Affine Extractors for Almost Logarithmic Entropy

Eshan Chattopadhyay
Department of Computer Science
Cornell University
Ithaca, NY, USA
eshan@cs.cornell.edu

Jesse Goodman
Department of Computer Science
Cornell University
Ithaca, NY, USA
jpmgoodman@cs.cornell.edu

Jyun-Jie Liao
Department of Computer Science
Cornell University
Ithaca, NY, USA
jjliao@cs.cornell.edu

2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS) | 978-1-6654-2055-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/FOCS52979.2021.00067

***Abstract*—We give an explicit construction of an affine extractor (over $\mathbb{F}_2$) that works for affine sources on n bits with min-entropy $k \geq \log n \cdot (\log\log n)^{1+o(1)}$. This improves prior work of Li (FOCS'16) that requires min-entropy at least $\mathrm{poly}(\log n)$.**

Our construction is based on the framework of using correlation breakers and resilient functions, a paradigm that was also used by Li. On a high level, the key sources of our improvement are based on the following new ingredients: (i) A new construction of an affine somewhere random extractor, that we use in a crucial step instead of a linear seeded extractor (for which optimal constructions are not known) that was used by Li. (ii) A near optimal construction of a correlation breaker for linearly correlated sources. The construction of our correlation breaker takes inspiration from an exciting line of recent work that constructs two-source extractors for near logarithmic min-entropy.

I. Introduction

The area of randomness extraction is concerned with producing truly random bits from defective sources of randomness. The motivation for this area stems from the fact that naturally occurring sources of randomness are typically defective, but applications in areas such as cryptography and distributed computing crucially require access to truly uniform bits.

A lot of research has gone into modeling weak sources of randomness, starting with the early work of von Neumann [1] who considered the problem of extracting randomness from a stream of independent, biased bits. By now, the standard way of measuring the quality of a weak source $\mathbf{X}$ is using the notion of *min-entropy* defined as $H_\infty(\mathbf{X}) = \min_x \log(1/\Pr[\mathbf{X} = x])$. Note that for a distribution $\mathbf{X}$ on $\{0,1\}^n$, we have $0 \leq H_\infty(\mathbf{X}) \leq n$. We define an (n,k)-source to be a distribution on n bits with min-entropy at least k.

We are now ready to define the notion of an extractor for a class of sources. We measure the quality of the output of the extractor using the notion of statistical distance between distributions D_1 and D_2 (on some universe Ω) defined as $|D_1 - D_2| := \frac{1}{2}\sum_{x\in\Omega} |D_1(x) - D_2(x)|$.

Definition I.1 (Deterministic extractors)**.** *An extractor $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ with error ϵ for a class of sources $\mathcal{X}$ satisfies the property that for any $\mathbf{X} \in \mathcal{X}$, we have $|\mathrm{Ext}(\mathbf{X}) - \mathbf{U}_m| \leq \epsilon$.*

A folklore result from the 80's rules out the possibility of an extractor (even with a single bit output) for the class of (n,k)-sources, for any $k \leq n-1$. Given this impossibility result, research on random extraction over the last four decades can be broadly classified into two directions: (i) Seeded extraction: the assumption in this setting is that the extractor has access to an independent seed that can be used to extract randomness from the source [2]. An impressive line of work has led to efficient constructions of seeded extractors with optimal parameters [3], [4], [5]. (See also the excellent survey by Shaltiel [6] for earlier results on seeded extraction.) (ii) Seedless (or deterministic) extraction: here one makes some additional assumption on the source $\mathbf{X}$ that enables the possibility of randomness extraction. Some examples include the class of bit-fixing sources [7], sources sampled by a computationally bounded algorithms [8], [9], and sources that comprise of multiple independent sources [10], [11].

In this paper we focus on the setting of deterministic extraction, and in particular, we study the problem of randomness extraction from *affine sources* defined as follows.

Definition I.2 (Affine sources)**.** *Fix a finite field $\mathbb{F}$ of size q, and parameters n, k. An $(n,k)_q$-affine source $\mathbf{X}$ is uniform over some (unknown) affine subspace of dimension k in $\mathbb{F}^n$. In other words, there exists linearly independent vectors $v_1, \ldots, v_k$ in $\mathbb{F}^n$ such that $\mathbf{X}$ is the distribution obtained by sampling $\lambda_1, \ldots, \lambda_k$*

2575-8454/21/$31.00 ©2021 IEEE
DOI 10.1109/FOCS52979.2021.00067

uniformly and independently from $\mathbb{F}$*, and outputting* $v_0 + \sum_{i=1}^{k} \lambda_i \cdot v_i$*, for some* $v_0 \in \mathbb{F}^n$.

We note that extracting from affine sources falls into the line of investigation that studies extraction from sources sampled by computationally bounded algorithms (since each output bit is computationally restricted to be an affine function of the input randomness, namely the λ_i's).

Thus, an affine extractor $\mathrm{AffExt} : \mathbb{F}^n \to \{0,1\}^m$ for entropy k and error ε is such that for any $(n,k)_q$-affine source $\mathbf{X}$ (over $\mathbb{F}$, where $q = |\mathbb{F}|$), we have $|\mathrm{AffExt}(\mathbf{X}) - U_m| \leq \varepsilon$. For intuition, consider the case of $m = 1$: in this case, an affine extractor is simply a 2-coloring of $\mathbb{F}^n$ such that every dimension k affine subspace of $\mathbb{F}^n$ is almost evenly colored.

A. Applications of Affine Extractors

The class of affine sources naturally generalize (oblivious) bit-fixing sources [7], where a bit-fixing source has some unknown set of coordinates that hold random bits, while the other coordinates are fixed to constants. Extractors for bit-fixing sources have applications in exposure resilient cryptography [7], [9], and the generalization of bit-fixing sources to affine sources played a key role in obtaining the best known extractors for bit-fixing sources [12].

Affine extractors have also found applications in deterministic extraction from many other models of weak sources. Viola [13] proved that sources sampled by small circuits are close to a convex combination of affine sources (and thus an affine extractor can be used to extract from such *circuit sources*). In a recent work, Chattopadhyay and Goodman [14] proved a similar result for sources sampled by algorithms with limited memory [15]. Such *small-space sources* are known to capture a wide variety of weak source models that have been considered for randomness extraction, including: finite Markov chain sources [16], symbol-fixing sources [9], and (short) independent sources. Ben-Sasson and Zewi [17] demonstrated another application by showing how to use any affine extractor to construct low-error two-source extractors in a black-box way, under plausible conjectures from additive combinatorics. Cohen and Tal [18] used affine extractors to extract randomness from *variety sources*, which are distributions that are uniform on the set of common zeroes of a system of polynomial equations. Thus, affine extractors provide a unified way to construct extractors for a wide range of well-studied models of weak sources.

Finally, affine extractors over $\mathbb{F}_2$ have important applications in circuit lower bounds: building on the work of Demenkov and Kulikov [19], a breakthrough work of Find, Golovnev, Hirsch, and Kulikov [20] used affine extractors to bypass a longstanding barrier in circuit lower bounds. To date, the best known circuit lower bounds (of roughly $3.1n$ by Li and Yang [21]) are for affine extractors.

B. Prior Work

A probabilistic argument shows the existence of excellent affine extractors: for example, setting the error ε to a constant and the output length to $m = k - O(1)$, a random function is an affine extractor for min-entropy $k > 2\log n$. However, for applications it is desirable to find *explicit* constructions of such extractors (i.e., an extractor that has running time which is polynomial in the parameters n, q).

Prior work on explicitly constructing affine extractors can be classified into two settings depending on the size of the field:

- The large field setting ($q = \mathrm{poly}(n)$): In this setting, Gabizon and Raz [22] in fact constructed an affine extractor for lines (i.e., $k = 1$).[1] More generally, they showed how to extract most of the entropy out of any $(n,k)_q$-affine source in this large field setting. A construction with improved error was given by Bourgain, Dvir, and Leeman [23] assuming q is a prime, and further that $q-1$ does not have too many prime factors.
- The small field setting ($q = O(1)$): The task of constructing affine extractors is generally more challenging as the field size gets smaller. In the setting of $q = O(1)$, Bourgain [24] and subsequent works of Yehudayoff [25] and Li [26] gave explicit constructions for $k \geq n/\sqrt{\log\log n}$. DeVos and Gabizon [27] obtained a trade-off between the field size and entropy, and gave explicit affine extractors for fields with characteristic $\Omega(n/k)$ and size $q = \Omega((n/k)^2)$. Thus, they require linear entropy (i.e., $k = \Omega(n)$) for extraction from affine sources on fields with constant characteristic.

  A vastly improved result was obtained by Li [28] who gave an explicit affine extractor that works for $q = 2$ (which is generally considered the hardest setting) and $k \geq C(\log n)^C$, for some large enough constant C.

[1] When n is large enough compared to q, the Hales-Jewett theorem rules out the possibility of an extractor for lines.

Li's construction in [28] is closely related to the two-source extractor construction by Chattopadhyay and Zuckerman [29], which works for two independent sources with min-entropy $O(\log^C(n))$. The min-entropy requirement for two-source extractors was later reduced to $\log^{1+o(1)}(n)$ by Ben-Aroya, Doron and Ta-Shma [30] and was further improved in subsequent works [31], [32], [33]. It is natural to ask whether a similar improvement can be obtained for affine extractors.

C. Our Result

Our main result is a further improvement over the result of Li [28], and thus nearly matching the random construction in terms of the min-entropy requirement.

Theorem I.3. *For any constant $\varepsilon > 0$, there exists a constant $C > 0$ such that for all $n, k \in \mathbb{N}$ such that $k \leq n$, there exists an efficient construction of an extractor* $\mathrm{AffExt} : \mathbb{F}_2^n \to \{0,1\}$ *with error ε for $(n,k)_2$-affine sources with min-entropy $k \geq C \cdot \log n \cdot \log\log n \cdot (\log\log\log n)^6$.*

In fact, in the $q = 2$ setting, it is impossible to construct an affine extractor for min-entropy $k < \log(n) - O(1)$, since every deterministic function $f : \mathbb{F}_2^n \to \{0,1\}$ is constant on some affine subspace of dimension $\log(n) - O(1)$ (see, e.g., [34, Lemma 6.7]). Therefore, the min-entropy requirement of our extractor is nearly optimal.

One drawback of our construction compared to [28] is the error of our extractor: while we can only achieve constant error, the construction in [28] achieves polynomially small error. We leave it as an interesting open question to reduce the error of our extractor construction.

D. Subsequent Work

In a subsequent work, Chattopadhyay and Liao [35] construct a "sumset source extractor" for min-entropy $O(\log(n)\log\log(n)\log\log\log^3(n))$, which also implies an affine extractor with the same parameters, and hence slightly improves over the result in this paper. The key ingredient for the improvement in [35] is a better construction of "affine correlation breakers" based on some ideas in this work.

II. Proof Overview

In this section, we give an overview of our affine extractor. The formal proof for Theorem I.3 can be found in the full version [36].

On a very high level, our construction follows the framework in [29], which has been used to construct deterministic extractors in many recent works. The framework works as follows: given a source $\mathbf{X}$, we first convert it into a non-oblivious bit-fixing (NOBF) source, which is a source on $N = \mathrm{poly}(n)$ bits such that $N - N^\delta$ of them are "good," meaning that they are t-wise independent. Then we apply an extractor for NOBF sources to get the output.

A general strategy to construct an NOBF source from multiple independent sources was initiated in [37]. The strategy works by first taking a *strong seeded extractor*, which is a function Ext that takes d bits of extra randomness (i.e. a *seed*) $\mathbf{S}$ and converts $\mathbf{X}$ into a close-to-uniform string $\mathrm{Ext}(\mathbf{X}, \mathbf{S})$, with high probability over the seed $\mathbf{S}$. Since in reality we do not have such a seed $\mathbf{S}$, we enumerate over all $D = 2^d$ possibilities of the random seed and get a somewhere-random source (SR-source), which is a collection of D different strings such that most of them are close to uniform. However, note that the strings which are close to uniform are arbitrarily correlated with each other. The second step is to take another independent source to "break the correlation" between these uniform strings and make them t-wise independent. A function which can complete this task is called a correlation breaker [38]. Recent constructions of such objects employ a technique known as alternating extraction [39], which uses a strong seeded extractor as a building block.

The setting of affine extractors is trickier since there is only one source. At first glance, it doesn't seem like the above framework can be used to construct an affine extractor. However, Li [28] showed that this framework can still be used, based on a crucial observation originating in the work of Rao [12]: if the strong seeded extractor Ext that we use is a linear function for *every* fixing of the seed s (such extractors are called *linear seeded extractor*), then there is still some "implicit independence" between the output $\mathbf{Y} = \mathrm{Ext}(\mathbf{X}, s)$ and the original source $\mathbf{X}$. Specifically, $\mathbf{X}$ can be written in the form $\mathbf{A} + \mathbf{B}$ such that $\mathbf{A}$ has some entropy and is independent of $(\mathbf{B}, \mathbf{Y})$. Then [28] showed that if we again use linear seeded extractors to construct the correlation breaker, then it is possible to exploit this implicit independence.

However, the affine extractor in [28] requires $\mathrm{polylog}(n)$ entropy, for the following reasons. First, to extract from an NOBF source, [28] used the derandomized Ajtai-Linial resilient function [29], [40], [28] in the last step, which requires the source to have poly-logarithmic entropy. Second, the correlation breaker in [28] also requires $\Omega(\log^2 n)$ entropy to work. In fact, even the state-of-the-art correlation breakers for

independent sources [41] required $\Omega(\log^2 n)$ entropy at the time, while in [28] the correlation breaker needs to work for two sources with linear correlation, which is even harder. Finally, many steps in this construction require a strong linear seeded extractor. However, the known constructions of strong linear seeded extractors usually require at least $\text{polylog}(n)$ entropy. Specifically, note that this framework heavily relies on strong linear seeded extractors for two different purposes:

1) To convert $\mathbf{X}$ into a SR-source, and
2) To construct the correlation breaker.

In both cases we need the error of extractor to be $1/\text{poly}(n)$, and in the first case we further need the seed to have length $d = O(\log(n))$ to make sure that there are only $2^d = \text{poly}(n)$ possibilities of seed to enumerate (since otherwise the running time of the affine extractor we construct will not be polynomial). The most commonly used strong linear seeded extractor is perhaps Trevisan's extractor [42], [43], but it requires a seed of length at least $\log^2(n)$ in this setting. The extractor constructed in [28], while having $O(\log(n))$ seed length, requires the source to have $\log^c(n)$ entropy for some constant $c > 4$.

A. Bypassing the linear seeded extractor barrier

To solve the problem of not having a good enough linear seeded extractor, we take different approaches in the two cases. We first discuss the task of turning $\mathbf{X}$ into a SR-source, and explain the construction of our correlation breaker in Section II-B.

In both cases, our starting point is a simple construction of strong linear seeded extractors which works as follows. To extract m uniform bits, our first step is to apply a strong lossless condenser on $\mathbf{X}$: this is a function that takes a seed and converts $\mathbf{X}$ into a shorter source $\mathbf{X}'$ of length $O(m)$ while still having roughly m bits of entropy. Using the celebrated "GUV condenser" [4], this step requires a seed of length $O(\log(n/\varepsilon))$ where ε is the error, and such a condenser can also be made linear [44]. Our second step is to apply a linear universal hash function on $\mathbf{X}'$ to get an m-bit uniform string by the Leftover Hash Lemma [45]. This "condense-then-hash" extractor has optimal entropy requirement, but the seed length is $O(m + \log(n))$.

Now, recall that in the first step we need the seed length to be $O(\log n)$, which means using this condense-then-hash extractor, we can only extract a string $\mathbf{Y}$ of length $O(\log n)$. However, this is not enough for a correlation breaker to work, even if we use the state-of-the-art correlation breaker for independent sources [33] (recall, we have to deal with the harder case of the linearly correlated sources).

To solve this problem, our observation is that while $|\mathbf{Y}| = O(\log n)$ is not enough for a correlation breaker to work, we require $\mathbf{Y}$ to be only slightly longer. In particular, we need $m = |\mathbf{Y}| = O(c(n) \cdot \log n)$ for some slowly growing function $c = c(n)$. (We will see that we can take $c = \log\log(n)$ when we discuss correlation breakers in Section II-B.) Our idea is to use a recursive approach based on block-source extraction, combined with an error reduction trick at the end, as follows:

- As before, we first use the GUV condenser to condense the source $\mathbf{X}$ into a source $\mathbf{X}'$ of length $n' = O(m)$ and entropy $0.9n'$. In other words, $\mathbf{X}'$ has *entropy rate* 0.9. This requires a seed of length $O(\log(n/\varepsilon))$.
- Next, we cut $\mathbf{X}'$ into two blocks, and a standard argument shows that each block still has entropy rate 0.8, even when conditioned on the other block. Then again we apply the GUV condenser on each block to condense the entropy rate to 0.9, but this time we use one seed to condense both blocks. Intuitively, this works because the GUV condenser is strong (which can be considered as "success with high probability" and hence we can apply the union bound on both blocks). Furthermore, note that this time the seed length is only $O(\log(m/\varepsilon))$.
- We again divide each block into two halves and get four blocks in total, and use one extra seed of length $O(\log(m/\varepsilon))$ to condense all four blocks. By repeating this step for $\log(c)$ times we eventually get c blocks, each having entropy roughly $O(\log(n))$.
- Finally, we use another seed of length $O(\log(n/\varepsilon))$ to sample a linear hash function and extract from every block. The total seed length is $O(\log(n) + \log(c)\log(m/\varepsilon))$, which is $O(\log(n) + \log(c)\log(1/\varepsilon))$ since m is short.[2] Now we get an extractor of seed length $O(\log(n))$, but with error $\varepsilon = n^{-O(1/\log(c))}$, which is slightly larger than what we need.
- To solve this problem, we apply the error reduction scheme in [30], which reduces the error to $1/\text{poly}(n)$ but only increases the seed length by a constant factor. A drawback of this scheme is that for every seed we get A different outputs such that only one of them is guaranteed to be uniform. In other words, we get a *somewhere random extractor* instead of an extractor.

[2] In fact, the actual seed length should be $O(\log(n) + \log^2(c)\log(1/\varepsilon))$. See the full version for more details.

We note that the weaker notion of a somewhere random extractor (instead of an extractor) suffices in our scheme of constructing affine extractors. Informally, we follow the approach of [30], and apply correlation breakers on all outputs of the somewhere random extractor. Using an idea from [38], we can simply merge these strings by taking the parity after we break their correlation.

Another possible concern is the following. In [30] they started from an extractor with error $1/\mathrm{poly}(n)$, and reduce it to $1/n^C$ for any constant C, and thus get the parameter A (the number of different outputs) to be a constant. In our setting, we start from an extractor with error slightly larger than $1/\mathrm{poly}(n)$, and thus require $A = O(\log(c))$. So, in our construction, we need a correlation breaker which breaks the correlation between more strings. This implies that we need the output of our somewhere extractor, $\mathbf{Y}$, to be longer, and thus it increases the seed length of our extractor correspondingly. Nevertheless, we only need to increase the length of $\mathbf{Y}$ by a factor of A^2. Since A only has logarithmic dependence on the length of $\mathbf{Y}$, this will not be a problem.

B. Correlation breakers for linearly correlated sources

In this section, we give a brief description of the main ideas that go into our correlation breaker construction, assuming some familiarity with the techniques that are used in recent constructions of correlation breakers. In Section III, we present a much more detailed account of our correlation breaker construction.

Many recent works successfully construct correlation breakers for independent sources with error $1/\mathrm{poly}(n)$ which only require $\log^{1+o(1)}(n)$ entropy [46], [47], [32], [31], [33]. In fact, the state-of-the-art construction by Li [33] only requires $O(\log n \cdot \frac{\log\log n}{\log\log\log n})$ entropy. As we pointed out above, if we try to adapt these constructions to the setting of linearly correlated sources using Trevisan's extractor or the linear seeded extractor in [28], the entropy requirement is at least $\log^2(n)$.

A natural idea is to use the linear seeded extractor, based on the "condense-then-hash" approach discussed in Section II-A, in the correlation breaker construction. However, a problem of the extractor in Section II-A is the seed length depends on the output length. Such a dependence makes the analysis much more complicated, for the following reasons. In correlation breaker constructions, the output length of an extractor Ext_1 usually depends on the seed length of some other extractor Ext_2, and the output length of Ext_2 might also depend on the seed length of another extractor Ext_3, and so on. If the seed length of each extractor also depends on its own output length, then the seed length of Ext_1 might depend on the parameters of some other extractor Ext_ℓ after $\ell = \omega(1)$ levels of propagation, and it is not clear whether this will cause a loss in the parameters.

To solve this problem, we observe that it's actually not necessary to use a strong linear seeded extractor all the time. To see why this is the case, first we recap why we need a linear seeded extractor when considering linearly correlated sources. As a toy example, we consider the two-step alternating extraction between $\mathbf{X}, \mathbf{Y}$, in which we first take a prefix of $\mathbf{Y}$ to extract from $\mathbf{X}$, and then use the extracted output to extract from $\mathbf{Y}$. Recall that $\mathbf{X}$ can be written as $\mathbf{A} + \mathbf{B}$, where $\mathbf{A}$ is independent of $(\mathbf{B}, \mathbf{Y})$. Now let LExt denote a strong linear seeded extractor, and Ext be another strong seeded extractor. If we take a prefix $\mathbf{Q}$ from $\mathbf{Y}$ and compute $\mathbf{W} = \mathrm{LExt}(\mathbf{X}, \mathbf{Q})$, then $\mathbf{W} = \mathbf{W}_A + \mathbf{W}_B$, where $\mathbf{W}_A = \mathrm{LExt}(\mathbf{A}, \mathbf{Q})$ and $\mathbf{W}_B = \mathrm{LExt}(\mathbf{B}, \mathbf{Q})$. Now note that conditioned on the fixing of $\mathbf{Q}$, $\mathbf{W}_A$ is uniform with high probability and is independent of $\mathbf{W}_B$. Therefore $\mathbf{W}$ is also uniform with high probability. When extracting from the $\mathbf{Y}$ side, we again use the fact that $\mathbf{W}$ can be written as $\mathbf{W}_A + \mathbf{W}_B$ where $\mathbf{W}_A$ is uniform and independent of $(\mathbf{W}_B, \mathbf{Y})$. Note that conditioned on $\mathbf{W}_B$, $\mathbf{W}$ should still be uniform and independent of $\mathbf{Y}$, and $\mathbf{Y}$ only loses a small amount of entropy (proportional to the length of $\mathbf{W}_B$). This ensures that $\mathrm{Ext}(\mathbf{Y}, \mathbf{W})$ is still uniform with high probability over $\mathbf{W}$ if $\mathbf{Y}$ has enough entropy. Observe that this argument does not require Ext to be linear.

Based on this observation, we can do the alternating extraction in an "asymmetric" way: when we extract from $\mathbf{X}$, we use the condense-then-hash extractor, which takes a d_X-bit seed and outputs a uniform string with d_Y bits. Note that $d_X = c \cdot d_Y$ for some constant $c > 1$. Then when we extract from $\mathbf{Y}$, we use a optimal non-linear strong seeded extractor (e.g. the GUV extractor [4]) which has a *fixed* seed length d_Y regardless of the output length. Therefore we can repeat this alternating extraction step for many rounds without creating any propagated dependence.

Finally, we note that the idea above can be generalized to give a modular way for adapting correlation breakers to the affine setting. Suppose a correlation breaker takes a weak source $\mathbf{X}$ and a uniform seed $\mathbf{Y}$ as input. We observe that previous constructions of correlation breakers can be viewed as alternatively executing "sub-protocols" on $\mathbf{X}$ and on $\mathbf{Y}$. A sub-protocol on $\mathbf{X}$ is a function which takes a seed correlated with $\mathbf{Y}$ and runs some functions on random variables correlated with

$\mathbf{X}$. Similarly, a sub-protocol on $\mathbf{Y}$ is a function which takes a seed correlated with $\mathbf{X}$ and runs some functions on random variables correlated with $\mathbf{Y}$. The simplest example of a sub-protocol is a strong seeded extractor, and more examples can be found in Section III. By an argument similar to the alternating extraction case, we observe that a correlation breaker will work in the affine setting if the sub-protocols satisfy the following conditions:

- If f is a sub-protocol on $\mathbf{X}$, f should be linear.
- If g is a sub-protocol on $\mathbf{Y}$, g does not need to be linear, but should work properly when all the random variables correlated with $\mathbf{Y}$ are weak sources.

Usually the second case is easier to deal with: the same construction (or a slight change) of sub-protocols should still work most of the time. Therefore the construction will be simple if we minimize the amount of sub-protocols in the first case.

In fact, in our construction, all the functions we need in the first case are simply strong seeded extractors, and thus we can replace them with the condense-then-hash extractor. However, it is still not clear how to use previous results in a black-box fashion. Thus, we give a much more detailed explanation of the main ideas of our correlation breaker construction in Section III.

C. Extracting from NOBF sources

The final step is to extract from an NOBF source $\mathbf{Z}$ on N bits that we obtain from the affine source $\mathbf{X}$ (by first converting it into an SR-source and then applying the correlation breaker). The derandomized Ajtai-Linial function [48], [29], [40] was shown to be an extractor for such sources with at least $N - N/\log^2 N$ good bits. However, this extractor needs min-entropy at least $\mathrm{polylog}(n)$ in the NOBF source, since good bits are required to be $\mathrm{polylog}(n)$-wise independent.

To circumvent this barrier, we recall a result of Viola [13], who proved that majority can extract from an $O(1)$-wise independent NOBF source with constant error. Thus, this is better suited for our goal of constructing affine extractors for near logarithmic entropy. In fact, this resilient function is also used in recent constructions of two-source extractors that work for near logarithmic min-entropy based on the two-source framework of Ben-Aroya, Doron, and Ta-Shma [30]. However, to use the majority function we require the NOBF source $\mathbf{Z}$ to have at least $N - N^{\delta}$ good bits, for $\delta < 1/2$. In fact, as pointed out in [30], $\delta = 1/2$ is actually a barrier in the two-source setting if the SR-source is created using a seeded extractor.

Interestingly, for our setting, a general seeded extractor is not required and the above barrier does not hold. Indeed, since we just wish to produce an SR-source from an *affine* source $\mathbf{X}$, it suffices to use a (linear) seeded extractor that only works for affine sources - and one can use the probabilistic method to show the existence of such an object with appropriate parameters that can bypass this barrier. However, we do not have explicit constructions of these objects. Instead, we show that our somewhere random extractor from Section II-A can be used to construct the SR source with desired parameters.

D. Summary of our construction

Finally we summarize our construction. Given an affine source $\mathbf{X}$, we run the following steps to extract a bit:

1) Take a strong linear seeded somewhere random extractor LSRExt with seed length $d = O(\log n)$, and for every $s \in \{0,1\}^d$ compute $(\mathbf{Y}_{s,1}, \ldots, \mathbf{Y}_{s,A}) := \mathrm{LSRExt}(\mathbf{X}, s)$.
   This step guarantees that for most of $s \in \{0,1\}^d$ there exists $i \in [A]$ such that $\mathbf{Y}_{s,i}$ is uniform.
2) Take a "correlation breaker" ACB and compute $\mathbf{Z}_{s,j} := \mathrm{ACB}(\mathbf{X}, \mathbf{Y}_{s,j}, (s,j))$ for every $s \in \{0,1\}^d, j \in [A]$. Here (s,j) serves as "advice" to ACB (more details can be found in the next section).
   Roughly speaking, the correlation breaker ACB converts uniform strings $\{\mathbf{Y}_{s,j}\}$ to t-wise independent bits $\{\mathbf{Z}_{s,j}\}$, for some proper choice of t. It is guaranteed that for most of s, there exists a "good bit" $\mathbf{Z}_{s,i}$ which is uniform.
3) For every $s \in \{0,1\}^d$, compute $\mathbf{P}_s := \bigoplus_{j=1}^{A} \mathbf{Z}_{s,j}$. After this step, most of $\mathbf{P}_s$ are uniform, and the uniform bits in $\{\mathbf{P}_1, \ldots, \mathbf{P}_{2^d}\}$ remain (t/A)-wise independent.
4) Compute the majority of $\mathbf{P}_1, \ldots, \mathbf{P}_{2^d}$.

III. Correlation-Breaking Games

In this section, we present a detailed explanation of the main ideas used in our correlation breaker. We explain these ideas using a few (related) two-party games that we introduce below.

Recall that our goal is to use an independent (or linearly correlated) source $\mathbf{X}$ to break the correlation between $\mathrm{poly}(n)$ strings $\mathbf{Y}_1, \ldots, \mathbf{Y}_D$ and make them t-wise independent. The high-level idea is that we compute the same function f, that is called a *correlation breaker*, on $\mathbf{X}$ and every $\mathbf{Y}_i$, to produce strings $\mathbf{Z}_1 = f(\mathbf{X}, \mathbf{Y}_1), \ldots, \mathbf{Z}_D = f(\mathbf{X}, \mathbf{Y}_D)$. The property we desire from f is the following: if there is a set

$T \subset [D]$, such that for any $i \in T$, $\mathbf{Y}_i$ is uniform, then for any $i \in T$ and any $i_1, \ldots, i_t \in [D]$ distinct from i, the random variable $\mathbf{Z}_i$ looks uniform conditioned on $\{\mathbf{Z}_{i_j}\}_{j=1}^t$. For constructing the function f, it helps to think of it as a two-party game that we discuss in detail below.

Given the above discussion, it is enough to consider the following setting: let $\mathbf{Y}$ be a uniform string, and let $\mathbf{Y}^1, \mathbf{Y}^2, \ldots, \mathbf{Y}^t$ be random variables that are arbitrarily correlated with $\mathbf{Y}$. $\mathbf{Y}^i$ is called the i^{th} *tampering* of $\mathbf{Y}$. As before, let $\mathbf{X}$ be a random variable that is independent (or linearly correlated) with $\mathbf{Y}, \{\mathbf{Y}^i\}_{i=1}^t$. We want to construct a correlation breaker f with the guarantee that the output $f(\mathbf{X}, \mathbf{Y})$ is uniform conditioned on the outputs computed using all of its t tamperings $\{f(\mathbf{X}, \mathbf{Y}^i)\}_{i=1}^t$. Before discussing how to construct correlation breakers, we first introduce some convenient notation.

As alluded to above, a useful perspective is to think of the computation of the correlation breaker as a two-party communication between $\mathbf{X}$ and $\mathbf{Y}$. This is because of the following reason: if $\mathbf{Z}$ is the transcript of a two-party communication between $\mathbf{X}$ and $\mathbf{Y}$, then $\mathbf{X} \leftrightarrow \mathbf{Z} \leftrightarrow \mathbf{Y}$ forms a Markov chain. This ensures that at any point of the computation we have two independent sources $\mathbf{X}, \mathbf{Y}$ to work with, conditioned on any fixing of $\mathbf{Z}$. Therefore, at each step of the computation one party can send a message as independent randomness to help the other party complete some tasks. However, $\mathbf{Z}$ leaks some information about $\mathbf{X}$ and $\mathbf{Y}$; so ideally we want the length of $\mathbf{Z}$, i.e. the *communication complexity*, to be as small as possible.

We now introduce some convenient notation.

Notation: Throughout this paper, we use $\mathbf{Y}^{[t]}$ to denote all the t tamperings of $\mathbf{Y}$, and for any set $S \subseteq [t]$ such that $S = \{i_1, \ldots, i_k\}$, we also use $\mathbf{Y}^S$ to denote the collection $\mathbf{Y}^{i_1}, \ldots, \mathbf{Y}^{i_k}$. For any random variable $\mathbf{R}$ computed, we use $\mathbf{R}^i$ to denote the i^{th} tampered version of $\mathbf{R}$ which is computed using $\mathbf{Y}^i$.

In Sections III-A to III-H, we discuss the two-party games and relevant techniques that are used in recent constructions of correlation breakers for independent sources, in increasing order of complexity. Along the way, we explain how we adapt some of these techniques to construct our correlation breaker in the affine source setting. In Section III-I, we define the correlation breaking game in the affine source setting, and in Section III-J we summarize our construction.

We start with describing the correlation breaking game in the independent source setting, in the more general case that $\mathbf{X}$ also has its tampered versions $\mathbf{X}^1, \ldots, \mathbf{X}^t$.

A. Correlation-breaking game for independent sources

The setup is as follows: Quentin has a source $\mathbf{X}$ which is uniform and Wendy has a source $\mathbf{Y}$ which has some entropy. Further, Quentin and Wendy hold some tampered sources $(\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^t)$ and $(\mathbf{Y}^1, \mathbf{Y}^2, \ldots, \mathbf{Y}^t)$ respectively such that $\mathbf{X}^{[t]}$ can be arbitrarily correlated with $\mathbf{X}$ and $\mathbf{Y}^{[t]}$ can be arbitrarily correlated with $\mathbf{Y}$. The assumption is that $(\mathbf{X}, \mathbf{X}^{[t]})$ is independent of $(\mathbf{Y}, \mathbf{Y}^{[t]})$. Quentin and Wendy are going to run a two-party game as follows. The game starts with a public transcript $\mathbf{Z}$ and some "tampered transcripts" $(\mathbf{Z}^1, \ldots, \mathbf{Z}^t)$ which are all empty at the beginning. They need to choose a deterministic two-party communication protocol P, which is a sequence of deterministic functions $(f_1, g_1, f_2, g_2, \ldots)$ so that in the first round Quentin sends a message $\mathbf{Q}_1 := f_1(\mathbf{X}, \mathbf{Z})$, and then $\mathbf{Q}_1$ is added to the transcript $\mathbf{Z}$. In the next round Wendy sends a message $\mathbf{W}_1 := g_1(\mathbf{Y}, \mathbf{Z})$, and then $\mathbf{W}_1$ is added to the transcript $\mathbf{Z}$. They keep sending messages computed with $f_2, g_2, \ldots$ until the protocol ends. However, there are also t "tampered communications" that are run in parallel. When Quentin sends $\mathbf{Q}_1 := f_1(\mathbf{X}, \mathbf{Z})$, a tampered message $\mathbf{Q}_1^j := f_1(\mathbf{X}^j, \mathbf{Z}^j)$ is also sent and added to the tampered transcript $\mathbf{Z}^j$ for every $j \in [t]$. Similarly when Wendy sends a message there will also be t tampered messages sent simultaneously. At the end of the protocol, one of the parties computes an output, which we denote as $\mathbf{R} = P(\mathbf{X}, \mathbf{Y})$, and $\mathbf{R}$ will not be added to the transcript $\mathbf{Z}$. Quentin and Wendy win the game if $\mathbf{R}$ is uniform conditioned on all the tampered outputs $\mathbf{R}^{[t]}$ where $\mathbf{R}^j = P(\mathbf{X}^j, \mathbf{Y}^j)$ and all the (tampered) transcripts $\mathbf{Z}, \mathbf{Z}^{[t]}$.

B. Alternating extraction

It is easy to see that Quentin and Wendy can never win the correlation-breaking game if $\mathbf{X}^1 = \mathbf{X}$ and $\mathbf{Y}^1 = \mathbf{Y}$, since this implies $\mathbf{R}^1 = \mathbf{R}$. However, it is possible to win a weaker game which we call a *look-ahead game*. In an ℓ-look-ahead game, Quentin and Wendy need to output multiple messages $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_\ell$. We say a message $\mathbf{R}$ has the *look-ahead property* [49] if $\mathbf{R}$ is uniform conditioned on all the transcripts $\mathbf{Z}, \mathbf{Z}^{[t]}$ (but not necessarily on the tampered output $\mathbf{R}^{[t]}$). Quentin and Wendy win the look-ahead game if the output $\mathbf{R}$ has the look-ahead property. Winning the look-ahead game with one output is actually not very interesting since Quentin can just output a prefix of $\mathbf{X}$ while all the transcripts are empty. Now consider the ℓ-look-ahead game so that Quentin and Wendy need to *sequentially* compute and send $\mathbf{R}_1, \ldots, \mathbf{R}_\ell$ such that every $\mathbf{R}_i$ satisfies the look-ahead property at the moment it is computed. Note

that the transcripts of $\mathbf{R}_i$ contain the previous outputs $\mathbf{R}_1, \ldots, \mathbf{R}_{i-1}$ and their tampered versions. This game is winnable with the alternating extraction [39] protocol, which works as follows. Observe that at any moment of the game, $(\mathbf{X}, \mathbf{X}^{[t]}) \leftrightarrow (\mathbf{Z}, \mathbf{Z}^{[t]}) \leftrightarrow (\mathbf{Y}, \mathbf{Y}^{[t]})$ is a Markov chain. Moreover, conditioned on $(\mathbf{Z}, \mathbf{Z}^{[t]})$, $\mathbf{Y}$ only loses roughly $(t+1)\ell$ bits of entropy where ℓ is the total length of messages from Wendy. Therefore, if $\mathbf{Y}$ has high enough entropy at the beginning, it will still have some entropy remaining if the total length of messages (i.e. the *communication complexity*) from Wendy is not too long. If Quentin sends a string $\mathbf{Q}_1 = f(\mathbf{X}, \mathbf{Z})$ which is uniform condition on $(\mathbf{Z}, \mathbf{Z}^{[t]})$ (i.e. $\mathbf{Q}_1$ satisfies the look-ahead property), then Wendy can also get a uniform string $\mathbf{W}_1 = \mathrm{Ext}(\mathbf{Y}, \mathbf{Q}_1)$ conditioned on $(\mathbf{Z}, \mathbf{Z}^{[t]})$ by applying a seeded extractor. Moreover, if Ext is strong, $\mathbf{W}_1$ remains uniform even after $(\mathbf{Q}_1, \mathbf{Q}_1^{[t]})$ is added to $(\mathbf{Z}, \mathbf{Z}^{[t]})$, since the entropy of $\mathbf{W}_1$ comes purely from $\mathbf{Y}$. Therefore, $\mathbf{W}_1$ also satisfies the look-ahead property. This observation gives the alternating extraction protocol: Quentin sends $\mathbf{Q}_1$ which is a prefix of $\mathbf{X}$, Wendy sends $\mathbf{W}_1 := \mathrm{Ext}(\mathbf{Y}, \mathbf{Q}_1)$, Quentin sends $\mathbf{Q}_2 := \mathrm{Ext}(\mathbf{X}, \mathbf{W}_1)$, and so on. As long as $\mathbf{X}$ and $\mathbf{Y}$ still have enough entropy left, every message sent in this protocol satisfies the look-ahead property. The entropy requirement is roughly $O(\ell t \log n)$, where the $\log n$ comes from the seed length of each randomness extractor.

C. Breaking correlation with advice

Now we change the correlation-breaking game a little bit to get an actually winnable game. Suppose Quentin and Wendy further get some advice $(\alpha, \alpha^1, \ldots, \alpha^t) \in [2^a]$ such that $\alpha \neq \alpha^j$ for every $j \in [t]$. Then the actual communication is run with a protocol P_α chosen from a family of protocols $\{P_1, \ldots, P_{2^a}\}$. Moreover, for every $j \in [t]$, the j^{th} tampered communication is run with the protocol P_{α^j}. Since the actual protocol is different from all the tampered protocols, now it's possible that $\mathbf{R}$ is independent of $\mathbf{R}^{[t]}$ even if $\mathbf{X} = \mathbf{X}^j$ and $\mathbf{Y} = \mathbf{Y}^j$ for every $j \in [t]$. This is called a *correlation breaker with advice* [41], [50]. In fact, consider the family of protocols such that P_i runs alternating extraction for i rounds and outputs the i^{th} message from Wendy. Then if $\alpha > \alpha^j$ for every $j \in [t]$, the output $\mathbf{R} := P_\alpha(\mathbf{X}, \mathbf{Y})$ is actually independent of $\mathbf{R}^{[t]} := P_{\alpha^{[t]}}(\mathbf{X}^{[t]}, \mathbf{Y}^{[t]})$ by the look-ahead property. This idea first came in [37] by Li. When the order of advice is unknown, there was a beautiful idea by Cohen [38] called the "flip-flop" construction which resolves the issue. However, note that with only this idea $\mathbf{X}$ and $\mathbf{Y}$ need entropy roughly $O(2^a \cdot t \log n)$, and hence the protocol is only good enough when a is small (e.g. 1 bit).

D. Merging independence

To reduce the entropy requirement, a nice *independence-preserving* property of strong seeded extractors comes to the rescue. Suppose there is a source $\mathbf{Y}$ and a seed $\mathbf{Q}$, and each of them have a tampered version $\mathbf{Y}^1, \mathbf{Q}^1$. Now suppose $\mathbf{Q}$ is uniform conditioned on $\mathbf{Q}^1$. Then if one applies a seeded extractor and gets $\mathrm{Ext}(\mathbf{Y}, \mathbf{Q})$, this string is also uniform conditioned on $\mathrm{Ext}(\mathbf{Y}^1, \mathbf{Q}^1)$. Too see why this is true, note that when conditioned on $\mathbf{Q}^1$ and $\mathrm{Ext}(\mathbf{Y}^1, \mathbf{Q}^1)$, $\mathbf{Q}$ is still uniform and independent of $\mathbf{Y}$, while $\mathbf{Y}$ only loses a small amount of entropy. Therefore $\mathrm{Ext}(\mathbf{Y}, \mathbf{Q})$ is still uniform. In other words, $\mathrm{Ext}(\mathbf{Y}, \mathbf{Q})$ preserves the independence of $\mathbf{Q}$ from its tampering $\mathbf{Q}^1$. This idea was used in [37] to get a better entropy requirement, which is only linear in a. We will see more details later. Furthermore, Ext can also preserve the independence on the other side: if $\mathbf{Y}$ has high entropy conditioned on $\mathbf{Y}^1$, then $\mathrm{Ext}(\mathbf{Y}, \mathbf{Q})$ is uniform conditioned on $\mathrm{Ext}(\mathbf{Y}^1, \mathbf{Q}^1)$. This can be proven using a similar argument. Based on this observation, Cohen and Schulman [51] suggested a protocol which works as follows. Suppose Quentin has two uniform strings $\mathbf{X}_1, \mathbf{X}_2$ such that either $\mathbf{X}_1$ is independent of $\mathbf{X}_1^1$ or $\mathbf{X}_2$ is independent of $\mathbf{X}_2^1$. (Note that $\mathbf{X}_1$ and $\mathbf{X}_2$ might be correlated.) Let $\mathbf{Q}_1$ be a prefix of $\mathbf{X}_1$. Now they can do two rounds of alternating extraction to compute $\mathbf{W}_1 = \mathrm{Ext}(\mathbf{Y}, \mathbf{Q}_1)$ and then $\mathbf{R} = \mathrm{Ext}(\mathbf{X}_2, \mathbf{W}_1)$. The output $\mathbf{R}$ should be independent of $\mathbf{R}^1$ by the independence-preserving property. In other words, they *merge* $\mathbf{X}_1, \mathbf{X}_2$ and *preserve the independence* of $\mathbf{X}_1$ or $\mathbf{X}_2$ from its tampered version. Chattopadhay and Li [46] showed that it's also possible to merge ℓ strings $\mathbf{X}_1, \ldots, \mathbf{X}_\ell$ by doing more rounds of alternating extraction. This protocol is called a non-malleable independence-preserving merger (NIPM) [51], [46].

In this paper we show that a stronger *independence merging* property holds. That is, suppose there exist $S, T \subseteq [t]$ such that $\mathbf{X}$ is uniform and independent of $\mathbf{X}^S$, and $\mathbf{Y}$ has high min-entropy conditioned on $\mathbf{Y}^T$. Then by taking a prefix $\mathbf{Q}$ of $\mathbf{X}$ and computing $\mathbf{W} = \mathrm{Ext}(\mathbf{Y}, \mathbf{Q})$, $\mathbf{W}$ is actually uniform conditioned on $\mathbf{W}^{S \cup T}$. In other words, the strong seeded extractor Ext *merges the independence* of $\mathbf{X}$ and $\mathbf{Y}$ from their tampered versions. To prove this, we can simply apply the argument from both cases of independence preservation, together. Therefore, we can use the same alternating extraction protocol to merge the independence

of $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_\ell$ from their tampered versions, even if the independence is scattered on multiple different $\mathbf{X}_i$. This stronger property will help us deal with the t-tampering case directly.

E. Strongness of protocols

Suppose $\mathbf{R}$ is the output of some two-party communication protocol $P(\mathbf{X}, \mathbf{Y})$, $\mathbf{Z}$ is the transcript, and $\mathbf{R}$ is computed by Wendy using some deterministic function $g(\mathbf{Y}, \mathbf{Z})$. Then $(\mathbf{X}, \mathbf{X}^{[t]}) \leftrightarrow (\mathbf{Z}, \mathbf{Z}^{[t]}) \leftrightarrow (\mathbf{R}, \mathbf{Y}, \mathbf{Y}^{[t]})$ forms a Markov chain. In all the two-party games we consider in this paper, $\mathbf{R}$ is uniform conditioned on the transcripts $(\mathbf{Z}, \mathbf{Z}^{[t]})$ and some of the tampered output. Therefore even if the whole source $\mathbf{X}$ (and $\mathbf{X}^{[t]}$) is sent by Quentin and added to the transcript, $\mathbf{R}$ is still uniform. In other words, the protocol P is *strong* in $\mathbf{X}$. Now observe that when a protocol P is strong in $\mathbf{X}$, we can actually re-design P in the following way: Quentin simply sends $\mathbf{X}$, and Wendy simulates the output of P using $\mathbf{X}, \mathbf{Y}$. When the protocol is re-designed in this way we say $\mathbf{X}$ is the *seed* of the protocol. Similarly we can let Wendy send $\mathbf{Y}$ and Quentin simulate P if P is strong in $\mathbf{Y}$. It's also not hard to switch the strongness of a protocol using the idea of alternating extraction: if Wendy produces the output $\mathbf{R}$, we can let Wendy send $\mathbf{R}$ and let Quentin output $\mathrm{Ext}(\mathbf{X}, \mathbf{R})$ instead. Now the protocol becomes strong in $\mathbf{Y}$. The advantage of strongness is we can run many different protocols *in parallel*. That is, suppose Wendy holds many correlated sources $\mathbf{Y}_1, \ldots, \mathbf{Y}_r$, Quentin holds $\mathbf{X}$ and another source $\mathbf{Q}$ correlated with $\mathbf{X}$, and they want to run many protocols $P_1(\mathbf{Q}, \mathbf{Y}_1), \ldots, P_\ell(\mathbf{Q}, \mathbf{Y}_\ell)$. Then Quentin can simply send $\mathbf{Q}$ (and $\mathbf{Q}^{[t]}$) and let Wendy simulate everything. This ensures that the total *communication complexity* is low, so that the source $\mathbf{X}$ in Quentin's hand only loses roughly $(t+1)|\mathbf{Q}|$ bits of entropy regardless of how many protocols there are. Moreover, Wendy doesn't lose any entropy. This idea plays a crucial role in [46].

Another advantage of strongness is, if we let Quentin send his whole source in a look-ahead game, Wendy doesn't need to send any of her output. Therefore all of Wendy's outputs $\mathbf{W}_1, \ldots, \mathbf{W}_\ell$ remain uniform but still have the look-ahead property (i.e. $\mathbf{W}_i$ is uniform conditioned on $\mathbf{W}_1, \ldots, \mathbf{W}_{i-1}$ and their tamperings). Therefore these strings can be saved for later use. This is called a look-ahead extractor [49]. There's only one drawback: to run any protocol based on alternating extraction, usually the length of $\mathbf{Q}$ needs to be proportional to t. Therefore if we need $\mathbf{X}$ to still have some entropy left after sending $\mathbf{Q}$, the total entropy requirement for $\mathbf{X}$ becomes proportional to t^2. Nevertheless t is usually small compared to other parameters so this is not a big deal.

F. Correlation breakers based on somewhere independence

Now we are ready to introduce the general strategy for the correlation-breaking game. First Quentin and Wendy run a 2-look-ahead extractor and create two strings $\mathbf{W}_0, \mathbf{W}_1$ on Wendy's side. Now suppose the advice is $\alpha \in \{0,1\}^a$, and we use α_j to denote the j^{th} bit of α. For every $j \in [a]$, define $\mathbf{V}_{2j-1} := \mathbf{W}_{\alpha_j}$ and $\mathbf{V}_{2j} := \mathbf{W}_{1-\alpha_j}$. Note that the pair $(\mathbf{V}_{2j-1}, \mathbf{V}_{2j})$ is defined in the "flip-flop" way [38] so that it will either be $(\mathbf{W}_0, \mathbf{W}_1)$ or $(\mathbf{W}_1, \mathbf{W}_0)$, depending on α_j. If $\alpha_j \neq \alpha'_j$, then in the position $\mathbf{V}_{2j-\alpha_j}$ where $\mathbf{W}_1$ is placed, the corresponding tampered version $\mathbf{V}^1_{2j-\alpha_j}$ should be $\mathbf{W}^i_0$. Therefore we get independence of $\mathbf{W}_1$ from $\mathbf{W}^i_0$ based on the look-ahead property. Now observe that $(\mathbf{V}_1, \ldots, \mathbf{V}_{2a})$ is *somewhere-independent* [51] from its tamperings. That is, for every $i \in [t]$ there exists some $j \in [2a]$ such that $\mathbf{V}_j$ is independent from $\mathbf{V}^i_j$. If they then use the independence merging protocol described above to merge these strings, they get $\mathbf{R}$ which is uniform conditioned on $\mathbf{R}^{[t]}$, and hence win the correlation-breaking game. The entropy requirement is then proportional to $O(a)$ instead of 2^a. Moreover, the entropy requirement can be further improved by running the independence merging protocol in parallel. That is, if they merge every two strings in parallel and repeat for $\log(2a)$ rounds, eventually all the strings will be merged into one which collects all the independence. Intuitively the entropy requirement should be proportional to $\log(a)$. This is the main idea in [46]. However, as pointed out by Li [32], there were two obstacles in [46] which prevented them from getting $O(\log a)$ dependence. We explain each of them in the next two paragraphs, respectively.

G. Preparing seeds for sub-protocols

The strategy we described above runs many independence merging protocols in parallel for $\log(a)$ rounds. In the i^{th} round Quentin needs to prepare a seed $\mathbf{Q}_i$ with enough entropy conditioned on the transcripts. The strategy in [46] is to take a prefix of $\mathbf{X}$ as $\mathbf{Q}_i$ in each round. However, suppose in the first round Quentin sends a prefix $\mathbf{Q}_1$. Then there are t tampered messages $\mathbf{Q}_1^{[t]}$ sent at the same time. Therefore $\mathbf{X}$ loses about $(t+1)|\mathbf{Q}_1|$ entropy. In the next round, to ensure that the prefix $\mathbf{Q}_2$ of $\mathbf{X}$ still has some entropy, the length of $\mathbf{Q}_2$ needs to be $O(t|\mathbf{Q}_1|)$. Therefore the entropy requirement

for $\mathbf{X}$ grows exponentially in the number of rounds. To solve this problem, Li [32] observed that they can first run an ℓ-look-ahead extractor which is strong in $\mathbf{Y}$ to help Quentin prepare $\ell = O(\log a)$ look-ahead strings $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_\ell$. Then in the i^{th} round Quentin simply sends $\mathbf{Q}_i$.

In our setting of linearly correlated sources (discussed below in Section III-I), we find it slightly cumbersome to define a look-ahead protocol which is strong in $\mathbf{Y}$ because of the linear correlation between sources. Therefore, we take a slightly different strategy: we construct an NIPM in a way such that it can take a uniform seed $\mathbf{Q}_1$ and merge every two *weak* sources $(\mathbf{V}_1, \mathbf{V}_2), (\mathbf{V}_3, \mathbf{V}_4), \ldots, (\mathbf{V}_{2a-1}, \mathbf{V}_{2a})$ into uniform strings $\mathbf{W}_1, \ldots, \mathbf{W}_a$. Then before the start of next round, we take a prefix $\mathbf{P}$ of $\mathbf{W}_1$ and extract $\mathbf{Q}_2 = \mathrm{Ext}(\mathbf{X}, \mathbf{P})$. If $\mathbf{P}$ is short compared to each $\mathbf{W}_i$, then each $\mathbf{W}_i$ should still have enough entropy, and the merging process can continue. Essentially this is like running the look-ahead protocol "on the fly."

H. Entropy recycling

Second, when merging a block $(\mathbf{V}_1, \mathbf{V}_2)$ into a single source $\mathbf{W}$ using the alternating extraction protocol described above, the length of $\mathbf{W}$ is only $\beta|\mathbf{V}_1|$ for some constant $\beta < 1$. Therefore the entropy requirement for $\mathbf{Y}$ also grows exponentially in the number of rounds. To solve this problem, Li [33] observed that one can try to "recycle entropy" from $\mathbf{Y}$, which actually contains all the entropy Wendy currently has. Since $\mathbf{Y}$ can contain entropy much larger than $|\mathbf{V}_1|$, it is possible to restore the length of $\mathbf{W}$ to $|\mathbf{V}_1|$. Li [33] did this by using an extra seed from Quentin as a buffer to extract from $\mathbf{Y}$.

However, as we pointed out in the previous paragraph, preparing multiple seeds for Quentin using a look-ahead extractor is cumbersome in the linearly correlated source setting. Therefore we choose to embed this approach into each NIPM sub-protocol. That is, we change the protocol $\mathrm{NIPM}(\mathbf{Q}, (\mathbf{V}_1, \mathbf{V}_2))$ to $\mathrm{NIPM}_{rec}(\mathbf{Q}, (\mathbf{V}_1, \mathbf{V}_2, \mathbf{Y}))$ in the following way. First, run the original $\mathrm{NIPM}(\mathbf{Q}, (\mathbf{V}_1, \mathbf{V}_2))$ to get $\mathbf{P}$ which merges the independence of $\mathbf{V}_1$ and $\mathbf{V}_2$. Suppose $\mathbf{P}$ is computed on Wendy's side. Then Wendy sends $\mathbf{P}$ to Quentin, Quentin sends $\mathbf{S} = \mathrm{Ext}(\mathbf{Q}, \mathbf{P})$ to Wendy, and Wendy computes $\mathbf{W} = \mathrm{Ext}(\mathbf{Y}, \mathbf{S})$ as output. Note that this protocol is still strong in $\mathbf{X}$. Moreover, if $\mathbf{P}$ is uniform conditioned on $\mathbf{P}^T$ for some $T \subseteq [t]$, then based on the independence preserving property of strong seeded extractors, $\mathbf{W}$ should still be uniform conditioned on $\mathbf{W}^T$. Therefore if the entropy of $\mathbf{Q}$ can afford one more round of alternating extraction, NIPM_{rec} preserves every property we want for NIPM, but it also has output length $|\mathbf{W}| = |\mathbf{V}_1|$.

I. Two-party games for linearly correlated sources

Finally we state how to modify the definition of correlation-breaking games to work for $(\mathbf{X} = \mathbf{A}+\mathbf{B}, \mathbf{Y})$ where $(\mathbf{A}, \mathbf{A}^{[t]})$ is independent of $(\mathbf{B}, \mathbf{B}^{[t]}, \mathbf{Y}, \mathbf{Y}^{[t]})$, $\mathbf{A}$ has some entropy and $\mathbf{Y}$ is uniform. In this modified game, Quentin holds $\mathbf{A}$, and Wendy holds $(\mathbf{B}, \mathbf{Y})$. Furthermore, there are two transcripts (and their t tampered versions): the normal transcript $\mathbf{Z}$, and a "write-only" transcript $\mathbf{Z}_B$. Then Quentin and Wendy run the game as if they are simulating a two-party communication between $\mathbf{X}$ and $\mathbf{Y}$. This works as follows. First, whenever Quentin wants to send or output $\mathbf{Q} = f(\mathbf{X}, \mathbf{Z})$, f must be a linear function for any fixed $\mathbf{Z}$, and Wendy must send $\mathbf{Q}_B = f(\mathbf{B}, \mathbf{Z})$ first. In addition, $\mathbf{Q}_B$ will be added to the write-only transcript $\mathbf{Z}_B$. After receiving $\mathbf{Q}_B$, Quentin sends or outputs $\mathbf{Q} = f(\mathbf{A}, \mathbf{Z}) + \mathbf{Q}_B$, and then $\mathbf{Q}$ is added to the transcript $\mathbf{Z}$. Second, Wendy's message should always be in the form $g(\mathbf{Y}, \mathbf{Z})$ for some deterministic function g, which means Wendy doesn't have access to $\mathbf{B}$ (except when helping Quentin compute $f(\mathbf{X}, \mathbf{Z})$). Then we want the output $\mathbf{R}$ to be uniform conditioned on $(\mathbf{R}^{[t]}, \mathbf{Z}, \mathbf{Z}^{[t]}, \mathbf{Z}_B, \mathbf{Z}_B^{[t]})$. Note that $(\mathbf{A}, \mathbf{A}^{[t]}) \leftrightarrow (\mathbf{Z}, \mathbf{Z}^{[t]}, \mathbf{Z}_B, \mathbf{Z}_B^{[t]}) \leftrightarrow (\mathbf{Y}, \mathbf{Y}^{[t]}, \mathbf{B}, \mathbf{B}^{[t]})$ is always a Markov chain.

Effectively, this modified game is similar to a normal game between $\mathbf{A}$ and $\mathbf{Y}$, except for two things. First, f must be linear. In our construction we actually make sure that every function f Quentin uses is simply a strong linear seeded extractor. Second, whenever Quentin wants to send a message to help Wendy run some protocols, Wendy is forced to send $\mathbf{Q}_B$ which leaks some information about her source. This means every source Wendy holds in her hand loses some entropy. Nevertheless, we will make sure that every sub-protocol Wendy simulates still works even if she only has weak sources. Finally, another slight difference on Wendy's side is she can only run some deterministic function using $\mathbf{Q} = \mathbf{Q}_A + \mathbf{Q}_B$ instead of $\mathbf{Q}_A$. But this is not a problem, since $\mathbf{Q}_B$ is independent of $\mathbf{Q}_A$ conditioned on the transcripts, which means the conditional entropy of $\mathbf{Q}$ is the same as $\mathbf{Q}_A$. Moreover, $\mathbf{Q}$ is still independent from Wendy's side since $\mathbf{Q}_B$ is already in the transcript.

J. Summary of our correlation breaker construction

Finally, we summarize the construction of our correlation breaker using the two-party game from Section III-I. We use the following building blocks:

- A strong linear seeded extractor LExt, which is the "condense-then-hash" extractor.
- A 2-look-ahead extractor laExt, which takes a weak source $\mathbf{Y}$ and an independent uniform seed $\mathbf{Q}$, then outputs two strings $(\mathbf{R}_0, \mathbf{R}_1)$ such that $\mathbf{R}_1$ is uniform conditioned on $(\mathbf{R}_0, \mathbf{R}_0^{[t]})$.
- An NIPM which takes two weak sources $\mathbf{V}_1, \mathbf{V}_2$, an entropy pool $\mathbf{Y}$ and an independent uniform seed $\mathbf{Q}$, then outputs a string $\mathbf{W}$ which merges the independence of $\mathbf{V}_1, \mathbf{V}_2$ from their tamperings.

Given a source $\mathbf{X} = \mathbf{A} + \mathbf{B}$, a uniform seed $\mathbf{Y}$, their tamperings $\mathbf{A}^{[t]}, \mathbf{B}^{[t]}, \mathbf{Y}^{[t]}$ such that $(\mathbf{A}, \mathbf{A}^{[t]})$ is independent of $(\mathbf{B}, \mathbf{B}^{[t]}, \mathbf{Y}, \mathbf{Y}^{[t]})$, an advice $\alpha \in \{0,1\}^a$ and the tampered advice $\alpha^{[t]} \neq \alpha$, the construction works as follows:

1) Wendy sends $\mathbf{W}_0$ which is a prefix of $\mathbf{Y}$.
2) Wendy sends $\mathbf{Q}_{0B} = \mathrm{LExt}(\mathbf{B}, \mathbf{W}_0)$, then Quentin sends $\mathbf{Q}_0 = \mathrm{LExt}(\mathbf{X}, \mathbf{W}_0) = \mathrm{LExt}(\mathbf{A}, \mathbf{W}_0) + \mathbf{Q}_{0B}$.
3) Wendy computes $(\mathbf{R}_1, \mathbf{R}_0) = \mathrm{laExt}(\mathbf{Y}, \mathbf{Q}_0)$, and gets a sequence of $2a$ somewhere-independent strings $\mathbf{V} = (\mathbf{V}_1, \ldots, \mathbf{V}_{2a})$ by assigning each string to be $\mathbf{R}_0$ or $\mathbf{R}_1$ based on α (see the discussion in Section III-F).
4) Repeat the following steps for i from 1 to $\log(2a)$:
   I Wendy sends $\mathbf{W}_i$ which is a prefix of $\mathbf{V}_1$.
   II Wendy sends $\mathbf{Q}_{iB} = \mathrm{LExt}(\mathbf{B}, \mathbf{W}_i)$, then Quentin sends $\mathbf{Q}_i = \mathrm{LExt}(\mathbf{A}, \mathbf{W}_i) + \mathbf{Q}_{iB}$.
   III Wendy merges each pair $(\mathbf{V}_{2j-1}, \mathbf{V}_{2j})$ into a single string $\mathbf{V}_j$ with the NIPM, using $\mathbf{Q}_i$ as the uniform seed and $\mathbf{Y}$ as the entropy pool. Note that the number of strings in $\mathbf{V}$ decreases by a factor of 2 after this step.
5) Now there is only one string in $\mathbf{V}$. Output $\mathbf{V}$, which is uniform conditioned on $\mathbf{V}^{[t]}$.

Note the construction above has the following features:

- The only message from Quentin is the uniform seed $\mathbf{Q}_i$ in each round, which is computed by a strong linear seed extractor. The length of $\mathbf{Q}_i$ should be $O(t \log(n))$ for each sub-protocol to work.
- In each round Wendy sends a message $\mathbf{W}_i$ to be used as the seed of Quentin's extraction, and also $\mathbf{Q}_{iB}$ to help Quentin compute $\mathbf{Q}_i$. Both messages have length $O(|\mathbf{Q}_i|)$ and cause the sources in Wendy's hand to lose $O(t\,|\mathbf{Q}_i|)$ bits of entropy. However, both the look-ahead extractor and the NIPM still work even if Wendy only has weak sources.

Finally, observe that in each of the $O(\log(a))$ rounds, both parties need to send a message of length $O(t \log(n))$. Since in each round there are t tampered messages sent simultaneously, the entropy requirement of each side is $O(t^2 \log(a) \log(n))$. Moreover, the length of each $|\mathbf{V}_j|$ needs to be $O(t\,|\mathbf{Q}_i|) = O(t^2 \log(n))$ to tolerate the entropy loss in each round, so there is an extra $O(t^3 \log(n))$ entropy requirement on $\mathbf{Y}$ in the look-ahead extractor. Nevertheless, this is dominated by $O(t^2 \log(a) \log(n))$ in our application.

ACKNOWLEDGEMENT

We thank David Zuckerman for discussions about affine extractors, which led to this work. We thank FOCS reviewers for many helpful comments. We thank Jason Gaitonde for helpful discussions.

Eshan Chattopadhyay, Jesse Goodman and Jyun-Jie Liao are supported by NSF CAREER award 2045576.

REFERENCES

[1] J. von Neumann, "Various techniques used in connection with random digits," *Applied Math Series*, vol. 12, pp. 36–38, 1951, notes by G.E. Forsythe, National Bureau of Standards. Reprinted in *Von Neumann's Collected Works*, 5:768-770, 1963.

[2] N. Nisan and D. Zuckerman, "Randomness is linear in space," *Journal of Computer and System Sciences*, vol. 52, pp. 43–52, 1996.

[3] C.-J. Lu, O. Reingold, S. Vadhan, and A. Wigderson, "Extractors: Optimal up to constant factors," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 2003, pp. 602–611.

[4] V. Guruswami, C. Umans, and S. P. Vadhan, "Unbalanced expanders and randomness extractors from parvaresh-vardy codes," *J. ACM*, vol. 56, pp. 20:1–20:34, 2009.

[5] Z. Dvir, S. Kopparty, S. Saraf, and M. Sudan, "Extensions to the method of multiplicities, with applications to kakeya sets and mergers," *SIAM Journal on Computing*, vol. 42, pp. 2305–2328, 2013.

[6] R. Shaltiel, "Recent developments in explicit constructions of extractors," in *Current Trends in Theoretical Computer Science: The Challenge of the New Century*. World Scientific, 2004, pp. 189–228.

[7] B. Chor, O. Goldreich, J. Hasted, J. Freidmann, S. Rudich, and R. Smolensky, "The bit extraction problem or t-resilient functions," in *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. IEEE, 1985, pp. 396–407.

[8] L. Trevisan and S. Vadhan, "Extracting randomness from samplable distributions," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000, pp. 32–42.

[9] J. Kamp and D. Zuckerman, "Deterministic extractors for bit-fixing sources and exposure-resilient cryptography," *SIAM Journal on Computing*, vol. 36, pp. 1231–1247, 2007.

[10] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity," *SIAM J. Comput.*, vol. 17, pp. 230–261, 1988.

[11] B. Barak, R. Impagliazzo, and A. Wigderson, "Extracting randomness using few independent sources," *SIAM Journal on Computing*, vol. 36, pp. 1095–1118, 2006.

[12] A. Rao, "Extractors for low-weight affine sources," in *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009*. IEEE Computer Society, 2009, pp. 95–101.

[13] E. Viola, "Extractors for circuit sources," *SIAM J. Comput.*, vol. 43, pp. 655–672, 2014.

[14] E. Chattopadhyay and J. Goodman, "Improved extractors for small-space sources," in *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, 2021, to appear.

[15] J. Kamp, A. Rao, S. Vadhan, and D. Zuckerman, "Deterministic extractors for small-space sources," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 2006, pp. 691–700.

[16] M. Blum, "Independent unbiased coin flips from a correlated biased source—a finite state markov chain," *Combinatorica*, vol. 6, pp. 97–108, 1986.

[17] E. Ben-Sasson and N. Ron-Zewi, "From affine to two-source extractors via approximate duality," *SIAM Journal on Computing*, vol. 44, pp. 1670–1697, 2015.

[18] G. Cohen and A. Tal, "Two structural results for low degree polynomials and applications," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015*, 2015, pp. 680–709.

[19] E. Demenkov and A. S. Kulikov, "An elementary proof of a 3n-o (n) lower bound on the circuit complexity of affine dispersers," in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 2011, pp. 256–265.

[20] M. G. Find, A. Golovnev, E. A. Hirsch, and A. S. Kulikov, "A better-than-3n lower bound for the circuit complexity of an explicit function," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, 2016, pp. 89–98.

[21] J. Li and T. Yang, "3.1n-o(n) circuit lower bounds for explicit functions." in *Electron. Colloquium Comput. Complex.*, vol. 28, 2021, p. 23.

[22] A. Gabizon and R. Raz, "Deterministic extractors for affine sources over large fields," *Combinatorica*, vol. 28, pp. 415–440, 2008.

[23] J. Bourgain, Z. Dvir, and E. Leeman, "Affine extractors over large fields with exponential error," *computational complexity*, vol. 25, pp. 921–931, 2016.

[24] J. Bourgain, "On the construction of affine extractors," *GAFA Geometric And Functional Analysis*, vol. 17, pp. 33–57, 2007.

[25] A. Yehudayoff, "Affine extractors over prime fields," *Combinatorica*, vol. 31, pp. 245–256, 2011.

[26] X. Li, "A new approach to affine extractors and dispersers," in *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011*. IEEE Computer Society, 2011, pp. 137–147.

[27] M. DeVos and A. Gabizon, "Simple affine extractors using dimension expansion," in *2010 IEEE 25th Annual Conference on Computational Complexity*. IEEE, 2010, pp. 50–57.

[28] X. Li, "Improved two-source extractors, and affine extractors for polylogarithmic entropy," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*. IEEE Computer Society, 2016, pp. 168–177.

[29] E. Chattopadhyay and D. Zuckerman, "Explicit two-source extractors and resilient functions," *Annals of Mathematics*, vol. 189, pp. 653–705, 2019.

[30] A. Ben-Aroya, D. Doron, and A. Ta-Shma, "An efficient reduction from two-source to nonmalleable extractors: achieving near-logarithmic min-entropy," *SIAM Journal on Computing*, pp. STOC17–31, 2019.

[31] G. Cohen, "Towards optimal two-source extractors and ramsey graphs," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*. ACM, 2017, pp. 1157–1170.

[32] X. Li, "Improved non-malleable extractors, non-malleable codes and independent source extractors," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*. ACM, 2017, pp. 1144–1156.

[33] ——, "Non-malleable extractors and non-malleable codes: Partially optimal constructions," in *34th Computational Complexity Conference, CCC 2019*, ser. LIPIcs, vol. 137. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 28:1–28:49.

[34] D. Aggarwal, H. Bennett, A. Golovnev, and N. Stephens-Davidowitz, "Fine-grained hardness of CVP(P) - everything that we can prove (and nothing else)," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*. SIAM, 2021, pp. 1816–1835.

[35] E. Chattopadhyay and J.-J. Liao, "Extractors for sum of two sources," *arXiv preprint arXiv:2110.12652*, 2021.

[36] E. Chattopadhyay, J. Goodman, and J. Liao, "Affine extractors for almost logarithmic entropy," *Electron. Colloquium Comput. Complex.*, p. 75, 2021.

[37] X. Li, "Extractors for a constant number of independent sources with polylogarithmic min-entropy," in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*. IEEE Computer Society, 2013, pp. 100–109.

[38] G. Cohen, "Local correlation breakers and applications to three-source extractors and mergers," *SIAM J. Comput.*, vol. 45, pp. 1297–1338, 2016.

[39] S. Dziembowski and K. Pietrzak, "Intrusion-resilient secret sharing," in *48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007*. IEEE Computer Society, 2007, pp. 227–237.

[40] R. Meka, "Explicit resilient functions matching ajtai-linial," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*. SIAM, 2017, pp. 1132–1148.

[41] E. Chattopadhyay, V. Goyal, and X. Li, "Non-malleable extractors and codes, with their many tampered extensions," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2016, pp. 285–298.

[42] L. Trevisan, "Extractors and pseudorandom generators," *J. ACM*, vol. 48, pp. 860–879, 2001.

[43] R. Raz, O. Reingold, and S. P. Vadhan, "Extracting all the randomness and reducing the error in trevisan's extractors," *J. Comput. Syst. Sci.*, vol. 65, pp. 97–128, 2002.

[44] M. Cheraghchi, "Applications of derandomization theory in coding," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2010.

[45] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM J. Comput.*, vol. 28, pp. 1364–1396, 1999.

[46] E. Chattopadhyay and X. Li, "Explicit non-malleable extractors, multi-source extractors, and almost optimal privacy amplification protocols," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*. IEEE Computer Society, 2016, pp. 158–167.

[47] G. Cohen, "Making the most of advice: New correlation breakers and their applications," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*. IEEE Computer Society, 2016, pp. 188–196.

[48] M. Ajtai and N. Linial, "The influence of large coalitions," *Combinatorica*, vol. 13, pp. 129–145, 1993.

[49] Y. Dodis and D. Wichs, "Non-malleable extractors and symmetric key cryptography from weak secrets," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*. ACM, 2009, pp. 601–610.

[50] G. Cohen, "Non-malleable extractors - new tools and improved constructions," in *31st Conference on Computational Complexity, CCC 2016*, ser. LIPIcs, vol. 50. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 8:1–8:29.

[51] G. Cohen and L. J. Schulman, "Extractors for near logarithmic min-entropy," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*. IEEE Computer Society, 2016, pp. 178–187.