CS 880: Pseudorandomness and Derandomization 5/8/2013

Lecture 29: Space Generators from Communication Complexity

Instructors: Holger Dell and Dieter van Melkebeek Scribe: Xi Wu

DRAFT

In previous lectures we saw two pseudorandom generators for bounded-space machines: Nisan's generator and the Impagliazzo–Nisan–Wigderson generator. Both generators are of the form $G_k : \{0,1\}^{(k+1)s} \mapsto \{0,1\}^{2^k}$ such that

$$G_k\Big(x, y\Big) = G_{k-1}\Big(x\Big)\ ,\ G_{k-1}\Big(\Gamma_k(x,y)\Big) \tag{1}$$

where $x \in \{0,1\}^{ks}$, $y \in \{0,1\}^s$, and $\Gamma_k : \{0,1\}^{ks} \times \{0,1\}^s \to \{0,1\}^{ks}$ is some function that can "recycle the randomness in x". In the INW-generator, Γ is the neighbor function of a suitable expander graph or it is a suitable extractor. Nisan's generator can also be cast in the framework of (1), but the definition is a bit subtle. Here the string x is of length $(2k-1) \cdot s$ and of the form $x = \sigma, h_1, \ldots, h_{k-1}$ and y is of the form $y = h_k$. The string σ is of length s and corresponds to Nisan's "inner seed" as discussed in Lecture 15. The "outer seed" consists of the k strings $h_1, \ldots, h_k$ of length $2s$ each, and they determine k pair-wise uniform hash functions from $\{0,1\}^s \to \{0,1\}^s$. Then we obtain Nisan's generator G_k when Γ_k is of the form

$$\Gamma_k(\sigma, h_1, \ldots, h_{k-1}; h_k) = h_k(\sigma), h_1, \ldots, h_{k-1}\,.$$

One way to look at generators of type (1) is that they exploit the bounded amount of communication that is passed from one party to another: Two players Alice and Bob want to compute a function $f(X,Y)$ where Alice has an input X and Bob has an input Y that is independent from X. In the communication model, we require Alice to do some computation on X and to then send a message to Bob. Bob receives Alice's message, continues the computation on Y, and outputs the result.

Given such a communication protocol, we want to replace the uniform distributions X and Y with distributions that require less randomness, while still approximating the overall behavior of the protocol under uniform input distributions. If the communication between Alice and Bob is limited, then intuitively we should be able to "recycle" the randomness that is used to create Alice's input and pass it on to Bob. In the case of read-once branching programs (ROBP) of small width, Nisan's generator and the INW-generator exploit the communication communication bottleneck in the following way: The branching program is split into half. Alice looks at the input of the first half of the program and computes the state in the middle layer of the program that is reached on the given input. She communicates this state to Bob, who can use it as the start state of the second half of the program and compute the final state of the program that is reached when reading the second half of the input. Since the number of bits used in the communication is only logarithmic in the width of the program, all but a logarithmic number of random bits from the first half can be reused for Bob's input, which leads to the framework (1). The program has a highly hierarchical structure since each half of the program is again an oblivious read-once branching program, and thus, the generator can be applied recursively.

Note that a similar intuition applies to the shrinkage-based pseudorandom generator that we discussed in the last few lectures. In this situation, the branching programs can be non-oblivious in that the order in which the input is read is not known to the generator and each variable can be read multiple times. The intuition for the pseudorandom generator was as follows: We divide the input into two parts: the first half is determined by a pseudorandom restriction X and the second half Y consists of the variables not set by the random restriction. Alice computes the restricted circuit $C|_X$ and simplifies it as much as possible; by the arguments in the last lecture, the circuit size shrinks significantly compared to the original circuit. Alice then communicates the shrunk circuit to Bob, who evaluates the received circuit on its input variables Y. Again, if we condition on the communication between Alice and Bob, we can recycle most of the randomness from X to generate Y.

1 Number-on-Forehead model of communication

In this lecture, we will see a pseudorandom generator for bounded-space machines that exploits communication protocols such as the above, except that we allow for more than two players, i.e., we consider *multiparty* communication. In particular, the pseudorandom generator will be based on a function that requires a large amount of communication between the players in a certain model of multiparty communication, namely one-way unicast communication. This is a more restrictive model than the *number-on-forehead* model, which we will introduce first.

Definition 1 (Number-on-Forehead (NOF) Model). *An NOF-protocol P that computes a function $f : \{0,1\}^{np} \mapsto \{0,1\}$ consists of the following parts:*

1. Players*: There are p players, denoted by $1, \ldots, p$.*
2. Input*: The overall input is $z_1, \ldots, z_p \in \{0,1\}^n$. Each player $j \in [p]$ gets access to all input strings* except for z_j.
3. Communication channel*: The players share a write-once blackboard to communicate and every player can see what is written on the blackboard.*
4. Goal*: Compute $f(z_1, \ldots, z_p)$.*

The input string z_i can be thought of a being written on the forehead of player i. That is, player i can see everyone else's input but not their own. The number of bits written down on the blackboard in an execution of the protocol is the amount of communication used in an execution. The *communication complexity* of a function f in the NOF-model is the minimum amount of communication that an NOF-protocol requires to compute f correctly on all inputs of length np. A trivial upper bound on the communication complexity of any function is n: Player 1 can write the string z_2 on the blackboard so that player 2 knows all input strings and can compute $f(z_1, \ldots, z_p)$.

We write $\mathrm{NOF}_P(f) = C$ to denote that P computes f correctly on all inputs with communication cost at most C. More generally, we say that P computes f with advantage ε if

$$\Pr_{z_1 \cdots z_p \sim U_{np}} \left[P(z_1, \ldots, z_p) = f(z_1, \ldots, z_p) \right] \geqslant \frac{1}{2} + \varepsilon$$

In this case, we write $\mathrm{NOF}^{\varepsilon}_{P}(f) = C$ to denote the fact that P computes f with advantage ε and cost at most C. Finally we define $\mathrm{NOF}^{\varepsilon}(f) = \min_P \mathrm{NOF}^{\varepsilon}_{P}(f)$ as the minimum worst-case communication complexity over all NOF-protocols P that compute f with advantage ε.

Let us consider an example of a function that is hard in the NOF-model. We view each z_i as the characteristic vector of some set $S_i \subseteq [n]$. That is, for every $j \in [n]$, we have $z_i[j] = 1$ if and only if $j \in S_i$. Define

$$f(z_1, \ldots, z_p) = \begin{cases} 1 & \text{if there is a } j \in [n] \text{ such that } j \in \bigcap_{k=1}^{p} S_k\,, \\ 0 & \text{otherwise.} \end{cases}$$

This function is called the *set intersection problem*, and it requires $\Omega\left(n^{1/(p+1)}/2^{2p}\right)$ bits of communication in the p-player NOF-model []. The *generalized inner product* is related to the set intersection, but it asks the players to compute the *parity* of the number of elements in their shared intersection, that is,

$$f(z_1, \ldots, z_p) = \Big| \bigcap_{j=1}^{p} S_j \Big| \bmod 2\,.$$

The generalized inner product is one of the hardest problems known in the NOF-model, and it requires $\Omega\left(n/2^{2p}\right)$ bits of communication []. On the other hand, there is also a corresponding upper bound of $O\left(pn/2^{p}\right)$ []. It is open whether there is a problem that requires $\Omega\left(n/\operatorname{poly}(p)\right)$ bits of communication; Raz [] gives a candidate for such a problem, and a proof implies circuit lower bounds for ACC^0 [].

2 Construction of the BNS-Generator

In this section, we will use a function f that p players want to compute using an NOF-protocol to construct a pseudorandom generator. If the function f is hard in a suitable model of communication, we will later show that the generator is pseudorandom for branching programs of bounded width. We remark that set intersection or generalized inner product could be used as a hard function here, but it would still give significantly worse seed length than Nisan's generator and the INW-generator. We will later introduce a function that is more suitable in our setting and gives the same asymptotic seed length as the best known generators.

Construction 1. ***(Babai, Nisan, and Szegedy [BNS92])***
Given a function $f : \{0,1\}^{np} \mapsto \{0,1\}$ and an enumeration $S_1, \ldots, S_{\binom{t}{p}}$ of all subsets of $[t]$ that have size p, the BNS-generator *is the function $G : \{0,1\}^{nt} \mapsto \{0,1\}^{\binom{t}{p}}$ with*

$$G(x) \doteq f\Big(x|_{S_1}\Big) \circ f\Big(x|_{S_2}\Big) \circ \cdots \circ f\Big(x|_{S_{\binom{t}{p}}}\Big)\,.$$

Here, for $x = (x_1, \ldots, x_t) \in (\{0,1\}^n)^t$, we let $x|_S$ denote the string $(x_i)_{i \in S}$ of length np.

To get a seed length of $O(\log^2 r)$ for $r = \binom{t}{p}$, we will later choose $p, t, n \in \Theta(\log r)$ with $p \ll t \ll n$. Note that the construction above depends not only on the hard function f but also on the ordering of all subsets of $[t]$ that have size p. On a high level, we have two requirements on the ordering and the function f:

(i) Since we want to use G to derandomize small-space machines, G itself should be computable in space $O(|x|) = O(nt)$.

(ii) The ordering should be such that we can use the hardness of f to prove that the output of G looks pseudorandom to small-width branching programs. In particular, the following property will suffice: if there is a small-width branching program that can distinguish the output of G from the uniform distribution with non-trivial probability, then we can use it to construct a low-cost NOF-protocol that computes f with non-trivial advantage.

Satisfying (i) is straightforward since the function f that we will choose is computable in space $O(|x|)$ and the order will be so that S_{i+1} can be computed in space $O(|x|)$ from S_i. It is (ii) that requires a bit more insight. In the next section, we will sketch the idea of how to use a distinguisher for G to obtain a low-cost protocol for f, and we will pick a suitable ordering along the way.

3 From Distinguisher to NOF-protocol

In the last section, we introduced a candidate generator G based on a hard function f. Now we want to show how a small-space machine D that can distinguish the output of G from the uniform distribution can be used to construct a fairly good NOF-protocol Π for f. If f is *hard* to compute by NOF-protocols, this hardness transfers to the inability to distinguish G from uniform by small-space machines.

Lemma 1 ([BNS92]). *Let $p, t, n \in \mathbb{N}$, $f : \{0,1\}^{np} \to \{0,1\}$, and $G : \{0,1\}^{nt} \to \{0,1\}^r$ be the BNS-generator. If $G(U_{nt})$ is not ϵ-pseudorandom for space S machines, then f has a p-player NOF-protocol that computes it with advantage ϵ/r and with communication cost at most pS.*

Proof. If G is not ϵ-pseudorandom, then it has a distinguisher. Recall that a distinguisher D is a space-S machine that takes as input a string of length $r = \binom{t}{p}$ and outputs a single bit such that

$$\left|\Pr\left[D(G(U_{nt})) = 1\right] - \Pr\left[D(U_r)) = 1\right]\right| > \frac{1}{2} + \varepsilon\,.$$

In Lecture 23, we used a *hybrid argument* to convert any distinguisher to a *predictor* P with slightly worse parameters. In particular, a predictor is a machine that takes a string of length $\ell < r$ as input and outputs a single bit such that

$$\Pr_{\rho \sim G(U_{nt})}\left[P\big(\rho_{\leq \ell}\big) = \rho_{\ell+1}\right] > \frac{1}{2} + \varepsilon/r\,. \tag{2}$$

Here, $\rho_{\leq \ell}$ denotes the prefix of ρ of length ℓ. It is this predictor that we are going to construct the protocol Π from. For this, let us first realize that $\rho_i \doteq (G(x))_i = f\left(x|_{S_i}\right)$ holds for $x \in \{0,1\}^{nt}$. In particular, the bit $\rho_{\ell+1}$ depends only on the pn-length substring of x selected by $S_{\ell+1}$. Therefore, we can fix some setting for the rest of x in (2). More precisely, there is a setting for $x|_{\overline{S_{\ell+1}}} \in \{0,1\}^{(t-p)n}$ such that

$$\Pr_{x|_{S_{\ell+1}} \sim U_{np}}\left[P\big(\rho_{\leq \ell}\big) = \rho_{\ell+1}\right] > \frac{1}{2} + \varepsilon/r\,. \tag{3}$$

Since $\rho_{\leq \ell} = \rho_{\leq \ell}(z)$ is a function of $z \doteq x|_{S_{\ell+1}}$ and $x|_{\overline{S_{\ell+1}}}$ is fixed to a constant, we can rewrite (3) as

$$\Pr_{z \sim U_{np}} \left[P\big(\rho_{\leq \ell}(z)\big) = f(z) \right] > \frac{1}{2} + \varepsilon/r \,. \tag{4}$$

Note that, while $\rho_{\leq \ell} = \rho_{\leq \ell}(z)$ is a function of z, it is not clear that it can be computed in space $O(S)$ since we do not in general require f to be computable in that space. However, ρ_i for $i \leq \ell$ does not depend on *all* z_j for $j \in [p]$. This is because S_i is different from $S_{\ell+1}$ for all $i \neq \ell + 1$. So, while $\rho_{\leq \ell}(z)$ may not be computable in small space, an NOF-protocol can simulate P on input $\rho_{\leq \ell}$ since the players themselves are computationally unbounded, and player j can compute all ρ_i that do not depend on z_j. The black-board communication then consists of the state in the branching program P, and there is always *some* player who can continue the computation of P and produce the next state. Formally, we get the following framework for a p-player NOF-protocol Π:

> *Input:* Player j gets as input $z_1 \dots z_{j-1} z_{j+1} \dots z_p$.
>
> *Last item written on the blackboard:* A state (i, s) of the branching program P (initially the start state of P).
>
> Some player j who can compute ρ_i from its input (i.e., $\rho_i = \rho_i(z)$ does not depend on z_j) goes next and simulates P: the branching program is started at (i, s) and fed the input ρ_i; the player continues to feed subsequent ρ_i's for as long as she can compute them without knowing z_j; finally, player j writes the state that P ends up in on the blackboard.

The fact that Π computes f with advantage ϵ/r follows from (4).

To analyze the communication cost, note that Π only ever writes down a state of the branching program if the players switch. In general, it could be that $\ell = r - 1$ and the sequence $S_1, \dots, S_r$ is such that the players end up switching $\sim r/p$ times, which is too large for our purposes. Therefore, it is natural to choose the order of the subsets so that each player can simulate P for as long as possible.

If we imagine for a moment that we knew the positions of $z_1 \dots z_p$ inside of x in advance, we could construct such an ordering greedily as follows: we let $S_1, \dots, S_{r/2}$ be all sets that do *not* contain z_1, in an arbitrary order. Then player 1 can simulate P until the middle layer $r/2$ is reached. Furthermore, it is natural to order the remaining $r/2$ sets so that the first half $S_{r/2+1}, \dots, S_{r/2+r/4}$ of the remaining sequence does not contain z_2; then player 2 can simulate P until layer $r/2 + r/4$. This continues in p iterations until all of the sequence has been constructed and the entire branching program can be simulated by having the players take turns, and each player goes only once.

The problem with the greedy approach above is that the positions of $z_1 \dots z_p$ correspond to the set $S_{\ell+1}$, which depends on the parameter ℓ that comes out of the hybrid argument, and we have no control over it. However, there is a simple ordering of the subsets that looks like one of the greedy ones above for *all* choices of $S_{\ell+1}$: the lexicographical ordering. We simply order the sets such that their corresponding indicator vectors $\chi_{S_i} \in \{0,1\}^t$ are sorted in lexicographical order. For clarity, we give the following inductive definition in terms of the sets directly.

Definition 2 (Lexicographical ordering). *Let $U \subset \mathbb{N}$ be a non-empty set. The sequence of distinct sets $S_1, \dots, S_r$ in $\binom{U}{p}$ with $r = \binom{|U|}{p}$ and $p \leq |U|$ is the lexicographical ordering of $\binom{U}{p}$ if $r = 1$ or if, for $u = \min U$, we have:*

- $u \notin S_1, \ldots, S_{r/2}$ *and* $u \in S_{r/2+1}, \ldots, S_r$, *and*
- *the sequences* $S_1, \ldots, S_{r/2}$ *and* $S_{r/2+1} \backslash u, \ldots, S_r \backslash u$ *are both equal to the lexicographical ordering of* $\binom{U \backslash u}{p-1}$.

If G is constructed with the above order of the subsets, it is clear that every player in the protocol Π only has to go once. Furthermore, the layer i of the communicated state (i, s) becomes redundant because each player can easily deduce it from the number of states communicated thus far. So the only information that the players have to write down is s, the position in the layer, which requires S bits since that is the space required to run P. The total communication on the blackboard is $p \cdot S$. □

Lemma 1 says that communication hardness of f in the NOF-model transfers to pseudorandomness of G for branching programs. Let us formulate this contrapositive explicitly below.

Corollary 2. *Let* $f : \{0,1\}^{np} \to \{0,1\}$ *be a function that does not have a p-player NOF-protocol that computes it with advantage* $\geq \epsilon$ *and communication at most* C. *Then the BNS-generator* $G : \{0,1\}^{nt} \to \{0,1\}^r$ *constructed from* f *is* ϵr*-pseudorandom for space-*C/p *machines.*

The problem with Corollary 2 is that we do not not of any function f that is sufficiently hard in the NOF-model. The hardest functions we know in this model only give a seed length that is not even poly-logarithmic in r. We therefore need a more restrictive model of communication if we want to use the general framework of Lemma 1.

4 Conservative One-way Unicast Model

We will now inspect the proof of Lemma 1 more closely; it constructs an NOF-protocol Π that has a very special structure, and we will formalize what this structure is to define a model of communication that is more restrictive than the NOF-model. Later we will construct a function that we can prove is very hard in the new model.

The protocol in the proof of Lemma 1 has the following properties:

(1) Each player acts only once and they go in a fixed order, so the protocol is *one-way*.

(2) Each player sends a message only to the next player in the fixed order (and not to all players), so the protocol is *unicast* (as opposed to multicast).

The third and last property is more subtle and used to reduce the dependencies between the players: The player $j \in [p]$ does not need to know the input $z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_p$ in its entirety. To understand what exactly we mean by this, consider what player j is doing: She picks up the simulation of the branching program P in some state (i, s) and will end the simulation in some state (i', s'). For this, she clearly needs to know the branching program P and receive the state s from player $j-1$; as we already discussed she can deduce the layer i from her position j in the order of players. Recall that the players simulate P on input $\rho_{\leq \ell}$. She therefore also needs access to the string $\rho_i \ldots \rho_{i'}$ used as the input to P when started at state (i, s); she does *not* need access to the string $\rho_1 \ldots \rho_{i-1}$ or the string $\rho_{i'+1} \ldots \rho_\ell$. Since $\rho_k = f(x|_{S_k})$ and all coordinates of x are constant except for the ones that correspond to $z = x|_{S_{\ell+1}}$, this means that $\rho'_j \doteq \rho_i \ldots \rho_{i'}$ is a function

of z. So player j needs access to the positions $A \doteq (S_i \cup \cdots \cup S_\ell) \cap S_{\ell+1}$ of x, which correspond to positions of z. The choice of the sequence S_k guarantees that z_j does not occur in A and player j does not need access to this part of the input. Thus $\rho'_j = \rho'_j(z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_p)$ is a function of z but independent from z_j. Unfortunately, we have that $z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_p \in A$ may hold in general, so player j needs information from the entire input to determine ρ'_j. However, the way this information is accessed is very special: We actually have that $z_1, \ldots, z_{j-1} \in S_k$ holds *for all* $k \in [i, i']$, whereas the appearance of the $z_{j+1}, \ldots, z_p$ varies in the sets S_k for $k \in [i, i']$. That is, for $k \in [i, i']$, we have $f(x|_{S_k}) = f(\tilde{z}_1, \ldots, \tilde{z}_p)$ where $\tilde{z}_1 = z_1$ up to $\tilde{z}_{j-1} = z_{j-1}$ holds because of the lexicographical ordering of the S_k, and furthermore, the $\tilde{z}_{j'}$ for $j' \geq j$ are either some constant part of x, or they are contained in the set $\{z_{j+1}, \ldots, z_p\}$. Therefore, the protocol constructed in Lemma 1 satisfies the following additional restriction of the NOF-model.

(3) Each player j's access to z can be divided into the following two components:

   (a) Player j has *direct access* to the string $z_{j+1} \ldots z_p$
   (b) Player j has *oracle access* to the function $f_j : (\{0,1\}^n)^{p-j+1} \to \{0,1\}$ with

$$f_j\Big(\tilde{z}_j, \ldots, \tilde{z}_p\Big) \doteq f\Big(z_1, \ldots, z_{j-1}, \tilde{z}_j, \tilde{z}_{j+1}, \ldots, \tilde{z}_p\Big).$$

   A communication protocol with this type of access is called *conservative*.

The reason why the latter restriction on the communication model makes it less difficult to prove the hardness of a function f in the restricted model is that the oracle f_j will in general not leak too much information about the string $z_1 \ldots z_{j-1}$.

Any NOF-protocol for a function f that also satisfies the conditions (1)–(3) is called a *conservative one-way unicast (COWU) communication protocol*. When talking about the COWU communication model, we make a small adjustment to how we measure the amount of communication used by a COWU-protocol: We say that a protocol uses C bits of communication if the longest message that was sent from one player to the next has length C. Then the following lemma connects the pseudorandomness of the BNS-generator G with the communication complexity of f in the COWU-model. As discussed, the proof is essentially the same as the one of Lemma 1.

Corollary 3. *Let $f : \{0,1\}^{np} \to \{0,1\}$ be a function that does not have a p-player COWU-protocol that computes it with advantage $\geq \epsilon$ and communication at most C per player. Then the BNS-generator $G : \{0,1\}^{nt} \to \{0,1\}^r$ constructed from f is ϵr-pseudorandom for space-C machines.*

5 Parameters in the Logspace Setting

By Corollary 3, we know that a hard function in the COWU-model gives rise to a pseudorandom generator for space-bounded computation. Let us consider the case of logspace-machines, i.e., machines whose space bound $C = O(\log r)$ is at most logarithmic its input length r. Then we need to choose $\epsilon \leq O(1/r)$ in Corollary 3 in order to get a non-trivial result. The construction of the BNS-generator furthermore requires that $\binom{t}{p} \geq r$, which can be achieved by setting $p \geq \log r$ and $t \geq 2p$. This is because $\binom{t}{p} \geq (t/p)^p \geq 2^p \geq r$. If furthermore $p \leq O(\log r)$, then the condition on ϵ can be rewritten as $\epsilon \leq O(2^{-p})$ and the condition on C as $C \leq O(p)$. The function f can only be hard if $n > C$ and, as we will see in the rest of this document, is possible to achieve with $n \leq O(p)$. Overall, we get $nt \sim np \sim \log^2 r$ as the seed length of the BNS-generator.

6 Construction of the Hard Function

We construct a function $f : \{0,1\}^{np} \to \{0,1\}$ that requires a lot of communication in the COWU model to compute or to approximate. The construction is based on an extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^k$ that is applied iteratively p times. The extractor is going to be a *strong* extractor that is able to extract $2k$ random bits when given access to a source of *average min-entropy* at least k and a uniform seed of length k. The precise extractor property that we will need is as follows:

Definition 3. $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^k$ *is a* (k,ϵ)-extractor *if, for all random variables* X *on* $\{0,1\}^n$*,* $Y = U_k$*, and* Z *on* $\{0,1\}^*$ *with* $H_\infty(X|Z) \geq k$*, we have*

$$d_{\mathrm{stat}}\Big(Z\ Y\ \mathrm{Ext}(X,Y)\ ,\ Z\ Y\ U_k\Big) \leq \epsilon\,. \tag{5}$$

Here $H_\infty(X|Z)$ denotes the *conditional average min-entropy*, the definition of which we leave at an intuitive level until later.

We define the functions $E^j : \{0,1\}^k \times \{0,1\}^{nj} \to \{0,1\}^k$ for $j \in [p]$ as follows:

$$\begin{aligned} E^1(y;x_1) &\doteq \mathrm{Ext}(x_1,y) \\ E^2(y;x_1,x_2) &\doteq \mathrm{Ext}(x_2,E^1(y;x_1)) \\ &\vdots \\ E^j(y;x_1,\ldots,x_j) &\doteq \mathrm{Ext}\Big(x_j,E^{j-1}(y;x_1,\ldots,x_{j-1})\Big)\,. \end{aligned}$$

That is, E^1 takes a seed y and x_1 to produce a new seed $y' = \mathrm{Ext}(x_1,y)$. Then E^2 takes the seed y' and x_2 to produce a new seed $y'' = \mathrm{Ext}(x_2,y')$ and so on. Finally, we let f be the first bit of the output of E^p. Note that there is a small mismatch in the input length because of the global seed y.

We extend the COWU model so that there can be an additional input y that *no* player has direct access to, and that is only revealed indirectly by oracle access to the function $f_j(\tilde{z}_j,\ldots,\tilde{z}_p) = f(y;x_1,\ldots,x_{j-1},\tilde{z}_j,\ldots,\tilde{z}_p)$ for player j. The following theorem then captures the hardness result for f.

Theorem 4 (Ganor and Raz [GR13]). *Let* $E^p : \{0,1\}^{k+np} \to \{0,1\}^k$ *be the function above, constructed from a* (k,ϵ)*-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^k$ *in the sense of Definition 3, and let* $f = (E^p)_1$ *be the first bit of this function. Then every COWU-protocol* Π *for* f *that uses at most* $n-k$ *bits of communication per player satisfies*

$$\Pr_{yx\sim U_{k+np}}\Big[\Pi(y,x) = f(y,x)\Big] \leq \frac{1}{2} + 2^p\epsilon \tag{6}$$

6.1 Slight modification of the BNS-generator

For the above hard function with a hidden input y to lead to a pseudorandom generator, we need to add this seed to the BNS-generator G as well. That is, we redefine $G : \{0,1\}^k \times \{0,1\}^{nt} \to \{0,1\}^{\binom{t}{p}}$ as

$$G(y;x) \doteq f\Big(y;x|_{S_1}\Big) \circ f\Big(y;x|_{S_2}\Big) \circ \cdots \circ f\Big(y;x|_{S_{\binom{t}{p}}}\Big)\,.$$

As long as $k \leq O(nt)$, this has no adverse effect on the seed length of the generator, and the proof of Corollary 3 goes through as before.

7 Proof of Theorem 4

The intuition for why the function f is hard is as follows: Player 1 does not have direct access to y or x_1, but only to the function table of f_1. This function table does not reveal *any* information about x_1 and little about y, for which reason it will be very difficult for player 1 to compute $y' = E^1(y; x_1)$ with any significant advantage. That is, y' is close to uniform and player 1 can send only very little information about y' to player 2. Furthermore, if the communication is bounded, player 1 can also not send a lot of information about x_2 to player 2. That is, even conditioned on the information that player 2 receives about x_2 from player 1, the min-entropy of x_2 will still be pretty large. Large enough in fact so that the almost uniform seed y' can make the output of $E^2(y; x_2)$ look almost uniform again, and player 2 has little chance of computing that value. This idea for the analysis goes through till player p and we get that the first bit of E^p cannot be predicted with any significant advantage. Theorem 4 follows immediately from the following lemma.

Lemma 5. *Let* $\mathrm{E}^j : \{0,1\}^{k+nj} \to \{0,1\}^k$ *be the sequence of functions above, constructed from a* (k,ϵ)*-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^k$ *in the sense of Definition 3. Let* Π *be a COWU-protocol for* f *that uses at most* $n-k$ *bits of communication per player. Furthermore, let* $Y = U_k$ *and* $X_j = U_n$ *for* $j \in [p]$ *be independent distributions, and let* $M_j = M_j(Y, X)$ *be the distribution of messages sent by player* j *on input* Y, X*. Then*

$$d_{\mathrm{stat}}\Big(M_j \; X_{\geq j+1} \; \mathrm{E}^j(Y, X_{\leq j}) \; , \; M_j \; X_{\geq j+1} \; U_k\Big) \leq (2^j - 1)\epsilon \, . \tag{7}$$

In particular, for $j = p$, we have

$$d_{\mathrm{stat}}\Big(M_p \; \mathrm{E}^p(Y, X) \; , \; M_p \; U_k\Big) \leq (2^p - 1)\epsilon \, ,$$

which means that M_p (and hence the output of the protocol Π) is fairly uncorrelated with E^p (and hence f). In particular, it implies (6).

Proof (of Lemma 5). We prove the claim by induction on j. For convenience, we define the case $j = 0$ as $\mathrm{E}^0 : \{0,1\}^k \to \{0,1\}^k$ with $\mathrm{E}^0(y) = y$ and we define M_0 as the empty string. Then the two distributions in (7) are identical, and we have for $j = 0$:

$$d_{\mathrm{stat}}\Big(M_0 \; X \; \mathrm{E}^0(Y) \; , \; M_0 \; X \; U_k\Big) = 0 \, .$$

Now let $j \geq 1$. We want to rewrite (7) so that we can apply the induction hypothesis. For this, we express M_j in terms of M_{j-1} as follows: By definition, M_j is the message that player j passes to player $j+1$ when

- receiving message M_{j-1} from player $j-1$,
- reading the string $X_{\geq j+1}$ it has access to, and
- inspecting the truth table of $f_j(Y; X_{\leq j-1}, \tilde{z}_j, \ldots, \tilde{z}_p)$ that it has access to.

In particular, since each player in the protocol is internally deterministic, M_j is a deterministic function $M_j = g(M_{j-1}, X_{\geq j+1}, f_j)$ depending on the three items above. A crucial observation

is now that, because of the recursive structure of E^p, the function table of f_j can be fully reconstructed from f and the string $\mathrm{E}^{j-1} = \mathrm{E}^{j-1}(X_{\leq j-1}, Y)$. Therefore, M_j can be written as $M_j = g(M_{j-1}, X_{\geq j+1}, \mathrm{E}^{j-1})$. Thus, we can rewrite the statistical distance in (7) as

$$\begin{aligned}
&d_{\mathrm{stat}}\Big(g(M_{j-1}, X_{\geq j+1}, \mathrm{E}^{j-1})\ X_{\geq j+1}\ \mathrm{E}^j(Y, X_{\leq j})\ ,\\
&\qquad g(M_{j-1}, X_{\geq j+1}, \mathrm{E}^{j-1})\ X_{\geq j+1}\ U_k\Big)\\
&\leq d_{\mathrm{stat}}\Big(M_{j-1}\ X_{\geq j+1}\ \mathrm{E}^{j-1}\ \mathrm{E}^j(Y, X_{\leq j})\ ,\\
&\qquad M_{j-1}\ X_{\geq j+1}\ \mathrm{E}^{j-1}\ U_k\Big)\ .\\
&= d_{\mathrm{stat}}\Big(M_{j-1}\ X_{\geq j+1}\ \mathrm{E}^{j-1}\ \mathrm{Ext}(X_j, \mathrm{E}^{j-1})\ ,\\
&\qquad M_{j-1}\ X_{\geq j+1}\ \mathrm{E}^{j-1}\ U_k\Big)\\
&\leq d_{\mathrm{stat}}\Big(M_{j-1}\ X_{\geq j+1}\ \mathrm{E}^{j-1}\ \mathrm{Ext}(X_j, \mathrm{E}^{j-1})\ ,\ M_{j-1}\ X_{\geq j+1}\ U'_k\ \mathrm{Ext}(X_j, U'_k)\Big)\\
&+d_{\mathrm{stat}}\Big(M_{j-1}\ X_{\geq j+1}\ U'_k\ \mathrm{Ext}(X_j, U'_k)\ ,\ M_{j-1}\ X_{\geq j+1}\ U'_k\ U_k\Big)\\
&+d_{\mathrm{stat}}\Big(M_{j-1}\ X_{\geq j+1}\ U'_k\ U_k\ ,\ M_{j-1}\ X_{\geq j+1}\ E^{j-1}\ U_k\Big) =: \delta_1 + \delta_2 + \delta_3\ .
\end{aligned}$$

The first inequality follows because of the general fact that applying the same deterministic function to two random variables can only decrease their statistical distance (we remark that this also works if the function g is a randomized mapping and the players have access to private randomness). The equality follows by definition of E^j. And the second inequality is the triangle inequality.

Now we can bound $\delta_1 \leq d_{\mathrm{stat}}\Big(M_{j-1}\ X_{\geq j}\ \mathrm{E}^{j-1}\ ,\ M_{j-1}\ X_{\geq j}\ U'_k\Big) \leq (2^{j-1}-1)\epsilon$ and $\delta_3 \leq (2^{j-1}-1)\epsilon$ by induction hypothesis. Furthemore, $\delta_2 \leq \epsilon$ follows because Ext is a (k,ϵ)-extractor in the sense of Definition 3, which we apply with $X = X_j$, $Y = U'_k$, and $Z = M_{j-1}\ X_{\geq j+1}$. This is possible since even conditioned on M_{j-1}, which is a random variable with support at most 2^{n-k}, the random source X_j has enough average min-entropy left, that is, $H_\infty(X|Z) = H_\infty(X_j|M_{j-1}\ X_{\geq j+1}) \geq H_\infty(X_j|X_{\geq j+1}) - (n-k) = n-(n-k) = k$ and the extractor works for the given average k-source and produces a distribution that is ϵ-close to the uniform distribution in the sense above.

Overall, we get $\delta_1 + \delta_2 + \delta_3 \leq (2^j - 2)\epsilon + \epsilon = (2^j - 1)\epsilon$. □

References

[BNS92] László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45(2):204–232, 1992.

[GR13] Anat Ganor and Ran Raz. Space pseudorandom generators by communication complexity lower bounds. *Electronic Colloquium on Computational Complexity*, 2013. TR13-064.

A Some Basics of Min-Entropy

Our construction of hard function in COWU model requires the use of strong extractors. In this section we list some important basic knowledge about min-entropy, which is central to the

understanding of extractors. As usual, given a random variable X, its min-entropy is defined to be $\mathrm{H}_\infty(X) = -\log \max_x \Pr[X = x]$. It satisfies the following properties, as we would expect for a quantity that measures randomness.

Fact 6. *1. (Joint distribution always has more randomness)* $\mathrm{H}_\infty(X, Y) \geqslant \mathrm{H}_\infty(X)$ *for any random variables* X, Y*.*

2. (Randomness is additive if independent) if X, Y *are independent then* $\mathrm{H}_\infty(X, Y) = \mathrm{H}_\infty(X) + \mathrm{H}_\infty(Y)$*.*

Similarly to conditional Shannon entropy we define conditional min-entropy: Given two random variables X, Y

$$\begin{aligned}\mathrm{H}_\infty(X|Y) &= -\log \mathrm{E}_{y\sim Y}\left[\max_x \Pr[X = x|Y = y]\right] \\ &= -\log \mathrm{E}_{y\sim Y}\left[2^{-\mathrm{H}_\infty(X|Y=y)}\right]\end{aligned}$$

Compare this to the definition of conditional Shannon entropy $\mathrm{H}(X|Y) = \mathrm{E}_{y\sim Y}[\mathrm{H}(X|Y = y)]$. For conditional Shannon entropy we have the nice equality $\mathrm{H}(X|Y) = \mathrm{H}(X, Y) - \mathrm{H}(Y)$, which is known as the *chain rule for Shannon entropy*. If X, Y are independent, then for both notions we have

Fact 7. *If* X, Y *are independent,* $\mathrm{H}_\infty(X|Y) = \mathrm{H}_\infty(X)$*, and* $\mathrm{H}(X|Y) = \mathrm{H}(X)$*.*

Either of these two equalities can be verified directly from corresponding definition. But for Shannon entropy, it also follows from chain rule and that $\mathrm{H}(X, Y) = \mathrm{H}(X) + \mathrm{H}(Y)$ for independent X, Y. Note that in this case, we still have for min-entropy $\mathrm{H}_\infty(X|Y) = \mathrm{H}_\infty(X, Y) - \mathrm{H}_\infty(Y)$.

However, when X, Y are *not* independent, the relationship between $\mathrm{H}_\infty(X|Y)$ and $\mathrm{H}_\infty(X, Y)$ become more subtle. One could construct a joint distribution X, Y such that $\mathrm{H}_\infty(X, Y) \geqslant n$, $\mathrm{H}_\infty(Y) = O(1)$, but $\mathrm{H}_\infty(X|Y) = O(1)$. One such construction is via the following characterization of conditional entropy,

Proposition 8. *Given a joint distribution* X, Y *over* $\mathcal{X} \times \mathcal{Y}$*, average min-entropy* $\mathrm{H}_\infty(X|Y)$ *measures the predictability of* X *from* Y*:*

$$\mathrm{H}_\infty(X|Y) = -\log \max_{P:\mathcal{Y}\mapsto\mathcal{X}} \Pr_{(x,y)\sim(X,Y)}\Big[P(y) = x\Big]$$

Fortunately, it turns out that if the support of Y is small, then we could still have something meaningful.

Fact 9. *If* $|\mathrm{supp}(Y)| \leqslant 2^\ell$*, then* $\mathrm{H}_\infty(X|Y) \geqslant \mathrm{H}_\infty(X, Y) - \ell$*.*

A direct averaging argument gives the following,

Fact 10. *If* $\mathrm{H}_\infty(X|Y) \geqslant k$*, then with probability at least* $1 - \varepsilon$ *over* $y \sim Y$*,*

$$\mathrm{H}_\infty(X|Y = y) \geqslant k - \log \frac{1}{\varepsilon}$$

Combining these two facts we can deduce the *chain rule for min-entropy*.

Fact 11. *Suppose that* $|\mathrm{Y}| \leqslant 2^\ell$. *Then with probability at least* $1-\varepsilon$ *over* $y \sim Y$,

$$\mathrm{H}_\infty(X|Y=y) \geqslant \mathrm{H}_\infty(X,Y) - \ell - \log\frac{1}{\varepsilon} \geqslant \mathrm{H}_\infty(X) - \ell - \log\frac{1}{\varepsilon}$$

Note that in this chain-rule, we condition on specific $y \sim Y$, and an additional $\log(1/\varepsilon)$ min-entropy is lost. Finally, we will also work with multiple conditioning.

Proposition 12. *Suppose that* $|\mathrm{supp}(Z)| \leqslant 2^\ell$. $\mathrm{H}_\infty(X|Y,Z) \geqslant \mathrm{H}_\infty(X,Z|Y) - \ell \geqslant \mathrm{H}_\infty(X|Y) - \ell$.

This follows from that double expectation is a double summation

$$\begin{aligned}
\mathrm{E}_{y,z}\left[\max_x \Pr[X=x|Y=y,Z=z]\right] &= \mathrm{E}_y\left[\mathrm{E}_z\left[\max_x \Pr\left[(X=x|Z=z)\middle|Y=y\right]\right]\right] \\
&= \mathrm{E}_y\left[2^{-\mathrm{H}_\infty\left((X|Z)\middle|Y=y\right)}\right] \\
&\leqslant \mathrm{E}_y\left[2^{-\left(\mathrm{H}_\infty\left((X,Z)\middle|Y=y\right)-\ell\right)}\right] \\
&= 2^\ell \mathrm{E}_y\left[2^{-\mathrm{H}_\infty\left((X,Z)\middle|Y=y\right)}\right]
\end{aligned}$$

Now take $-\log$ on both sides (note $\leqslant$ becomes $\geqslant$) and we are done.

B Hardness in the COWU Model

We are left to analyze the communication complexity of E^p we constructed in the last section. For this we need strong average min-entropy extractor, defined as follows,

Definition 4. *A* (k,ε)-average min-entropy extractor *is a function* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \mapsto \{0,1\}^m$ *such that for any joint random variable* (X,Y) *with* $H(X|Y) \geqslant k$, *we have that* $(\mathrm{Ext}(X,U_d),Y)$ *is* ε*-close to* (U_m,Y). *Similar to usual extractor, an average min-entropy extractor is* strong *if* $(U_d,\mathrm{Ext}(X,U_d),Y)$ *is* ε*-close to* (U_d,U_m,Y).

Proposition 13. *A* (k,ε)*-extractor is a* $(k,4\sqrt{\varepsilon})$*-average min-entropy extractor.*

Proof. In the high level, the proof involves two steps.

1. We show that for any $k \leqslant n-1$, and any $t > 0$, a (k,ε)-extractor is also a $(k-t, 2^{t+2}\varepsilon)$-extractor.
2. We choose a proper t so that (k,ε)-extractor gives a $(k,4\sqrt{\varepsilon})$-average min-entropy extractor.

Assume step 1 for a while, we show how to do step 2. Suppose that $\mathrm{H}_\infty(X|Y) \geqslant k$. Then with probability at least $(1-\delta)$ over $y \sim Y$, $\mathrm{H}_\infty(X|Y=y) \geqslant k - \log(1/\delta)$. Now step 1 says that applying a (k,ε)-extractor gives error $2^{\log(1/\delta)+2}\varepsilon$. Therefore the total error is

$$\delta + \frac{4\varepsilon}{\delta}$$

Set $\delta = 2\sqrt{\varepsilon}$ gives the required bound.

It is left to complete step 1. We first make an observation about extractor. By definition, Ext is a (k, ε)-extractor means

$$\max_{X:k\text{-source}} \max_{T\subseteq\{0,1\}^m} \Big|\Pr[\mathrm{Ext}(X, U_d) \in T] - \mu(T)\Big| \leqslant \varepsilon$$

Therefore we can swap the order the quantifier, which is

$$\max_{T\subseteq\{0,1\}^m} \max_{X:k\text{-source}} \Big|\Pr[\mathrm{Ext}(X, U_d) \in T] - \mu(T)\Big| \leqslant \varepsilon$$

Assume for contradiction that Ext is not a $(k-t, 2^{t+2}\varepsilon)$-extractor. Therefore for some $T \subseteq \{0,1\}^m$ and $(k-t)$-source Y,

$$\Big|\Pr[\mathrm{Ext}(Y, U_d) \in T] - \mu(T)\Big| > 2^{t+2}\varepsilon$$

Without loss of generality, suppose that

$$\Pr[\mathrm{Ext}(Y, U_d) \in T] - \mu(T) > 2^{t+2}\varepsilon$$

Our plan now has two steps:

1. Construct a k-source X, such that $X = \lambda Y + (1-\lambda)Z$, where Z is also a k-source.
2. Show that $\Pr[\mathrm{Ext}(X, U_d) \in T] - \mu(T) > \varepsilon$.

Let p_X, p_Y, p_Z be the probability mass functions of X, Y and Z. In order to do step 1, we need that for every $x \in \{0,1\}^n$,

$$p_X(x) = \lambda p_Y(x) + (1-\lambda)p_Z(x) \leqslant 2^{-k}$$

Because Y is a $(k-t)$-source, so it suffices that

$$\begin{aligned}
&\lambda 2^{-(k-t)} + (1-\lambda)p_Z(x) \leqslant 2^{-k} \\
\Leftrightarrow\ &(1-\lambda)p_Z(x) \leqslant (1-2^t\lambda)2^{-k} \\
\Leftarrow\ &p_Z(x) \leqslant (1-2^t\lambda)2^{-k}
\end{aligned}$$

Therefore it suffices to set $2^t\lambda < 1$, and so Z exists.

To do step 2, we have

$$\begin{aligned}
&\Pr[\mathrm{Ext}(X, U_d) \in T] - \mu(T) \\
=&\Pr[\mathrm{Ext}(\lambda Y + (1-\lambda)Z, U_d) \in T] - \mu(T) \\
=&\lambda \Pr[\mathrm{Ext}(Y, U_d) \in T] + (1-\lambda)\Pr[\mathrm{Ext}(Z, U_d) \in T] - \mu(T) \\
>&\lambda\Big(\mu(T) + 2^{t+2}\varepsilon\Big) + (1-\lambda)\Big(\mu(T) \pm \varepsilon\Big) - \mu(T) \\
=&\lambda 2^{t+2}\varepsilon \pm (1-\lambda)\varepsilon
\end{aligned}$$

Therefore it suffices to set that $\lambda 2^{t+2}\varepsilon - (1-\lambda)\varepsilon \geqslant \varepsilon$, which is that $\lambda \geqslant 2/(2^{t+2}+1)$. Set $\lambda = 1/2^{t+1}$, note that $2^t\lambda < 1$ is satisfied. Now $p_Z(x) \leqslant (1-2^t\lambda)2^{-k} = (1/2)2^{-k} = 2^{-(k+1)}$. Because $k \leqslant n-1$, $k+1 \leqslant n$, so D exists. The proof is complete. □

How to show 3ε? This is problem 6.8 in Vadhan's monograph.

As we will see the constant 3 is not that important in our analysis, so I will drop it, and pretend that we have a (k, ε)-average min-entropy extractor. Note that the reduction used in this proposition preserves strongness of extractors. Therefore

Corollary 14. *A (k, ε)-strong extractor is a $(k, 3\varepsilon)$-strong average min-entropy extractor.*

Proposition 15. *Let X be a random variable over $\mathcal{U}$ and Y, Z be random variables over $\mathcal{V}$. If (X, Y) and (X, Z) has statistical distance at most ε, then for any deterministic function $P : \mathcal{U} \mapsto \mathcal{V}$:*

$$\left| \Pr_{(u,v)\sim(X,Y)} \left[P(u) = v \right] - \Pr_{(u,v)\sim(X,Z)} \left[P(u) = v \right] \right| \leqslant \varepsilon$$

Proof. Let T be the event of all pairs (u, v) such that $P(u) = v$. □