2-Source Extractors Under Computational Assumptions and Cryptography with Defective Randomness

Yael Tauman Kalai
Microsoft Research New England
yael@microsoft.com

Xin Li
Department of Computer Science
University of Texas at Austin
lixints@cs.utexas.edu

Anup Rao
Institute for Advanced Study
arao@ias.edu

Abstract— We show how to efficiently extract truly random bits from two independent sources of linear min-entropy, under a computational assumption. The assumption we rely on is the existence of an efficiently computable permutation f[1], such that for any source $X \in \{0,1\}^n$ with linear min-entropy, any circuit of size $\mathrm{poly}(n)$ cannot invert $f(X)$ with non-negligible probability.

Under the stronger assumption that $f(X)$ cannot be inverted even by circuits of size $\mathrm{poly}(n^{log n})$ with non-negligible probability, we design a lossless computational network extractor protocol. Namely, we design a protocol for a set of players, each with access to an independent source of linear min-entropy, with the guarantee that at the end of the protocol, each honest player is left with bits that are computationally indistinguishable from being uniform and private. Our protocol succeeds as long as there are at least two honest players. Our results imply that if such one-way permutations exist, and enhanced trapdoor permutations exist, then secure multiparty computation with imperfect randomness *is possible* for any number of players, as long as at least two of them are honest.

We also construct a network extractor protocol for the case where each source has only *polynomially-small* min-entropy (n^{δ} for some constant $\delta > 0$). For this we need at least a constant $u(\delta)$ (which depends on δ) number of honest players, and we need that the one-way permutation is hard to invert even on polynomially small min-entropy sources.

1. Introduction

Randomness is a useful resource for solving many problems in computer science. In some situations, such as in algorithm design, the use of randomness can lead to simpler and more efficient solutions than can be done deterministically. In other situations, such as in cryptography and distributed computing, randomness can be used to give solutions where deterministic solutions are simply impossible. Thus, it is worthwhile to understand what can be done with and without randomness, and to find the minimal assumptions on the randomness under which these randomized solutions are still viable.

In computer science, it is typically assumed that we have access to perfectly uniform bits. Moreover, in cryptography and distributed computing it is usually assumed that protocol participants have access to randomness that is not only truly uniform, but also private. In this work, we seek to weaken both types of assumptions.

One generic way to convert schemes that assume perfect randomness, into schemes which only assume weak randomness, is to design a *randomness extractor*. This is an algorithm that takes as input a single sample drawn from a *weak source* of randomness, and outputs bits that are close to uniform. A necessary condition on the source is that it must have some entropy: we think of the weak source as a bit-string of length n with some entropy[2] k, and call such a source an (n,k)-source. Unfortunately, it can be shown that having high entropy alone is not enough for extraction to be feasible — there is no *single* function that can extract a random bit from *every* $(n, n-1)$-source.

A natural model under which randomness extraction is feasible, is to assume that we have access to two or more *independent* sources, each of which has sufficient entropy. Based on past work [1], [2], [3], [4], [5], [6], [7], we now know how to extract randomness from 2 sources when the entropy in each is at least $.4999n$ [5], from 3 sources if the entropy in each is at least $n^{0.99}$ [6], and from $O(1/\gamma)$ sources if the entropy in each is at least n^{γ} [6], [7]. Although the probabilistic method can be used to show that there is a function that can extract randomness from 2 independent sources even when they

Supported in part by NSF Grant CCF-0634811 and THECB ARP Grant 003658-0113-2007.

Partially supported by NSF grant CCF 0832797. Part of this work was done while this author was visiting Microsoft Research New England.

[1]We can relax the assumption that f is a permutation, and only require that f is not "too lossy", in the sense that for every linear min-entropy source X the source $f(X)$ still has linear min-entropy. See section 2 for details.

[2]We use a standard measure of entropy called *min-entropy*: a distribution has min-entropy k if all strings have probability at most 2^{-k}.

have logarithmic entropy, we know of no *efficient* 2-source extractor for linear entropy. In this work, we make progress towards closing this gap, constructing an efficient 2-source extractor for linear entropy, under a computational assumption.

Although it is impossible to deterministically extract randomness from a single weak source, a sequence of works showed that it is possible to simulate every randomized algorithm with access to a single weak source [8], [1], [9], [10], [11]. However, it is not known how to get an analogous result for protocols in distributed computing or cryptography, where it is essential that each of the processors in the protocol have access to *private* random bits. In fact, the work of Dodis et al. [12] shows that almost all of the classic cryptographic tasks, including encryption, bit commitment, secret sharing, and secure two-party computation (for nontrivial functions), are actually impossible even with a single $(n, .99n)$-source. This leads us to ask the question: what can be done if each of the processors in the protocol has access to an independent weak source?

One hope is to use extractors for independent sources. However, it is not immediately clear that such extractors can be applied, since we need to tolerate adversarial behavior at an unknown subset of the processors, and the extracted bits must remain private even given the information exchanged during the extraction. This question, of whether it is possible to do distributed computing with imperfect randomness, was first considered by Goldwasser et al. [13], who showed how to run a Byzantine agreement protocol when each of the processors only has access to a specific kind of independent defective source of randomness (the sources they considered were more restricted than weak sources). In subsequent work, [14] showed how to build efficient *network extractor protocols*. These are protocols that have the property that if all the honest processors have access to independent (n, k)-sources, then at the end of the protocol *most* of these honest processors are left with private random bits. Under a non-standard cryptographic assumption, [14] designed protocols where *every* honest processor ends up with private randomness, assuming the number of honest processors is larger than $\mathrm{polylog}(n)$, where n is the security parameter and a constant fraction of the players are honest. Such a network extractor can be used to do secure multiparty computation [15], even if each party has access only to a weak source of randomness, using trapdoor permutations and the above assumptions.

The work of [14] left open the question of whether secure multiparty computation with imperfect randomness is possible for a constant number of processors (or even for $O(\log n)$ processors). In this work, we give a positive answer to this question, again under a computational assumption.

The rest of the paper is organized as follows. We state our results in section 2. In section 3 we give an overview of our constructions. In section 4 we give a formal description of our 2-source extractor. The formal description of network extractor protocols are deferred to the full version.

2. Our Results

All of our results in this work are based on the assumption that there exist one-way permutations[3] that are (very) hard to invert, even when the input is sampled from a weak source.

Definition 2.1 (One-Way Permutations for Weak Sources). We call a family of polynomial time computable permutations $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}$ **one-way for k-sources** if for every (n, k) source X, and every circuit $\mathcal{A}$ of size $2^{O(\log^2 n)}$, $\Pr[\mathcal{A}(f_n(X)) = X]$ is negligible.

We note this is equivalent to saying that f is exponentially hard to invert under the uniform distribution.

Proposition 2.2. *If $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}$ is one way for k-sources, then for every circuit $\mathcal{A}$ of size $2^{O(\log^2 n)}$, $\Pr_{x \leftarrow U}[\mathcal{A}(f_n(x)) = x] = \mathrm{negl}(n)2^{-(n-k)}$, and vice versa.*

About our assumption:

- Ideally, we would like to achieve our goals assuming that it is hard to invert these permutations with polynomial sized circuits. This is actually enough for the construction of 2-source extractors, but the error of the output would be $1/\mathrm{poly}(n)$. For our cryptographic applications we need the error to be negligible, thus we need the stronger definition above.
- We actually do not need the function f to be a permutation. All we need is that the function f is not "too lossy." More formally, for any constant $\delta > 0$ we assume the existence of a family of functions $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}_{n \in \mathbb{N}}$ and a constant $\gamma = \gamma(\delta)$ such that for every $(n, \delta n)$-source X, $f(X)$ is a $(n, \gamma n)$-source.

A candidate function is suggested by Goldreich in [16]. We note that we are not the first to assume the existence of one-way functions w.r.t. any weak sources.

[3] As was noted before, we can weaken the assumption that the function is a permutation. See elaboration below.

Canetti [17] conjectured that the discrete-log function is one-way w.r.t. any weak source with sufficient min-entropy. This assumption was later used in [14] to construct network extractors. Assumptions of a similar flavor were also used in several other crypto results, e.g., [18], [19], [20].

Our first result shows that such one-way permutations can be used to obtain 2-source extractors:

Theorem 2.3 (2-Source Extractor). *Let $\delta > 0$ be any constant. Suppose there exists a family of one-way permutations for $(n, 0.3\delta n)$-sources. Then there is a polynomial time computable 2-source extractor for 2 independent $(n, \delta n)$-sources that extracts $n^{\Omega(1)}$ bits that are computationally indistinguishable from uniform.*

Note that the first $O(\log n)$ bits of the output of our extractor are actually guaranteed to be statistically indistinguishable from uniform, since every statistical test on such a small number of bits can be efficiently simulated. Thus, even though we make a computational assumption, our conclusion is of an information theoretic nature. Moreover, our 2-source extractor is a strong extractor in the sense that the output is close to uniform even given one of the inputs X (see Theorem 4.11). Thus we can use the output as a seed and apply a seeded extractor to X, and output $\Omega(n)$ bits that are close to uniform. Specifically, we get the following corollary:

Corollary 2.4 (2-Source Extractor). *Let $\delta > 0$ be any constant. Suppose there exists a family of one-way permutations for $(n, 0.3\delta n)$-sources. Then there is a polynomial time computable 2-source extractor for 2 independent $(n, \delta n)$-sources that extracts $\Omega(n)$ random bits with error $1/\mathrm{poly}(n)$.*

Our next result shows that secure multiparty computation with imperfect randomness is possible for *any* number of processors, as long as there are at least *two* honest processors and each processor has access to an independent source with linear entropy. Secure multiparty computation is also possible if each processor has access to an independent source with polynomially small entropy (n^δ for some constant $\delta > 0$), as long as there are at least $u(\delta)$ honest parties, where here $u = u(\delta)$ is a constant that depends only on δ. We note that this is in contrast with the strong negative results given by Dodis et al. [12], who showed that most of the classic cryptographic tasks are impossible even with a single $(n, .99n)$-source. Thus, in some sense, our results are quite tight.

We obtain these results by constructing computational network extractor protocols (defined in [14]) where *every* honest player ends up with a private random-looking string. A computational network extractor is a protocol in which a set of players each starts with an independent weak random source, and despite the presence of faulty players controlled by a PPT adversary, at the end of the protocol some of the honest players end up with random bits that are computationally indistinguishable from being uniform and private to the adversary who sees the entire transcript of the protocol. We refer the reader to [14] for the formal definition of computational network extractor protocols.

Theorem 2.5 (Network Extractors for Linear Min-Entropy). *Let $\delta > 0$ be any constant. Suppose there exists a family of one-way permutations for $(n, 0.3\delta n)$-sources. Then, there is a network extractor protocol where each player takes as input an independent $(n, \delta n)$-source, and as long as there are at least 2 honest players, all the honest players end up with a string that is computationally indistinguishable from being uniform and private.*

Theorem 2.6 (Network Extractors for Polynomial Min-Entropy). *Let $\delta > 0$ be any constant. Suppose there exists a family of one-way permutations for $(n, 0.3n^\delta)$-sources. Then, there exists a constant $u = u(\delta)$ such that there is a network extractor protocol where each player takes as input an independent (n, n^δ)-source, and as long as there are at least u honest players, all the honest players end up with a string that is computationally indistinguishable from being uniform and private.*

Our network extractor constructions establish that as long as such one-way permutations exist, weak sources are the same as true randomness for the purpose of running multiparty cryptographic protocols, as formalized below.

Corollary 2.7. *Let $\delta > 0$ be any constant. Assume that there exists a family of one-way permutations for $(n, 0.3\delta n)$-sources, and assume that there exists a family of enhanced trapdoor permutations. Then any functionality can be computed securely even if each party has only access to an (independent) $(n, \delta n)$-source, as long as there are at least two honest parties.*

Corollary 2.8. *Let $\delta > 0$ be any constant. Assume that there exists a family of one-way permutations for $(n, 0.3n^\delta)$-sources, and assume that there exists a family of enhanced trapdoor permutations. Then there exists a constant $u = u(\delta)$ such that any functionality can be computed securely even if each party has only access to an (independent) (n, n^δ)-source, as long as there are at least u honest parties.*

3. Overview of the Constructions

In this section we shall be slightly inaccurate, in order to easily convey what we consider to be the key ideas in our work. All of our constructions share the same basic structure. They all consist of a sequence of rounds, where in each round i, a string R_i is generated. We will be able to show that there must exist a "good" round j for which R_j is uniform. However, we do not know which of the rounds is the good round. We shall then output the xor of all the R_i's, and claim that this output is computationally indistinguishable from uniform. To this end, we would like to simply say that R_j is independent of all the other R_i's, which would immediately imply that their xor is indeed uniform. However, we are not able to establish this independence. Instead we prove that under our computational assumption, R_j is computationally indistinguishable from being independent of all the other R_i's, which is enough to get the result we want.

3.1. 2-Source Extractor

We first note that it is well known how to extract randomness from two independent sources, assuming one of them is a *block source*. A block source X is a source that can be partitioned into two parts $X = (X_1, X_2)$ in such a way that X_1 has entropy δn, and X_2 has entropy δn *even conditioned on any fixing of* $X_1 = x_1$. The entropy in such a source is spread out, and it is known how to extract randomness from a block source $X = (X_1, X_2)$ using an independent weak source Y, as long as the blocks X_1, X_2 and the weak source Y each have entropy δn [7], [21], [3].

To make use of block sources, one can show that *every* source X with linear entropy δn, is exponentially close to a convex combination of somewhere block sources. In other words, one can assume without loss of generality that if X is broken into a sufficiently large (say $t = 100/\delta$) number of blocks $X = (X_1, X_2, \ldots, X_t)$, then there is some index $j \in [t]$ for which (X_j, X) is a block source where each block has linear entropy, even conditioned on any fixing of $(X_1, \ldots, X_{j-1})$. Intuitively, this is because each block in the source has at most $\delta n/100$ bits, and so cannot contain all δn bits of entropy.

This alone is not enough to apply extractors for block sources, since the index j above is not known. Still, we might be tempted to try the following approach:

Naive 2-Source Extractor for (X,Y):

1) Let BExt be an extractor for a block source and an independent weak source
2) Partition $x = (x_1, \ldots, x_t)$.
3) For every i, compute $r_i = \mathsf{BExt}(x_i, x, y)$.
4) Output the bitwise xor $r_1 \oplus \cdots \oplus r_t$.

Since it is no loss of generality to assume that there is some index j for which (X_j, X) is a block source, $R_j = \mathsf{BExt}(X_j, X, Y)$ must be uniform. Unfortunately, this algorithm does not work because the rest of the strings R_i are not independent of R_j, and so the output could be a fixed constant even though R_j is uniform.

Our actual construction is a variation of the above construction, where we use computational assumptions to enforce that R_j is independent (in some sense) from the other R_i's. More specifically, we use a one-way permutation for δn-sources to generate independence. To this end we use reconstructive extractors, a concept that originally appeared in the work of Goldreich-Levin [22] and was subsequently formalized and refined in a sequence of works [23], [24], [25], [26], [27], [28]. We do not define this concept here, but the property we use is that the reconstructive extractor RExt satisfies the property that if f is one-way with respect to a weak source X, then

$$(\mathsf{RExt}(X, R), R, f(X)) \approx (\mathsf{Uniform}, R, f(X)),$$

where $\approx$ denotes computational indistinguishability. Thus, if we take the xor of $\mathsf{RExt}(X, R)$ and $\mathsf{RExt}(f(X), R)$, the output still looks uniform.

Given such an object, here is how we use it to build a 2-source extractor.

Our 2-Source Extractor:

1) Let BExt be an extractor for a block source and an independent weak source, and let RExt be a reconstructive (seeded) extractor. Let f be a one-way permutation for weak sources.
2) Partition $x = (x_1, \ldots, x_t)$.
3) For every i, compute $r_i = \mathsf{BExt}(x_i, x, f^i(y))$.
4) Set $z_i = \mathsf{RExt}(f^i(y), r_i)$.
5) Output the bitwise xor $z_1 \oplus \cdots \oplus z_t$.

Here $f^i(y) = f(f(\cdots f(y) \cdots))$, where f is applied i times. The purpose of f here is to break the dependence (at least computationally) between the Z_i's.

3.2. Network Extractor Protocols

We construct two network extractor protocols. One for the case where each player has an independent source with linear entropy δn, and one for the case where each player has an independent source with entropy n^δ. We start with the former.

We first present a protocol where all the honest players *except one* end up with private randomness. Note that if we knew of one player j that is honest, then the protocol would be simple: Player j will simply

reveal his source, and all the other players would apply the 2-source extractor TExt (presented above) to this source and their own sources. The fact that TExt is a strong extractor immediately implies that all the honest players except player j, would end up with private randomness. However, we don't know of any honest player. Thus it is tempting to try the following approach.

Naive Network Extractor for Linear Min-Entropy:

1) The protocol proceeds in p rounds, where p is the number of players. In round i:
   a) Player i sends x_i to all other players.
   b) Each player ℓ computes $r^i_\ell = \mathsf{TExt}(x_i, x_\ell)$.
2) Each player i outputs the bitwise xor $r^1_i \oplus \cdots \oplus r^p_i$.

Let j denote the first honest player. Then, for every player i different than j, r^j_i is uniform. Despite this, the output of player i may not be random, and may even be a fixed constant. As before, the problem is that the random variables $r^1_i, \ldots, r^p_i$ are not independent, and a malicious adversary can actually force the output to be a fixed constant. As in our 2-source extractor construction, the idea is to get around this problem by using computational assumptions. To this end, each player ℓ, instead of using the same source X_ℓ in each round, will use the source $f^i(X_\ell)$ in round i. However, for this approach to work we need the guarantee that

$$(\mathsf{TExt}(X_i, X_\ell), X_i, f(X_\ell)) \approx (\mathsf{Uniform}, X_i, f(X_\ell)).$$

Our extractor TExt does not satisfy this, but luckily our extractor does satisfy the following (similar) guarantee:

$$(\mathsf{TExt}(X_i, X_j), X_i, f^{t+1}(X_j)) \approx (\mathsf{Uniform}, X_i, f^{t+1}(X_j)),$$

where t is a constant that depends on δ.

So, instead we consider the following network extractor protocol, which has the guarantee that all the honest players, except for the first one, end up with private random-looking strings.

Lossy Network Extractor for Linear Min-Entropy:

1) Let $g = f^{t+1}$. The protocol proceeds in p rounds, where p is the number of players. In round i:
   a) Player i sends $g^i(x_i)$ to all other players.
   b) Each player ℓ computes $r^i_\ell = \mathsf{TExt}(g^i(x_i), g^i(x_\ell))$.
2) Each player i outputs the bitwise xor $r^1_i \oplus \cdots \oplus r^p_i$.

Now we can prove that all the honest players, except player j (who is the first honest player), end up with private random-looking strings. The analysis proceeds in three steps.

1) We first fix all sources sent before round j, and we fix $\{r^i_\ell\}_{\ell \in [p], i<j}$, which were all computed before round j. We claim that even conditioned on all these fixings, the sources are still independent, and with high probability they all have "enough" entropy left.
2) Next, we claim that the strings $\{r^j_\ell\}$ of all the honest players $\ell \neq j$, are independent and close to uniform.
3) Finally, we claim that the rest of the r^i_ℓ for $i > j$ are (computationally) independent of $\{r^j_\ell\}$, which implies that the output of all the honest players, except player j, are computationally indistinguishable from random. For this we use the fact that for any two independent variables Y_i and Y_j with "sufficient" entropy,

$$(\mathsf{TExt}(Y_i, Y_j), Y_i, g(Y_j)) \approx (\mathsf{Uniform}, Y_i, g(Y_j)). \quad (1)$$

Note that in the protocol above, player j, the first honest player, may not end up with private randomness. To fix this, we add another phase. So, the protocol consists of two phases. In the first phase, the players run the (lossy) protocol presented above. In the second phase, all the honest players use their (supposedly) random string, generated in the first phase, to run a coin flipping protocol and generate a public random-looking seed V. Recall that we assume that there are at least two honest players, therefore there is at least one honest player besides player j. Thus, V is indeed random-looking. Finally, each player i will extract randomness from his own source X_i using the seed V[4].

This approach would indeed work if there were at least *three* honest players, since in that case we could argue that V is random-looking and is *independent* of each of the sources X_i. Thus, we could use it to extract (private) randomness from each source.

However, if there are only *two* honest players then this approach does not seem to work, since in this case we cannot argue that V is independent of all the sources. Indeed if there is a single honest player ℓ besides player j (the first honest player) then it may be the case that V depends on the source of player ℓ. This is the case since player ℓ maybe the only player who used "good" randomness for the coin-flipping protocol. As before, we get around this dependence by using reconstructive extractors.

Final Network Extractor for Linear Min-Entropy:

[4]Note all the sources still have high entropy left

1) **Phase 1.** This phase proceeds in p rounds, where p is the number of players. In round i:
   a) Player i sends $g^{2i}(x_i)$ to all other players.
   b) Each player ℓ computes $r^i_\ell = \mathsf{TExt}(g^{2i}(x_i), g^{2i}(x_\ell))$.

   At the end of this phase, each player i outputs $r_i = r^1_i \oplus \cdots \oplus r^p_i$.
2) **Phase 2.** Each player i uses its (supposedly) random string r_i, generated in Phase 1, to run a secure coin flipping protocol and generate v. Finally, each player i outputs $z_i = \mathsf{RExt}(g^{(2i-1)}(x_i), v)$.

We claim that even if there are only two honest players ℓ and j, player ℓ (and player j) still end up with private randomness. The reason why player j ends up with private randomness is quite straightforward: It follows from the fact that V is random and is independent of X_j. The reason why player ℓ ends up with private randomness is more involved. On a very high-level, the proof follows the following two steps.

1) First, we use the property of our 2-source extractor TExt (Equation 1), to argue that R_ℓ looks random, even conditioned on all the sources of all the other players *and* conditioned on $g^{2j+1}(X_\ell)$. We stress that the above statement is a computational statement, and cannot hold information theoretically, since R_ℓ is uniquely determined by $g^{2j+1}(X_\ell)$ and all the other sources.
2) Then, we use the reconstructive property of RExt, together with the conclusion above, to argue that indeed player ℓ ends up with private randomness.

Our Network Extractor Protocol for Polynomially-Small Min-Entropy.: Finally, we construct a network extractor for the case where each player has an independent (n, n^δ)-source (for some constant $\delta > 0$). This protocol is very similar (in spirit) to the protocol above, though is significantly more complicated. The reason for the complication is that in this setting we cannot use a 2-source extractor, since we do not know of such an extractor for the case where the entropy is polynomially small. Thus, instead, we use a multi-source extractor IExt that extracts randomness from $u = u(\delta)$ independent (n, n^δ)-sources [6], [7].

As before, the protocol proceeds in two phases. In the first phase, all the honest players except u of them, end up with private randomness. In the second phase, all the players run a secure coin-tossing protocol using the (supposedly) random string they obtained in the first phase, to get a public random seed V. Then, each player i will use a reconstructive extractor to extract randomness from his source (more precisely, from a function of his source), using the random seed V. To argue that all honest players end up with private randomness we need to assume that there are at least $u + 1$ honest players.

4. Two-Source Extractor

In this section we present our construction of a two-source extractor under computational assumptions. We assume one of the sources has linear min-entropy, while the other one can have smaller min-entropy.

4.1. Ingredients

First we need the following definition.

Definition 4.1. A source $X = (X_1, \cdots, X_t)$ is $t \times r$ **somewhere-random** if each X_i takes values in $\{0,1\}^r$ and there is an i such that X_i is uniform. It is $(t \times r)$ **k-somewhere-random** if each X_i takes values in $\{0,1\}^r$ and there is an i such that X_i has min-entropy k.

We need to use the following objects constructed in previous works:

Theorem 4.2 ([29]). *For every constant $0 < \alpha < 1$ there exists a function*

$$\mathrm{Zuc} : \{0,1\}^n \to \{0,1\}^{c \times \ell}$$

where $c = \mathrm{poly}(1/\alpha)$ is a constant and $\ell = \frac{n}{\mathrm{poly}(1/\alpha)}$, such that for every $(n, \alpha n)$-source X, $\mathrm{Zuc}(X)$ is $2^{-\Omega(n)}$-close to a convex combination of $(c \times \ell)0.9\ell$-somewhere random source.

Theorem 4.3 ([7]). *There exist constants $\alpha, \beta < 1$ such that for every $n, k(n)$ with $k > \log^{10} n$, and constant $0 < \gamma < 1/2$, there is a polynomial time computable function $\mathsf{BasicExt} : \{0,1\}^n \times \{0,1\}^{k^{\gamma+1}} \to \{0,1\}^m$ s.t. if X is an (n,k) source and Y is a $(k^\gamma \times k)$ $(k - k^\beta)$ somewhere random source,*

$$|(Y, \mathsf{BasicExt}(X,Y)) - (Y, U_m)| < \epsilon$$

and

$$|(X, \mathsf{BasicExt}(X,Y)) - (X, U_m)| < \epsilon$$

where U_m is independent of X, Y, $m = k - k^{\Omega(1)}$ and $\epsilon = 2^{-k^\alpha}$.

Theorem 4.4 ([4]). *For any n_1, n_2, k_1, k_2, m and any $0 < \delta < 1/2$ with*

- $n_1 \geq 6 \log n_1 + 2 \log n_2$
- $k_1 \geq (0.5 + \delta) n_1 + 3 \log n_1 + \log n_2$
- $k_2 \geq 5 \log(n_1 - k_1)$
- $m \leq \delta \min[n_1/8, k_2/40] - 1$

There is a polynomial time computable strong 2-source extractor

$$\mathsf{Raz} : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^m$$

for min-entropy k_1, k_2 with error $2^{-1.5m}$.

The last ingredient we need is a family of extractors known as reconstructive extractors. Following [28], we define these extractors:

Definition 4.5. A $(n, t, m, d, a, \epsilon, \delta)$-reconstructive extractor is a triple of functions:

- A polynomial time computable extractor function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$
- An advice function $\mathsf{A} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^a$
- A $\mathrm{poly}(n, 1/\epsilon)$ time randomized oracle reconstruction procedure $\mathcal{R} : \{0,1\}^a \to \{0,1\}^n$

That satisfy the property that for every $x \in \{0,1\}^n$ and $\mathcal{D} : \{0,1\}^m \to \{0,1\}$ for which

$$|\Pr[\mathcal{D}(\mathsf{Ext}(x, U_t), U_t) = 1] - \Pr[\mathcal{D}(U_m, U_t) = 1]| \geq \epsilon,$$

we have

$$\Pr_w[\mathcal{R}^{\mathcal{D}}(A(x, w)) = x] \geq \delta.$$

We have the following theorem, which follows from the discussion in section 6 of [28]:

Theorem 4.6 ([28]). *There is a constant $\beta > 0$ such that for every n, a, ϵ with $a = n^{\Omega(1)}$, there exists $(n, t = O(\log(n/\epsilon))), m = n^\beta, d = O(\log(n/\epsilon)), a, \epsilon, 1/2)$ reconstructive extractor.*

An almost immediate consequence of this theorem is the following theorem, whose proof we omit here due to lack of space:

Theorem 4.7. *For every n, k, ϵ with $k = n^{\Omega(1)}$ and $\epsilon = \frac{1}{\mathrm{poly}(n^{\log n})}$, there is a polynomial time computable function $\mathsf{REXt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ such that $d = O(\log(n/\epsilon))$, $m = k^{\Omega(1)}$ and if f is a one way function for $k/3$ sources and X is an (n, k) source, then for every distinguisher $\mathcal{A}$ of size $\mathrm{poly}(n^{\log n})$,*

$$|\Pr[\mathcal{A}(f(X), \mathsf{REXt}(X, U_d), U_d) = 1] - \Pr[\mathcal{A}(f(X), U_m, U_d) = 1]| = \mathrm{negl}(n)$$

We also need the following two lemmas, with both proofs omitted here and deferred to the full version:

Lemma 4.8. *Let X, Y be two independent random variables on $\{0,1\}^n$ and Z be a random variable on $\{0,1\}^m$ that is independent of (X, Y). Let $f : \{0,1\}^n \to \{0,1\}^d$ and $g : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be two deterministic functions. Let $R = f(X)$. If there exists a non-uniform adversary $\mathcal{A}$ that distinguishes between $(g(Y, R), R, X, Y)$ and (Z, R, X, Y) with probability ϵ, then there exists another non-uniform adversary $\mathcal{B}$ of size $2^d \cdot n \cdot Size(\mathcal{A})$ that distinguishes between $(g(Y, R), R, Y)$ and (Z, R, Y) with probability at least ϵ.*

Lemma 4.9. *Let X be an $(n, \alpha n)$ source for some constant $0 < \alpha < 1$. Let $t = \frac{4}{\alpha}$. Divide X evenly into t blocks $X = X_1 \circ X_2 \circ \cdots \circ X_t$. Then X is $2^{-\Omega(n)}$-close to being a convex combination of sources $\{X^j\}_{j \in J}$ such that for every j there exists $g \in [t]$ for which*

- *$X_1^j, \ldots, X_{g-1}^j$ is fixed.*
- *$H_\infty(X_g^j) \geq \frac{\alpha^2}{6} n$.*
- *$\forall x \in \mathsf{supp}(X_g^j), H_\infty(X | X_g^j = x) \geq \frac{\alpha^2}{6} n$.*

We use the following standard lemma about conditional min-entropy. (For a proof, we refer the reader to the proof of Lemma 5 in [30]).

Lemma 4.10. *Let X and Y be random variables and let $\mathcal{Y}$ denote the range of Y. Then for all $\epsilon > 0$*

$$\Pr_Y\left[H_\infty(X|Y = y) \geq H_\infty(X) - \log\left(\frac{|\mathcal{Y}|}{\epsilon}\right)\right] \geq 1 - \epsilon$$

4.2. Construction

Let X and Y be two independent weak sources, where X is an $(n, \alpha n)$-source for some constant $\alpha > 0$; and Y is an (n, k) source, where $k \geq n^{\Omega(1)}$.

We first define an extractor for a block source and an independent weak source, as in [21], [7]:

$$\mathsf{BExt} : \{0,1\}^{n_1} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$$

- **Ingredients**:
  - $\mathrm{Zuc} : \{0,1\}^{\alpha n/2} \to \{0,1\}^{c \times \ell}$ the function as in Theorem 4.2, where $c = \mathrm{poly}(1/\alpha)$ and $\ell = \frac{n}{\mathrm{poly}(1/\alpha)}$.
  - $\mathsf{Raz} : \{0,1\}^\ell \times \{0,1\}^n \to \{0,1\}^s$ the strong 2-source extractor as in Theorem 4.4, where one source is a $(\ell, 0.9\ell)$-source and the other is an $(n, 0.9k)$-source (where k is the min-entropy of Y).
  - $\mathsf{BasicExt} : \{0,1\}^n \times \{0,1\}^{s^\gamma \times s} \to \{0,1\}^d$ the extractor as in Theorem 4.3, where $s = \mathrm{polylog}(n)$.
- $\mathsf{BExt}(x_1, x, y)$:
  1) Compute $v = \mathrm{Zuc}(x_1)$.
  2) Apply the strong 2-source extractor Raz to each row of v and y. Denote the resulting matrix by sr. Note each row of sr is of length $s = \mathrm{polylog}(n)$.

3) Output $\mathsf{BasicExt}(x, sr)$

Next we use this block source extractor to construct

$$\mathsf{TExt} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$$

- **Ingredients**:
  - $\mathsf{RExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ an extractor for entropy $0.9k$ as in Theorem 4.7, where $\epsilon = \frac{1}{\mathrm{poly}(n^{\log n})}$ and $d = O(\log(n/\epsilon)) = \mathrm{polylog}(n)$.
  - f, a one way permutation for $0.3k$-sources
- $\mathsf{TExt}(x, y)$:
  Partition x, into $t = \frac{4}{\alpha}$ equally sized parts[5] $x = (x_1, \ldots, x_t)$.
  For every $i = 1, \ldots, t$,
  1) Compute $r_i = \mathsf{BExt}(x_i, x, f^{(i)}(y))$. Let the output r_i be of length d where $d = \mathrm{polylog}(n)$ is the seed length for RExt.
  2) Compute $z_i = \mathsf{RExt}(f^{(i)}(y), r_i)$. Let the output z_i be of length $m = k^{\Omega(1)}$.

  Output the bitwise xor $z = \oplus_{i=1}^{t} z_i$.

Theorem 4.11. *Fix a constant $\alpha > 0$ and parameters $t = \frac{4}{\alpha}$ and $k \geq n^{\Omega(1)}$. Assume that there exists a one way permutation $f : \{0,1\}^n \to \{0,1\}^n$ for $0.3k$-sources. Then $\mathsf{TExt} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ described above is a computational 2-source extractor such that for any $(n, \alpha n)$-source X, and any (n,k)-source Y that is independent of X,*

$$(\mathsf{TExt}(X,Y), X) \approx (U_m, X)$$

Remark 4.12. Rather than proving Theorem 4.11, we prove the following (stronger) statement:

$$(\mathsf{TExt}(X,Y), X, h(X), f^{(t+1)}(Y)) \approx (U_m, X, h(X), f^{(t+1)}(Y)), \quad (2)$$

where h is any deterministic function (not necessarily computable in polynomial time) on $\{0,1\}^n$. The reason is that we need this stronger variant for our network extractor protocol.

To prove Equation (2) we first prove the following lemma(in the analysis we use capital letters to denote the corresponding strings viewed as random variables).

Lemma 4.13. *Divide X into $X = (X_1, \ldots, X_t)$ as in the construction of TExt, and let $Z = \mathsf{TExt}(X,Y)$. Suppose there exists $g \in [t]$ such that*

- *$X_1, \ldots, X_{g-1}$ are fixed.*
- *$H_\infty(X_g) \geq \frac{\alpha^2}{6} n$.*
- *$\forall x \in \mathsf{supp}(X_g^j), H_\infty(X | X_g = x) \geq \frac{\alpha^2}{6} n$.*

Then

$$(Z, X, h(X), f^{(t+1)}(Y)) \approx (U_m, X, h(X), f^{(t+1)}(Y))$$

Proof of Lemma 4.13.: Let sr_i denote the string sr computed in $\mathsf{BExt}(x_i, x, f^{(i)}(Y))$. Fix

$$(sr_1, \ldots, sr_{g-1}) \leftarrow (SR_1, \ldots, SR_{g-1})$$
$$(r_1, \ldots, r_{g-1}) \leftarrow (R_1, \ldots, R_{g-1})$$

and

$$(z_1, \ldots, z_{g-1}) \leftarrow (Z_1, \ldots, Z_{g-1}).$$

For any random variable Z, we denote by Z' the random variable Z conditioned on these fixings.

Let TYPICAL denote the event that conditioned on these fixings, the following conditions are satisfied:

- X' and Y' are independent
- $H_\infty(Y') \geq k - k^{\Omega(1)}$
- $H_\infty(X'_g) \geq \frac{\alpha^2}{8} n$
- With probability $1 - \mathrm{negl}(n)$ over $(x'_g \leftarrow X'_g)$, $H_\infty(X' | X'_g = x'_g) \geq \frac{\alpha^2}{8} n$

Claim 4.14.

$$\Pr[\mathrm{TYPICAL}] = 1 - \mathrm{negl}(n).$$

Proof of Claim 4.14[Sketch]: Since $X_1, \ldots, X_{g-1}$ are fixed, $(SR_1, \ldots, SR_{g-1})$ is a deterministic function of Y. Thus, conditioned on $(sr_1, \ldots, sr_{g-1}) \leftarrow (SR_1, \ldots, SR_{g-1})$, X and Y are still independent. Next, we further condition on $(r_1, \ldots, r_{g-1}) \leftarrow (R_1, \ldots, R_{g-1})$. Note that now $(R_1, \ldots, R_{g-1})$ is a deterministic function of X and thus conditioned on this fixing, X and Y are still independent. Finally, we further condition on $(z_1, \ldots, z_{g-1}) \leftarrow (Z_1, \ldots, Z_{g-1})$. Note that now $(Z_1, \ldots, Z_{g-1})$ is a deterministic function of Y. Thus, conditioned on this fixing, X and Y are still independent.

In each step above, conditioning on the random variables reduces the entropies of X_g, X and Y. However, note that the size of $(sr_1, \ldots, sr_{g-1})$ is bounded by $tcs = \mathrm{polylog}(n)$, the size of $(r_1, \ldots, r_{g-1})$ is bounded by $t|r_i| = \mathrm{polylog}(n)$ and the size of $(z_1, \ldots, z_{g-1})$ is bounded by $t|z_i| = k^{\Omega(1)}$. Thus by Lemma 4.10, with probability $1 - \mathrm{negl}(n)$ over these fixings, the entropies of X_g, X and Y still satisfy the conditions. ∎

Now fix

$$(x'_g, sr'_g) \leftarrow (X'_g, SR'_g).$$

For every random variable Z', denote by

$$Z'' = Z' | (X'_g = x'_g, SR'_g = sr'_g).$$

Let TYPICAL2 denote the event that conditioned on all the above fixings, the following holds:

[5]For the sake of simplicity, we assume t is an integer.

- X'' and Y'' are independent, R''_g is a deterministic function of X''
- $H_\infty(Y'') \geq 0.9k$
- $|(R''_g, Y'') - (U_d, Y'')| = \text{negl}(n)$

Claim 4.15. *If* TYPICAL *holds, then*

$$\Pr[\text{TYPICAL2}] = 1 - \text{negl}(n)$$

Proof of Claim 4.15.: First note that when TYPICAL holds, X' and Y' are independent, and X'_g has min-entropy rate $\geq \frac{\alpha}{2}$. Therefore by Theorem 4.2 M'_g is $2^{-\Omega(n)}$-close to a convex combination of $(c \times \ell)0.9\ell$-somewhere random source. Theorem 4.4 implies that there exists a (convex combination of) somewhere-random source SR with c rows, each row of length s, such that

$$|(M'_g, SR'_g) - (M'_g, SR)| = \text{negl}(n).$$

A standard averaging argument shows that with probability $1 - \text{negl}(n)$ over the fixing of M'_g, we still have

$$|SR'_g - SR| = \text{negl}(n).$$

Moreover, X'_g is a deterministic function of X', thus conditioned on the fixing of X'_g (and thus M'_g), X' and Y' are still independent. Note once conditioned on M'_g, SR'_g is a deterministic function of Y', and is independent of X'. Also, with probability $1 - \text{negl}(n)$ over the fixing of X'_g, $H_\infty(X') \geq \frac{\alpha^2}{8}n$. The probability that both these two events happen is $1 - \text{negl}(n)$, and when this happens, Theorem 4.3 implies that

$$|(SR'_g, R_g) - (SR'_g, U_d)| < 2^{-n^{\Omega(1)}} + \text{negl}(n) = \text{negl}(n).$$

Since this happens with probability $1 - \text{negl}(n)$, we actually have that

$$|(SR'_g, R_g) - (SR'_g, U_d)| = \text{negl}(n).$$

Again, by a standard averaging argument, with probability $1 - \text{negl}(n)$ over the fixing of SR'_g, we still have

$$|R_g - U_d| = \text{negl}(n).$$

Note since SR'_g is a deterministic function of Y', conditioning on it still leaves X' and Y' independent. Moreover, since the size of sr_g is small, by Lemma 4.10 with probability $1 - \text{negl}(n)$ over the fixings of SR'_g, $H_\infty(Y') \geq k - k^{\Omega(1)} - \text{polylog}(n) > 0.9k$. Finally, conditioned on the fixing of SR'_g, R_g is a deterministic function of X', and is thus independent of Y'. Note $|R_g - U_d| = \text{negl}(n)$, therefore

$$|(R_g, Y'') - (U_d, Y'')| = \text{negl}(n)$$

The probability that all of the above are satisfied is $1 - \text{negl}(n)$. ∎

Next we prove the following claim.

Claim 4.16. *If both* TYPICAL *and* TYPICAL2 *hold, then*

$$(\oplus_{i=g}^{t} Z''_i, X'', h(X''), f^{t+1}(Y'')) \approx (U_m, X'', h(X''), f^{t+1}(Y''))$$

Proof of Claim 4.16: Assume for the sake of contradiction that there exists a non-uniform PPT adversary $\mathcal{A}_1$ and a polynomial q such that for infinitely many n's

$$|\Pr[\mathcal{A}_1(\oplus_{i=g}^{t} Z''_i, X'', h(X''), f^{t+1}(Y'')) = 1] - \Pr[\mathcal{A}_1(U_m, X'', h(X''), f^{t+1}(Y'')) = 1]| \geq \frac{1}{q(n)}.$$

Since $Z''_{g+1}, \cdots, Z''_t$ and $f^{t+1}(Y'')$ can be computed from $(X'', f^{g+1}(Y''))$ in polynomial time, there exists another non-uniform PPT adversary $\mathcal{A}_2$ such that

$$|\Pr[\mathcal{A}_2(Z''_g, R''_g, X'', h(X''), f^{g+1}(Y'')) = 1] - \Pr[\mathcal{A}_2(U_m, R''_g, X'', h(X''), f^{g+1}(Y'')) = 1]| \geq \frac{1}{q(n)}.$$

Recall that

$$Z''_g = \mathsf{REXt}(f^{(g)}(Y''), R''_g)$$

and $f^{(g)}(Y'')$ is a deterministic function of $f^{(g+1)}(Y'')$(though not efficiently computable). Thus Z''_g is a deterministic function of $f^{(g+1)}(Y'')$ and R''_g. Next note that R''_g is a deterministic function of X'', and X'', Y'', U_m are independent. Thus Lemma 4.8 implies that there exists another non-uniform adversary $\mathcal{A}_3$ that runs in time $2^d \cdot n \cdot \text{poly}(n) = \text{poly}(n, \frac{1}{\epsilon})$ such that

$$|\Pr[\mathcal{A}_3(Z''_g, R''_g, f^{g+1}(Y'')) = 1] - \Pr[\mathcal{A}_3(U_m, R''_g, f^{g+1}(Y'')) = 1]| \geq \frac{1}{q(n)}.$$

Note that the fact that TYPICAL2 holds implies that $|(R_g, Y'') - (U_d, Y'')| = \text{negl}(n)$. Thus

$$|\Pr[\mathcal{A}_3(\mathsf{REXt}(f^{(g)}(Y''), U_d), U_d, f^{g+1}(Y'')) = 1] - \Pr[\mathcal{A}_3(U_m, U_d, f^{g+1}(Y'')) = 1]| > \frac{1}{2q(n)},$$

Note Y'' has min-entropy $0.9k$ thus $f^{(g)}(Y'')$ also has min-entropy $0.9k$. Note f is one way for $0.3k$-sources and $\mathcal{A}_3$ runs in time $\text{poly}(n, \frac{1}{\epsilon}) = \text{poly}(n^{\log n})$, thus this contradicts Theorem 4.7. ∎

Note the event that both TYPICAL and TYPICAL2 hold happens with probability $1 - \mathrm{negl}(n)$. Thus

$$(\oplus_{i=g}^{t} Z_i, \{Z_i\}_{i\in[g-1]}, X, h(X), f^{t+1}(Y)) \approx$$
$$(U_m, \{Z_i\}_{i\in[g-1]}, X, h(X), f^{t+1}(Y)).$$

Note that $Z = \oplus_{i=1}^{t} Z_i$, thus

$$(Z, X, h(X), f^{t+1}(Y)) \approx (U_m, X, h(X), f^{t+1}(Y)).$$

■

Proof of Theorem 4.11: Since we divide X into $t = \frac{4}{\alpha}$ blocks, Lemma 4.9 says that X is $2^{-n^{\Omega(1)}}$-close to being a convex combination of $\{X^j\}_{j\in J}$ such that for every $j \in J$, X^j satisfies the conditions in Lemma 4.13. For every $j \in J$, let $Z^j = \mathsf{TExt}(X^j, Y)$, then

$$(Z^j, X^j, h(X^j), f^{(t+1)}(Y)) \approx (U_m, X^j, h(X^j), f^{(t+1)}(Y)).$$

Thus, by Lemma 4.9 and the property of convex combination, the theorem holds. ■

ACKNOWLEDGEMENTS

We thank David Zuckerman for many helpful discussions. We also thank Boaz Barak, Iftach Haitner, Amit Sahai, Chris Umans for useful discussions and the anonymous reviewers for valuable comments.

REFERENCES

[1] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 230–261, 1988.

[2] B. Barak, R. Impagliazzo, and A. Wigderson, "Extracting randomness using few independent sources," in *Proc. of 45th FOCS*, 2004, pp. 384–393.

[3] B. Barak, G. Kindler, R. Shaltiel, B. Sudakov, and A. Wigderson, "Simulating independence: New constructions of condensers, Ramsey graphs, dispersers, and extractors," in *Proc. of 37th STOC*, 2005, pp. 1–10.

[4] R. Raz, "Extractors with weak random seeds," in *Proc. of 37th STOC*, 2005, pp. 11–20.

[5] J. Bourgain, "More on the sum-product phenomenon in prime fields and its applications," *International Journal of Number Theory*, vol. 1, pp. 1–32, 2005.

[6] A. Rao, "Extractors for a constant number of polynomially small min-entropy independent sources," in *Proc. of 38th STOC*, 2006.

[7] B. Barak, A. Rao, R. Shaltiel, and A. Wigderson, "2 source dispersers for $n^{o(1)}$ entropy and Ramsey graphs beating the Frankl-Wilson construction," in *Proc. of 38th STOC*.

[8] U. V. Vazirani and V. V. Vazirani, "Random polynomial time is equal to slightly-random polynomial time," in *Proc. of 26th FOCS*, 1985, pp. 417–428.

[9] D. Zuckerman, "Simulating BPP using a general weak random source," *Algorithmica*, vol. 16, pp. 367–391, 1996.

[10] M. Saks, A. Srinivasan, and S. Zhou, "Explicit OR-dispersers with polylog degree," *Journal of the ACM*, vol. 45, pp. 123–154, 1998.

[11] A. E. Andreev, A. E. F. Clementi, J. D. P. Rolim, and L. Trevisan, "Weak random sources, hitting sets, and BPP simulations," *SIAM Journal on Computing*, vol. 28, no. 6, pp. 2103–2116, 1999.

[12] Y. Dodis, S. J. Ong, M. Prabhakaran, and A. Sahai, "On the (im)possibility of cryptography with imperfect randomness," in *FOCS04*, 2004, pp. 196–205.

[13] S. Goldwasser, M. Sudan, and V. Vaikuntanathan, "Distributed computing with imperfect randomness," in *DISC 2005*, 2005.

[14] Y. T. Kalai, X. Li, A. Rao, and D. Zuckerman, "Network extractor protocols," in *Proc. of 49th FOCS*, 2008.

[15] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. of 19th STOC*, 1987.

[16] O. Goldreich, "A candidate counterexample to the easy cylinders conjecture."

[17] R. Canetti, "Towards realizing random oracles: Hash functions that hide all partial information," in *Advances in Cryptology — CRYPTO '97, 17th Annual International Cryptology Conference, Proceedings*, 1997.

[18] H. Wee, "On obfuscating point functions," in *Proc. of 37th STOC*, 2005.

[19] S. Dziembowski and K. Pietrzak, "Leakage-resilient cryptography," in *Proc. of 49th FOCS*, 2008.

[20] K. Pietrzak, "A leakage-resilient mode of operation," in *Advances in Cryptology — EUROCRYPT 2009*, 2009.

[21] A. Rao and D. Zuckerman, "Extractors for 3 uneven length sources," in *Random 2008*, 2008.

[22] O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions," in *Proc. of 21st STOC*, 1989, pp. 25–32.

[23] N. Nisan and A. Wigderson, "Hardness vs randomness," *Journal of Computer and System Sciences*, vol. 49, no. 2, pp. 149–167, Oct. 1994.

[24] L. Trevisan, "Extractors and pseudorandom generators," *Journal of the ACM*, pp. 860–879, 2001.

[25] A. Ta-Shma and D. Zuckerman, "Extractor codes," *IEEE Transactions on Information Theory*, vol. 50, 2004.

[26] A. Ta-Shma, C. Umans, and D. Zuckerman, "Loss-less condensers, unbalanced expanders, and extractors," in *Proc. of 33rd STOC*, 2001, pp. 143–152.

[27] R. Shaltiel and C. Umans, "Simple extractors for all min-entropies and a new pseudorandom generator," *Journal of the ACM*, vol. 52, pp. 172–216, 2005.

[28] C. Umans, "Reconstructive dispersers and hitting set generators," in *APPROX-RANDOM*, vol. 3624, 2005, pp. 460–471.

[29] D. Zuckerman, "Linear degree extractors and the inapproximability of max clique and chromatic number," in *Theory of Computing*, 2007, pp. 103–128.

[30] U. Maurer and S. Wolf, "Privacy amplification secure against active adversaries," in *Advances in Cryptology — CRYPTO '97*, vol. 1294, Aug. 1997, pp. 307–321.