}$, the i^{th} *composition of* f is a function $f^{(i)} : \{0,1\}^{m+in} \to \{0,1\}^m$ defined for every $a_0 \in \{0,1\}^m$, $a_1, \ldots, a_i \in \{0,1\}^n$ as

$$f^{(i)}(a_0, a_1, \ldots, a_i) = f(f^{(i-1)}(a_0, a_1, \ldots, a_{i-1}), a_i)$$

where $f^{(1)}(a_0, a_1) = f(a_0, a_1)$. In addition, we define $f^{(0)}(a_0) = a_0$.

Let $f^{(k)}$ be the k^{th} composition of a function $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^m$. Note that for every input $(a_0, a_1, \ldots, a_k) \in \{0,1\}^{m+kn}$ and every $i \in [k]$, if the i^{th} player knows $f^{(i-1)}(a_0, a_1, \ldots, a_{i-1})$, then she also knows the function $f_i^{(k)}$ defined in item (1) in Definition 1. Therefore, for the sake of proving lower bounds for the communication complexity of conservative communication protocols with respect to $f^{(k)}$, it is enough to assume that for an input $(a_0, a_1, \ldots, a_k) \in \{0,1\}^{m+kn}$, for every $i \in [k]$, before the beginning of the protocol, the i^{th} player only knows $a_{i+1}, \ldots, a_k$ and $f^{(i-1)}(a_0, a_1, \ldots, a_{i-1})$.

In our paper, we prove lower bounds for the communication complexity of conservative one-way unicast communication protocols with respect to a certain composed function $f^{(k)}$. Therefore, we replace item (1) in Definition 1 by the assumption that for an input $(a_0, a_1, \ldots, a_k) \in \{0,1\}^{m+kn}$, for every $i \in [k]$, before the beginning of the protocol, the i^{th} player only knows $a_{i+1}, \ldots, a_k$ and $f^{(i-1)}(a_0, a_1, \ldots, a_{i-1})$.

An Example – The Pointer Jumping Problem

In the pointer jumping problem for k players, the input is k functions $\Pi_1, \ldots, \Pi_k : [r] \to [r]$ and an additional input $i_0 \in [r]$. The players need to output $\Pi_k \circ \cdots \circ \Pi_1(i_0)$. Let $\mathcal{S}_r$ denote the set of all functions from $[r]$ to $[r]$ and let $f : [r] \times \mathcal{S}_r \to [r]$ be the function defined by

$f(i, \Pi) = \Pi(i)$ for every $i \in [r]$ and $\Pi \in S_r$. Note that the pointer jumping problem for k players is the k^{th} composition of f. In a conservative communication protocol (satisfying item (1) in Definition 1) for the pointer jumping problem, for every $i \in [k]$, before the beginning of the protocol, the i^{th} player only knows $\Pi_{i+1}, \ldots, \Pi_k$ and $f^{i-1}(i_0, \Pi_1, \ldots, \Pi_{i-1}) = \Pi_{i-1} \circ \cdots \circ \Pi_1(i_0)$.[2]

1.1 Main Result

We say that a communication protocol P computes a function $f : \{0,1\}^n \to \{0,1\}^m$ with bias $\delta > 0$ if

$$\Pr_{x \in_R \{0,1\}^n}[f(x) = P(x)] \geq 2^{-m} + \delta$$

We denote the length of the longest message sent during the execution of P by $L(P)$ (on the worst case input, not including the last message which is the output of the protocol).

Let $Ext : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^m$ be a (t, ε) strong extractor (see Definition 10). We refer to the k^{th} composition of Ext, denoted $Ext^{(k)}$, as a (t, ε) *composed strong extractor*. Composed strong extractors are closely related to alternating extractors, which are used in [10], with cryptographic applications.

Our lower bound is for the length of the longest message communicated during any conservative one-way unicast communication protocol that computes a composed strong extractor with bias $\delta > 0$.

▶ **Theorem 3.** *Let $Ext^{(k)} : \{0,1\}^{m+nk} \to \{0,1\}^m$ be a (t, ε) composed strong extractor and let P be a conservative one-way unicast communication protocol with respect to $Ext^{(k)}$ that computes $Ext^{(k)}$ with bias $\delta > 0$, such that $\varepsilon < \delta \cdot 2^{-(k+2)}$. Then,*

$$L(P) \geq n - t - k - \log \frac{1}{\delta} - 2$$

In fact, we prove a slightly stronger version of Theorem 3 in which we consider projections of the composed strong extractor (see Theorem 17). Using this lower bound together with the pseudorandom generator construction of Babai, Nisan and Szegedy [2], we obtain a space pseudorandom generator that converts $O(\log^2 n)$ random bits to $poly(n)$ bits that look random to any logspace machine (see Section 4).

Comparison with [19] and [15]

The pseudorandom generator construction of [15] is also based on a recursive composition of extractors. However, their generator is different from the one presented here. The recursive composition used in [15] is different from the composition in Definition 2. Moreover, [15] use extractors that output $O(\log^2 n)$ bits, whereas here we use extractors that output $O(\log n)$ bits.

The pseudorandom generator construction of [19] is based on a recursive composition of hash functions. This is done by a composition similar to the one in Definition 2. We note that hashing can be viewed as an application of an extractor. However, when viewing the hashing as an application of an extractor, the composition of [19] does not fit our definition of a composed extractor. In particular, in our definition of a composed extractor, the recursion

[2] In this case, knowing $\Pi_{i-1} \circ \cdots \circ \Pi_1(i_0)$ is equivalent to knowing the function f_i defined in item (1) in Definition 1.

is done by replacing the seed of the extractor with the output of the extractor from the previous composition, whereas in [19], the recursion is done by replacing the source of the extractor with the output of the extractor from the previous composition.

2 Preliminaries

2.1 General Notation

Let $[n]$ be the set of numbers $\{1, 2, \ldots, n\}$. For a binary string $x \in \{0,1\}^*$ and an index $i \in \mathbb{N}$, let x_i be the i^{th} bit of x. For a set of indexes $S = \{i_1, \ldots, i_k\} \subseteq [|x|]$, let x_S be the string $x_{i_1}, \ldots, x_{i_k}$.

2.1.1 Functions

For a function $f : \{0,1\}^n \to \{0,1\}^m$ and a subset $S \subseteq [m]$ of size m', where $m' \leq m$, the projection of f on S, denoted f_S, is a function from $\{0,1\}^n$ to $\{0,1\}^{m'}$ defined as $f_S(x) = (f(x))_S$ for every $x \in \{0,1\}^n$. To simplify notation, for $i \in [m]$, we define $f_i = f_{\{i\}}$. For two functions $f : \mathcal{A} \to \mathcal{B}$ and $h : \mathcal{B} \to \mathcal{C}$, let $h \circ f$ be the function from $\mathcal{A}$ to $\mathcal{C}$ defined as $h(f(a))$ for every $a \in \mathcal{A}$.

2.1.2 Distributions and Random Variables

We write $x \in_R \mathcal{X}$ if x is chosen uniformly at random from $\mathcal{X}$. For a distribution D and a subset S of the support of D, let $D(S)$ be the sum $\sum_{s \in S} D(s)$. For a random variable X and an event E, we write $X|E$ to denote X conditioned on E. We write $X \in \mathcal{X}$ if X is distributed over the set $\mathcal{X}$. For two random variables X and Y, we write $X \sim Y$ if X and Y have the same distribution. Slightly abusing notation, given a random variable X, we let $x \sim X$ indicate the sampling of x from the distribution of X.

2.2 Statistical Distance

▶ **Definition 4** (Statistical Distance)**.** Let D_1 and D_2 be two distributions over the same space Ω. Their *statistical distance* is

$$\|D_1 - D_2\| = \max_{S \subseteq \Omega} |D_1(S) - D_2(S)| = \frac{1}{2} \sum_{x \in \Omega} |D_1(x) - D_2(x)|$$

For two random variables $X_1, X_2 \in \Omega$ distributed according to D_1 and D_2 respectively, we define $\|X_1 - X_2\| = \|D_1 - D_2\|$.

▶ **Proposition 5.** *Let $X, X' \in \mathcal{X}$ be two random variables and let $f : \mathcal{X} \to \mathcal{Y}$ be any deterministic function. Then,*

$$\|f(X) - f(X')\| \leq \|X - X'\|$$

▶ **Proposition 6.** *Let $X \in \mathcal{X}$, $Y \in \mathcal{Y}$ and $Z \in \mathcal{Z}$ be three random variables, and let U be uniform over $\mathcal{X}$, independent of X, Y and Z. Then,*

$$\|(Z, X) - (Z, U)\| \leq \|(Y, Z, X) - (Y, Z, U)\|$$

2.3 Space Pseudorandom Generators

A deterministic, space $s(n)$ Turing machine uses $s(n)$ space on any input of size n. A non-uniform, space $s(n)$ statistical test is a deterministic, space $s(n)$ Turing machine M and an infinite sequence of binary strings $a = (a_1, \ldots, a_n, \ldots)$ called the advice strings, where the length of a_n is $\exp(s(n))$, for every $n \in \mathbb{N}$. The result of the test on input x, denoted $M^a(x)$, is the result of running M on x when it has access to the advice $a_{|x|}$. The machine M reads the advice as if it is on a normal input tape, and it has a one-way access to the input x (i.e., it can access the next bit of x but it cannot go "back" and review bits it already read). A pseudorandom generator for space bounded computations is required to produce strings that can be used instead of truly random strings in randomized, space bounded computations (while introducing only small additional error). Therefore, a pseudorandom generator must produce strings that look random to any non-uniform, bounded space statistical test. The following is a formal definition. For more information see e.g. [2].

▶ **Definition 7.** $G = \left\{G_n : \{0,1\}^{m(n)} \to \{0,1\}^n\right\}$ is an ε pseudorandom generator for space $s(n)$ if for every non-uniform, space $s(n)$ statistical test M^a it holds that

$$\left| \Pr_{x \in_R \{0,1\}^n}[M^a(x) = 1] - \Pr_{y \in_R \{0,1\}^{m(n)}}[M^a(G(y)) = 1] \right| \leq \varepsilon$$

The following is an alternative definition (which is equivalent upto a multiplicative factor of n change in ε).

▶ **Definition 8.** $G = \left\{G_n : \{0,1\}^{m(n)} \to \{0,1\}^n\right\}$ is an ε pseudorandom generator for space $s(n)$ if for every $i \in [n]$ and for every non-uniform, space $s(n)$ statistical test M^a it holds that

$$\left| \Pr_{y \in_R \{0,1\}^{m(n)}}[M^a(\text{first } i-1 \text{ bits of } G(y)) = i^{th} \text{ bit of } G(y)] - \frac{1}{2} \right| \leq \varepsilon$$

In this paper, we use Definition 8.

2.4 Strong Extractors

The notion of weak source was first defined by Nisan and Zuckerman [21].

▶ **Definition 9** (Min-Entropy)**.** For a random variable X, the *min-entropy* of X is

$$\mathbf{H}_\infty(X) = -\log \max_x \Pr[X = x]$$

An (n,t) source is a random variable in $\{0,1\}^n$ that has min-entropy at least t.

▶ **Definition 10** (Strong Extractor [22])**.** A function $Ext : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^\ell$ is a (t,ε) strong extractor if for every (n,t) source X and every seed S uniformly distributed over $\{0,1\}^m$ it holds that

$$\|(S, Ext(S,X)) - U\| \leq \varepsilon$$

where U is uniformly distributed over $\{0,1\}^{m+\ell}$.

2.4.1 Average Min-Entropy and Average-Case Extractors

The following definitions and lemmas appear in [9].

▶ **Definition 11** (Average Min-Entropy). For two random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$, the *average min-entropy* of X given Y is

$$\tilde{\mathbf{H}}_\infty(X|Y) = -\log \mathop{\mathrm{E}}_{y\sim Y} \max_{x\in\mathcal{X}} \Pr[X = x|Y = y] = -\log \mathop{\mathrm{E}}_{y\sim Y} \left[2^{-\mathbf{H}_\infty(X|Y=y)}\right]$$

▶ **Lemma 12.** *Let X, Y and Z be random variables. If Y has at most 2^ℓ possible values, then*

$$\tilde{\mathbf{H}}_\infty(X|(Y,Z)) \geq \tilde{\mathbf{H}}_\infty((X,Y)|Z) - \ell \geq \tilde{\mathbf{H}}_\infty(X|Z) - \ell$$

▶ **Definition 13** (Average-case Strong Extractor). A function $Ext : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^\ell$ is an average-case (t,ε) strong extractor if for every pair of random variables (W, I) such that $W \in \{0,1\}^n$ and $\tilde{\mathbf{H}}_\infty(W|I) \geq t$, and every seed S uniformly distributed over $\{0,1\}^m$, it holds that

$$\|(I, S, Ext(S, W)) - (I, U)\| \leq \varepsilon$$

where U is uniformly distributed over $\{0,1\}^{m+\ell}$.

▶ **Lemma 14.** *For any $\gamma > 0$, if Ext is a $(t - \log 1/\gamma, \varepsilon)$ strong extractor, then Ext is also an average-case $(t, \varepsilon + \gamma)$ strong extractor.*

3 Lower Bounds for Conservative One-way Unicast Communication Protocols

For the construction of the pseudorandom generator in Section 4, we will need a lower bound for the communication complexity of a function that outputs a single bit. To this end we consider also projections of composed strong extractors (see notation for projections in Section 2.1).

▶ **Definition 15.** A function $g : \{0,1\}^{m+nk} \to \{0,1\}^{m'}$ is called a (t,ε) *projection of a composed strong extractor (PCSE)* if $g = Ext^{(k)}_S$, where $S \subseteq [m]$ is a subset of size m' for $m' \leq m$ and $Ext^{(k)} : \{0,1\}^{m+nk} \to \{0,1\}^m$ is a (t,ε) composed strong extractor.

The following lemma is the main technical part of our paper. The proof is related to the proof of the "alternating extraction theorem" in [10], which uses ideas from [11]. See also lecture notes [29].

▶ **Lemma 16.** *Let $Ext^{(k)}_S : \{0,1\}^{m+nk} \to \{0,1\}^{m'}$ be a (t,ε) PCSE and let $A_0 \in \{0,1\}^m$, $A_1, \ldots, A_k \in \{0,1\}^n$ be uniformly and independently distributed. Let P be a conservative one-way unicast communication protocol with respect to $Ext^{(k)}$, and let $M_1, \ldots, M_k$ be the messages sent during the execution of P on inputs $A_0, \ldots, A_k$, where the i^{th} player sends M_i, for $i \in [k]$. Fix $\gamma > 0$ and assume that $M_1, \ldots, M_{k-1} \in \{0,1\}^\ell$, for $\ell \leq n - t - \log \frac{1}{\gamma}$. Then,*

$$\|(M_k, Ext^{(k)}_S(A_0, A_1, \ldots, A_k)) - (M_k, U')\| \leq 2^{k+1}(\varepsilon + \gamma)$$

where U' is uniformly distributed over $\{0,1\}^{m'}$.

Proof. To simplify notation, for every $i \in [k]$ we write $Ext^{(i)}$ instead of $Ext^{(i)}(A_0, A_1, \ldots, A_i)$ and let $\overline{A}_i = (A_i, \ldots, A_k)$. For $i > k$ we define $\overline{A}_i$ to be the empty string. Let U be uniformly distributed over $\{0,1\}^m$. Recall that U' is uniformly distributed over $\{0,1\}^{m'}$. Then, for every $z \in \{0,1\}^{m'}$, by Proposition 5,

$$\|(Ext_S^{(k)}|M_k = z) - U'\| \leq \|(Ext^{(k)}|M_k = z) - U\|$$

Therefore, it is enough to prove that

$$\|(M_k, Ext^{(k)}) - (M_k, U)\| \leq 2^{k+1}(\varepsilon + \gamma) \tag{1}$$

By the definition of a conservative one-way unicast communication protocol with respect to $Ext^{(k)}$, for every $i \in [k]$ it holds that

$$M_i = g_i(\overline{A}_{i+1}, M_{i-1}, Ext^{(i-1)}) \tag{2}$$

where g_i is some (deterministic) function, and $M_0 = 0^\ell$. We prove by induction on i, that for every $0 \leq i \leq k$,

$$\|(\overline{A}_{i+1}, M_i, Ext^{(i)}) - (\overline{A}_{i+1}, M_i, U)\| \leq \sum_{j=0}^{i} 2^j(\varepsilon + \gamma)$$

Substituting $i = k$ we get equation (1) as required. For $i = 0$ we have that $\|(\overline{A}_1, M_0, Ext^{(0)}) - (\overline{A}_1, M_0, U)\| = 0$, and the claim holds. Assume that the claim holds for some $0 \leq i < k$ and let $\Delta = \|(\overline{A}_{i+2}, M_{i+1}, Ext^{(i+1)}) - (\overline{A}_{i+2}, M_{i+1}, U)\|$. By equation (2) and Proposition 5,

$$\Delta \leq \|(\overline{A}_{i+2}, M_i, Ext^{(i)}, Ext^{(i+1)}) - (\overline{A}_{i+2}, M_i, Ext^{(i)}, U)\|$$

By the definition of $Ext^{(i+1)}$,

$$\Delta \leq \|(\overline{A}_{i+2}, M_i, Ext^{(i)}, Ext(Ext^{(i)}, A_{i+1})) - (\overline{A}_{i+2}, M_i, Ext^{(i)}, U)\|$$

Let S be uniformly distributed over $\{0,1\}^m$. By the triangle inequality,

$$\begin{aligned}
&\|(\overline{A}_{i+2}, M_i, Ext^{(i)}, Ext(Ext^{(i)}, A_{i+1})) - (\overline{A}_{i+2}, M_i, Ext^{(i)}, U)\| \leq \\
&\quad \|(\overline{A}_{i+2}, M_i, Ext^{(i)}, Ext(Ext^{(i)}, A_{i+1})) - (\overline{A}_{i+2}, M_i, S, Ext(S, A_{i+1}))\| + && (3)\\
&\quad \|(\overline{A}_{i+2}, M_i, S, Ext(S, A_{i+1})) - (\overline{A}_{i+2}, M_i, S, U)\| + && (4)\\
&\quad \|(\overline{A}_{i+2}, M_i, S, U) - (\overline{A}_{i+2}, M_i, Ext^{(i)}, U)\| && (5)
\end{aligned}$$

By Lemma 14, Ext is also an average-case $(t + \log 1/\gamma, \varepsilon + \gamma)$ strong extractor. By Lemma 12,

$$\tilde{\mathbf{H}}_\infty(A_{i+1}|\overline{A}_{i+2}, M_i) \geq \tilde{\mathbf{H}}_\infty(A_{i+1}|\overline{A}_{i+2}) - \ell = \mathbf{H}_\infty(A_{i+1}) - \ell = n - \ell \geq t + \log 1/\gamma$$

and therefore, by Definition 13, $(4) \leq \varepsilon + \gamma$. By Propositions 5 and 6,

$$(3), (5) \leq \|(\overline{A}_{i+1}, M_i, Ext^{(i)}) - (\overline{A}_{i+1}, M_i, S)\|$$

By the inductive hypothesis, $\|(\overline{A}_{i+1}, M_i, Ext^{(i)}) - (\overline{A}_{i+1}, M_i, S)\| \leq \sum_{j=0}^{i} 2^j(\varepsilon + \gamma)$. Putting it together we get that

$$\Delta \leq \varepsilon + \gamma + 2 \cdot \sum_{j=0}^{i} 2^j(\varepsilon + \gamma) = \sum_{j=0}^{i+1} 2^j(\varepsilon + \gamma)$$

as required. ◀

Finally, we give a lower bound for the length of the longest message in a conservative one-way unicast communication protocol that computes a projection of a composed strong extractor.

▶ **Theorem 17.** *Let* $Ext_S^{(k)} : \{0,1\}^{m+nk} \to \{0,1\}^{m'}$ *be a* (t,ε) *PCSE and let* P *be a conservative one-way unicast communication protocol with respect to* $Ext^{(k)}$ *that computes* $Ext_S^{(k)}$ *with bias* $\delta > 0$*, such that* $\varepsilon < \delta \cdot 2^{-(k+2)}$*. Then,*

$$L(P) \geq n - t - k - \log \frac{1}{\delta} - 2$$

Proof. Let $A_0 \in \{0,1\}^m$, $A_1, \ldots, A_k \in \{0,1\}^n$ be uniformly and independently distributed and let $M_1, \ldots, M_k$ be the messages sent during the execution of the protocol P on inputs $A_0, \ldots, A_k$, where the i^{th} player sends M_i, for $i \in [k]$. To simplify notation, we write $Ext_S^{(k)}$ instead of $Ext_S^{(k)}(A_0, A_1, \ldots, A_k)$. Since the protocol P computes $Ext_S^{(k)}$ with bias δ,

$$\delta + 2^{-m'} \leq \Pr_{\bar{A} \in_R \{0,1\}^{m+nk}} \left[M_k = Ext_S^{(k)} \right]$$

Let U' be uniformly distributed over $\{0,1\}^{m'}$. Since $M_k = Ext_S^{(k)}$ is a statistical test on the distribution $(M_k, Ext_S^{(k)})$, and the same statistical test on (M_k, U') passes with probability $2^{-m'}$,[3]

$$\left| \Pr_{\bar{A} \in_R \{0,1\}^{m+nk}} \left[M_k = Ext_S^{(k)} \right] - 2^{-m'} \right| \leq \|(M_k, Ext_S^{(k)}) - (M_k, U')\|$$

Assume for simplicity and without loss of generality, that all messages $M_1, \ldots, M_{k-1}$ have the same length, denoted ℓ. Fix $\gamma = \delta \cdot 2^{-(k+2)}$ and assume towards a contradiction that $\ell < n - t - \log \frac{1}{\gamma}$. Then, by Lemma 16,

$$\|(M_k, Ext_S^{(k)}) - (M_k, U')\| \leq 2^{k+1}(\varepsilon + \gamma)$$

We get that $\delta \leq 2^{k+1}(\varepsilon + \gamma)$ and therefore, $\gamma \geq \delta \cdot 2^{-(k+1)} - \varepsilon > \delta \cdot 2^{-(k+2)}$, which contradicts our choice of γ. ◀

4 Logspace Pseudorandom Generators

We review the construction of the pseudorandom generator of Babai, Nisan and Szegedy [2]. The generator is based on a function f that takes k arguments, each r bits long, and has high multiparty communication complexity. The ε multiparty communication complexity of f, denoted $C_\varepsilon(f)$, is the communication complexity of the best deterministic communication protocol in the NOF model with blackboard communication that computes f with bias at least ε.

The input to the generator consists of t random strings of length r each. Fix $k \leq t$ and let $S_1, S_2, \ldots, S_{\binom{t}{k}}$ be all k-subsets of the input strings in anti-lexicographic order (i.e., each S_i is a set of k strings, each string is r bits long, and S_i appears before S_j if the last string in the symmetric difference of S_i and S_j belongs to S_j). The output of the generator is $f(S_1), f(S_2), \ldots, f(S_{\binom{t}{k}})$.

The proof of the following lemma appears in [2]. We give it for completeness in Appendix A.

[3] We can assume, without loss of generality, that M_k is of length m'.

▶ **Lemma 18.** *For every $\varepsilon > 0$, every function $f : \{0,1\}^{rk} \to \{0,1\}$ and every $s < C_\varepsilon(f)/k$, the above construction gives an ε pseudorandom generator for space s (see Definition 8).*

We make few observations on Lemma 18 and its proof, that will allow us to use our lower bound from Section 3:

1. In the multiparty communication protocol used for the proof, the players communicate in a fixed order. Hence, we can consider communication protocols that satisfy item (2) in Definition 1.
2. In the multiparty communication protocol used for the proof, for every $i < k$, the i^{th} message, sent by the i^{th} player, is used only by player $i+1$. Therefore, the blackboard is not required and we can consider communication protocols that satisfy item (3) in Definition 1.
3. In the multiparty communication protocol used for the proof, for every $j \in [k]$, if the j^{th} player needs to compute $f(T)$ during the simulation, then it holds that the set T comes before the set S in the anti-lexicographic order, and $y_{i_1}, \ldots, y_{i_{j-1}} \in T$ and $y_{i_j} \notin T$. For every such a set T, the j^{th} player can compute $f(T)$ without knowing $y_{i_1}, \ldots, y_{i_{j-1}}$. It suffices that she knows the strings that were fixed, the input strings $y_{i_{j+1}}, \ldots, y_{i_k}$ and the function $f(y_{i_1}, \ldots, y_{i_{j-1}}, z_j, \ldots, z_k)$ for every $z_j, \ldots, z_k \in \{0,1\}^r$. Hence, we can consider communication protocols that satisfy item (1) in Definition 1.
4. In the multiparty communication protocol used for the proof, all messages have the same length. Hence, we can use a lower bound for the length of the longest message sent during the execution of the protocol.

Note that the function f from Definition 1 has an additional input string $b \in \mathcal{B}$. We can think of b as if it is added to all subsets $S_1, S_2, \ldots, S_{\binom{t}{k}}$. Formally, our **adjusted construction** is as follows. The input to the generator consists of t random strings of length r each and an additional random string $b \in \{0,1\}^m$. Let $S_1, S_2, \ldots, S_{\binom{t}{k}}$ be all k-subsets of the input strings (not including the string b) in anti-lexicographic order, as in the original construction. For every $1 \le j \le \binom{t}{k}$, let $S_j = \{y_{i_{j,1}}, \ldots, y_{i_{j,k}}\}$, where $i_{j,1} > i_{j,2} > \cdots > i_{j,k}$. Then, the j^{th} bit in the output of the generator is $f_1(b, y_{i_{j,1}}, \ldots, y_{i_{j,k}})$. Recall that f_1 returns the first bit of the function f (see notation for projections in Section 2.1).

We get the following lemma.

▶ **Lemma 19.** *Fix $\varepsilon > 0$ and a function $f : \{0,1\}^{m+kr} \to \{0,1\}^m$, such that for every conservative one-way unicast communication protocol with respect to f that computes f_1 with bias ε, the length of the longest message is at least C. Then, for every $s < C$, the adjusted construction gives an ε pseudorandom generator for space s.*

▶ **Corollary 20.** *For every constant $c > 0$, there exists an (explicitly given) n^{-c} pseudorandom generator for logspace which converts $O(\log^2 n)$ random bits to poly(n) bits.*

Proof. Let $m = O(\log n)$, $r = O(\log n)$ and let $f : \{0,1\}^{m+r} \to \{0,1\}^m$ be a (t', ε) strong extractor, such that $t' < r - 2\log n - c\log n - 2$ and $\varepsilon < 1/4n^{c+1}$. For an explicit construction of a strong extractor with such parameters see Theorem 4.2 in [13] (for more information see e.g. [20], [27] and [28]). Let $\delta = n^{-c}$, $k = \log n$ and let P be a conservative one-way unicast communication protocol with respect to $f^{(k)}$ that computes $f_1^{(k)}$ with bias δ. Since $\varepsilon < \delta \cdot 2^{-(k+2)} = 1/4n^{c+1}$, Theorem 17 guarantees that $L(P) \ge r - t' - k - \log\frac{1}{\delta} - 2 = r - t' - \log n - c\log n - 2 > \log n$. By Lemma 19, using the adjusted construction with the PCSE $f_1^{(k)}$ and $t = k \cdot 2^{c'}$ for any constant $c' > 1$, we get a δ pseudorandom generator for space $\log n$, that on a seed of length $m + tr = O(\log^2 n)$ produces a pseudorandom string of length $\binom{t}{k} \ge n^{c'}$. ◀

References

1. Miklos Ajtai, Janos Komlos, and Endre Szemeredi. Deterministic simulation in logspace. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC'87, pages 132–140, New York, NY, USA, 1987. ACM.
2. László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992.
3. Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9:283–293, 2013.
4. Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. Tight bounds for set disjointness in the message passing model. *CoRR*, abs/1305.4696, 2013.
5. Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. In *FOCS*, pages 40–47, 2010.
6. Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In *FOCS*, pages 30–39, 2010.
7. Amit Chakrabarti. Lower bounds for multi-player pointer jumping. In *IEEE Conference on Computational Complexity*, pages 33–45, 2007.
8. Carsten Damm, Stasys Jukna, and Jiri Sgall. Some bounds on multiparty communication complexity of pointer jumping. *Computational Complexity*, 7(2):109–127, 1998.
9. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
10. Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In *STOC*, pages 601–610, 2009.
11. Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237, 2007.
12. Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985.
13. Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Struct. Algorithms*, 11(4):315–343, 1997.
14. Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *FOCS*, pages 120–129, 2012.
15. Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *STOC*, pages 356–364, 1994.
16. Richard M. Karp, Christian Schindelhauer, Scott J. Shenker, and Berthold Vocking. Randomized rumor spreading. In *In IEEE Symposium on Foundations of Computer Science*, pages 565–574, 2000.
17. David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'03, pages 482–, Washington, DC, USA, 2003. IEEE Computer Society.
18. Michal Koucky, Prajakta Nimbhorkar, and Pavel Pudlak. Pseudorandom generators for group products. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:113, 2010.
19. Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.
20. Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. *J. Comput. Syst. Sci.*, 58(1):148–173, 1999.

21 Noam Nisan and David Zuckerman. More deterministic simulation in logspace. In *STOC*, pages 235–244, 1993.

22 Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

23 Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *STOC*, pages 159–168, 1999.

24 Omer Reingold, Thomas Steinke, and Salil P. Vadhan. Pseudorandomness for regular branching programs via fourier analysis. *CoRR*, abs/1306.3004, 2013.

25 Michael E. Saks and Shiyu Zhou. $\mathrm{Bp}_h\mathrm{space(s)} \subseteq \mathrm{dspace}(s^{3/2})$. *J. Comput. Syst. Sci.*, 58(2):376–403, 1999.

26 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

27 Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.

28 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

29 Instructor: Leo Reyzin Scribers: Drew Wolpert and Sophia Yakoubov. Alternating extractors and leakage-resilient stream ciphers. New Developments in Cryptography, MIT, 2011.

A Proof of Lemma 18

Fix $f : \{0,1\}^{rk} \to \{0,1\}$, $\varepsilon > 0$ and $s < {}^{C_\varepsilon(f)}/_k$. Assume towards a contradiction that the i^{th} bit of the output of the generator can be predicted by a non-uniform, space s statistical test. That is, there exists a non-uniform, space s statistical test M^a such that

$$\Pr_{y \in_R \{0,1\}^{tr}}[M^a(\text{first } i-1 \text{ bits of } G(y)) = i^{th} \text{ bit of } G(y)] - \frac{1}{2} > \varepsilon$$

where G is the generator that is defined by the construction above. Fix $y = (y_1, \ldots, y_t) \in (\{0,1\}^r)^t$ and let the i^{th} bit of the output of the generator on input y be $f(S)$, where $S = \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ and $i_1 > i_2 > \cdots > i_k$. By an averaging argument, we can fix all input strings from y that are not in S, such that the prediction bias of M^a is preserved. We describe a multiparty communication protocol for k players, that computes $f(S)$ with bias ε. The model of this multiparty communication protocol is the NOF model with blackboard communication, in which the j^{th} player knows all input strings except y_{i_j}, for $j \in [k]$, and the players broadcast their messages. The players simulate the running of M^a on the first $i-1$ bits of $G(y)$ as follows. The first player starts the simulation and continues it for as long as she can, that is, as long as she has access to the input bits that the test reads. Then, the first player sends the state of M^a (i.e., all the memory space used by the machine) to the second player. The second player continues the simulation for as long as she can, and so on. Note that for every $j \in [k]$, the j^{th} player can simulate M^a until the simulation requires the value $f(T)$ for a set T that contains y_{i_j}. Moreover, because the sets used to compute the bits of the generator are ordered in anti-lexicographic order, every set that appears after T, until S appears, contains y_{i_j}. Therefore, the k^{th} player can continue the simulation until it reaches a set that contains $y_{i_1}, y_{i_2}, \ldots, y_{i_k}$, which must be the set S, when the simulation ends and the prediction is made. Sending the space used by the machine $k-1$ times, by each of the first $k-1$ players, results in less than ks communicated bits. Since $ks < C_\varepsilon(f)$, we get a contradiction.