Thesis for the degree
Doctor of Philosophy

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

עבודת גמר (תזה) לתואר
דוקטור לפילוסופיה

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

By
Gil Cohen

מאת
גיל כהן

בניות חדשות ושימושים של עצמים פסאודו-אקראיים
New Constructions and Applications of Pseudorandom Objects

Advisor: Ran Raz

מנחה: רן רז

April 2015

ניסן תשע"ה

Dedicated to Orit and Yahli

Abstract

Pseudorandomness is the subfield of theoretical computer science which studies explicit constructions of objects that share desired properties with random objects. Results and techniques from pseudorandomness have found applications in many research fields, such as privacy, cryptography, coding theory, and data structures.

In this thesis we study pseudorandomness from both ends. First, we give several novel constructions of pseudorandom objects, such as three-source extractors, non-malleable extractors and pseudorandom generators for low-degree polynomials. Second, we apply known and original results from the field for solving fundamental problems in other research areas, such as secure multiparty computation, privacy amplification, and circuit lower bounds. In some cases, a priori, the connection to pseudorandomness is unclear.

Acknowledgements

Towards the end of my undergraduate studies at the Technion, I was participating in a reading group on Ran Raz's parallel repetition theorem. In the midst of the proof, I recall wondering how would it be like to work with a person that is able to produce such a tour de force result? Luckily for me, this is one open problem solved! Having Ran as my advisor was a priceless opportunity for me to learn how truly great research is done. Over the years, I learned from Ran (sometimes in the hard way) never to forget asking myself which problems are worthy of attack; the realization that it is not about finding a solution but about finding the right solution; and the evasive equilibrium between limitless technical strength and a single pinpointed razor-sharp observation. I am certain that having Ran as my advisor made me a better researcher than I would have ever been otherwise. I wish to thank Ran for his faith in me, for giving me the freedom and courage to pave my own research path, for the enlightening and enjoyable discussions we had over the years, either at Weizmann or at IAS (well, at "small world coffee" to be precise), and most of all for his friendship and genuine care.

I would like to thank Oded Goldreich, Robert Krauthgamer and Adi Shamir for serving as my PhD committee, and for their advice during the past few years.

In the last couple of years I had the privilege to work with Amnon Ta-Shma – my algebraic geometry codes mentor. Learning the subject from Amnon at first hand and working together on fascinating problems in the field were extremely enjoyable and challenging.

During graduate school, I had an enjoyable and fruitful internship as an intern at MSR Silicon Valley, hosted by Guy Rothblum. I wish to thank Guy for his warm hospitality and guidance and for many insightful discussions. During the internship we had many interesting discussions with Raghu Meka and Omer Reingold for whom I wish to thank.

I had the fortune to work with many gifted researchers and it was interesting to see again and again how each great mind is great in its own unique way. Some of the collaborations ended up in a nicely wrapped paper, but for me this is merely a nice bonus. Special thanks to Noga Alon, Avraham Ben-Aroya, Itai Benjamini, Anat Ganor, Yuval Ishai, Raghu Meka, Ran Raz, Omer Reingold, Ron D. Rothblum, Guy Rothblum, Gil Segev, Igor Shinkar, Avishay Tal, and Amnon Ta-Shma for sharing their knowledge and wisdom with me.

I wish to thank the theory group at Weizmann for setting a great research atmosphere. In particular, I am thankful to Irit Dinur, Oded Goldreich and Shafi Goldwasser, which had a strong affect, even if indirect, on my work.

At least a dozen of promising yet failed ideas separate between each pair of fruitful research ideas, and these dozen failures can be very discouraging. In such times, especially, which have accumulated to a $1 - o(1)$ fraction of my graduate studies, I am grateful to

have a wonderful wife and son that keep me on track thanks to their love and faith in me. I cannot thank Orit and Yahli enough for their understanding of my attitude towards research – it is a common theme that at a festive family dinner, I wonder off thinking about the current problem which I am taken by. When I “return”, there is always a nice dessert on the plate and beautiful smiles on their faces

Contents

Chapter 1

Introduction

In combinatorics, and to some extent in mathematics in general, one often wishes to prove the existence of an object with some desired properties. It is usually preferred to have some explicit description of such an object, if it exists. A classical, perhaps pre-computational, interpretation of the word "explicit" is one in which the object is described using a succinct mathematical formula.

In his three pages seminal paper, Erdös [Erd47] lays down one of the first, and certainly most influential example of the probabilistic method. This method enables one to prove the existence of an object by probabilistic arguments. The outcome is a deterministic no-error proof for the existence of the object, even though the argument itself is partially based on a probabilistic reasoning. The strength of the probabilistic method usually lies in its simplicity, yet the method comes with an inherent cost – although the existence of the desired object is proven, one usually gains almost no information about how such an object might look like, and certainly an explicit description cannot be extracted.

Explicitness got a new meaning in the computational era. While, classically, only a succinct mathematical formula was considered to be an explicit description, computer science suggests a more relaxed, and arguably more natural interpretation. An object is explicit if one can construct that object from scratch using a small amount of resources, where the central resource is, as always in life, time. To be a bit more focused, an explicit construction of an object is an efficient deterministic algorithm that produces this object.

Pseudorandomness is the subfield of theoretical computer science which studies explicit constructions, in the computational interpretation discussed above, of objects with properties that are shared by random objects. That is, roughly speaking, in pseudorandomness one wants to design efficient deterministic algorithms that produce desired objects, which are guaranteed to exist by the probabilistic method. This can also be interpreted as understanding the role of randomness in computation, as in most cases, the probabilistic method gives an efficient randomized algorithm for the construction of desired objects. Pseudorandomness has found applications in many research fields, such as computation complexity, privacy, cryptography, coding theory and data structures.

This thesis covers works that were done during the graduate studies of the author. Each of these works either presents a novel construction of a pseudorandom object (that

is, a contribution to the field of pseudorandomness) or otherwise gives an application of results and techniques from pseuodrandomness to other fields. In the next section we give a brief account to each of these works.

1.1 Our Results

In this section we briefly review for the main results that are included in this thesis.

1.1.1 Non-malleable extractors and applications to privacy amplification protocols

Non-malleable extractors were introduced by Dodis and Wichs [DW09], motivated by the classical problem of privacy amplification. Informally speaking, a non-malleable extractor is a seeded extractor with a very strong pseudorandom property – the output of a non-malleable extractor obtained using a typical seed does not reveal information about the output that one would get using any different seed. In [DW09] it is shown how to simplify and significantly improve upon a long line of research on privacy amplification using a non-malleable extractor, though the construction of the latter was left as an open problem.

In joint work with Raz and Segev [CRS14] we give the first unconditional construction of a non-malleable extractor that outputs more than a logarithmic number of output bits, as required by privacy amplification protocols. Prior to this work, the only known construction [DLWZ11a] was based on a long standing conjecture on prime numbers in order to achieve this result. Moreover, unlike the work of [DLWZ11a], that required a seed of linear length, our extractor uses a seed of logarithmic length. This in turn significantly improves the communication complexity of the privacy amplification protocol. Chapter 3 covers this work.

1.1.2 Local correlation breakers and applications to multi-source extractors and mergers

In Chapter 4, which covers the paper [Coh15], we introduce and construct a pseudorandom object that we call a *local correlation breaker (LCB)*. Informally speaking, an LCB is a function that gets as input a sequence of r (arbitrarily correlated) random variables and an independent weak-source. The output of the LCB is a sequence of r random variables with the following property. If the i^{th} input random variable is uniform then the i^{th} output variable is uniform even given a bounded number of any other output variables. That is, an LCB uses the weak-source to "break" local correlations between random variables. Using our construction of LCBs, we obtain several new and improved constructions of certain pseudorandom objects which we now briefly describe.

- We construct a three-source extractor where one of the sources is only assumed to have a double-logarithmic entropy. More precisely, for any integer n and con-

stant $\delta > 0$, we construct a three-source extractor for entropies δn, $O(\log n)$ and $O(\log \log n)$. This result improves the three-source extractor of Raz [Raz05] and is incomparable with the recent three-source extractor by Li [Li15]. As the third source is required to have tantalizingly low entropy, we hope that further ideas can be used to eliminate the need for this source altogether, solving a major open problem.

- We construct a merger with weak-seeds that merges r random variables using an independent (n, k)-weak-source with $k = \widetilde{O}(r) \cdot \log \log n$. A previous construction, that appears in the celebrated paper by Barak *et al.* [BRSW12], assumes $k \geq \Omega(r^2) + \text{polylog}(n)$.

- We introduce the notion of a *two-source non-malleable extractor*. This is a relaxation of a non-malleable extractor, discussed in Chapter 3, where two weak-sources, rather than one, are given. We construct a two-source non-malleable extractor for entropy $O(\log^2 n)$ with a logarithmic seed length. This should be compared with "standard" non-malleable extractors for which the known explicit constructions require entropy roughly $n/2$.

1.1.3 Zero-fixing extractors for sub-logarithmic entropy

An (n, k)-bit-fixing source [Vaz85, CGH+85, BBR85] is a distribution on n bit strings, that is fixed on $n - k$ of the coordinates, and jointly uniform on the remaining k bits. Explicit constructions of bit-fixing extractors by Gabizon, Raz and Shaltiel [GRS06] and Rao [Rao09b], extract $(1 - o(1)) \cdot k$ bits for $k = \text{polylog}(n)$, almost matching the probabilistic argument. Intriguingly, unlike other well-studied sources of randomness, a result of Kamp and Zuckerman [KZ06] shows that, for *any* k, some small portion of the entropy in an (n, k)-bit-fixing source can be extracted. Although the extractor does not extract all the entropy, it does extract $0.5 \log k$ bits.

In a joint work with Igor Shinkar [CS15], covered in Chapter 5, we prove that when the entropy k is small enough compared to n, this exponential entropy-loss is unavoidable. More precisely, one cannot extract more than $0.5 \log(k) + O(1)$ bits from (n, k)-bit-fixing sources. The remaining entropy is inaccessible, information theoretically. By the Kamp-Zuckerman construction, this negative result is tight. For small enough k, this strengthens a result by Reshef and Vadhan [RV13], who proved a similar bound for extractors computable by space-bounded streaming algorithms.

Our impossibility result also holds for what we call *zero-fixing* sources. These are bit-fixing sources where the fixed bits are set to 0. We partially complement our negative result, by giving an explicit construction of an (n, k)-zero-fixing extractor, that outputs $\Omega(k)$ bits, even for $k = \text{poly} \log \log n$.

Finally, we give a construction of an (n, k)-bit-fixing extractor, that outputs $k - O(1)$ bits, for entropy $k = (1 + o(1)) \cdot \log \log n$, with running-time $n^{O((\log \log n)^2)}$. This answers an open problem by Reshef and Vadhan [RV13].

1.1.4 Efficient multiparty protocols via log-depth threshold formulae

Secure multiparty computation (MPC) enables a set of parties to accomplish distributed computational tasks, while maintaining the secrecy of the inputs and the correctness of the outputs in the presence of coalitions of dishonest parties. Originated from the seminal works of [Yao82a, GMW87, BGW88, CCD88], secure MPC is a classical problem in cryptography that has been the subject of an enormous body of work. Nevertheless, MPC protocols remained quite complicated and their security was difficult to prove. In Chapter 6, based on a joint work with Damgård *et al.* [CDI+13], we propose a new modular approach for the construction of MPC protocols. Using our approach we obtain conceptually simple proofs for known classical results in cryptography [BGW88, CCD88] and in distributed computing [PSL80, Dol82], and resolve several open problems.

Our approach is to reduce the problem of constructing MPC protocols for n players to constructing MPC protocols for just a constant number of players – a significantly easier task. To this end, we rely on a seemingly unrelated object from complexity theory – a log-depth formula, composed of constant fan-in threshold gates, that computes a threshold function. We use techniques and results from pseudorandomness, such as expander graphs and pseudorandom generators for read-once branching programs, to give deterministic constructions of such formulas.

The new approach we put forward for the design of efficient MPC protocols is as follows:

1. Design a protocol π for a small number of parties (say, 3 or 4) which achieves security against a *single* corrupted party. Such protocols are typically easy to construct, as they may employ techniques that do not scale well with the number of corrupted parties.

2. Recursively compose π with itself to obtain an efficient n-party protocol which achieves security against a constant fraction of corrupted parties.

The second step of our approach combines the "player emulation" technique of Hirt and Maurer [HM00] with constructions of logarithmic-depth formulae which compute threshold functions using only constant fan-in threshold gates. Using this approach, we simplify and improve on previous results in cryptography and distributed computing. In particular:

- We provide conceptually simple constructions of efficient protocols for Secure MPC in the presence of an honest majority, as well as broadcast protocols from point-to-point channels and a 2-cast primitive.

- We obtain new results on MPC over blackbox groups and other algebraic structures.

The above results rely on the following complexity-theoretic contributions, which we believe is of independent interest:

- We show that for every $j, k \in \mathbb{N}$ such that $m \triangleq \frac{k-1}{j-1}$ is an integer, there is an explicit (poly(n)-time) construction of a logarithmic-depth formula which computes a good approximation of an (n/m)-out-of-n threshold function using only j-out-of-k threshold gates and no constants.

- For the special case of n-bit majority from 3-bit majority gates, a non-explicit construction follows from the work of Valiant [Val77]. For this special case, we provide an explicit construction with a better approximation than for the general threshold case, and also an *exact* explicit construction based on standard complexity-theoretic or cryptographic assumptions.

1.1.5 On rigid matrices and U-polynomials

In a joint work with Alon [AC13], covered in Chapter 7, we introduce a class of polynomials, which we call *U-polynomials* and show that the problem of explicitly constructing a rigid matrix can be reduced to the problem of explicitly constructing a small hitting set for this class. We prove that small-bias sets are hitting sets for the class of U-polynomials, though their size is larger than desired. Furthermore, we give two alternative proofs for the fact that small-bias sets induce rigid matrices.

Further, we construct rigid matrices from unbalanced expanders, with essentially the same size as the construction via small-bias sets.

1.1.6 Gradual small-bias sample spaces

A (k, ε)-biased sample space is a distribution over $\{0,1\}^n$ that ε-fools every nonempty linear test of size at most k. Since they were introduced by Naor and Naor [NN93], these sample spaces have become a central notion in theoretical computer science with a variety of applications. When constructing such spaces, one usually attempts to minimize the seed length as a function of n, k and ε. Given such a construction, if we reverse the roles and consider a fixed seed length, then the smaller we pick k, the better the bound on the bias ε becomes. However, once the space is constructed we have a *single* bound on the bias of all tests of size at most k.

In a joint work with Ben-Aroya [BAC12], covered in Chapter 8, we initiate the study of a new pseudorandom object, which we call a *gradual* (k, ε)-biased sample space. Roughly speaking, this is a sample space that ε-fools linear tests of size *exactly* k and moreover, the bound on the bias for linear tests of size $i \leq k$ decays as i gets smaller. We show how to construct gradual (k, ε)-biased sample spaces of size comparable to the (non-gradual) spaces constructed by Alon *et al.* [AGHP92], and prove a lower bound on their size. Our construction is based on the lossless expanders of Guruswami *et al.* [GUV09], combined with the Quadratic Character Construction of Alon *et al.* [AGHP92].

1.2 Results Not Included in This Thesis

In this section we give a brief account for results that were obtained during the author's graduate studies, that we chose not to include in this thesis.

1.2.1 Bi-Lipschitz bijection between the Boolean cube and the Hamming ball

In a joint work with Benjamini and Shinkar [BCS14] we construct a bi-Lipschitz bijection from the Boolean cube to the Hamming ball of equal volume. More precisely, we show that for all even $n \in \mathbb{N}$ there exists an explicit bijection $\psi\colon \{0,1\}^n \to \{x \in \{0,1\}^{n+1}\colon |x| > n/2\}$ such that for every $x \neq y \in \{0,1\}^n$ it holds that

$$\frac{1}{5} \leq \frac{\mathsf{distance}(\psi(x),\psi(y))}{\mathsf{distance}(x,y)} \leq 4,$$

where $\mathsf{distance}(\cdot,\cdot)$ denotes the Hamming distance. In particular, this implies that the Hamming ball is bi-Lipschitz transitive.

This result gives a strong negative answer to an open problem of Lovett and Viola [LV12], who raised the question in the context of sampling distributions in low-level complexity classes. The conceptual implication is that the problem of proving lower bounds in the context of sampling distributions requires ideas beyond the sensitivity-based structural results of Boppana [Bop97].

We study the mapping ψ further and show that it (and its inverse) are computable in DLOGTIME-uniform TC^0, but not in AC^0. Moreover, we prove that ψ is "approximately local" in the sense that all but the last output bit of ψ are essentially determined by a single input bit.

1.2.2 Two sides of the coin problem

In the *coin problem*, one is given n independent flips of a coin that has bias $\beta > 0$ towards either Head or Tail. The goal is to decide which side the coin is biased towards, with high confidence. An optimal strategy for solving the coin problem is to apply the majority function on the n samples. This simple strategy works as long as $\beta > \Omega(1/\sqrt{n})$. However, computing majority is an impossible task for several natural computational models, such as bounded width read once branching programs and $\mathbf{AC^0}$ circuits.

Brody and Verbin [BV10] proved that a length n, width w read once branching program cannot solve the coin problem for $\beta < O(1/(\log n)^w)$. This result was tightened by Steinberger [Ste13] to $O(1/(\log n)^{w-2})$. The coin problem in the model of $\mathbf{AC^0}$ circuits was first studied by Shaltiel and Viola [SV10], and later by Aaronson [Aar10] who proved that a depth d size s Boolean circuit cannot solve the coin problem for $\beta < O(1/(\log s)^{d+2})$.

In a joint work with Ganor and Raz [CGR14] we obtained the following results:

- We strengthen Steinberger result and show that any Santha-Vazirani source with bias $\beta < O(1/(\log n)^{w-2})$ fools length n, width w read once branching programs. In other words, the strong independence assumption in the coin problem is completely redundant in the model of read once branching programs, assuming the bias remains small. That is, the exact same result holds for a much more general class of sources.

- We tighten Aaronson's result and show that a depth d, size s Boolean circuit cannot solve the coin problem for $\beta < O(1/(\log s)^{d-1})$. Moreover, our proof technique is different and we believe that it is simpler and more natural.

1.2.3 Two structural results for low degree polynomials and applications

In a joint work with Tal [CT14] we obtained two structural results concerning low degree polynomials over finite fields. The first states that over any finite field $\mathbb{F}$, for any polynomial f on n variables with degree $d \leq \log(n)/10$, there exists a subspace of $\mathbb{F}^n$ with dimension $\Omega(d \cdot n^{1/(d-1)})$ on which f is constant. This result is shown to be tight. Stated differently, a degree d polynomial cannot compute an affine disperser for dimension smaller than $\Omega(d \cdot n^{1/(d-1)})$. Using a recursive argument, we obtain our second structural result, showing that any degree d polynomial f induces a partition of $\mathbb{F}^n$ to affine subspaces of dimension $\Omega(n^{1/(d-1)!})$, such that f is constant on each part.

We extend both structural results to more than one polynomial. We further prove an analog of the first structural result to sparse polynomials (with no restriction on the degree) and to functions that are close to low degree polynomials. We also consider the algorithmic aspect of the two structural results.

Our structural results have various applications, two of which are:

- Dvir [Dvi12] introduced the notion of extractors for varieties, and gave explicit constructions of such extractors over large fields. We show that over any finite field any affine extractor is also an extractor for varieties with related parameters. Our reduction also holds for dispersers, and we conclude that Shaltiel's affine disperser [Sha11] is a disperser for varieties over $\mathbb{F}_2$.

- Ben-Sasson and Kopparty [BSK12] proved that any degree 3 affine disperser over a prime field is also an affine extractor with related parameters. Using our structural results, and based on the work of Kaufman and Lovett [KL08] and Haramaty and Shpilka [HS10], we generalize this result to any constant degree.

1.2.4 The complexity of DNF of parities

Joint with Shinkar [CS14] we study depth 3 circuits of the form $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{XOR}$, or equivalently – DNF of parities. This model was first explicitly studied by Jukna [Juk06] who obtained a $2^{\Omega(n)}$ lower bound for explicit functions in this model. Several related

models have gained attention in the last few years, such as parity decision trees, the parity kill number and $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuits.

For a function $f\colon \{0,1\}^n \to \{0,1\}$, we denote by $\mathsf{DNF}_\oplus(f)$ the least integer s for which there exists an $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{XOR}$ circuit, with OR gate of fan-in s, that computes f. We summarize some of our results:

- For any affine disperser $f\colon \{0,1\}^n \to \{0,1\}$ for dimension k, it holds that $\mathsf{DNF}_\oplus(f) \geq 2^{n-2k}$. By plugging Shaltiel's affine disperser (FOCS'11) we obtain an explicit $2^{n-n^{o(1)}}$ lower bound.

- We give a non-trivial general upper bound by showing that $\mathsf{DNF}_\oplus(f) \leq O(2^n/n)$ for any function f on n bits. This bound is shown to be tight up to an $O(\log n)$ factor.

- We show that for any symmetric function f it holds that $\mathsf{DNF}_\oplus(f) \leq 1.5^n \cdot \mathrm{poly}(n)$. Furthermore, there exists a symmetric function f for which this bound is tight up to a polynomial factor.

- For symmetric threshold functions we show tighter bounds. For example, we show that the majority function has $\mathsf{DNF}_\oplus$ complexity of $2^{n/2} \cdot \mathrm{poly}(n)$. This is also tight up to a polynomial factor.

- For the inner product function IP on n inputs we show that $\mathsf{DNF}_\oplus(\mathsf{IP}) = 2^{n/2} - 1$. Previously, Jukna gave a lower bound of $\Omega(2^{n/4})$ for the $\mathsf{DNF}_\oplus$ complexity of this function. We further give bounds for any low degree polynomial.

- Finally, we obtain a $2^{n-o(n)}$ average case lower bound for the parity decision tree model using affine extractors.

1.3 Organization

The thesis is organized in chapters, where each chapter (with the exception of Chapter 2) covers a published paper or a paper that is currently in submission. Hence, each chapter may be read independently of all other chapters. In Chapter 2 we give basic definitions and results from the field of pseudorandomness that are used throughout the thesis.

1.4 Papers On Which This Thesis is Based On

In this section we give an account for the papers on which each chapter in this thesis is based on.

- Chapter 3 is based on a joint work with Raz and Segev, published at the SIAM Journal on Computing [CRS14]. A preliminary version of this paper appeared in the IEEE 27th annual conference on computational complexity (CCC 2012) [CRS12].

- Chapter 4 is based on a recent work [Coh15], that is to appear in the IEEE 56th annual symposium on foundations on computer science (FOCS 2015).
- Chapter 5 is based on a joint work with Shinkar [CS15], published at the 42nd international colloquium on automata, languages, and programming (ICALP 2015).
- Chapter 6 is based on a joint work with Damgard *et al.* [CDI+13], published at advances in cryptology (CRYPTO 2013).
- Chapter 7 is based on a joint work with Alon that appeared at the IEEE 28th annual conference on computational complexity (CCC 2013) [AC13]. A final version will appear at the journal of computational complexity 2015.
- Chapter 8 is based on a joint work with Ben-Aroya [BAC12] that is currently under submission.

Chapter 2

Basic Definitions and Results from Pseudorandomness

This chapter lists formal definition and results from the literature used repeatedly throughout the thesis. We recommend the reader to skim this chapter and return to it when a definition in later chapters needs clarification.

We start by setting some notation that will be used throughout the thesis. The logarithm in this thesis is always taken base 2. For every natural number $n \geq 1$, define $[n] = \{1, 2, \ldots, n\}$. Throughout the thesis we almost always avoid the use of floor and ceiling in order not to make the equations cumbersome.

2.1 Basic Notions in Probability

2.1.1 Random variables and distributions

We sometimes abuse notation and syntactically treat random variables and their distribution as equal, specifically, we denote by U_m a random variable that is uniformly distributed over $\{0,1\}^m$. Furthermore, if U_m appears in a joint distribution (U_m, X) then U_m is independent of X. When m is clear from context, we omit it from the subscript and write U.

Let X, Y be two random variables. We say that Y is a *deterministic function of* X if the value of X determines the value of Y. Namely, there exists a function f such that $Y = f(X)$. Let $X, Y, Z_1, \ldots, Z_r$ be random variables. We introduce the following shorthand notation and write $(X, Z_1, \ldots, Z_r) \approx_\varepsilon (Y, \cdot)$ for $(X, Z_1, \ldots, Z_r) \approx_\varepsilon (Y, Z_1, \ldots, Z_r)$.

2.1.2 Statistical distance

The *statistical distance* between two distributions X, Y on the same domain D is defined by

$$\mathsf{SD}(X, Y) = \max_{A \subseteq D} \{ |\Pr[X \in A] - \Pr[Y \in A]| \}.$$

If $\mathsf{SD}(X,Y) \leq \varepsilon$ we write $X \approx_\varepsilon Y$ and say that X is ε-close to Y.

2.1.3 Min-entropy

The *min-entropy* of a random variable X is defined by

$$H_\infty(X) = \min_{x \in \sup(X)} \log_2 \left(\frac{1}{\Pr[X = x]} \right).$$

If X is supported on $\{0,1\}^n$, we define the *min-entropy rate* of X by $H_\infty(X)/n$. In such case, if X has min-entropy k or more, we say that X is an (n,k)-weak-source. We sometimes abbreviate and simply say X is an (n,k)-source.

2.1.4 Flat sources

Let X be an (n,k)-source. We say that the source X is flat if it is uniformly distributed over a set $S_X \subseteq \{0,1\}^n$ of size 2^k. The following lemma, proved by Chor and Goldreich [CG88], shows that the distribution of any (n,k)-source is a convex combination of distributions of flat (n,k)-sources. Hence, in most cases, it will be enough to consider flat sources rather than general weak sources.

Lemma 2.1. *The distribution of any (n,k)-source is a convex combination of distributions of flat (n,k)-sources.*

2.1.5 Average conditional min-entropy

Definition 2.2. *Let X, W be two random variables. The* average conditional min-entropy *of X given W is defined as*

$$\begin{aligned} \widetilde{H}_\infty(X \mid W) &= -\log_2 \left(\mathop{\mathbf{E}}_{w \sim W} \left[\max_x \Pr\left[X = x \mid W = w\right] \right] \right) \\ &= -\log_2 \left(\mathop{\mathbf{E}}_{w \sim W} \left[2^{-H_\infty(X \mid W = w)} \right] \right). \end{aligned}$$

Lemma 2.3 ([DORS08])**.** *Let X, Y, Z be random variables such that Y has support size at most 2^ℓ. Then,*

$$\widetilde{H}_\infty(X \mid (Y,Z)) \geq \widetilde{H}_\infty((X,Y) \mid Z) - \ell \geq \widetilde{H}_\infty(X \mid Z) - \ell.$$

In particular, $\widetilde{H}_\infty(X \mid Y) \geq H_\infty(X) - \ell$.

Lemma 2.4 ([DORS08])**.** *For any two random variables X, Y and any $\varepsilon > 0$, it holds that*

$$\Pr_{y \sim Y} \left[H_\infty(X \mid Y = y) < \widetilde{H}_\infty(X \mid Y) - \log(1/\varepsilon) \right] \leq \varepsilon.$$

Lemma 2.5 ([DORS08])**.** *Let $\delta > 0$ and let $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a strong seeded extractor for entropy k, with error ε. Let X be a random variable on n bit strings, such that $\widetilde{H}_\infty(X \mid Z) \geq k + \log(1/\delta)$, where Z is some random variable. Then,*

$$(\mathsf{Ext}(X,S), S, Z) \approx_{\varepsilon+\delta} (U_m, S, Z),$$

where S is uniformly distributed over $\{0,1\}^d$ and is independent of the joint distribution (X,Z).

We also need the following simple lemma.

Lemma 2.6. *Let X, Y, Z be random variables such that for any $y \in \sup(Y)$ it holds that $(X \mid Y = y)$ and $(Z \mid Y = y)$ are independent. Then, $\widetilde{H}_\infty(X \mid (Y,Z)) = \widetilde{H}_\infty(X \mid Y)$. In particular, if X and Z are independent then $\widetilde{H}_\infty(X \mid Z) = H_\infty(X)$.*

2.2 Basic Lemmata in Probability

Throughout the thesis we make a frequent use of the following simple and well-known lemmata.

Lemma 2.7. *Let X, Y be two independent random variables on a common domain D. Then,*

$$\mathsf{SD}(X,Y) = \frac{1}{2} \cdot \sum_{z \in D} \left| \Pr[X = z] - \Pr[Y = z] \right|.$$

Lemma 2.8. *Let X, Y be two independent random variables on a common domain D. Let f be a function with domain D. Then, $\mathsf{SD}(f(X), f(Y)) \leq \mathsf{SD}(X,Y)$. Moreover, the inequality above holds also for f which is a random function, where the internal randomness of f is independent of (X,Y).*

Lemma 2.9. *For all random variables X, Y, Z, it holds that*

$$\mathsf{SD}((X,Y),(Z,Y)) = \mathop{\mathbf{E}}_{y \sim Y} [\mathsf{SD}((X \mid Y = y), (Z \mid Y = y))].$$

Lemma 2.10. *Let X, Y, Z be random variables such that X is independent of Y and Z is independent of Y. Then, $\mathsf{SD}((X,Y),(Z,Y)) = \mathsf{SD}(X,Z)$. In particular, if X is supported on $\{0,1\}^a$ then $\mathsf{SD}((X,Y),(U_a,Y)) = \mathsf{SD}(X,U_a)$.*

Lemma 2.11. *Let X, Y, Z be random variables such that for any $y \in \sup(Y)$, the random variables $(X \mid Y = y)$ and $(Z \mid Y = y)$ are independent. Assume that X is supported on $\{0,1\}^a$. Then,*

$$\mathsf{SD}((X,Y,Z),(U_a,Y,Z)) = \mathsf{SD}((X,Y),(U_a,Y)).$$

Lemma 2.12. *Let X, Z be random variables on a common domain. Let Y be some random variable. Then, $\mathsf{SD}(X,Z) \leq \mathsf{SD}((X,Y),(Z,Y))$.*

Lemma 2.13. *Let X, Y be two random variables on a common domain D. Let $f\colon D \to \mathbb{R}$ be a function with non-negative range, that is, $f(z) \geq 0$ for all $z \in D$. Then,*

$$\left| \mathop{\mathbf{E}}_{x\sim X}[f(x)] - \mathop{\mathbf{E}}_{y\sim Y}[f(y)] \right| \leq \max_{z\in D} |f(z)| \cdot \mathsf{SD}(X,Y).$$

Lemma 2.14. *Let X, Y be two random variable over a common domain D. Let $E \subseteq D$ be an event. Then,*

$$\mathsf{SD}\left(X \mid E, Y \mid E\right) \leq \frac{1}{\Pr[E]} \cdot \mathsf{SD}\left(X, Y\right).$$

Lemma 2.15 ([Li12b])**.** *Let (X, Y) be a joint distribution. Let Z be a random variable with the same range as X. Then, there exists a joint distribution (Z, Y) such that $\mathsf{SD}\left((X,Y),(Z,Y)\right) = \mathsf{SD}(X,Z)$.*

2.3 Extractors

Definition 2.16 (Seeded-Extractor)**.** *A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ε)-*seeded-extractor *if for every (n,k)-source W and an independent random variable S uniformly distributed over $\{0,1\}^d$, the distribution of $\mathsf{Ext}(W,S)$ is ε-close to U_m. A (k,ε)-seeded-extractor is* strong *if for X and S as above, the distribution of $(\mathsf{Ext}(X,S), S)$ is ε-close to (U_m, U_d).*

Throughout this thesis we make use of the following explicit constructions of strong seeded-extractors.

Theorem 2.1 ([ILL89])**.** *For all integers $n \geq k > m > 0$ and for any $\varepsilon > 0$ such that $m \leq k - 2\log(1/\varepsilon)$, there is an explicit construction of a strong (k,ε)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, where $d = O(n)$.*

Theorem 2.2 ([GUV09])**.** *For every constant $\beta > 0$, for all integers $n \geq k > m > 0$ such that $m \leq (1-\beta)k$, and for any $\varepsilon > 0$, there is an explicit construction of a strong (k,ε)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, where $d = O(\log n + \log(1/\varepsilon))$.*

Theorem 2.3 ([GUV09])**.** *For all integers $n \geq k > m > 0$, and for any $\varepsilon > 0$ such that $m \leq k - 2\log(1/\varepsilon) - O(1)$, there is an explicit construction of a strong (k,ε)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, where $d = \log n + O(\log k \cdot \log(k/\varepsilon))$.*

Definition 2.17 (Multi-source extractors)**.** *A function $\mathsf{Ext}\colon (\{0,1\}^n)^t \to \{0,1\}^m$ is called a t-*source extractor *for entropies $k_1, \ldots, k_t$, with error ε, if for any t independent n-bit weak-sources $X_1, \ldots, X_t$, where $H_\infty(X_i) \geq k_i$, it holds that $\mathsf{Ext}(X_1, \ldots, X_t) \approx_\varepsilon U_m$. For a subset $I \subseteq [t]$, we say that Ext is strong in I if $(\mathsf{Ext}(X_1, \ldots, X_t), \{X_i\}_{i\in I}) \approx_\varepsilon (U_m, \cdot)$.*

Theorem 2.4 ([Li13])**.** *For every constant $\mu > 0$ and all integers n, k with $k \geq \log^{2+\mu} n$, there exists an explicit function $\mathsf{Li}\colon (\{0,1\}^n)^c \to \{0,1\}^m$, with $m = \Omega(k)$ and $c = O(1/\mu)$, such that the following holds. If $X_1, \ldots, X_c$ are independent (n,k)-weak sources, then*

$$\mathsf{Li}(X_1, \ldots, X_c) \approx_\varepsilon U_m,$$

where $\varepsilon = n^{-\Omega(1)} + 2^{-k^{\Omega(1)}}$.

2.4 Small-Bias Sets

A random variable Z over $\{0,1\}$ is *ε-biased* if $\text{bias}(Z) = \left|\Pr[Z=0] - \Pr[Z=1]\right| \leq \varepsilon$, that is, if its distribution is ε-close to uniform. A sequence of 0-1 random variables $Z_1, \ldots, Z_N$ is *ε-biased for linear tests of size k* if the exclusive-or of any nonempty set of cardinality at most k of these variables is ε-biased, that is, for any nonempty $\tau \subseteq [N]$, such that $|\tau| \leq k$, the random variable $Z_\tau = \bigoplus_{i \in \tau} Z_i$ is ε-biased. We say in this case, that the sequence $Z_1, \ldots, Z_N$ *ε-fools linear tests of size k*.

Explicit constructions of small probability spaces on N random variables that are ε-biased for linear tests of size k were given in [NN93, AGHP92, ABN+92, BT09]. In particular, [AGHP92] showed that for every k, $N \geq 2$, variables $Z_1, \ldots, Z_N$ as above can be explicitly constructed using $2 \cdot \lceil \log(1/\varepsilon) + \log k + \log\log N \rceil$ random bits.

2.5 Unbalanced Expander Graphs

In this section we give the definition and basic facts about unbalanced expander graphs. For more information we refer the reader to [HLW06]. Let $G = (L, R, E)$ be a bipartite graph with $|L| = m$, $|R| = n$, and left-degree d. For a set $S \subseteq L$ define

$$\Gamma(S) = \{r \in R : \exists s \in S \text{ such that } sr \in E\},$$

and

$$\Gamma_1(S) = \{r \in R : \exists! s \in S \text{ such that } sr \in E\}.$$

G is called *$(k_{max}, 1-\varepsilon)$-bipartite-expander* if for every $S \subseteq L$ with size at most $k_{\max}$, it holds that $|\Gamma(S)| \geq (1-\varepsilon)d|S|$. G is called *$(k_{max}, 1-\varepsilon)$-unique neighbor expander* if for every $S \subseteq L$ with size at most $k_{\max}$ it holds that $|\Gamma_1(S)| \geq (1-\varepsilon)d|S|$. The following simple well-known fact relates the two definitions.

Fact 2.18. *Every $(k_{max}, 1-\varepsilon)$-bipartite expander is a $(k_{max}, 1-2\varepsilon)$-unique neighbor expander.*

We will be interested in the case where $m = \omega(n)$. Such bipartite expanders are called *unbalanced expanders*. The following fact shows that given any plausible $n, d, k_{\max} \in \mathbb{N}$ and $\varepsilon \in (0,1)$, there exist highly unbalanced expanders, that is, $(k_{\max}, 1-\varepsilon)$-bipartite expanders with large m.

Fact 2.19. *Let $n, d \in \mathbb{N}$, and let $\frac{1}{d} < \varepsilon < 1$. For any $k_{max} \leq e^{-2/\varepsilon} \cdot \frac{n}{d}$ there exists a $(k_{max}, 1-\varepsilon)$-bipartite expander with $m = \Omega(k_{max} \cdot e^d)$.*

In particular, for constant ε, 7.12 implies that for large enough n, d, there exist $(k_{\max}, 1-\varepsilon)$-bipartite expanders with $k_{\max} = \Omega(\frac{n}{d})$, and $m = \Omega(\frac{n}{d} \cdot e^d)$.

Chapter 3

Non-Malleable Extractors with Short Seeds and Applications to Privacy Amplification

3.1 Non-Malleable Extractors and Privacy Amplification

Among the wide variety of settings in which randomness extractors play an instrumental role is the classical problem of *privacy amplification* [BBR88, Mau92, BBCM95]. This problem considers a setting in which two parties, Alice and Bob, begin by sharing a secret $W \in \{0,1\}^n$ whose distribution may be far from uniform. The parties interact over a public communication channel in the presence of an adversary, Eve, and would like to securely agree on a nearly uniform secret $R \in \{0,1\}^m$.

In various applications, the secret W is often chosen, for example, as a human-memorizable password or some biometric data, both of which are typically of rather low min-entropy, or even as a truly uniform secret which may have been partially leaked to Eve. The formal way of modeling such sources of randomness is to consider weak-sources (for the formal definition, see Chapter 2). In this chapter, which is based on a joint work with Raz and Segev [CRS14], we consider the information-theoretic setting of the problem where no computational assumptions are made. In particular, Eve is assumed to be computationally unbounded.

In the presence of a *passive* adversary that is assumed to only observe the communication channel between the parties, any strong extractor provides an elegant solution to the privacy amplification problem. We recall here an informal description of strong seeded extractors (see Definition 2.16 for the formal definition). A strong seeded-extractor is a function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that takes two inputs, a weak source W and an independent uniform seed S, and outputs a string $\mathsf{Ext}(W,S)$ that is nearly uniform given the seed S. Using a strong extractor, Alice simply sends Bob a seed S that is chosen uniformly at random, and they both compute $R = \mathsf{Ext}(W,S)$ which is guaranteed to be

nearly uniform from Eve's point of view (who sees only the seed S).

In the presence of an *active* adversary that fully controls the communication channel between the parties, however, privacy amplification is significantly more challenging. One reason is that in addition to preventing Eve from learning essentially any information on the resulting secret R, the protocol should also prevent Eve from causing the parties to output different secrets R and R'.

In this light, extensive research has been devoted for designing privacy amplification protocols that are secure under active attacks (see [Mau97, MW97, Wol98, MW03, RW03, DKRS06, DW09, KR09, CKOR10] and the references therein), with the natural goal of optimizing the efficiency of such protocols. The main measures of efficiency that have been studied in this line of research are the following:

1. **Required entropy rate:** The ratio between the required min-entropy of the weak secret W and its length.

2. **Entropy loss:** The difference between the entropy of the weak secret W and the length of the resulting secret R.

3. **Communication complexity:** The number of bits exchanged between the two parties in the protocol.

4. **Round complexity:** The number of rounds in the protocol.

A major progress in the design of privacy amplification protocols was made by Dodis and Wichs [DW09]. Their approach relies on introducing the new and elegant notion of a *non-malleable extractor*, significantly strengthening the notion of a strong extractor. Informally, a non-malleable extractor is a function $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that takes two inputs: a weak source W and an independent uniform seed S, and outputs a string $\mathsf{nmExt}(W,S)$ that is nearly uniform given the seed S as well as the value $\mathsf{nmExt}(W,S')$ for any seed $S' \neq S$ that may be determined as an arbitrary function of S.

Definition 3.1 (Adversarial Function)**.** *Let $\mathcal{A}\colon \{0,1\}^d \to \{0,1\}^d$. We say that $\mathcal{A}$ is an* adversarial function *if it has no fixed points. That is, for every $s \in \{0,1\}^d$ it holds that $\mathcal{A}(s) \neq s$.*

Definition 3.2 (Non-Malleable Extractor)**.** *A function $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ε)-*non-malleable extractor *if for every (n,k)-source W and independent random variable S uniformly distributed over $\{0,1\}^d$, and for every adversarial function $\mathcal{A}\colon \{0,1\}^d \to \{0,1\}^d$,*

$$\mathsf{SD}\left((\mathsf{Ext}(W,S), \mathsf{Ext}(W,\mathcal{A}(S)), S), (U_m, \mathsf{Ext}(W,\mathcal{A}(S)), S)\right) \leq \varepsilon.$$

We also consider a natural generalization of non-malleable extractors, in which the adversary has the value of the extractor not only on *one* correlated seed $\mathcal{A}(S)$ of her choice, but rather on *many* correlated seeds $\mathcal{A}_1(S), \ldots, \mathcal{A}_t(S)$ of her choice.

Definition 3.3 (t-Adversarial Function)**.** *Let $t \in \mathbb{N}$. Let $\mathcal{A}\colon \{0,1\}^d \to \{0,1\}^{td}$. We think of the output of $\mathcal{A}$ as t concatenated binary strings, each of length d. That is, we think of $\mathcal{A}(s)$ as $\mathcal{A}(s) = (\mathcal{A}_1(s), \ldots, \mathcal{A}_t(s))$, where for all $i \in [t]$, $\mathcal{A}_i$ is a function of the form $\mathcal{A}_i\colon \{0,1\}^d \to \{0,1\}^d$. We say that $\mathcal{A}$ is a* t-adversarial function *if for every $i \in [t]$ the function $\mathcal{A}_i$ is an adversarial function.*

Definition 3.4 (t-Non-Malleable Extractor)**.** *A function $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ε)-t-*non-malleable extractor *if for every (n,k)-source W and independent random variable S uniformly distributed over $\{0,1\}^d$, and for every t-adversarial function $\mathcal{A}\colon \{0,1\}^d \to \{0,1\}^{td}$,*

$$\mathsf{SD}\left(\left(\mathsf{Ext}(W,S), \{\mathsf{Ext}(W,\mathcal{A}_i(S))\}_{i=1}^t, S\right), \left(U_m, \{\mathsf{Ext}(W,\mathcal{A}_i(S))\}_{i=1}^t, S\right)\right) \leq \varepsilon.$$

3.1.1 Constructions of non-malleable extractors prior to our work

Although the approach of Dodis and Wichs [DW09] seems very promising, they were in fact unable to present an explicit construction of a non-malleable extractor (not even one with poor parameters). They showed, however, using the probabilistic method, that such extractors, with excellent parameters, exist. More specifically, Dodis and Wichs proved the existence of a (k,ε)-non-malleable extractor $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, as long as

$$\begin{aligned} d &> \log(n-k-1) + 2\log(1/\varepsilon) + 5, \\ k &> 2m + 2\log(1/\varepsilon) + \log d + 6. \end{aligned}$$

Recently, Dodis *et al.* [DLWZ11b] presented the first explicit construction of a non-malleable extractor. They showed that an extractor introduced by Chor and Goldriech in the context of two-source extractors [CG88] is non-malleable as long as the weak source has min-entropy rate $1/2 + \delta$ for an arbitrarily small constant $\delta > 0$. Their extractor outputs up to a linear number of bits, but suffers from two drawbacks. First, the seed used by their extractor is of length $d = \Omega(n)$ bits, even for the purpose of extracting a single bit. Second, the construction is conditional: when outputting more than a logarithmic number of bits (as required for privacy amplification protocols) its efficiency relies on a longstanding conjecture on the distribution of prime numbers.

3.1.2 Privacy amplification via non-malleable extractors

Using a non-malleable extractor, Dodis and Wichs constructed the first 2-round privacy amplification protocol for any min-entropy rate that is secure against active attacks[1]. Specifically, Dodis and Wichs demonstrated that the idea underlying the simple privacy amplification protocol discussed above for passive attacks can be implemented also

[1] They also showed that 1-round protocols do not exist when the weak secret W has min-entropy $k \leq n/2$, and are inherently inefficient in terms of communication when $n/2 < k \ll n$.

in the setting of active attacks. Moreover, when instantiating their approach with a non-malleable extractor that enjoys sufficiently good parameters (as well as with an essentially optimal strong extractor), the resulting privacy amplification protocol in turn enjoys asymptotically optimal entropy loss $O(\log(n/\varepsilon))$ and communication complexity $O(\log(n/\varepsilon))$, where ε is the security parameter of the protocol (i.e., the protocol error).

As discussed above, Dodis and Wichs were in fact unable to present an explicit construction of a non-malleable extractor. Nevertheless, they were still able to construct an explicit privacy amplification protocol by introducing the weaker notion of a *look-ahead extractor*[2], for which they were able to provide an explicit construction. It is worth noting that in [Coh15], which is covered in Chapter 4, we make an extensive use of look-ahead extractor for a completely different purpose. Using their look-ahead extractor, Dodis and Wichs constructed an explicit 2-round privacy amplification protocol with entropy loss $\beta k + O(\log^2 n + \log^2(1/\varepsilon))$, for an arbitrarily small constant $\beta > 0$, and communication complexity $O(\log^2 n + \log^2(1/\varepsilon))$, both of which are somewhat far from optimal[3].

Dodis *et al.* [DLWZ11b] showed that when instantiating the protocol of Dodis and Wichs with their explicit non-malleable extractor one obtains an explicit 2-round privacy amplification protocol for weak sources of min-entropy rate $1/2 + \delta$, for an arbitrarily small constant $\delta > 0$, with entropy loss $O(\log(n/\varepsilon))$. However, since the seed of the extractor is of length $\Omega(n)$ bits, the resulting privacy amplification protocol suffers from communication complexity of $\Omega(n)$ bits.

Thus, although the approach of Dodis and Wichs [DW09] for privacy amplification indeed seems very promising, due to the difficulties in constructing explicit non-malleable extractors, protocols prior to the work covered in the chapter are rather far from optimal either in their entropy loss or communication complexity.

Finally, we note that privacy amplification protocols can also be constructed using various other techniques and tools (and not only using non-malleable extractors). For example, the privacy amplification protocol of Chandran et al. [CKOR10] uses a somewhat different approach that crucially utilizes repeated interaction between the parties, and is essentially optimal in all parameters except for its rather high round complexity of $O(\log(1/\varepsilon))$ rounds.

3.2 Our Contribution

In a joint work with Raz and Segev [CRS14], we give an *unconditional* construction of a non-malleable extractor with *short seeds*. More precisely, we prove the following.

[2]Informally, a look-ahead extractor is a function $\mathsf{laExt} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that takes two inputs: a weak source W and a uniform seed S, and outputs a string $\mathsf{laExt}(W,S)$ whose any suffix is nearly uniform given the seed S and the complementing prefix of $\mathsf{laExt}(W,S')$ for some seed $S' \neq S$ that may be determined as an arbitrary function of S. Note that any non-malleable extractor is in particular also a look-ahead extractor.

[3]In fact, the dependence on the min-entropy k of the weak source can be eliminated from the entropy loss in their protocol. This can be done, for example, simply by instantiating the strong extractor in their protocol with a different and more suitable extractor.

Theorem 3.1. *For any integers n and d such that $2.01 \log n \leq d \leq n$, and for any constant $\delta > 0$, there exists an explicit $((1/2 + \delta) \cdot n, 2^{-m})$-non-malleable extractor $\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, with $m = \Omega(d)$.*

In particular, setting $d = 2.01 \log n$ yields the first explicit construction of a non-malleable extractor that uses a seed of length $O(\log n)$ bits [4]. This should be compared with the original extractor of [DLWZ11b] that uses a seed of length $\Omega(n)$ bits. This improvement in the seed length is crucial for the communication complexity of the resulting privacy amplification protocols.

In addition, setting $d = n$ yields the first *unconditional* explicit construction of a non-malleable extractor that outputs $\Omega(n)$ bits. This should be compared with the extractor of [DLWZ11b] whose efficiency relies on an unproven conjecture (when outputting $\omega(\log n)$ bits). In fact, our extractor is the first non-malleable extractor that outputs $\omega(\log n)$ bits unconditionally.

The result is in fact more general, and in fact we show an explicit construction of a t-non-malleable-extractor with essentially optimal parameters[5], as long as the min-entropy rate of the weak-source is any constant larger than $1/2$ and the output length is shorter than the seed length.

Theorem 3.2. *For any integers n, d, m and t, and for any $0 < \delta < 1/2$ such that*

$$\begin{aligned} d \geq & \frac{23}{\delta} \cdot tm + 2 \log n, \\ n \geq & \frac{160}{\delta} \cdot tm, \\ \delta \geq & 10 \cdot \frac{\log(nd)}{n}, \end{aligned}$$

there exists an explicit $((1/2 + \delta) \cdot n, 2^{-m})$-$t$-non-malleable extractor $\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$.

3.2.1 Applications to privacy amplification

By instantiating the framework of Dodis and Wichs [DW09] with the non-malleable extractor from Theorem 3.1 we obtain an explicit 2-round privacy amplification protocol for weak sources of min-entropy rate $1/2 + \delta$ for an arbitrarily small constant $\delta > 0$. The protocol offers a trade-off between its entropy loss and communication complexity, resulting from instantiating it with different explicit constructions of strong extractors. Specifically, it offers asymptotically optimal entropy loss $O(\log(n/\varepsilon))$ with communication complexity $O(\min\left\{\log^2 n + \log n \cdot \log(1/\varepsilon), n\right\})$, or entropy loss $\beta n + O(\log(n/\varepsilon))$ for an arbitrarily small constant $\beta > 0$ with communication complexity $O(\log(n/\varepsilon))$, where the hidden constant in the big-O notation depends on β. In particular, we prove the following theorem:

[4] The constant 2.01 can, in fact, be replaced by any constant strictly greater than 2.

[5] We made no attempt to optimize the constants in the theorem as they depend on each other.

Theorem 3.3. *For any integer n, constant $\delta > 0$, and security parameter $\varepsilon = 2^{-O(n)}$, there exists an explicit and efficient 2-round privacy amplification protocol for $(n, (1/2 + \delta)n)$-sources with entropy loss $O(\log n + \log(1/\varepsilon))$, and communication complexity $O(\min\left\{\log^2 n + \log n \cdot \log(1/\varepsilon), n\right\})$.*

This is the first explicit 2-round privacy amplification protocol for min-entropy rate $1/2 + \delta$ with asymptotically optimal entropy loss and poly-logarithmic communication complexity. This should be compared to the previously known 2-round protocols: the protocol of [DW09] whose entropy loss is not asymptotically optimal, and the protocol of [DLWZ11b] whose communication complexity is linear in the length of the weak secret.

3.2.2 Subsequent work

Subsequent to the paper [CRS14], on which this chapter builds on, there has been an extensive research in the construction of non-malleable extractors, and their applications. Based on ideas from [CRS14], Dodis *et al.* [DLWZ11c] showed how to reduce the seed length of their original extractor from [DLWZ11b] so to match the parameters of Theorem 3.1. [6] However, their analysis remains conditional when outputting $\omega(\log n)$ bits.

Li [Li12a] uses the extractor presented in this chapter, combined with a combinatorial object called *design extractor*, and obtains a non-malleable extractor with a logarithmic length seed that can output $\Omega(n)$ bits when fed with an n-bit source with min-entropy larger than $n/2$. In [Li12c], Li proves that Bourgain's two-source extractor [Bou05, Rao07] is non-malleable, thus achieving a non-malleable extractor that works for sources with min-entropy $1/2 - \varepsilon_0$ for some small universal constant $\varepsilon_0 > 0$. In the same paper Li proves that a substantial improvement in the construction of non-malleable extractors will yield (a modest) improvement in the construction of two-source extractors. A simplified and uniform approach for constructing, among other objects, non-malleable extractors, was suggested by Dodis and Yu [DY13].

The problem of constructing privacy amplification protocols against an active adversary has been further studied as well. In [Li12b] Li gives new and improved constructions of 2-round privacy amplification protocols based on the notion of non-malleable condensers, introduced in [Li12a]. In particular, Li presents a protocol for min-entropy $k = \Omega(\log^2(n/\varepsilon))$ with an optimal entropy loss $O(\log(n/\varepsilon))$, and with communication complexity $O(\log^2(n/\varepsilon))$. An interesting open problem in this area is to construct a 2-round privacy amplification protocol for the minimal min-entropy $k = O(\log(n/\varepsilon))$ with an optimal entropy loss $O(\log(n/\varepsilon))$.

3.3 Overview of the Construction

In this section we overview the main ideas underlying our constructions. We begin with the construction of the non-malleable extractor in Section 3.3.1, and then proceed with

[6] In fact, as pointed out in [DLWZ11c], the construction of Dodis *et al.* appears to be a special case of our construction, at least for the one-bit case.

the resulting privacy amplification protocol in Section 3.3.2. The full proofs can be found in the paper [CRS14], on which this chapter is based.

3.3.1 The non-malleable extractor

Raz [Raz05] gave an explicit construction of seeded-extractors, based on small-bias sets (see Section 2), or more precisely, on small probability spaces of 0-1 random variables that have small bias for linear tests of bounded size. We begin by describing the construction of these extractors, starting with extractors that output one bit, and then turn to describe our approach.

The Extractor of Raz [Raz05]. Let $D = 2^d$, and let $Z_1, \ldots, Z_D$ be 0-1 random variables that are ε-biased for linear tests of size k, and assume that the random variables can be constructed using n random bits. We define $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ by $\mathsf{Ext}(w,s) = Z_s(w)$. That is, $\mathsf{Ext}(w,s)$ is the value of the random variable Z_s when using w as the value of the n bits needed to produce $Z_1, \ldots, Z_D$. In other words, w is used to choose the point in the probability space, and s is used to choose the variable from $Z_1, \ldots, Z_D$ that we evaluate.

Extracting many bits is done similarly: Let $D = m \cdot 2^d$, and let $Z_1, \ldots, Z_D$ be 0-1 random variables, constructed using n random bits, that are ε-biased for linear tests of size k. We interpret the set of indices $\{1, \ldots, D\}$ as the set $\{(i,s) : i \in [m], s \in \{0,1\}^d\}$. We define $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ by $\mathsf{Ext}_i(w,s) = Z_{(i,s)}(w)$, where $\mathsf{Ext}_i(w,s)$ denotes the i^{th} bit of $\mathsf{Ext}(w,s)$. In other words, w is used to choose the point in the probability space, and the pair (i,s) is used to choose the variable from $Z_1, \ldots, Z_D$ that we evaluate.

Raz showed that the above extractor, based on *any* small probability space of 0-1 random variables that have small bias for linear tests of bounded size, is an excellent extractor. Specifically, using any of the probability spaces from [AGHP92], one gets a $((1/2 + \delta) \cdot n, 2^{-m})$-seeded extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $m = \min(\Omega(\delta n), \Omega(d))$ as long as $d = \Omega(\log n)$. In the same paper, among other results, Raz showed that this extractor is a strong seeded-extractor.

Our Approach. We show that the extractor of Raz is in fact non-malleable with essentially the same parameters. Moreover, we show that it is t-non-malleable with optimal dependency on t. We now present the proof strategy. For simplicity, we focus on the case $m = t = 1$. The proof strategy for the t-non-malleability of the extractor that extracts m bits follows by the same logic, but it is more technical.

Assume for a contradiction that Ext as defined above is not non-malleable. This implies the existence of a weak-source W and an adversarial function $\mathcal{A} : \{0,1\}^d \to \{0,1\}^d$ such that for a typical seed $s \in \{0,1\}^d$, the value $\mathsf{Ext}(W,s)$ is correlated to $\mathsf{Ext}(W, \mathcal{A}(s))$. We can then find a large set of seeds $S \subseteq \{0,1\}^d$ such that for every $s \in S$, the random variable $Y_s = \mathsf{Ext}(W,s) \oplus \mathsf{Ext}(W, \mathcal{A}(s))$ is biased.

At this point we consider the directed graph $G = (S \cup \mathcal{A}(S), E)$ where $E = \{(s, \mathcal{A}(s)) : s \in S\}$. We note that G has no self loops, but it might be the case that G contains cycles. We prove the existence of a large subset $S' \subseteq S$ such that the induced graph of G by $S' \cup \mathcal{A}(S')$ is acyclic. To this end we prove a simple lemma about graphs that we prove.

For every $s \in S'$, define $Y'_s = Z_s \oplus Z_{\mathcal{A}(s)}$. In the next step of the proof we prove that the set of random variables $(Y'_s)_{s \in S'}$ is ε-biased for linear tests of size at most $k/2$. This follows easily by the acyclicity of the above mentioned graph and by the fact that for every $s \in S'$, it holds that Y'_s is a parity of two random variables from a probability space that ε-fools linear tests of size k.

Now we consider the extractor that is built upon the random variables $(Y'_s)_{s \in S'}$ as described in the beginning of the section (where the $(Y'_s)_{s \in S'}$ play the role of $(Z_i)_{i=1}^d$). The result of Raz, which holds for *any* probability space that fools linear tests of bounded size, implies that this is a good seeded-extractor. This yields a contradiction (for an appropriate choice of parameters) when feeding the weak-source W to this extractor, because the random variables $(Y_s)_{s \in S'}$ are all biased.

3.3.2 The privacy amplification protocol

As discussed in Section 3.2.1, by instantiating the framework of Dodis and Wichs [DW09] with our non-malleable extractor we obtain the first explicit 2-round privacy amplification protocol for weak sources of min-entropy rate $1/2 + \delta$, for an arbitrarily small constant $\delta > 0$, with asymptotically optimal entropy loss and poly-logarithmic communication complexity. In what follows we first overview the main idea underlying the Dodis-Wichs protocol, and then discuss the parameters that we obtain by instantiating it with our non-malleable extractor.

The Dodis-Wichs Protocol. In the presence of a *passive* adversary that is assumed to only observe the communication channel between the parties, the privacy amplification problem is well-understood. Specifically, any strong exactor Ext yields the following elegant solution: Alice sends Bob a uniform seed S for Ext, and they both compute $R = \mathsf{Ext}(W, S)$, where W is their shared weak secret. The property of the strong extractor guarantees that the resulting value R is nearly uniform from the adversary's point of view.

The main idea underlying the approach of Dodis and Wichs is that a non-malleable extractor nmExt can be used for implementing the above elegant solution in the presence of an *active* adversary. Specifically, the non-malleable extractor is used for authenticating the seed S, and as long as the communication complexity involved in the authentication is rather small. Since only a small number of bits are revealed to the adversary, W still has sufficient min-entropy that can be extracted as $R = \mathsf{Ext}(W, S)$.

For authenticating the seed S, in the first round of the protocol Alice chooses a uniform seed Y for a non-malleable extractor nmExt, sends Y to Bob, and computes a key $\mathsf{key} = \mathsf{nmExt}(W, Y)$ for a one-time message-authentication code MAC. The adversary may modify Y to any value Y', and in this case Bob might compute a different authentication key $\mathsf{key}' = \mathsf{nmExt}(W, Y')$. Then, in the second round of the protocol, Bob samples

a uniform seed S' for a strong extractor Ext, and sends it to Alice together with the authentication tag $\sigma' = \mathsf{MAC}_{\mathsf{key}'}(S')$. At this point Bob concludes his part of the protocol by outputting the value $R' = \mathsf{Ext}(W, S')$. The adversary may modify the pair (S', σ') to any pair (S, σ), and Alice verifies that $\sigma = \mathsf{MAC}_{\mathsf{key}}(S)$. If the verification fails then Alice aborts, and otherwise Alice outputs $R = \mathsf{Ext}(W, S)$.

Note that if the adversary does not modify the seed Y that is chosen by Alice, then Alice and Bob share the same authentication $\mathsf{key} = \mathsf{key}'$, which is nearly uniform from the adversary's point of view. Thus, the adversary has only a negligible probability of computing a valid authentication tag σ for any seed $S \neq S'$. In addition, if the adversary does modify the seed Y to a different seed Y', then the property of the non-malleable extractor guarantees that the authentication key key computed by Alice is nearly uniform from the adversary's point of view, even if she receives key' (and, in particular, if she receives σ' which is a deterministic function of S' and key'). Thus, again, the adversary has only a negligible probability of computing a valid authentication tag σ for any seed S with respect to key (and this holds even if $S = S'$). These two observations then easily imply the security of the protocol.

Our Instantiation. In the Dodis-Wichs protocol the output $\mathsf{key} = \mathsf{nmExt}(W, Y)$ of the non-malleable extractor is used as a key for a one-time message authentication code. It is well known that explicit and efficient constructions of message-authentication codes exist with keys and authentication tags of length $O(\log(n/\varepsilon))$ bits, where n is the length of the authenticated message and ε is the security parameter.

Using our non-malleable extractor we can set its seed Y to be of length $O(\log(n/\varepsilon))$ bits. Then, one can instantiate the strong extractor Ext with any explicit construction, where we choose the one provided in [GUV09]. It extracts $(1/2 + \delta)n - O(\log(n/\varepsilon))$ bits from the weak source W using a seed S of length $O(\log^2 n + \log n \cdot \log(1/\varepsilon))$ bits. When dealing with a very small security parameter ε one can instead use the extractor provided by the leftover hash lemma for extracting the same number of bits using a seed of length n bits. Thus, our protocol has entropy loss $O(\log(n/\varepsilon))$, and communication complexity $O(\min\left\{\log^2 n + \log n \cdot \log(1/\varepsilon), n\right\})$.

3.4 Privacy Amplification Protocols Preliminaries

In this section and the next we give the formal definition of privacy amplification protocols and the formal definition of message authenticated codes, respectively.

3.4.1 Privacy amplification protocols

Our definition of a privacy amplification protocol (also known as an information-theoretic key-agreement protocol) follows that of Dodis and Wichs [DW09]. In a privacy amplification protocol, two parties, Alice and Bob, begin by sharing a weak secret $W \in \{0,1\}^n$, that is, a string sampled from a weak-source W. The parties interact over a public communication channel in the presence of an adversary, Eve, and would like to securely agree on

a nearly uniform secret $R \in \{0,1\}^m$. In this paper we consider the information-theoretic setting of the problem where no computational assumptions are made (in particular, Eve is assumed to be computationally unbounded), and the weak secret W may be sampled from any publicly known distribution subject to a pre-specified min-entropy rate. In addition, we assume that Eve is an *active* adversary that fully controls the communication channel between the parties.

At the beginning of the protocol Alice and Bob each have candidate keys R_A and R_B, respectively, which are initially set to the special value $\perp$. At some point during the execution of the protocol one party can reach a KeyDerived state and the other party can reach a KeyConfirmed state. Upon reaching either of these states, a party sets its candidate key to some m-bit value and does not modify it afterwards. Informally, the KeyDerived and KeyConfirmed states should be interpreted as follows:

1. If Alice reaches the KeyDerived state, then she possesses a uniformly random candidate key R_A, which remains private no matter how Eve acts during the remainder of the protocol execution. However, she is not sure if her key is shared with Bob, or if Bob is even involved in the protocol execution at all.

2. If Bob reaches the KeyConfirmed state and obtains a candidate key R_B, then Alice must have been involved in the protocol execution, she must have reached the KeyDerived state, and the two parties share the same key $R_A = R_B$ which is nearly uniform from Eve's point of view.

For formally defining the security of privacy amplification protocols, we first introduce the following random variables for any adversary Eve:

- We denote by $\mathsf{KeyDerived}_A$ and $\mathsf{KeyDerived}_B$ the indicators of the events in which Alice and Bob reach the KeyDerived state, respectively.

- We denote by $\mathsf{KeyConfirmed}_A$ and $\mathsf{KeyConfirmed}_B$ the indicators of the events in which Alice and Bob reach the KeyConfirmed state, respectively.

- We denote by V_E the random variable corresponding to the transcript of the protocol's execution as seen by Eve (i.e., Eve's view).

Definition 3.5 (Privacy amplification protocol)**.** *In an (n, k, m, ε)-privacy amplification protocol Alice and Bob share a weak secret $W \in \{0,1\}^n$ and have candidate keys $R_A, R_B \in \{0,1\}^m \cup \{\perp\}$, respectively. We require that for any n-bit weak secret W with min-entropy at least k the protocol satisfies the following properties:*

1. **Correctness:** *If Eve is passive then one party must reach the* KeyDerived *state, the other party must reach the* KeyConfirmed *state, and $R_A = R_B \in \{0,1\}^m$.*

2. **Privacy for Alice:** *For any adversary Eve, if $\Pr[\mathsf{KeyDerived}_A] > 0$ then*

$$\mathsf{SD}\left((R_A, V_E \mid \mathsf{KeyDerived}_A), (U_m, V_E \mid \mathsf{KeyDerived}_A)\right) \leq \varepsilon.$$

3. **Privacy for Bob:** *For any adversary Eve, if* $\Pr[\mathsf{KeyDerived}_B] > 0$ *then*

$$\mathsf{SD}\left((R_B, V_E \mid \mathsf{KeyDerived}_B), (U_m, V_E \mid \mathsf{KeyDerived}_B)\right) \leq \varepsilon.$$

4. **Authenticity:** *For any adversary Eve it holds that*

$$\Pr[(\mathsf{KeyConfirmed}_A \vee \mathsf{KeyConfirmed}_B) \wedge R_A \neq R_B] \leq \varepsilon.$$

Given an (n, k, m, ε)-privacy amplification protocol we refer to $k - m$ as its *entropy loss*, and to ε as its *security parameter*. We now elaborate and explain the different requirements in the above definition.

- First, the correctness requirement naturally asks that whenever the protocol is executed without any adversarial interference, then Alice and Bob output the same key (i.e., $R_A = R_B$) and this key is indeed an m-bit value (and not the symbol $\perp$). Note that at this point the definition does not ask for the resulting key to be uniformly distributed (it will follow from the privacy requirements that in this case the key is ε-close to uniform).

- Next, the privacy requirements for Alice and Bob ask that when focusing on executions in which Alice (respectively, Bob) derives a key R_A (respectively, R_B), then from Eve's point of view, this key is ε-close to an independently and uniformly sampled m-bit key. In particular, in such executions, Eve may be able to completely determine the key of the other party, but she learns essentially nothing about the (essentially uniform) key of the party that reached the $\mathsf{KeyDerived}$ state.

- Finally, the authenticity requirement asks that if one of the parties reaches the $\mathsf{KeyConfirmed}$ state, then the parties output the same key (i.e., $R_A = R_B$) except with probability at most ε. For understanding this requirement, assume without loss of generality that Alice is the party that reaches the $\mathsf{KeyConfirmed}$ state. By the semantics of privacy amplification protocols, as discussed above, Bob may either reach the $\mathsf{KeyDerived}$ state, or output $R_B = \perp$. In the first case (i.e., when Bob reaches the $\mathsf{KeyDerived}$ state), the privacy requirement for Bob states that from Eve's point of view, their resulting key is ε-close to an independently and uniformly sampled m-bit key. In the second case (i.e., when Bob outputs $\perp$), both Alice and Bob output $\perp$ and therefore they are both aware of the fact that Eve tried to interfere.

3.4.2 Message authentication codes

One-time message authentication codes (MACs) provide assurance to the receiver of a message that it was sent by a specified legitimate sender, even in the presence of an active and computationally unbounded adversary who controls the communication channel. A message-authentication code for messages of length n, keys of length ℓ, and authentication tags of length τ is defined via a family of deterministic and efficiently computable

functions $\{\mathsf{MAC}_{\mathsf{key}} : \{0,1\}^n \to \{0,1\}^\tau\}_{\mathsf{key}\in\{0,1\}^\ell}$. In terms of security the requirement is that any adversary that obtains an authentication tag on a single message m of her choice with respect to a uniform key, should have only a negligible probability of computing a valid authentication tag on a different message with respect to the same key.

Definition 3.6. *A family* $\{\mathsf{MAC}_{\mathsf{key}} : \{0,1\}^n \to \{0,1\}^\tau\}_{\mathsf{key}\in\{0,1\}^\ell}$ *of deterministic and efficiently computable functions is an* ε-secure one-time message authentication code *if for any message* $m \in \{0,1\}^n$ *and function* $A : \{0,1\}^\tau \to \{0,1\}^n \times \{0,1\}^\tau$ *it holds that*

$$\Pr_{\mathsf{key}\leftarrow\{0,1\}^\ell}[\mathsf{MAC}_{\mathsf{key}}(m') = \sigma' \ \wedge \ m \neq m' \mid (m',\sigma') = A(\mathsf{MAC}_{\mathsf{key}}(m))] \leq \varepsilon \ .$$

For the construction of our privacy amplification protocol we rely on the existence of message-authentication codes with the following parameters (see, for example, [KR09]):

Theorem 3.4. *For any integer* n *and* $\varepsilon > 0$ *there exists an explicit* ε*-secure message-authentication code* $\{\mathsf{MAC}_{\mathsf{key}} : \{0,1\}^n \to \{0,1\}^\tau\}_{\mathsf{key}\in\{0,1\}^\ell}$*, where* $\tau \leq \log n + \log(1/\varepsilon)$ *and* $\ell \leq 2\tau$*.*

3.5 A Central Lemma from [Raz05]

The following lemma is one of the main components that were used in the construction of two-sources-extractors in [Raz05]. We state the lemma for the special case of seeded-extractors, and prove it for completeness.

Lemma 3.7. *Let* $D = 2^d$*. Let* $Z_1, \ldots, Z_D$ *be* 0-1 *random variables that are* ε*-biased for linear tests of size* k' *that are constructed using* n *random bits. Define* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ *by* $\mathsf{Ext}(w,s) = Z_s(w)$*, that is,* $\mathsf{Ext}(w,s)$ *is the random variable* Z_s*, when using* w *as the value of the* n *bits needed to produce* $Z_1, \ldots, Z_D$*. Then, for any* $0 < \delta < 1/2$ *and even integer* $k \leq k'$ *such that* $k \cdot (1/\varepsilon)^{1/k} \leq D^{1/2}$*, the function* Ext *is a* $((1/2+\delta)\cdot n, \gamma)$*-seeded-extractor, with*

$$\gamma = \left(\varepsilon \cdot 2^{(1/2-\delta)n+1}\right)^{1/k}.$$

Proof. Let W be a $(n, (1/2+\delta)\cdot n)$-source. Let S be a random variable that is uniformly distributed over $\{0,1\}^d$ and is independent of W. We will show that the distribution of $\mathsf{Ext}(W,S)$ is γ-close to uniform. As in [CG88] (see Section 2.1.4), it is enough to consider the case where W is uniformly distributed over a set $W' \subseteq \{0,1\}^n$ of size $2^{(1/2+\delta)n}$. For every $w \in \{0,1\}^n$ and $s \in \{0,1\}^d$ denote

$$e(w,s) = (-1)^{Z_s(w)}.$$

Claim 3.7.1. *For any* $r \in [k]$ *and any different* $s_1, \ldots, s_r \in \{0,1\}^d$*,*

$$\sum_{w\in\{0,1\}^n} \prod_{j=1}^{r} e(w,s_j) \leq \varepsilon \cdot 2^n.$$

Proof.

$$\sum_{w\in\{0,1\}^n}\prod_{j=1}^{r}e(w,s_j)=\sum_{w\in\{0,1\}^n}\prod_{j=1}^{r}(-1)^{Z_{s_j}(w)}=\sum_{w\in\{0,1\}^n}(-1)^{Z_{s_1}(w)\oplus\cdots\oplus Z_{s_r}(w)},$$

and since $Z_{s_1}(w)\oplus\cdots\oplus Z_{s_r}(w)$ is ε-biased, the last sum is at most $\varepsilon\cdot 2^n$. □

Denote by $\gamma(W,S)$ the expectation of $e(W,S)$. We will show that $|\gamma(W,S)|\leq\gamma$. Obviously, this means that $\mathsf{Ext}(W,S)$ is γ-close to uniform, as required.

By the definition

$$2^{(1/2+\delta)n}\cdot 2^d\cdot\gamma(W,S)=\sum_{w\in W'}\sum_{s\in\{0,1\}^d}e(w,s).$$

Hence, by a convexity argument and since k is even,

$$2^{(1/2+\delta)n}\cdot\left(2^d\cdot\gamma(W,S)\right)^k\leq\sum_{w\in W'}\left(\sum_{s\in\{0,1\}^d}e(w,s)\right)^k\leq$$

$$\sum_{w\in\{0,1\}^n}\left(\sum_{s\in\{0,1\}^d}e(w,s)\right)^k=\sum_{w\in\{0,1\}^n}\sum_{s_1,\ldots,s_k\in\{0,1\}^d}\prod_{j=1}^{k}e(w,s_j)$$

$$=\sum_{s_1,\ldots,s_k\in\{0,1\}^d}\sum_{w\in\{0,1\}^n}\prod_{j=1}^{k}e(w,s_j).$$

We will break the sum over $s_1,\ldots,s_k\in\{0,1\}^d$ into two sums. The first sum is over $s_1,\ldots,s_k\in\{0,1\}^d$ such that at least one s_j is different than all other elements in $\{s_1,\ldots,s_k\}$, and the second sum is over $s_1,\ldots,s_k\in\{0,1\}^d$ such that every s_j is identical to at least one other element in $\{s_1,\ldots,s_k\}$. The number of summands in the first sum is trivially bounded by $2^{d\cdot k}$, and by Claim 3.7.1 each summand is bounded by $2^n\cdot\varepsilon$. The number of summands in the second sum is bounded by $2^{d\cdot k/2}\cdot(k/2)^k$, and each summand is trivially bounded by 2^n. Hence,

$$\begin{aligned}2^{(1/2+\delta)\cdot n}\cdot 2^{d\cdot k}\cdot\gamma(W,S)^k&\leq 2^n\cdot\varepsilon\cdot 2^{d\cdot k}+2^n\cdot 2^{d\cdot k/2}\cdot(k/2)^k\\&\leq 2\cdot 2^n\cdot\varepsilon\cdot 2^{d\cdot k},\end{aligned}$$

where the last inequality follows by the assumption that $k\cdot(1/\varepsilon)^{1/k}\leq D^{1/2}$. That is,

$$|\gamma(W,S)|\leq\left(\varepsilon\cdot 2^{(1/2-\delta)n+1}\right)^{1/k}.$$

□

3.6 A Simple Lemma about Graphs

The following simple lemma about graphs is another ingredient we need for the proof of Theorem 3.1.

Lemma 3.8. *Let $G = (V, E)$ be a directed graph without self-loops. Assume that the out-degree of each vertex is exactly t, where parallel edges are allowed. Let $w\colon V \to \mathbb{R}$ be a weight function on the vertices of G. Denote by ω the average vertex weight, that is, $\omega = \frac{1}{|V|} \cdot \sum_{v \in V} w(v)$. Then, there exists a subset of the vertices $V' \subseteq V$, such that the induced graph $H = (V', E')$ of G by the set of vertices V' has the following properties:*

1. *H is acyclic,*
2. *The average vertex weight of H is at least $\omega/(t+1)$, that is, $\frac{1}{|V'|} \cdot \sum_{v \in V'} w(v) \geq \omega/(t+1)$,*
3. *$|V'| \geq |V|/(t+1)$.*

Proof. We construct H by a greedy algorithm. During the running of the algorithm, every vertex in V has one of the following statuses: *available*, *chosen* or *forbidden*. We say that a vertex is available if it has an available status. Similarly, we say that a vertex is chosen / forbidden if it has a chosen / forbidden status. For every vertex $v \in V$ we denote by $\mathsf{status}(v)$ the status of the vertex v. For every vertex v, let $v^+ = \{u \in V : (v, u) \in E\}$. The greedy algorithm is defined as follows:

1. For every vertex $v \in V$ initialize $\mathsf{status}(v) \leftarrow \mathsf{available}$.
2. While there exists an available vertex,
   (a) Let v be an available vertex such that $w(v) \geq w(v')$ for any available vertex v'.
   (b) Set $\mathsf{status}(v) \leftarrow \mathsf{chosen}$.
   (c) For every vertex $v' \in v^+$, if $\mathsf{status}(v') = \mathsf{available}$ set $\mathsf{status}(v') \leftarrow \mathsf{forbidden}$.
3. Return $V' = \{v\colon \mathsf{status}(v) = \mathsf{chosen}\}$.

Assume for contradiction that H contains a cycle C, that is, C is a cycle of chosen vertices. Let v be the first chosen vertex in C. Let $v' \in v^+$ be the vertex that follows v in C. At the time v was chosen, v' was available, and so the algorithm set the status of v' to forbidden. A contradiction is then met as a forbidden vertex is never chosen and so v' cannot be in C.

We now prove property 2. Once the algorithm terminates, the status of every vertex is either chosen or forbidden. For every chosen veretx v, let $v^A \subseteq v^+$ be the set of vertices that were available at the time v was chosen. The vertices of the graph G can be partitioned as follows:

$$V = \bigcup_{v \in V'} \left(\{v\} \cup v^A\right). \tag{3.1}$$

By Equation (3.1) and by the fact that all (at most t) vertices in v^A have a weight which is no more than $w(v)$, we get

$$\begin{aligned}\sum_{v\in V} w(v) &= \sum_{v\in V'} \Big(w(v) + \sum_{v'\in v^A} w(v')\Big) \\ &\leq \sum_{v\in V'} \Big(w(v) + w(v)\cdot|v^A|\Big) \\ &\leq (t+1)\sum_{v\in V'} w(v).\end{aligned}$$

Hence,

$$\begin{aligned}\frac{1}{|V'|}\cdot\sum_{v\in V'} w(v) &\geq \frac{1}{|V'|}\cdot\frac{1}{t+1}\cdot\sum_{v\in V} w(v) \\ &\geq \frac{1}{t+1}\cdot\frac{1}{|V|}\cdot\sum_{v\in V} w(v) \\ &= \frac{\omega}{t+1}.\end{aligned}$$

This proves property 2. By Equation (3.1)

$$|V| = \sum_{v\in V'}\big(1+|v^A|\big) \leq \sum_{v\in V'}\big(1+t\big) = (t+1)\cdot|V'|,$$

which proves property 3. □

3.7 A Conditional Parity Lemma

The following lemma is a generalization of the Parity Lemma (usually attributed to Vazirani. See for example [NN93]). A similar lemma appears in [DLWZ11b]. Let Z be a random variable over $\{0,1\}^{m+n}$. Lemma 3.9 states that given the suffix of length n of Z, one can bound the statistical distance between the remaining length m prefix of Z and the uniform distribution in terms of appropriate biases. Setting $n = 0$ yields the Parity Lemma.

Lemma 3.9. *Let X be a random variable over $\{0,1\}^m$. Let Y be a random variable over $\{0,1\}^n$. Then,*

$$\big\|(X,Y) - (U_m,Y)\big\|_1 \leq \Big(\sum_{\substack{\emptyset\neq\sigma\subseteq[m]\\ \tau\subseteq[n]}} \mathrm{bias}(X_\sigma\oplus Y_\tau)^2\Big)^{1/2},$$

where $X_\sigma = \bigoplus_{i\in\sigma} X_i$ and $Y_\tau = \bigoplus_{i\in\tau} Y_i$.

We derive two corollaries from Lemma 3.9.

Corollary 3.5. *Let X be a random variable over $\{0,1\}^m$. Let Y be a random variable over $\{0,1\}^n$. Then,*

$$\left\|(X,Y)-(U_m,Y)\right\|_1 \leq \sum_{\substack{\emptyset\neq\sigma\subseteq[m]\\ \tau\subseteq[n]}} \mathrm{bias}(X_\sigma \oplus Y_\tau).$$

Corollary 3.6. *Let X be a random variable over $\{0,1\}^m$. Let Y be a random variable over $\{0,1\}^n$. Then,*

$$\left\|(X,Y)-(U_m,Y)\right\|_1 \leq ((2^m-1)\cdot 2^n)^{1/2} \cdot \max_{\substack{\emptyset\neq\sigma\subseteq[m]\\ \tau\subseteq[n]}} \mathrm{bias}(X_\sigma \oplus Y_\tau).$$

Deriving both corollaries from Lemma 3.9 can be done by applying basic norms inequalities.

Proof. Lemma 3.9 Let $D \in \mathbb{R}^{2^{m+n}}$. We index the entries of D by strings of length $m+n$ bits. For $x \in \{0,1\}^m$ and $y \in \{0,1\}^n$, we define

$$D(xy) = \Pr((X,Y)=(x,y)) - \Pr((U_m,Y)=(x,y)),$$

where xy is the concatenation of x and y. By Parseval and basic norms inequalities,

$$\begin{aligned}
\left\|(X,Y)-(U_m,Y)\right\|_1^2 &= \Big(\sum_{\substack{x\in\{0,1\}^m\\ y\in\{0,1\}^n}} \left|\Pr((X,Y)=(x,y)) - \Pr((U_m,Y)=(x,y))\right|\Big)^2 \\
&= \Big(\sum_{\substack{x\in\{0,1\}^m\\ y\in\{0,1\}^n}} |D(xy)|\Big)^2 \leq 2^{m+n}\cdot \sum_{\substack{x\in\{0,1\}^m\\ y\in\{0,1\}^n}} D(xy)^2 \qquad (3.2)\\
&= 2^{m+n}\cdot \|D\|_2^2 = 2^{2(m+n)}\cdot \|\widehat{D}\|_2^2, \qquad (3.3)
\end{aligned}$$

where $\widehat{D}$ is the Fourier transform of D (we refer the reader to the book of O'Donnell [O'D] for information regarding Fourier analysis of Boolean functions).

Claim 3.9.1. *For every $\sigma \subseteq [m]$ and $\tau \subseteq [n]$*[7]*,*

$$|\widehat{D}(\sigma\tau)| = \begin{cases} 2^{-(m+n)}\cdot \mathrm{bias}(X_\sigma \oplus Y_\tau), & \sigma \neq \emptyset; \\ 0, & \sigma = \emptyset. \end{cases}$$

[7]We slightly abuse notation and identify sets in $[m]$ with their characteristic vectors over $\{0,1\}^m$.

Proof. For every $\sigma \subseteq [m]$ and $\tau \subseteq [n]$,

$$\begin{aligned}\widehat{D}(\sigma\tau) &= \frac{1}{2^{m+n}} \cdot \sum_{\substack{x\in\{0,1\}^m \\ y\in\{0,1\}^n}} (-1)^{\langle \sigma\tau, xy\rangle} \cdot D(xy) \\ &= \frac{1}{2^{m+n}} \cdot \sum_{\substack{x\in\{0,1\}^m \\ y\in\{0,1\}^n}} (-1)^{\langle \sigma\tau, xy\rangle} \cdot \Pr\left((X,Y) = (x,y)\right) \\ &\quad - \frac{1}{2^{m+n}} \cdot \sum_{\substack{x\in\{0,1\}^m \\ y\in\{0,1\}^n}} (-1)^{\langle \sigma\tau, xy\rangle} \cdot \Pr\left((U_m,Y) = (x,y)\right).\end{aligned}$$

We note that

$$\sum_{\substack{x\in\{0,1\}^m \\ y\in\{0,1\}^n}} (-1)^{\langle \sigma\tau, xy\rangle} \cdot \Pr((U_m,Y) = (x,y)) = \sum_{y\in\{0,1\}^n} (-1)^{\langle \tau, y\rangle} \cdot \Pr(Y = y) \cdot \frac{1}{2^m} \sum_{x\in\{0,1\}^m} (-1)^{\langle \sigma, x\rangle}.$$

For $\sigma \neq \emptyset$, it holds that $\sum_{x\in\{0,1\}^m} (-1)^{\langle \sigma, x\rangle} = 0$. Hence, for $\sigma \neq \emptyset$,

$$\begin{aligned}|\widehat{D}(\sigma\tau)| &= \left| \frac{1}{2^{m+n}} \cdot \sum_{\substack{x\in\{0,1\}^m \\ y\in\{0,1\}^n}} (-1)^{\langle \sigma\tau, xy\rangle} \cdot \Pr((X,Y) = (x,y)) \right| \\ &= \frac{1}{2^{m+n}} \cdot \mathrm{bias}(X_\sigma \oplus Y_\tau).\end{aligned}$$

For $\sigma = \emptyset$, it holds that $\sum_{x\in\{0,1\}^m} (-1)^{\langle \sigma, x\rangle} = 2^m$. Therefore, for $\sigma = \emptyset$,

$$\begin{aligned}\widehat{D}(\sigma\tau) &= \frac{1}{2^{m+n}} \cdot \sum_{\substack{x\in\{0,1\}^m \\ y\in\{0,1\}^n}} (-1)^{\langle \tau, y\rangle} \cdot \Pr((X,Y) = (x,y)) \\ &\quad - \frac{1}{2^{m+n}} \cdot \sum_{y\in\{0,1\}^n} (-1)^{\langle \tau, y\rangle} \cdot \Pr(Y = y) \\ &= \frac{1}{2^{m+n}} \cdot \sum_{y\in\{0,1\}^n} (-1)^{\langle \tau, y\rangle} \left(\sum_{x\in\{0,1\}^m} \Pr((X,Y) = (x,y)) \;\; - \Pr(Y = y) \right) \\ &= 0.\end{aligned}$$

□

By Equation (3.2) and Claim 3.9.1,

$$\begin{aligned}\left\|(X,Y) - (U_m,Y)\right\|_1^2 &\leq 2^{2(m+n)} \cdot \|\widehat{D}\|_2^2 = 2^{2(m+n)} \cdot \sum_{\substack{\sigma\subseteq[m] \\ \tau\subseteq[n]}} \widehat{D}(\sigma\tau)^2 \\ &= \sum_{\substack{\emptyset\neq\sigma\subseteq[m] \\ \tau\subseteq[n]}} \mathrm{bias}(X_\sigma \oplus Y_\tau)^2,\end{aligned}$$

which concludes the proof of the lemma. □

3.8 Proof of Main Theorem

To ease the reading we restate the main theorem.

Theorem 3.7 (Theorem 8.1 – Restated)**.** *For any integers n, d, m and t, and for any $0 < \delta < 1/2$ such that*

$$\begin{aligned} d \geq & \frac{23}{\delta} \cdot tm + 2\log n, \\ n \geq & \frac{160}{\delta} \cdot tm, \\ \delta \geq & 10 \cdot \frac{\log(nd)}{n}, \end{aligned}$$

there exists an explicit $((1/2+\delta)\cdot n, 2^{-m})$-$t$-non-malleable extractor $\mathsf{nmExt}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$.

Proof. Theorem 3.7 Let $D = m \cdot 2^d$. Let $k' = \lceil \delta n/8 \rceil$. Let $\varepsilon = 2^{-n/2+r}$, where $r = 1 + \log(k') + \log\log(D)$. The explicit construction we present is the extractor constructed in [Raz05]. We now describe it. Let $Z_1, \ldots, Z_D$ be 0-1 random variables that are ε-biased for linear tests of size k' that are constructed using n random bits. It is easy to verify that

$$n \geq 2 \cdot \lceil \log(1/\varepsilon) + \log k' + \log\log D \rceil,$$

and so by [AGHP92] (see Section 2.4) such a construction is indeed possible.

We think of the set of indices $[D]$ as the set $\{(i,s)\colon i \in [m], s \in \{0,1\}^d\}$. We define $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ by $\mathsf{Ext}_i(w,s) = Z_{(i,s)}(w)$, where $\mathsf{Ext}_i(w,s)$ denotes the i^{th} bit of $\mathsf{Ext}(w,s)$. In other words, w is used to choose the point in the probability space and i, s are used to choose the variable from $Z_1, \ldots, Z_D$ that we evaluate.

Let S be a random variable uniformly distributed over $\{0,1\}^d$. Assume for contradiction that Ext is not a $((1/2+\delta)\cdot n, 2^{-m})$-$t$-non-malleable-extractor. Then, there exists a source W of length n and min-entropy $(1/2+\delta)\cdot n$, and a t-adversarial-function $\mathcal{A}\colon \{0,1\}^d \to \{0,1\}^{td}$ such that,

$$\begin{aligned} \| & \big(\mathsf{Ext}(W,S), \mathsf{Ext}(W,\mathcal{A}_1(S)), \ldots, \mathsf{Ext}(W,\mathcal{A}_t(S)), S\big) \\ & - \big(U_m, \mathsf{Ext}(W,\mathcal{A}_1(S)), \ldots, \mathsf{Ext}(W,\mathcal{A}_t(S)), S\big)\|_1 > 2^{-m}. \end{aligned} \tag{3.4}$$

As in [CG88] (see Section 2.1.4), we may assume that W is uniformly distributed over a set $W' \subseteq \{0,1\}^n$ of size $2^{(1/2+\delta)\cdot n}$.

For every $s \in \{0,1\}^d$ let X_s be the random variable $\mathsf{Ext}(W,s)$. By Equation (3.4) and Lemma 2.9,

$$\mathbb{E}_{s\sim S}\Big[\|\big(X_s, X_{\mathcal{A}_1(s)}, \ldots, X_{\mathcal{A}_t(s)}\big) - \big(U_m, X_{\mathcal{A}_1(s)}, \ldots, X_{\mathcal{A}_t(s)}\big)\|_1\Big] > 2^{-m}.$$

Hence, by Corollary 3.5,

$$\sum_{\substack{\emptyset\neq\sigma\subseteq[m]\\ \tau_1,\ldots,\tau_t\subseteq[m]}} \mathbb{E}_{s\sim S}\Big[\mathrm{bias}\Big((X_s)_\sigma \oplus \big(\bigoplus_{i\in[t]}(X_{\mathcal{A}_i(s)})_{\tau_i}\big)\Big)\Big] > 2^{-m}.$$

Let $\sigma^*, \tau_1^*, \ldots, \tau_t^* \subseteq [m]$ be the indices of (one of) the largest summands in the above sum. For every $s \in \{0,1\}^d$, let

$$Y_s = (X_s)_{\sigma^*} \oplus \big(\bigoplus_{i\in[t]}(X_{\mathcal{A}_i(s)})_{\tau_i^*}\big).$$

Then,

$$\mathbb{E}_{s\sim S}\big[\mathrm{bias}(Y_s)\big] > 2^{-(t+2)m}.$$

Let $G = (V, E)$ be a directed graph with $V = \{0,1\}^d$ and $E = \{(s, \mathcal{A}_j(s)) \colon s \in V, j \in [t]\}$. Since $\mathcal{A}$ is a t-adversarial-function, G has no self-loops. Equip G with a weight function on the vertices $w \colon V \to \mathbb{R}$ that is defined as follows: For every $s \in V$, $w(s) = \mathrm{bias}(Y_s)$.

By Lemma 3.8, there exists a subset $V' \subseteq V$ such that the induced graph, H, of G by V' has the properties mentioned in that lemma. In particular, by properties 2 and 3,

$$\mu \triangleq \frac{1}{|V'|} \cdot \sum_{s\in V'} \mathrm{bias}(Y_s) > \frac{2^{-(t+2)m}}{t+1},$$

and

$$|V'| \geq \frac{|V|}{t+1} = \frac{2^d}{t+1}.$$

By Markov's inequality, there exists a subset $S' \subseteq V'$ such that

$$|S'| \geq \frac{\mu}{2} \cdot |V'| \geq \frac{2^{d-(t+2)m-1}}{(t+1)^2},$$

and for all $s \in S'$

$$\mathrm{bias}(Y_s) \geq \frac{\mu}{2} > \frac{2^{-(t+2)m-1}}{t+1}.$$

Let $S'_0 = \{s \in S' \colon \Pr[Y_s = 0] \geq 1/2\}$. Let $S'_1 = S' \setminus S'_0$. There exists $b \in \{0,1\}$ such that $|S'_b| \geq |S'|/2$. Denote S'_b by S''. Then,

$$|S''| \geq \frac{|S'|}{2} \geq \frac{2^{d-(t+2)m-2}}{(t+1)^2},$$

and for all $s \in S''$

$$\Pr(Y_s = b) - \Pr(Y_s \neq b) > \frac{2^{-(t+2)m-1}}{t+1}. \tag{3.5}$$

Define a random variable $Y_{S''}$ over $\{0,1\}$ as follows: To sample a bit from $Y_{S''}$, uniformly sample a string s from S'', and then independently sample a string w uniformly from W'. The sampled value is $Y_s(w)$. We have that

$$\begin{aligned}\text{bias}(Y_{S''}) &= |\Pr(Y_{S''}=0) - \Pr(Y_{S''}=1)| \\ &= \frac{1}{|S''|}\cdot\Big|\sum_{s\in S''}\Pr(Y_s=0)-\Pr(Y_s=1)\Big| \\ &= \frac{1}{|S''|}\cdot\sum_{s\in S''}\big|\Pr(Y_s=0)-\Pr(Y_s=1)\big| \\ &> \frac{2^{-(t+2)m-1}}{t+1},\end{aligned} \tag{3.6}$$

where the inequality and equality before it follow by Equation (3.5).

For every $s \in S''$, let

$$Y'_s = \Big(\bigoplus_{j\in\sigma^*} Z_{(j,s)}\Big) \oplus \Big(\bigoplus_{i\in[t]}\Big(\bigoplus_{j\in\tau_i^*} Z_{(j,\mathcal{A}_i(s))}\Big)\Big).$$

Claim 3.9.2. *The set of random variables $\{Y'_s\}_{s\in S''}$ ε-fools linear tests of size $k'/((t+1)m)$.*

Proof. Let $A \subseteq S''$ be a nonempty set of size at most $\ell \triangleq k'/((t+1)m)$, and let $Y_A = \bigoplus_{s\in A} Y'_s$. We note that Y_A is a linear combination of random variables from $(Z_{(i,s)})_{i\in[m],s\in\{0,1\}^d}$, composed of at most k' summands. Since $(Z_{(i,s)})_{i\in[m],s\in\{0,1\}^d}$ ε-fools linear tests of size k', it is enough to show that this linear combination is non-trivial, that is, it is suffices to prove that Y_A is not the constant 0 random variable.

Assume for contradiction that $Y_A = 0$. Let e be an arbitrary element in σ^*. Such an element exists as $\sigma^* \neq \emptyset$. Let s_1 be an arbitrary element in A. Such an element exists as $A \neq \emptyset$. The random variable $Z_{(e,s_1)}$ is therefore a summand in Y_A.

By the assumption that $Y_A = 0$, $Z_{(e,s_1)}$ must appear an even number of times as a summand in Y_A. In particular, $Z_{(e,s_1)}$ must appear at least one more time as a summand in Y_A. Therefore, there exists some $s_2 \in A$ and $i_2 \in [t]$ such that $\mathcal{A}_{i_2}(s_2) = s_1$.

Since $\mathcal{A}$ is an adversarial-function, $s_2 \neq s_1$ and so the random variable $Z_{(e,s_2)}$ is a summand in Y_A which is different than $Z_{(e,s_1)}$. Following the same logic as above, since $Y_A = 0$, the random variable $Z_{(e,s_2)}$ must appear an even number of times as a summand in Y_A. In particular, it must appear at least one more time. Hence, there exists some $s_3 \in A$ and $i_3 \in [t]$ such that $\mathcal{A}_{i_3}(s_3) = s_2$.

We continue this way to get a sequence $s_1, s_2, s_3, \ldots$ of elements of A until two elements in the sequence are equal. Since A is finite, this is bound to happen. However, in such case, a directed cycle in the graph H is implied, contradicting its acyclicity. □

Let k be the largest even integer that is not larger than $k'/((t+1)m)$.

Claim 3.9.3.

$$\frac{1}{10}\cdot\frac{\delta n}{(t+1)m}\leq k\leq\frac{1}{5}\cdot\frac{\delta n}{(t+1)m}$$

Proof. As k is the largest even integer that is not larger than $k'/((t+1)m)$,

$$k\in\left\{\left\lfloor\frac{k'}{(t+1)m}\right\rfloor-1,\left\lfloor\frac{k'}{(t+1)m}\right\rfloor\right\}.$$

Therefore,

$$k\leq\left\lfloor\frac{k'}{(t+1)m}\right\rfloor\leq\frac{k'}{(t+1)m}=\frac{\lceil\delta n/8\rceil}{(t+1)m}\leq\frac{1}{8}\cdot\frac{\delta n}{(t+1)m}+\frac{1}{(t+1)m}\leq\frac{1}{5}\cdot\frac{\delta n}{(t+1)m},$$

where the last inequality follows by the assumption that $\delta\geq 160\cdot tm/n\geq 40/(3\cdot n)$. As for the lower bound on k,

$$k\geq\left\lfloor\frac{k'}{(t+1)m}\right\rfloor-1\geq\frac{k'}{(t+1)m}-2=\frac{\lceil\delta n/8\rceil}{(t+1)m}-2\geq\frac{1}{8}\cdot\frac{\delta n}{(t+1)m}-2\geq\frac{1}{10}\cdot\frac{\delta n}{(t+1)m},$$

where, again, the last inequality follows by the assumption that

$$\delta\geq 160\cdot\frac{tm}{n}\geq\frac{80(t+1)m}{n}.$$

□

We apply Lemma 3.7 on the random variables $\{Y'_s\}_{s\in S''}$[8]. The following claim confirms that the assumption of Lemma 3.7 is indeed met with the k defined above.

Claim 3.9.4.

$$k\cdot\left(\frac{1}{\varepsilon}\right)^{1/k}\leq\left(\frac{2^{d-(t+2)m-2}}{(t+1)^2}\right)^{1/2}. \tag{3.7}$$

Proof. Taking the $\log(\cdot)$ of both sides of Equation (3.7) and rearranging the terms, we see it is enough to prove that

$$d\geq(t+2)m+2\log k+\frac{2}{k}\log\frac{1}{\varepsilon}+2\log(t+1)+2. \tag{3.8}$$

By Claim 3.9.3

$$\frac{2}{k}\cdot\log\frac{1}{\varepsilon}=\frac{2}{k}\cdot\left(\frac{n}{2}-r\right)\leq\frac{n}{k}\leq\frac{10(t+1)m}{\delta}, \tag{3.9}$$

and

$$\log k\leq\log\left(\frac{\delta n}{5(t+1)m}\right)\leq\log\left(\frac{n}{20}\right). \tag{3.10}$$

[8]For simplicity of presentation we assume $|S''|$ is a power of 2. The exact same result can be obtained regardless of this assumption.

By Equations (3.8), (3.9) and (3.10), it is enough to show that

$$d \geq \left(\frac{10(t+1)}{\delta} + t + 2\right) m + 2\log n + 2\log(t+1) + 2 - 2\log 20.$$

Since for all $t \geq 1$

$$\frac{22}{\delta} \cdot t \geq \frac{10(t+1)}{\delta} + t + 2$$

and

$$2\log t \geq 2\log(t+1) + 2 - 2\log 20,$$

it is enough to prove that

$$d \geq \frac{22}{\delta} \cdot tm + 2\log n + 2\log t.$$

The above equation holds as we assume

$$d \geq \frac{23}{\delta} \cdot tm + 2\log n.$$

□

Consider the weak-source W. By Lemma 3.7, the distribution of $\mathbf{E}(W, S'')$ is γ-biased, for $\gamma = (\varepsilon \cdot 2^{1+(1/2-\delta)n})^{1/k} = 2^{(1+r-\delta n)/k}$. However, we note that $\mathbf{E}(W, S'')$ has the same distribution as $Y_{S''}$. In particular, both random variables have the same bias. Equation (3.6) yields that

$$2^{(1+r-\delta n)/k} \geq \text{bias}(\mathbf{E}(W, S'')) = \text{bias}(Y_{S''}) > \frac{2^{-(t+2)m-1}}{t+1}. \tag{3.11}$$

We conclude the proof of Theorem 3.7 by the following claim, that stands in contradiction to Equation (3.11).

Claim 3.9.5.

$$2^{(1+r-\delta n)/k} < \frac{2^{-(t+2)m-1}}{t+1}.$$

Proof. It is enough to prove that

$$\frac{\delta n}{k} > (t+2)m + \log\big(4(t+1)\big) + \frac{r}{k}.$$

By Claim 3.9.3, it is enough to show that

$$(4t+3)m > \log\big(4(t+1)\big) + \frac{r}{k}.$$

Since for all $n \geq 2$ (indeed $n \geq 160 \cdot tm/\delta \geq 320$),

$$k' = \left\lceil \frac{\delta n}{8} \right\rceil \leq \frac{n}{2},$$

we have that

$$r = 1 + \log k' + \log\log D = \log\left(2k'(d + \log m)\right) \leq \log(ndm). \tag{3.12}$$

By Equation (3.12) and Claim 3.9.3 we have that

$$\frac{r}{k} \leq 10(t+1)m\frac{\log(ndm)}{\delta n}.$$

It is therefore enough to prove that

$$(4t+3)m > \log\big(4(t+1)\big) + 10(t+1)m\frac{\log(ndm)}{\delta n}. \tag{3.13}$$

To prove Equation (3.13) for all $t \geq 1$, it is enough to show that

$$m \geq \frac{3}{7} + \frac{20}{7} \cdot m \cdot \frac{\log(ndm)}{\delta n},$$

which holds if

$$n \geq \frac{5}{\delta} \cdot \log(ndm). \tag{3.14}$$

Since $m < n$ (indeed, by the second assumption of Theorem 3.7, $m \leq \delta n/(160t) \leq n/320$), Equation (3.14) holds by the third assumption of Theorem 3.7. □

□

3.9 The Privacy Amplification Protocol

In the section we present the protocol that is obtained by instantiating the Dodis-Wichs protocol [DW09] with our non-malleable extractor. Given the length n of the weak source and parameters k, ε', $\varepsilon_{\mathsf{nmExt}}$, $\varepsilon_{\mathsf{Ext}}$, and $\varepsilon_{\mathsf{MAC}}$ that we will fix later, the protocol relies on the following building blocks:

1. A non-malleable $(k, \varepsilon_{\mathsf{nmExt}})$-extractor $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^{d_1} \to \{0,1\}^\ell$ (see Definition 3.2).
2. A strong $(k - (d_1 + \ell) - \log(1/\varepsilon'), \varepsilon_{\mathsf{Ext}})$-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{d_2} \to \{0,1\}^m$ (see Definition 2.16).
3. An $\varepsilon_{\mathsf{MAC}}$-secure message authentication code $\left\{\mathsf{MAC}_{\mathsf{key}} : \{0,1\}^{d_2} \to \{0,1\}^\tau\right\}_{\mathsf{key}\in\{0,1\}^\ell}$ (see Definition 3.6).

The protocol is described in Figure 1. The following theorem was proved in [DW09], and we provide here its proof for completeness.

Theorem 3.8. *Let* nmExt, Ext *and* MAC *be as specified above. Then for any integers n and $k < n$, the protocol in Figure 1 is a 2-round (n, k, m, ε)-privacy amplification protocol, with communication complexity $d_1 + d_2 + \tau$, where $\varepsilon = \max\{\varepsilon' + \varepsilon_{\mathsf{Ext}}, \varepsilon_{\mathsf{nmExt}} + \varepsilon_{\mathsf{MAC}}\}$.*

For obtaining explicit protocols we instantiate the building blocks nmExt, and MAC with those provided by Theorem 3.1, and Theorem 3.4, respectively. In addition, we instantiate the strong extractor Ext by either one of those provided by Theorem 2.1, Theorem 2.2, or Theorem 2.3. We obtain the following two theorems:

Theorem 3.9. *For any integer n, constant $\delta > 0$, and security parameter $\varepsilon = 2^{-O(n)}$, there exists an explicit and efficient 2-round $(n, (1/2+\delta)n, m, \varepsilon)$-privacy amplification protocol with entropy loss $O(\log(n/\varepsilon))$, and communication complexity $O(\min\left\{\log^2 n + \log n \cdot \log(1/\varepsilon), n\right\})$.*

Theorem 3.10. *For any integer n, constants δ and β such that $1/2+\delta > \beta > 0$, and security parameter $\varepsilon = 2^{-O(n)}$, there exists an explicit and efficient 2-round $(n, (1/2+\delta)n, m, \varepsilon)$-privacy amplification protocol with entropy loss $\beta n + O(\log(n/\varepsilon))$, and communication complexity $O(\log(n/\varepsilon))$.*

Shared input: Alice and Bob share a sample from an (n,k)-source W.

The protocol:

1. Alice samples $Y \leftarrow \{0,1\}^{d_1}$ uniformly at random, sends it to Bob, and computes $\mathsf{key} = \mathsf{nmExt}(W, Y)$.
2. Denote by Y' the value received by Bob, who then computes $\mathsf{key}' = \mathsf{nmExt}(W, Y')$.
3. Bob samples $S' \leftarrow \{0,1\}^{d_2}$ uniformly at random, computes $\sigma' = \mathsf{MAC}_{\mathsf{key}'}(S')$, and sends the pair (S', σ') to Alice.
4. Bob reaches the $\mathsf{KeyDerived}$ state and outputs $R_B = \mathsf{Ext}(W, S')$.
5. Denote by (S, σ) the pair received by Alice. If $\sigma = \mathsf{MAC}_{\mathsf{key}}(S)$ then Alice reaches the $\mathsf{KeyConfirmed}$ state and outputs $R_A = \mathsf{Ext}(W, S)$. Otherwise, Alice outputs $R_A = \perp$.

Figure 1: The Dodis-Wichs privacy amplification protocol.

In the remainder of this section we prove Theorems 3.8, 3.9, and 3.10.

Proof. Theorem 3.8 The correctness of the protocol and the parameters specified in the theorem follow directly from the description of the protocol. Thus, it only remains to argue the privacy and authenticity properties of the protocol. Since no assumptions are made on the computational capabilities of Eve, we assume without loss of generality that Eve is deterministic. Specifically, this implies that the value Y' is a deterministic function of the value Y, and then the pair (S, σ) is a deterministic function of the vector (Y, S', σ'). Therefore, without loss of generality, we refer to the view of Eve in the protocol as the vector $V_E = (Y, S', \sigma')$.

We argue the privacy property of the protocol in Lemma 3.10 and the authenticity property of the protocol in Lemma 3.11. For arguing the privacy of the protocol note that Bob always reaches the $\mathsf{KeyDerived}$ state and Alice never reaches the $\mathsf{KeyDerived}$ state

(i.e., $\Pr[\mathsf{KeyDerived}_A] = 0$ and $\Pr[\mathsf{KeyDerived}_B] = 1$). Therefore we only need to bound $\mathsf{SD}\left((R_B, V_E \mid \mathsf{KeyDerived}_B), (U_m, V_E \mid \mathsf{KeyDerived}_B)\right)$.

Lemma 3.10 (Privacy)**.** *It holds that*

$$\mathsf{SD}\left((R_B, V_E \mid \mathsf{KeyDerived}_B), (U_m, V_E \mid \mathsf{KeyDerived}_B)\right) \leq \varepsilon' + \varepsilon_{\mathsf{Ext}} \ .$$

Proof. The protocol specifies that Bob always reaches the $\mathsf{KeyDerived}$ state and outputs the value $R_B = \mathsf{Ext}(W, S')$. In addition, recall that Eve's view consists of $V_E = (Y, S', \sigma')$. Therefore, where the last inequality follows from the fact that the value σ' is a deterministic function of the values S' and key'. Thus, since the value S' is uniformly distributed and independent of W, Y, and key', in order to complete the argument we only need to prove that with high probability W has sufficient min-entropy given the pair (Y, key'). This follows from the fact that the latter pair (Y, key') is of total length $d_1 + \ell$ bits. Formally, Lemma 2.4 implies that with probability $1 - \varepsilon'$ over the choice of $(y, \kappa') \leftarrow (Y, \mathsf{key}')$ it holds that

$$H_\infty\left(W \mid Y = y, \mathsf{key}' = \kappa'\right) \geq k - (d_1 + \ell) - \log(1/\varepsilon').$$

In turn, the fact that Ext is a strong $(k - (d_1 + \ell) - \log(1/\varepsilon'), \varepsilon_{\mathsf{Ext}})$-extractor yields

$$\begin{aligned}
&\mathsf{SD}\left((R_B, V_E \mid \mathsf{KeyDerived}_B), (U_m, V_E \mid \mathsf{KeyDerived}_B)\right) \leq \\
&\mathsf{SD}\left((\mathsf{Ext}(W, S'), Y, S', \mathsf{key}'), (U_m, Y, S', \mathsf{key}')\right) \leq \\
&\varepsilon' + \varepsilon_{\mathsf{Ext}} \ .
\end{aligned}$$

□

Lemma 3.11 (Authenticity)**.** *It holds that*

$$\Pr\left[(\mathsf{KeyConfirmed}_A \vee \mathsf{KeyConfirmed}_B) \wedge (R_A \neq R_B)\right] \leq \varepsilon_{\mathsf{nmExt}} + \varepsilon_{\mathsf{MAC}} \ .$$

Proof. The protocol specifies that only Alice may reach the $\mathsf{KeyConfirmed}$ state, and therefore

$$\Pr\left[(\mathsf{KeyConfirmed}_A \vee \mathsf{KeyConfirmed}_B) \wedge (R_A \neq R_B)\right] = \Pr\left[\mathsf{KeyConfirmed}_A \wedge (R_A \neq R_B)\right].$$

We now consider two cases: one in which $S = S'$ (i.e., Eve does not modify S') and the other in which $S \neq S'$ (i.e., Eve does modify S').

Case 1: $S = S'$. In this case Alice either reaches the $\mathsf{KeyConfirmed}$ state and outputs $R_A = \mathsf{Ext}(W, S) = \mathsf{Ext}(W, S') = R_B$ or does not reach the $\mathsf{KeyConfirmed}$ state and outputs $R_A = \perp$. This implies that

$$\Pr\left[\mathsf{KeyConfirmed}_A \wedge (R_A \neq R_B) \mid S = S'\right] = 0 \ .$$

Case 2: $S \neq S'$. For Alice to reach the $\mathsf{KeyConfirmed}$ state Eve must compute a valid authentication tag σ on S with respect to the authentication key key. This implies that

$$\begin{aligned}\Pr\left[\mathsf{KeyConfirmed}_A \wedge (R_A \neq R_B) \mid S \neq S'\right] &\leq \Pr\left[\mathsf{KeyConfirmed}_A \mid S \neq S'\right] \\ &\leq \Pr\left[\sigma = \mathsf{MAC}_{\mathsf{key}}(S) \mid S \neq S'\right]\end{aligned}$$

For analyzing this case we consider two subcases: one in which $Y' = Y$ (i.e., Eve does not modify Y) and the other in which $Y' \neq Y$ (i.e., Eve does modify Y). We show that

$$\begin{aligned}\Pr\left[\sigma = \mathsf{MAC}_{\mathsf{key}}(S) \mid S \neq S' \wedge Y' = Y\right] &\leq \varepsilon_{\mathsf{nmExt}} + \varepsilon_{\mathsf{MAC}} \\ \Pr\left[\sigma = \mathsf{MAC}_{\mathsf{key}}(S) \mid S \neq S' \wedge Y' \neq Y\right] &\leq \varepsilon_{\mathsf{nmExt}} + \varepsilon_{\mathsf{MAC}} \ .\end{aligned}$$

Case 2.1: $Y = Y'$. In this case Alice and Bob share the same authentication key $\mathsf{key} = \mathsf{nmExt}(W, Y) = \mathsf{nmExt}(W, Y') = \mathsf{key}'$, which we will show to be statistically-close to a uniform authentication key due to the fact that the view, V_E, of Eve cannot significantly reduce the min-entropy of W. Therefore, the security of the message authentication code guarantees that even after viewing the authentication tag $\sigma' = \mathsf{MAC}_{\mathsf{key}}(S')$ she has only a negligible probably of computing a valid authentication tag $\sigma = \mathsf{MAC}_{\mathsf{key}}(S)$ for any $S \neq S'$.

Formally, the facts that: (1) nmExt is in particular a strong $(k, \varepsilon_{\mathsf{nmExt}})$-extractor, (2) S' is independent of W and Y, and (3) Y is uniformly distributed, guarantee that

$$\mathsf{SD}\left((\mathsf{key}, Y, S'), (U_\ell, Y, S')\right) \leq \varepsilon_{\mathsf{nmExt}} \ .$$

Therefore, the probability that Eve (after viewing $\sigma' = \mathsf{MAC}_{\mathsf{key}}(S')$) computes a valid authentication tag σ on any $S \neq S'$ with respect to authentication key key differs by at most $\varepsilon_{\mathsf{nmExt}}$ from the probably $\varepsilon_{\mathsf{MAC}}$ that Eve computes a valid authentication tag σ for any $S \neq S'$ with respect to a uniformly and independently chosen authentication key:

$$\Pr\left[\sigma = \mathsf{MAC}_{\mathsf{key}}(S) \mid S \neq S' \wedge Y' = Y\right] \leq \varepsilon_{\mathsf{nmExt}} + \varepsilon_{\mathsf{MAC}} \ .$$

Case 2.2: $Y \neq Y'$. In this case Eve views an authentication tag $\sigma' = \mathsf{MAC}_{\mathsf{key}'}(S')$ with respect to the authentication key $\mathsf{key}' = \mathsf{nmExt}(W, Y')$, and has to compute an authentication tag $\sigma = \mathsf{MAC}_{\mathsf{key}}(S)$ for some $S \neq S'$ with respect to the authentication key $\mathsf{key} = \mathsf{nmExt}(W, Y)$. The property of the non-malleable extractor nmExt guarantees that even if Eve was in fact given the authentication key key' then from her point of view, the authentication key key is $\varepsilon_{\mathsf{nmExt}}$-close to an independently and uniformly chosen key. For such a key Eve can compute such an authentication tag σ with probability at most $\varepsilon_{\mathsf{MAC}}$ (and this in fact holds even if $S = S'$).

Formally, the facts that: (1) nmExt is a non-malleable $(k, \varepsilon_{\mathsf{nmExt}})$-extractor, (2) S' is independent of W, Y, and key', and (3) Y is uniformly distributed, guarantee that

$$\mathsf{SD}\left((\mathsf{key}, Y, \mathsf{key}', S'), (U_\ell, Y, \mathsf{key}', S')\right) \leq \varepsilon_{\mathsf{nmExt}}$$

which implies

$$\Pr\left[\sigma = \mathsf{MAC}_{\mathsf{key}}(S) \mid S \neq S' \wedge Y' \neq Y\right] \leq \varepsilon_{\mathsf{nmExt}} + \varepsilon_{\mathsf{MAC}} \ .$$

Combining cases 2.1 and 2.2 we obtain

$$\Pr\left[\sigma = \mathsf{MAC}_{\mathsf{key}}(S) \mid S \neq S'\right] \leq \varepsilon_{\mathsf{nmExt}} + \varepsilon_{\mathsf{MAC}} \ .$$

□

□

Proof. Theorem 3.9 Given the length n of the weak source, the constant δ, and the security parameter ε, we let $\varepsilon' = \varepsilon_{\mathsf{Ext}} = \varepsilon_{\mathsf{nmExt}} = \varepsilon_{\mathsf{MAC}} = \varepsilon/2$, and instantiate our protocol with the following explicit constructions:

1. Theorem 3.1 guarantees a non-malleable $((1/2+\delta)n, \varepsilon_{\mathsf{nmExt}})$-extractor $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^{d_1} \to \{0,1\}^\ell$, where $d_1 = \Theta(\log(n/\varepsilon))$ and $\ell = \Theta(\log(n/\varepsilon))$.
2. Theorem 2.1 guarantees a strong $((1/2+\delta)n - (d_1+\ell) - \log(1/\varepsilon'), \varepsilon_{\mathsf{Ext}})$-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{d_2} \to \{0,1\}^m$, where $d_2 = \Theta(n)$ and $m = (1/2+\delta)n - \Theta(\log n + \log(1/\varepsilon))$. In addition, Theorem 2.3 guarantees such a strong extractor where $d_2 = \Theta(\log^2 n + \log n \cdot \log(1/\varepsilon))$, and therefore we in fact have $d_2 = \Theta(\min\left\{\log^2 n + \log n \cdot \log(1/\varepsilon), n\right\})$
3. Theorem 3.4 guarantees an $\varepsilon_{\mathsf{MAC}}$-secure MAC $\left\{\mathsf{MAC}_{\mathsf{key}} : \{0,1\}^{d_2} \to \{0,1\}^\tau\right\}_{\mathsf{key} \in \{0,1\}^\ell}$ where $\tau = \Theta(\log n + \log(1/\varepsilon))$ and $\ell = \Theta(\log n + \log(1/\varepsilon))$.

By combining the above explicit constructions, the resulting privacy amplification protocol has security parameter $\max\{\varepsilon' + \varepsilon_{\mathsf{Ext}}, \varepsilon_{\mathsf{nmExt}} + \varepsilon_{\mathsf{MAC}}\} = \varepsilon$, entropy loss $(1/2+\delta)n - m = \Theta(\log n + \log(1/\varepsilon))$, and communication complexity

$$d_1 + d_2 + \tau = \Theta(\min\left\{\log^2 n + \log n \cdot \log(1/\varepsilon), n\right\}).$$

□

Proof. Theorem 3.10 The proof is identical to the proof of Theorem 3.9, where the only difference is that we instantiate the strong extractor Ext using the one provided by Theorem 2.2. Specifically, for any constants δ and β such that $1/2 + \delta > \beta > 0$, Theorem 2.2 guarantees a strong $((1/2+\delta)n - (d_1+\ell) - \log(1/\varepsilon'), \varepsilon_{\mathsf{Ext}})$-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^{d_2} \to \{0,1\}^m$, where $d_2 = \Theta(\log n + \log(1/\varepsilon))$ and

$$m = \left(1 - \frac{\beta}{1/2+\delta}\right)\left((1/2+\delta)n - (d_1+\ell) - \log(1/\varepsilon')\right).$$

In turn, the resulting privacy amplification protocol has entropy loss $(1/2+\delta)n - m = \beta n + \Theta(\log n + \log(1/\varepsilon))$, and communication complexity $d_1 + d_2 + \tau = \Theta(\log n + \log(1/\varepsilon))$. □

Chapter 4

Local Correlation Breakers and Applications to Multi-Source Extractors and Mergers

4.1 Local Correlation Breakers

As can be seen again and again throughout this thesis, a central theme in pseudorandomness concerns the design of efficient algorithms that transform one or more sources of randomness to a source with a desired property. From the applications end, the most typical desired property from a source of randomness is simply for it to be uniformly distributed over some ambient support. Informally speaking, extractors accomplish exactly this task as they produce truly random bits given a sample from a weak source of randomness (see Section 2).

Constructing extractors of various types is challenging and many constructions in the literature rely on auxiliary pseudorandom primitives such as mergers and condensers. These auxiliary objects either have weaker guarantee on the output or further assumptions on the input, so they are not fit to serve as off-the-shelf objects and are therefore more intrinsic to the subfield of pseudorandomness. Nevertheless, such objects are very useful as building blocks for the construction of other pseudorandom objects.

When constructing pseudorandom objects such as extractors, one is often faced with the problem of correlations between random variables. Namely, at some point in the construction, a sequence of random variables $X_1, \ldots, X_r$ is obtained, such that one or more of these random variables is "well-behaved", yet the correlations between the variables prevent one from proceeding with the construction and analysis.

Based on the paper [Coh15], in this chapter we present a primitive that we call a *local correlation breaker* (LCB for short) that, as its name suggests, allows one to "break" local correlations between random variables. We further present applications of LCBs to the construction of three-source extractors, mergers with weak-seeds, and a variant of non-malleable extractors. As LCBs allows one to face the typical scenario above, we believe LCBs will find further applications in the future.

The "well-behaved" property that we consider is being uniformly distributed. Ideally, for some locality parameter t, an LCB would be an algorithm that gets as input a sequence of r (arbitrarily correlated) random variables, and outputs a sequence of r random variables with the following property: If the i^{th} input random variable is uniform then the i^{th} output variable is uniform even given any other $t-1$ output variables.

Unfortunately, regardless of efficiency, no deterministic algorithm can accomplish the task above. A natural suggestion would be to consider seeded-LCBs, namely, LCBs that have a short auxiliary string of truly random bits (that is independent of the input variables). Although this suggestion is natural and appealing, given our applications in mind, we consider a different (and more challenging) variant where the auxiliary source of randomness is a weak-source of randomness. For the formal definition of LCBs we make use of standard definitions from the literature such as min-entropy, statistical distance, and (n,k)-weak-sources (see Section 2).

Definition 4.1 (Local correlation breakers)**.** *A* t-local correlation breaker (t-LCB) *for min-entropy* k*, with error* ε*, is a function*

$$\mathsf{LCB}\colon \left(\{0,1\}^{\ell}\right)^{r} \times \{0,1\}^{n} \to \left(\{0,1\}^{m}\right)^{r},$$

with the following property. Let $X = (X_1, \ldots, X_r)$ *be a sequence of random variables, each supported on* $\{0,1\}^{\ell}$*. Let* Y *be an independent* (n,k)*-weak-source. Denote the output* $\mathsf{LCB}(X,Y)$ *by* $Z_1, \ldots, Z_r$*, where each* Z_i *is supported on* $\{0,1\}^{m}$*. Let* $g \in [r]$ *be such that* X_g *is uniform. Let* $I \subseteq [r] \setminus \{g\}$ *be any set of size* $t-1$*. Then,*

$$\left(Z_g, \{Z_i\}_{i\in I}\right) \approx_{\varepsilon} \left(U_m, \{Z_i\}_{i\in I}\right).$$

The main technical contribution of this work is an explicit construction of LCBs. For simplicity, the theorem below is stated for constant error.

Theorem 4.1 (Explicit LCBs; informal statement)**.** *For all integers* n, r, t*, there exists an explicit* t*-local correlation breaker* $\mathsf{LCB}\colon \left(\{0,1\}^{\ell}\right)^{r} \times \{0,1\}^{n} \to \left(\{0,1\}^{m}\right)^{r}$ *with*

$$\begin{aligned} \ell &= O\left(t^2 \cdot \log(nr) \cdot \log r\right), \\ m &= \Omega\left(\ell/(t \cdot \log r)\right), \end{aligned}$$

for entropy

$$k = O\left(t \cdot \log(r) \cdot \log(r \log n)\right).$$

We emphasize the surprising fact that, in our construction, the dependence of the entropy k in n is double-logarithmic. One can show that a random function has the same dependence of k in n, and so it is plausible that this is the right dependence. For a complete and formal statement of Theorem 4.1, see [Coh15].

A pseudorandom object related to LCBs appears (implicitly) in the analysis of Li's multi-source extractor [Li13]. The difference between LCBs and Li's pseudorandom object is that the latter only guarantees that an output variable Z_g that corresponds to a uniform

input variable X_g is statistically-close to uniform given output variables that correspond to $t-1$ other *uniform* input variables. In other words, in Li's pseudorandom object, the set I in Definition 4.1 is assumed to contain indices only of uniform input variables. For the applications we consider, it is crucial that Z_g is close to uniform even given output variables $\{Z_i\}$ that correspond to possibly non-uniform input variables.

Our proof builds on the work of [Li13], together with some new ideas required so to guarantee the stronger property. In terms of parameters, Theorem 4.1 gives LCBs with somewhat better parameters compared to Li's pseudorandom object (even though the guarantee is stronger).

We believe that LCBs are natural pseudorandom primitives as they allow one to face the recurrent difficulty of having correlation between random variables using (very) weak-sources of randomness. Next we exemplify the usefulness of LCBs by presenting several applications.

4.2 Applications of LCBs

4.2.1 Three-source extractors with a double-logarithmic entropy source

Recall that a function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ is called a *two-source extractor* if for any two independent sources X, Y over $\{0,1\}^n$, with sufficient min-entropy, it holds that $f(X,Y)$ is statistically-close to uniform. By a standard probabilistic argument, one can prove the existence of a two-source extractor for any entropies k_1, k_2 such that $\min(k_1,k_2) > \log n + O(1)$.

Chor and Goldreich [CG88], who introduced the notion of a two-source extractor, gave an explicit construction for any entropies k_1, k_2 such that $k_1 + k_2 > (1+\delta)\cdot n$, where $\delta > 0$ is an arbitrarily small constant. In particular, one can take $k_1 = k_2 > (1/2+\delta)\cdot n$, for any constant $\delta > 0$. This construction is far from optimal (ignoring the computational aspect). Nevertheless, it took almost 20 years before any improvement was made. Raz [Raz05] gave an explicit construction of a two-source extractor for sources with entropies k_1, k_2, with $k_1 = O(\log n)$ and $k_2 > (1/2+\delta)\cdot n$, where $\delta > 0$ is an arbitrarily small constant. An incomparable result was obtained by Bourgain [Bou05] who constructed a two-source extractor for entropies $k_1 = k_2 > (1/2-\alpha)\cdot n$, where $\alpha > 0$ is some (small) universal constant.

Given the difficulty of explicitly constructing two-source extractors for low entropy, a significant research effort was directed towards the construction of t-source extractors for $t > 2$. The next natural goal is constructing three-source extractors. A simple probabilistic argument can be used to prove the existence of an extractor for three independent $(n, \log(n)/2 + O(1))$-weak-sources. Barak *et al.* [BKS+05] gave an explicit construction of a three-source extractor, where the entropy of each of the sources is δn, for any constant $\delta > 0$. This was improved by Raz [Raz05], who requires only one of the sources to have entropy δn, while the other two sources can have entropy $O(\log n)$. Here, again, $\delta > 0$ is

an arbitrarily small constant. Raz's extractor supports a constant error, and in a subsequent work, Rao [Rao09a] showed how to support exponentially small error, assuming the second and third sources have entropy $O(\log^4 n)$. Furthermore, Rao [Rao09a] constructed a three-source extractor, where the entropy of each of the sources is $n^{0.9}$. This was later improved by Li [Li11] to $n^{1/2+\delta}$, where $\delta > 0$ is an arbitrarily small constant.

In a recent breakthrough, Li [Li15] constructed a three-source extractor for poly-logarithmic entropy. This exciting result sets the next natural goal in multi-source extractors on improving the constructions of two-source extractors by Raz [Raz05] and Bourgain [Bou05]. Towards this goal, as an application of LCBs, we construct a three-source extractor where one of the sources is only assumed to have double-logarithmic entropy.

Theorem 4.2 (Explicit three-source extractors; informal statement)**.** *For any integer n and $\delta > 0$, there exists an explicit three-source extractor* $\mathsf{3Ext}\colon (\{0,1\}^n)^3 \to \{0,1\}^m$ *for entropies*

$$\begin{aligned} k_1 &= \delta n, \\ k_2 &= \mathrm{poly}(1/\delta) \cdot \log n, \\ k_3 &= \mathrm{poly}(1/\delta) \cdot \log\log n, \end{aligned}$$

with $m = \mathrm{poly}(1/\delta) \cdot \log n$ output bits.

The extractor in Theorem 4.2 is incomparable with the extractor of [Li15] and improves Raz's and Rao's three-source extractors [Raz05, Rao09a] which assume the third source has entropy $\Omega(\log n)$. As the third source fed to our three-source extractor is required to have a tantalizingly low entropy, we hope that further ideas can be used to eliminate the need for this third source altogether.

Improved three-source extractors for poly-logarithmic entropy As mentioned, Li [Li15] constructed a three-source extractor for poly-logarithmic entropy. More precisely, the entropy required by Li's construction is $O(\log^{12} n)$. For his construction, Li uses a pseudorandom object that is related to LCBs, introduced in [Li13], as well as the merger with weak-seeds of [BRSW12]. As our construction of LCBs has better parameters than Li's related pseudorandom object, and since our merger with weak-seeds improves that of [BRSW12] (see the following section), by using our results as building blocks in Li's three-source extractor, one can obtain a three-source extractor for a somewhat lower entropy of $\widetilde{O}(\log^c n)$, where $c < 12$ is some constant. By a short calculation, one can show that $c = 7$. Nevertheless, this calculation was not verified as carefully as the other proofs, and should be trusted accordingly.

4.2.2 Mergers with weak-seeds

Motivated by the construction of seeded extractors, Ta-Shma [TS96] introduced the notion of a merger. Informally speaking, a merger is a function that gets as input a sequence

of (arbitrarily correlated) random variables, at least one of which is uniform. The goal of a merger is to "merge" the random variables into a single random variable that is statistically-close to uniform. [1] It is not hard to show that randomness is a necessity for merging.

Constructing mergers with short seeds (namely, short strings that are uniform and independent of the random variables we wish to merge) has been studied in several works [TS96, LRVW03, Raz05, DS07, Zuc07, DR08, DW11, DKSS09]. The state of the art construction of Dvir and Wigderson [DW11] merges r random variables, supported on $\{0,1\}^\ell$, using a seed of length $O(\log(r\ell))$. An incomparable result was obtained by Dvir, Kopparty, Saraf and Sudan [DKSS09], who use a seed of length $O(\log(r)/\delta)$ to output a string that has entropy-rate $1-\delta$. As a building block for their celebrated two-source disperser, Barak, Rao, Shaltiel and Wigderson [BRSW12] constructed, what we call, mergers with weak-seeds. [2]

Definition 4.2 (Mergers with weak-seeds)**.** *A* merger with weak-seeds *for entropy k, with error ε, is a function*

$$\mathsf{Merg}\colon \left(\{0,1\}^\ell\right)^r \times \{0,1\}^n \to \{0,1\}^m,$$

with the following property. Let $X = (X_1, \ldots, X_r)$ be a sequence of random variables, supported on $\{0,1\}^\ell$, such that at least one of them is uniform. Let Y be an independent (n,k)-weak-source. Then, $\mathsf{Merg}(X,Y) \approx_\varepsilon U_m$.

In [BRSW12], a construction of a merger with weak-seeds is given, assuming $k = \ell > \Omega(r^2) + \mathrm{polylog}(n)$. [3] A probabilistic argument can be used to show that there exists a merger with weak-seeds for parameters

$$\begin{aligned}\ell &= \log n + O(1),\\ k &= \log r + \log\log n + O(1).\end{aligned}$$

In particular, the entropy k is only required to be double-logarithmic in n.

We note that constructing mergers with weak-seeds given an LCB is trivial. Indeed, one can apply an r-LCB to $X_1, \ldots, X_r$ and Y so to obtain random variables $Z_1, \ldots, Z_r$. The output of the merger is simply the XOR of all Z_i's. To see that this reduction works, note that if X_g is uniform then, by the guarantee of the LCB, Z_g is statistically-close to uniform even given all other Z_i's. Therefore, the XOR of all Z_i's is statistically-close to uniform. We use Theorem 4.1 with this simple idea (together with a bit more work so to improve the output length) and obtain the following result.

[1] Variants of mergers (which are also called mergers in the literature) assume that one of the random variables is not necessarily uniform, yet has high entropy-rate.

[2] In [BRSW12] this object is called an extractor for a general source and a somewhere-random source.

[3] In fact, the construction of [BRSW12] works even assuming one of the X_i's has entropy-rate $1-o(1)$.

Theorem 4.3 (Explicit mergers with weak-seeds; informal statement)**.** *For all integers n, r, there exists an explicit merger with weak-seeds* $\mathsf{Merg}\colon \left(\{0,1\}^{\ell}\right)^{r} \times \{0,1\}^{n} \to \{0,1\}^{m}$, *with*

$$\begin{aligned} \ell &= O\left(r^2 \cdot \log(r) \cdot \log(nr)\right), \\ k &= O\left(r \cdot \log(r) \cdot \log(r \cdot \log n)\right), \\ m &= \Omega(\ell/r). \end{aligned}$$

The merger of Barak *et al.* [BRSW12] and ours are incomparable. On one hand, the merger of [BRSW12] works even if one of the rows has min-entropy rate $1 - o(1)$. On the other hand, Theorem 4.3 has a quadratically improved dependence of k in r, and more importantly, an exponentially improved dependence of k in n, which matches the probabilistic construction. This feature allows us to obtain a three-source extractor with double-logarithmic entropy source. Moreover, we believe our construction and analysis are somewhat simpler and more intuitive than the construction of [BRSW12] which uses a completely different set of ideas.

4.2.3 Two-source non-malleable extractors

As discussed in Chapter 3, non-malleable extractors were introduced by Dodis and Wichs [DW09] for the purpose of constructing privacy amplification protocols in the setting of an active adversary. In [DW09] it is shown that a random function of the form $\mathsf{NMExt}\colon \{0,1\}^{n} \times \{0,1\}^{d} \to \{0,1\}^{m}$ is, with high probability, a non-malleable extractor for entropy $k = 2m + \log d$, with seed length $d = \log n + O(1)$. However, explicit constructions of non-malleable extractors fall significantly behind the parameters obtained by the probabilistic construction, and the state of the art construction has a logarithmic seed length but only supports entropy $k = (1/2 - \alpha) \cdot n$, where $\alpha > 0$ is some small universal constant [Li12b].

Given the challenge of explicitly constructing non-malleable extractors, we consider a relaxation of non-malleable extractors, which we call *two-source non-malleable extractors*. More generally, we consider two-source t-non-malleable extractors, which are defined as follows.

Definition 4.3 (Two-source t-non-malleable extractors)**.** *A function* $f\colon \left(\{0,1\}^{n}\right)^{2} \times \{0,1\}^{d} \to \{0,1\}^{m}$ *is called a* two-source t-non-malleable extractor *for entropies k_1, k_2, with error ε, if for any (n, k_1)-weak-source X and an independent (n, k_2)-weak-source Y, the following holds. For any t functions $A_1, \ldots, A_t\colon \{0,1\}^{d} \to \{0,1\}^{d}$, where each A_i has no fixed points (that is, for all $i \in [t]$ and $s \in \{0,1\}^{d}$, $A_i(s) \neq s$), it holds that*

$$\left(f(X,Y,S), S, \{f(X,Y,A_i(S))\}_{i=1}^{t}\right) \approx_{\varepsilon} (U_m, \cdot).$$

We recall that computational aspects aside, if one has access to two independent weak-sources, then one can output a string that is statistically-close to uniform by applying a two-source extractor, without any auxiliary seed. However, currently we do not know

how to efficiently construct two-source extractors for poly-logarithmic entropy or even for entropy-rate 0.49, and two-source non-malleable extractors can be viewed as a relaxation both of non-malleable extractors and of two-source extractors. In particular, one can view a two-source non-malleable extractor with seed length d as a collection of 2^d two-source extractors with the following property: for any two independent weak-sources X, Y with sufficiently high entropy, most extractors in the collection are two-source extractors for X, Y, and moreover, the output of a "good" two-source extractor in the collection applied to X, Y is independent of the output of any other $t-1$ extractors from the collection applied to the same samples.

Building on our construction of LCBs from Theorem 4.1 and on ideas from [Li15], we construct a two-source non-malleable extractor for poly-logarithmic entropy, with a seed of logarithmic length.

Theorem 4.4 (Explicit two-source t-non-malleable extractors; informal statement)**.** *For all integers n, t, there exists an explicit two-source t-non-malleable extractor*

$$\mathsf{2NMExt}\colon \left(\{0,1\}^n\right)^2 \times \{0,1\}^d \to \{0,1\}^m$$

for entropy $O(t^2 \cdot \log^2 n)$, with $d = O(\log n)$ and $m = \Theta(t \cdot \log n)$ output bits.

While the current best explicit construction of "standard" non-malleable extractors (namely, non-malleable extractors with one source) only supports entropy roughly $n/2$, Theorem 4.4 states that using two independent weak-sources, one can support poly-logarithmic entropy. Furthermore, it is worth noting that the seed length d in Theorem 4.4 is independent of t.

4.3 (L, R)-Histories

In this section we introduce the notion of an (L, R)-*history* and some technical lemmata concerning it that we use repeatedly throughout the chapter.

Definition 4.4 ((L, R)-histories)**.** *Let L, R be two independent random variables. A sequence of random variables $\mathcal{H} = (H_t, H_{t-1}, \ldots, H_1)$ is called an* (L, R)-history *if for any $i \in [t]$, H_i is either a deterministic function of $H_{i-1}, \ldots, H_1, L$ or otherwise H_i is a deterministic function of $H_{i-1}, \ldots, H_1, R$.*

Some remarks and notations:

- Throughout the chapter we assume that each H_i is supported on bit strings of some common length, which we can then denote by $|H_i|$.
- We note that if H_{i+1}, H_i are two consecutive random variables in some (L, R)-history, such that H_i is a deterministic function of $H_{i-1}, \ldots, H_1, L$ (resp. $H_{i-1}, \ldots, H_1, R$) and H_{i+1} is a deterministic function of $H_i, \ldots, H_1, L$ (resp. $H_{i-1}, \ldots, H_1, R$), then

one can replace H_{i+1}, H_i by a single random variable which is their joint distribution. This yields a new (L,R)-history. We allow ourselves to apply this operation freely during the proofs.

- Given two (L,R)-histories $\mathcal{H} = (H_t, \ldots, H_1)$ and $\mathcal{H}' = (H'_{t'}, \ldots, H'_1)$, one can consider the (L,R)-history which is the concatenation of $\mathcal{H}, \mathcal{H}'$, namely, $H'_{t'}, \ldots, H'_1, H_t, \ldots, H_1$. When we do not want to refer to the random variables in $\mathcal{H}$ but do want to refer to the random variables in $\mathcal{H}'$ (which is quite frequent), we write this concatenated (L,R)-history as $(H'_{t'}, \ldots, H'_1, \mathcal{H})$.

The following lemma states that conditioned on any fixing of an (L,R)-history, the random variables L, R remain independent. We omit the proof, which is done by a straightforward induction.

Lemma 4.5. *Let L, R be two independent random variables, and let $\mathcal{H}$ be an (L,R)-history. Then, for any $h \in \sup(\mathcal{H})$, the random variables $(L \mid \mathcal{H} = h)$ and $(R \mid \mathcal{H} = h)$ are independent.*

In the rest of this section we state and prove two technical lemmata for (L,R)-histories. Before giving the formal statement of the first lemma, we present the lemma in an informal manner so to give some intuition about what the lemma aims to abstract. A common scenario in our proofs is the following. Let L, R be two independent random variables. We think of L, R as two independent sources of randomness from which we extract randomness again and again and preform various computations on the sequence. We denote by $\mathcal{H} = (H_t, \ldots, H_1)$ the (L,R)-history that captures the random variables obtained from L, R so far. Typically we will know that some random variable P is statistically-close to uniform even given $\mathcal{H}$, namely, $(P, \mathcal{H}) \approx (U, \mathcal{H})$. Furthermore, P is either a deterministic function of $L, \mathcal{H}$ or otherwise P is a deterministic function of $R, \mathcal{H}$. Assume, without loss of generality, that P is a deterministic function of $L, \mathcal{H}$. Let Ext be a strong seeded extractor. The following lemma states that if M is a deterministic function of $R, \mathcal{H}$ and $\widetilde{H}_\infty(M \mid \mathcal{H})$ is sufficiently high, then $(\mathsf{Ext}(M,P), P, \mathcal{H}) \approx (U, P, \mathcal{H})$.

The proof of this technical lemma is fairly simple. Nevertheless, we apply the lemma frequently and believe that our proofs are cleaner and conceptually simpler by identifying the operation that is described and analyzed by the lemma as an atomic operation.

Lemma 4.6. *Let L, R be two independent random variables, and let $\mathcal{H}$ be an (L,R)-history. Let P be a random variable over $\{0,1\}^p$ which is a deterministic function of $L, \mathcal{H}$.* [4] *Assume that*

$$(P, \mathcal{H}) \approx_\delta (U_p, \mathcal{H}) . \tag{4.1}$$

Let M be a random variable over $\{0,1\}^m$ which is a deterministic function of $R, \mathcal{H}$, such that

$$\widetilde{H}_\infty (M \mid \mathcal{H}) \geq k + \log(1/\varepsilon). \tag{4.2}$$

[4] Note that any (L,R)-history is also an (R,L)-history, and so an analog statement of the lemma in which P is a deterministic function of $R, \mathcal{H}$ readily follows.

Let $\mathsf{Ext}\colon \{0,1\}^m \times \{0,1\}^p \to \{0,1\}^f$ *be a strong seeded extractor for entropy* k *with error* ε*. Define* $F = \mathsf{Ext}(M, P)$*. Then,* $P, \mathcal{H}$ *is an* (L, R)*-history, and*

$$(F, P, \mathcal{H}) \approx_{\delta+2\varepsilon} (U_f, P, \mathcal{H}).$$

Proof. Let $h \in \sup(\mathcal{H})$. For the sake of readability, for a random variable T, we denote the random variable $(T \mid H = h)$ by T_h. Let $\delta_h = \mathsf{SD}(P_h, U_p)$. By Lemma 2.9 and Equation (4.1),

$$\mathop{\mathbf{E}}_{h\sim\mathcal{H}}[\delta_h] = \mathop{\mathbf{E}}_{h\sim\mathcal{H}}[\mathsf{SD}(P_h, U_p)] = \mathsf{SD}((P, \mathcal{H}), (U_p, \mathcal{H})) \leq \delta.$$

Note that the random variables M_h, P_h are independent. Indeed, Lemma 4.5 implies that the random variables L_h, R_h are independent, and M is a deterministic function of $R, \mathcal{H}$ whereas P is a deterministic function of $L, \mathcal{H}$. Therefore, by Lemma 2.9,

$$\mathsf{SD}((F_h, P_h), (U_f, P_h)) = \mathop{\mathbf{E}}_{s\sim P_h}[\mathsf{SD}(F_h \mid P_h = s, U_f)] = \mathop{\mathbf{E}}_{s\sim P_h}[\mathsf{SD}(\mathsf{Ext}(M_h, s), U_f)]. \quad (4.3)$$

Since $\mathsf{SD}(P_h, U_p) = \delta_h$ and since the range of the function $g(s) = \mathsf{SD}(\mathsf{Ext}(M_h, s), U_f)$ is contained in the interval $[0, 1]$, Lemma 2.13 implies that

$$\mathop{\mathbf{E}}_{s\sim P_h}[\mathsf{SD}(\mathsf{Ext}(M_h, s), U_f)] \leq \mathop{\mathbf{E}}_{s\sim U_p}[\mathsf{SD}(\mathsf{Ext}(M_h, s), U_f)] + \delta_h. \quad (4.4)$$

Equation (4.3) and Equation (4.4) imply that

$$\mathsf{SD}((F_h, P_h), (U_f, P_h)) \leq \mathop{\mathbf{E}}_{s\sim U_p}[\mathsf{SD}(\mathsf{Ext}(M_h, s), U_f)] + \delta_h.$$

As we assume that $\widetilde{H}_\infty(M \mid \mathcal{H}) \geq k + \log(1/\varepsilon)$, Lemma 2.4 implies that

$$\Pr_{h\sim\mathcal{H}}[H_\infty(M_h) \geq k] \geq 1 - \varepsilon.$$

We say that h is *good* if $H_\infty(M_h) \geq k$. Since Ext is a strong seeded extractor for entropy k with error ε, for any good h it holds that

$$\mathsf{SD}((F_h, P_h), (U_f, P_h)) \leq \varepsilon + \delta_h.$$

By Lemma 2.9,

$$\mathsf{SD}((F, P, \mathcal{H}), (U_f, P, \mathcal{H})) = \mathop{\mathbf{E}}_{h\sim\mathcal{H}}[\mathsf{SD}((F_h, P_h), (U_f, P_h))].$$

The right hand side is bounded above by

$$\mathop{\mathbf{E}}_{h\sim\mathcal{H}}\left[\mathsf{SD}((F_h, P_h), (U_f, P_h)) \;\middle|\; h \text{ is good}\right] \cdot \Pr_{h\sim\mathcal{H}}[h \text{ is good}] + \Pr_{h\sim\mathcal{H}}[h \text{ is not good}] \leq \delta + 2\varepsilon,$$

as stated. The fact that $P, \mathcal{H}$ is an (L, R)-history readily follows since $\mathcal{H}$ is an (L, R)-history and P is a deterministic function of $L, \mathcal{H}$. □

The following lemma is also used frequently in our proofs.

Lemma 4.7. *Let L, R be two independent random variables, and let $\mathcal{H}$ be an (L,R)-history. Let P be a random variable that is a deterministic function of $R, \mathcal{H}$. Let J be a random variable that is a deterministic function of $L, \mathcal{H}$. Then,*

$$\mathsf{SD}\left((P,J,\mathcal{H}),(U,J,\mathcal{H})\right) = \mathsf{SD}\left((P,\mathcal{H}),(U,\mathcal{H})\right).$$

Moreover, $J, \mathcal{H}$ is an (L,R)-history.

Proof. The fact that $J, \mathcal{H}$ is an (L,R)-history readily follows since $\mathcal{H}$ is an (L,R)-history and J is a deterministic function of $L, \mathcal{H}$. Now, by Lemma 2.9,

$$\mathsf{SD}\left((P,J,\mathcal{H}),(U,J,\mathcal{H})\right) = \mathop{\mathbf{E}}_{h\sim\mathcal{H}}\left[\mathsf{SD}\left(((P,J) \mid \mathcal{H}=h),(U,J \mid \mathcal{H}=h)\right)\right].$$

For any $h \in \sup(\mathcal{H})$, the random variable $(P \mid \mathcal{H}=h)$ is a deterministic function of R whereas $(J \mid \mathcal{H}=h)$ is a deterministic function of L. Since $(L \mid \mathcal{H}=h)$, $(R \mid \mathcal{H}=h)$ are independent, as guaranteed by Lemma 4.5, we have that $(P \mid \mathcal{H}=h)$ and $(J \mid \mathcal{H}=h)$ are independent. Thus, Lemma 2.10 implies that

$$\mathsf{SD}\left((P,J,\mathcal{H}),(U,J,\mathcal{H})\right) = \mathop{\mathbf{E}}_{h\sim\mathcal{H}}\left[\mathsf{SD}\left(P \mid (\mathcal{H}=h),U\right)\right],$$

which concludes the proof, as by Lemma 2.9, the right hand side of the above equation equals to $\mathsf{SD}\left((P,\mathcal{H}),(U,\mathcal{H})\right)$. □

4.4 Two-Steps Look-Ahead Extractors

In this section we present a restricted version of look-ahead extractors. Building on the idea of alternating extraction [DP07], Dodis and Wichs [DW09] introduced the notion of look-ahead extractors. Look-ahead extractors were further used by Li [Li13, Li15] for his multi-source extractors. In these cases, the look-ahead extractors were applied for some non-constant number of "steps" or "rounds". We construct our LCBs using look-ahead extractors with only two steps. This in turn allows us to present relatively simple constructions of LCBs, which are also easier to analyze. Since we need only this very restricted version, and since we use it in the analysis of our constructions in a white-box manner, we give in this section the construction for two-steps look-ahead extractors.

Let n, a, h be integers and let $\varepsilon > 0$ be such that $a = \Omega(\log(h/\varepsilon))$ and $h = \Omega(\log(n/\varepsilon))$. Set $s = \Theta(\log(n/\varepsilon))$, where some appropriately chosen large enough universal constant is hidden under the Θ notation. Let $\mathsf{Ext}_1 \colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^a$ and $\mathsf{Ext}_2 \colon \{0,1\}^h \times \{0,1\}^a \to \{0,1\}^s$ be strong seeded extractors from Theorem 2.2, both with error ε. Note that the choice of s and the assumption on a guarantee that the seed lengths of Ext_1 and Ext_2 are sufficient. Moreover, by Theorem 2.2, Ext_1 is an extractor for entropy $2a$ and Ext_2 is an extractor for entropy $2s$. Define the function

$$\mathsf{LookAheadExt} \colon \{0,1\}^h \times \{0,1\}^n \to \{0,1\}^a \times \{0,1\}^a$$

as follows. Given $W \in \{0,1\}^h$ and $Y \in \{0,1\}^n$, let

$$\begin{aligned} A &= \mathsf{Ext}_1(Y, W|_s), \\ Z &= \mathsf{Ext}_2(W, A), \\ B &= \mathsf{Ext}_1(Y, Z). \end{aligned}$$

Define

$$\mathsf{LookAheadExt}(W, Y) = (A, B).$$

With notations as above, we have the following lemma.

Lemma 4.8. *Let r be an integer. Let X, Y be two independent random variables, and let $\mathcal{H}$ be an (X, Y)-history such that*

$$\widetilde{H}_\infty(Y \mid \mathcal{H}) \geq (r+2)a + \log(1/\varepsilon). \tag{4.5}$$

Let W be a random variable of the form of an $r \times h$ matrix, which is a deterministic function of $X, \mathcal{H}$, where

$$h \geq (r+2)s + \log(1/\varepsilon). \tag{4.6}$$

Assume further that there exists $g \in [r]$ such that

$$(W_g, \mathcal{H}) \approx_\delta (U_h, \mathcal{H}). \tag{4.7}$$

For each $i \in [r]$, let (A_i, B_i) be the output $\mathsf{LookAheadExt}(W_i, Y)$. Then the following holds:

- *$\mathcal{H}' = (W, Z_g, \{A_i\}_{i=1}^r, W|_s, \mathcal{H})$ is an (X, Y)-history.*
- *$(B_g, \mathcal{H}') \approx_{2\delta+6\varepsilon} (U_a, \mathcal{H}')$.*
- *$\widetilde{H}_\infty(Y \mid \mathcal{H}') \geq \widetilde{H}_\infty(Y \mid \mathcal{H}) - ra$.*
- *For any random variable N which is a deterministic function of $X, \mathcal{H}$, it holds that $\widetilde{H}_\infty(N \mid \mathcal{H}') \geq \widetilde{H}_\infty(N \mid \mathcal{H}) - rh$.*

As mentioned, Lemma 4.8 is not new and general versions of it appear in the literature. Nevertheless, as we consider a restricted setting and since the lemma as stated uses the notion of (L, R)-histories (which is new), a direct proof for the lemma above does not appear in the literature (though existing proofs can be adopted in a straightforward manner). Thus, for completeness, we give a proof for Lemma 4.8 in the following section.

4.5 Proof of Lemma 4.8

For the proof of Lemma 4.8 we use the following simple lemma.

Lemma 4.9. *Let X, Y be two random variable over a common domain D. Let $E \subseteq D$ be an event. Then,*

$$\mathsf{SD}\left(X \mid E, Y \mid E\right) \leq \frac{1}{\Pr[E]} \cdot \mathsf{SD}\left(X, Y\right).$$

We start by proving the following lemma.

Lemma 4.10. *Let L, R be two independent random variables, and let $\mathcal{H}$ be an (L, R)-history. Let M be a random variable over $\{0,1\}^m$ which is a deterministic function of $\mathcal{H}, L$ such that*

$$(M, \mathcal{H}) \approx_{\delta_M} (U_m, \mathcal{H}). \tag{4.8}$$

Let J be a random variable over $\{0,1\}^j$ which is a deterministic function of $\mathcal{H}, L$. Let P be a random variable over $\{0,1\}^p$ which is a deterministic function of $\mathcal{H}, R, J$, such that

$$(P, J, \mathcal{H}) \approx_{\delta_P} (U_p, J, \mathcal{H}). \tag{4.9}$$

Let $\mathsf{Ext}\colon \{0,1\}^m \times \{0,1\}^p \to \{0,1\}^f$ be a strong seeded extractor for entropy $m - j - \log(1/\varepsilon)$, with error ε. Define $F = \mathsf{Ext}(M, P)$. Then, $P, J, \mathcal{H}$ is an (L, R)-history, and

$$(F, P, J, \mathcal{H}) \approx_{\delta_P + \delta_M + 2\varepsilon} (U_f, P, J, \mathcal{H}).$$

Proof of Lemma 4.10. For $h \in \sup(\mathcal{H})$ denote by J_h the random variable $(J \mid \mathcal{H} = h)$. For $j \in \sup(J_h)$ let $E_{h,j}$ be the event $\mathcal{H} = h$, $J_h = j$. For the sake of readability, we let $M_{h,j}$ denote the random variable $(M \mid E_{h,j})$. Similarly, we define $P_{h,j}$, $L_{h,j}$, $R_{h,j}$ to be $(P \mid E_{h,j})$, $(L \mid E_{h,j})$ and $(R \mid E_{h,j})$, respectively.

Since J is a deterministic function of $\mathcal{H}, L$, the random variable J_h is a deterministic function of L. Thus, even after further conditioning on the event $J_h = j$, the random variables L, R remain independent. That is, the random variables $L_{h,j}$ and $R_{h,j}$ are independent. Furthermore, since P is a deterministic function of $\mathcal{H}, R, J$ and M is a deterministic function of $\mathcal{H}, L$, we have that $M_{h,j}$ and $P_{h,j}$ are independent.

Let $\delta_{P;h,j} = \mathsf{SD}\left(P_{h,j}, U_p\right)$. By Equation (4.9) and Lemma 2.9 we have that

$$\mathop{\mathbf{E}}_{h \sim \mathcal{H}} \mathop{\mathbf{E}}_{j \sim J_h} \left[\delta_{P;h,j}\right] \leq \delta_P.$$

Since $M_{h,j}$ and $P_{h,j}$ are independent, Lemma 2.9 implies that

$$\mathsf{SD}\left(\left(\mathsf{Ext}\left(M_{h,j}, P_{h,j}\right), P_{h,j}\right)\left(U_f, P_{h,j}\right)\right) = \mathop{\mathbf{E}}_{s \sim P_{h,j}} \left[\mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), U_f\right)\right].$$

As the function $g(s) = \mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), U_f\right)$ attains values in the interval $[0,1]$, Lemma 2.13 yields

$$\mathop{\mathbf{E}}_{s \sim P_{h,j}} \left[\mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), U_f\right)\right] \leq \mathop{\mathbf{E}}_{s \sim U_p} \left[\mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), U_f\right)\right] + \delta_{P;h,j}.$$

Thus,

$$\begin{aligned}
\mathsf{SD}\left(\left(F, P, J, \mathcal{H}\right), \left(U_f, P, J, \mathcal{H}\right)\right) &= \mathop{\mathbf{E}}_{h \sim \mathcal{H}} \mathop{\mathbf{E}}_{j \sim J_h} \mathop{\mathbf{E}}_{s \sim P_{h,j}} \left[\mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), U_f\right)\right] \\
&\leq \mathop{\mathbf{E}}_{h \sim \mathcal{H}} \mathop{\mathbf{E}}_{j \sim J_h} \left(\mathop{\mathbf{E}}_{s \sim U_p} \left[\mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), U_f\right)\right] + \delta_{P;h,j} \right) \\
&\leq \mathop{\mathbf{E}}_{h \sim \mathcal{H}} \mathop{\mathbf{E}}_{j \sim J_h} \mathop{\mathbf{E}}_{s \sim U_p} \left[\mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), U_f\right)\right] + \delta_P.
\end{aligned}$$

Let $\delta_{M;h} = \mathsf{SD}\left((M \mid \mathcal{H} = h), M'\right)$, where M' is a random variable that is uniformly distributed over $\{0,1\}^m$ and is independent of $\mathcal{H}$. By Equation (4.8) and Lemma 2.9,

$$\mathop{\mathbf{E}}_{h\sim\mathcal{H}}\left[\delta_{M;h}\right] \leq \delta_M.$$

Lemma 4.9 implies that for every $j \in \sup(J_h)$, the distribution of the random variable $M_{h,j}$ is $\delta_{M;h,j}$-close to the distribution $(M' \mid J_h = j)$, where

$$\delta_{M;h,j} = \frac{\delta_{M;h}}{\Pr[J_h = j]}.$$

Let $M'_{h,j}$ be a random variable with distribution $(M' \mid J_h = j)$. By the triangle inequality and Lemma 2.8, it holds that

$$\begin{aligned}
\mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), U_f\right) &\leq \mathsf{SD}\left(\mathsf{Ext}\left(M_{h,j}, s\right), \mathsf{Ext}\left(M'_{h,j}, s\right)\right) + \mathsf{SD}\left(\mathsf{Ext}\left(M'_{h,j}, s\right), U_f\right) \\
&\leq \mathsf{SD}\left(M_{h,j}, M'_{h,j}\right) + \mathsf{SD}\left(\mathsf{Ext}\left(M'_{h,j}, s\right), U_f\right) \\
&= \delta_{M;h,j} + \mathsf{SD}\left(\mathsf{Ext}\left(M'_{h,j}, s\right), U_f\right),
\end{aligned}$$

and so,

$$\begin{aligned}
\mathsf{SD}\left((F, P, J, \mathcal{H}), (U_f, P, J, \mathcal{H})\right) &\leq \mathop{\mathbf{E}}_{h\sim\mathcal{H}} \mathop{\mathbf{E}}_{j\sim J_h} \mathop{\mathbf{E}}_{s\sim U_p}\left[\mathsf{SD}\left(\mathsf{Ext}\left(M'_{h,j}, s\right), U_f\right) + \delta_{M;h,j}\right] + \delta_P \\
&\leq \mathop{\mathbf{E}}_{h\sim\mathcal{H}} \mathop{\mathbf{E}}_{j\sim J_h} \mathop{\mathbf{E}}_{s\sim U_p}\left[\mathsf{SD}\left(\mathsf{Ext}\left(M'_{h,j}, s\right), U_f\right)\right] + \delta_M + \delta_P.
\end{aligned} \tag{4.10}$$

By Lemma 2.3, for any $h \in \sup(\mathcal{H})$, $\widetilde{H}_\infty(M'_{h,j}) = \widetilde{H}_\infty(M' \mid J_h = j) \geq m - j$. Thus, by Lemma 2.4, for any $h \in \sup(\mathcal{H})$,

$$\Pr_{j\sim J_h}\left[H_\infty(M'_{h,j}) \geq m - j - \log(1/\varepsilon)\right] \geq 1 - \varepsilon.$$

We say that a pair h, j, where $h \in \sup(\mathcal{H})$ and $j \in \sup(J_h)$, is *good* if $H_\infty(M'_{h,j}) \geq m - j - \log(1/\varepsilon)$. By the above, for every $h \in \sup(\mathcal{H})$, with probability $1 - \varepsilon$ over $j \sim J_h$ it holds that the pair h, j is good. Since Ext is a strong seeded extractor for entropy $m - j - \log(1/\varepsilon)$ and error ε, we have that the contribution of any good pair h, j to the expectation in Equation (4.10) is at most ε. Since $1 - \varepsilon$ fraction of the pairs are good, and since any pair contributes at most 1 to the expectation, we get that

$$\mathsf{SD}\left((F, P, J, \mathcal{H}), (U_f, P, J, \mathcal{H})\right) \leq 2\varepsilon + \delta_M + \delta_P,$$

as stated. To conclude the proof of the lemma, note that $P, J, \mathcal{H}$ is an (L, R)-history since $\mathcal{H}$ is an (L, R)-history, J is a deterministic function of $\mathcal{H}, L$, and P is a deterministic function of R and $J, \mathcal{H}$ (note that J precede P in the history and so P is allowed to depend on J). □

Proof of Lemma 4.8. We apply Lemma 4.6 to the (X,Y)-history $\mathcal{H}$ with $P = (W_g)|_s$, $M = Y$ and the extractor Ext_1. The hypothesis of Lemma 4.6 is met since the random variable $P = (W_g)|_s$ is a deterministic function of $X, \mathcal{H}$. Moreover, by the Equation (4.5), $\widetilde{H}_\infty(Y \mid \mathcal{H}) \geq 2a + \log(1/\varepsilon)$. Since Ext_1 is a strong seeded extractor for entropy $2a$, Equation (4.2) of Lemma 4.6 holds. Since $A_g = \mathsf{Ext}_1(Y, (W_g)|_s)$, Lemma 4.6 together with Equation (4.7) imply that

$$(A_g, (W_g)|_s, \mathcal{H}) \approx_{\delta+2\varepsilon} (U_a, \cdot).$$

Moreover, $(W_g)|_s, \mathcal{H}$ is an (X,Y)-history.

Note that A_g is a deterministic function of Y and $(W_g)|_s$, whereas the joint distribution of $\{(W_i)|_s\}_{i\in[r]\setminus\{g\}}$ is a deterministic function of X and $\mathcal{H}$. Thus, Lemma 4.7 applied with $P = A_g$, $J = \{(W_i)|_s\}_{i\in[r]\setminus\{g\}}$ and the (X,Y)-history $(W_g)|_s, \mathcal{H}$, implies that

$$(A_g, W|_s, \mathcal{H}) \approx_{\delta+2\varepsilon} (U_a, \cdot). \tag{4.11}$$

Furthermore, $W|_s, \mathcal{H}$ is an (X,Y)-history.

We apply Lemma 4.10 to the (X,Y)-history $\mathcal{H}$, with $M = W_g$, $P = A_g$, $J = W|_s$ and the extractor Ext_2. The hypothesis of Lemma 4.10 is met since $W|_s, W_g$ are deterministic functions of $\mathcal{H}, X$, and A_g is a deterministic function of $Y, W|_s$. Moreover, Equation (4.6) implies that

$$|W_g| - |(W|_s)| - \log(1/\varepsilon) = h - rs - \log(1/\varepsilon) \geq 2s.$$

Since Ext_2 is a strong seeded extractor for entropy $2s$ with error ε, Lemma 4.10 together with Equation (4.7) and Equation (4.11) imply that

$$(Z_g, A_g, W|_s, \mathcal{H}) \approx_{2\delta+4\varepsilon} (U_s, \cdot). \tag{4.12}$$

Furthermore, $A_g, W|_s, \mathcal{H}$ is an (X,Y)-history.

We apply Lemma 4.7 with $P = Z_g$, the (X,Y)-history $A_g, W|_s, \mathcal{H}$ and $J = \{A_i\}_{i\in[r]\setminus\{g\}}$, and conclude that

$$(Z_g, \{A_i\}_{i=1}^r, W|_s, \mathcal{H}) \approx_{2\delta+4\varepsilon} (U_s, \cdot). \tag{4.13}$$

This application of Lemma 4.7 is valid since $Z_g = \mathsf{Ext}_2(W_g, A_g)$ is a deterministic function of A_g, which is contained in the (X,Y)-history to which we apply the lemma, and W_g which, by assumption, is a deterministic function of $X, \mathcal{H}$ (and $\mathcal{H}$ is also contained in (X,Y)-history above). On the other hand, the joint distribution of $\{A_i\}_{i\in[r]\setminus\{g\}}$ is a deterministic function of $Y, W|_s$, and $W|_s$ is contained in the (X,Y)-history to which we apply the lemma. Lemma 4.7 further implies that $\{A_i\}_{i=1}^r, W|_s, \mathcal{H}$ is an (X,Y)-history.

Next, we apply Lemma 4.6 to the (X,Y)-history $\{A_i\}_{i=1}^r, W|_s, \mathcal{H}$ with $P = Z_g$, $M = Y$ and the extractor Ext_1. The hypothesis of Lemma 4.6 is met since the random variable $Z_g = \mathsf{Ext}_2(W_g, A_g)$ is a deterministic function of $X, \mathcal{H}, A_g$, and $\mathcal{H}, A_g$ are contained in the (X,Y)-history to which we apply the lemma. In terms of entropy,

$$\begin{aligned}
\widetilde{H}_\infty(Y \mid \{A_i\}_{i=1}^r, W|_s, \mathcal{H}) &\geq \widetilde{H}_\infty(Y \mid (W|_s), \mathcal{H}) - ra \\
&= \widetilde{H}_\infty(Y \mid \mathcal{H}) - ra \\
&\geq 2a + \log(1/\varepsilon),
\end{aligned} \tag{4.14}$$

where the first inequality follows by Lemma 2.3 and the fact that $\{A_i\}_{i=1}^r$ consists of ra bits. The second equality follows by Lemma 2.6, which is applicable in this case as conditioned on any fixing of $\mathcal{H}$, the random variables $W|_s, Y$ are independent. The last inequality follows by Equation (4.5). Since Ext_1 is a strong seeded extractor for entropy $2a$, Equation (4.2) of Lemma 4.6 holds. As $B_g = \mathsf{Ext}_1(Y, Z_g)$, Lemma 4.6 together with Equation (4.13) imply that

$$(B_g, Z_g, \{A_i\}_{i=1}^r, W|_s, \mathcal{H}) \approx_{2\delta+6\varepsilon} (U_a, \cdot).$$

Moreover, $Z_g, \{A_i\}_{i=1}^r, W|_s, \mathcal{H}$ is an (X,Y)-history.

Note that B_g is a deterministic function of Y, Z_g whereas W is a deterministic function of $X, \mathcal{H}$. Thus, we can apply Lemma 4.7 with $P = B_g$ and $J = W$ to the (X,Y)-history $Z_g, \{A_i\}_{i=1}^r, W|_s, \mathcal{H}$ and conclude that

$$(B_g, \mathcal{H}') \approx_{2\delta+6\varepsilon} (U_a, \cdot),$$

where $\mathcal{H}' = (W, Z_g, \{A_i\}_{i=1}^r, W|_s, \mathcal{H})$ is an (X,Y)-history. This proves the first and second items of the lemma.

As for the third item, since W and Y are independent conditioned on any fixing of $Z_g, \{A_i\}_{i=1}^r, W|_s, \mathcal{H}$, Lemma 2.6 implies that

$$\widetilde{H}_\infty(Y \mid \mathcal{H}') = \widetilde{H}_\infty(Y \mid Z_g, \{A_i\}_{i=1}^r, W|_s, \mathcal{H}).$$

Similarly, conditioned on any fixing of $\{A_i\}_{i=1}^r, W|_s, \mathcal{H}$, the random variables Y and Z_g are independent, and so

$$\widetilde{H}_\infty(Y \mid \mathcal{H}') = \widetilde{H}_\infty(Y \mid \{A_i\}_{i=1}^r, W|_s, \mathcal{H}).$$

By Equation (4.14), the right hand side of the equation above is bounded below by $\widetilde{H}_\infty(Y \mid \mathcal{H}) - ra$, which proves the third item of the lemma.

As for the fourth item, note that Z_g is a deterministic function of W, A_g, and $W|_s$ is a deterministic function of W, and so

$$\widetilde{H}_\infty(N \mid \mathcal{H}') = \widetilde{H}_\infty(N \mid W, \{A_i\}_{i=1}^r, \mathcal{H}).$$

Note that conditioned on any fixing of $W, \mathcal{H}$, the random variables $\{A_i\}_{i=1}^r$ are deterministic functions of Y. Since N is a deterministic function of $X, \mathcal{H}$, Lemma 2.6 further implies that

$$\widetilde{H}_\infty(N \mid \mathcal{H}') = \widetilde{H}_\infty(N \mid W, \mathcal{H}) \geq \widetilde{H}_\infty(N \mid \mathcal{H}) - rh,$$

where the last inequality follows by Lemma 2.3. □

4.6 A Warm Up – Merging Three Rows

In order to convey the ideas underling our LCBs, we present in this section a construction of a merger with weak-seeds for a somewhere-random source with only three rows. This

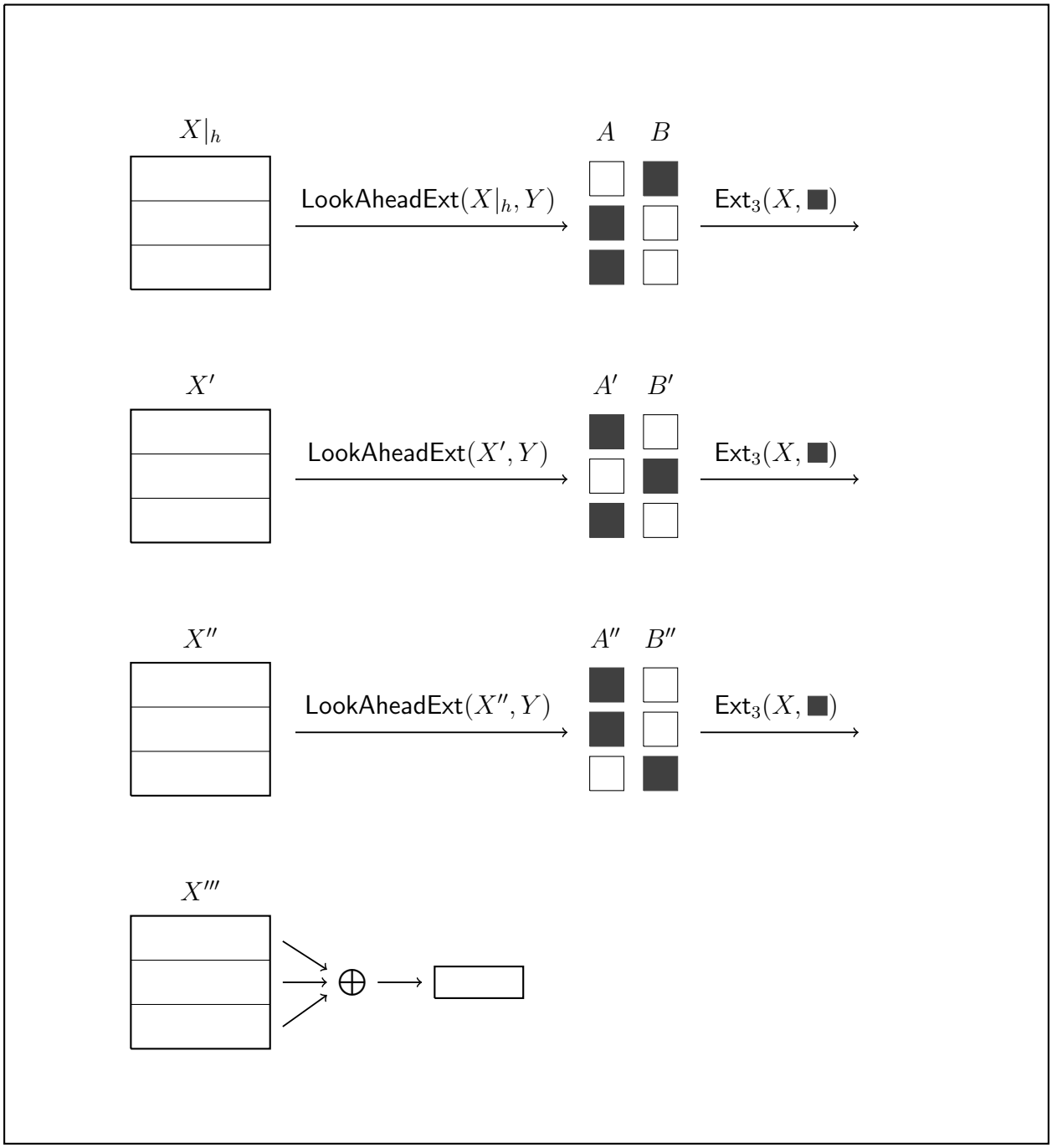

Figure 4.1: A schematic diagram of the three rows merger Merg_3.

toy example allows us to present some of the ideas used in the actual constructions of our mergers with weak-seeds (Theorem 4.3) and LCBs (Theorem 4.1). This section is meant only for building up intuition, and presenting the underling ideas behind our constructions without getting into all the details.

During this section we ignore the error analysis as this does not affect the parameters and slightly complicates the presentation. In particular, when applying Lemma 4.6 and Lemma 4.8 we ignore the expression $\log(1/\varepsilon)$ in Equation (4.2) and in Equation (4.5).

In this section we prove the following theorem which, roughly speaking, states that one can efficiently and deterministically merge the rows of a $3 \times \ell$ somewhere-random source, using an independent (n,k)-weak-source, even when $\ell = \Theta(\log n)$ and $k = \Omega(\log\log n)$.

Theorem 4.5 (Merging three rows)**.** *For any integer n, there exists a* poly(n)*-time computable function*

$$\mathsf{Merg}_3 \colon \left(\{0,1\}^{\ell}\right)^3 \times \{0,1\}^n \to \{0,1\}^m,$$

where $\ell = \Theta(\log n)$ and $m = \Omega(\ell)$, with the following property. Let X be a $3 \times \ell$ somewhere-random source. Let Y be an independent (n,k)-weak-source with $k = \Omega(\log\log n)$. Then, $\mathsf{Merg}_3(X,Y) \approx U_m$.

Proof. During the proof of this toy example we assume that the second row, X_2, is good. Of course, the algorithm Merg_3 will not rely on this assumption (or otherwise the algorithm can simply output X_2). We use the assumption that X_2 is uniform only for the analysis. We exemplify this with the good row being the second row just to avoid introducing more indices. Since the second row is not the first or the last row, this will enable us to demonstrate all the ideas needed to prove the theorem for any number of rows.

We turn to present the construction of Merg_3. For the reader's convenience, the construction is depict in Figure 4.1. As mentioned, the problem of merging the 3 rows X_1, X_2, X_3, with X_2 being the good row, is reduced to the problem of obtaining random variables X_1''', X_2''', X_3''', where each X_i''' is a function of X_i and Y, with the following property: $(X_2''', X_1''', X_3''') \approx (U, X_1''', X_3''')$, namely, constructing 3-LCBs for 3 rows. Once this independence is obtained, one can simply output

$$\mathsf{Merg}_3(X,Y) = X_1''' \oplus X_2''' \oplus X_3'''.$$

Set h to be just large enough for the two-steps look-ahead extractor from Section 4.4 with $r = 3$. Taking $h = \Theta(\log n)$ will do. Set $a = \Theta(\log\log n)$ and note that this choice of a satisfies the hypothesis of the two-steps look-ahead extractor. We now compute

$$\begin{aligned}
(A_1, B_1) &= \mathsf{LookAheadExt}\left((X_1)|_h, Y\right), \\
(A_2, B_2) &= \mathsf{LookAheadExt}\left((X_2)|_h, Y\right), \\
(A_3, B_3) &= \mathsf{LookAheadExt}\left((X_3)|_h, Y\right),
\end{aligned}$$

and then compute

$$X'_1 = \mathsf{Ext}_3(X_1, B_1),$$
$$X'_2 = \mathsf{Ext}_3(X_2, A_2),$$
$$X'_3 = \mathsf{Ext}_3(X_3, A_3),$$

where $\mathsf{Ext}_3 \colon \{0,1\}^\ell \times \{0,1\}^a \to \{0,1\}^h$ is the strong seeded extractor from Theorem 2.2 for entropy $2h$. [5] This was the first iteration of the algorithm Merg_3, in which we produced the random variables X'_1, X'_2 and X'_3 from X_1, X_2, X_3 and Y. In the second iteration we will compute X''_1, X''_2 and X''_3 from X'_1, X'_2, X'_3 and X, Y in a similar way. The difference will be that instead of taking the B variable as a seed for the first row and the corresponding A variables to the other rows, we will take the B variable as a seed for the *second* row and the corresponding A variables to the other two rows. More formally, we compute

$$(A'_1, B'_1) = \mathsf{LookAheadExt}\,(X'_1, Y)\,,$$
$$(A'_2, B'_2) = \mathsf{LookAheadExt}\,(X'_2, Y)\,,$$
$$(A'_3, B'_3) = \mathsf{LookAheadExt}\,(X'_3, Y)\,,$$

and then set

$$X''_1 = \mathsf{Ext}_3(X_1, A'_1),$$
$$X''_2 = \mathsf{Ext}_3(X_2, B'_2),$$
$$X''_3 = \mathsf{Ext}_3(X_3, A'_3).$$

The algorithm continues for one more iteration, and computes

$$(A''_1, B''_1) = \mathsf{LookAheadExt}\,(X''_1, Y)\,,$$
$$(A''_2, B''_2) = \mathsf{LookAheadExt}\,(X''_2, Y)\,,$$
$$(A''_3, B''_3) = \mathsf{LookAheadExt}\,(X''_3, Y)\,,$$

and then computes

$$X'''_1 = \mathsf{Ext}_3(X_1, A''_1),$$
$$X'''_2 = \mathsf{Ext}_3(X_2, A''_2),$$
$$X'''_3 = \mathsf{Ext}_3(X_3, B''_3).$$

Informally speaking, we want to show that if there is enough entropy in Y and in X_2 (namely, ℓ is large enough), then X'''_2 is close to uniform even given X'''_1, X'''_3. This is formalized by the following claim. By the discussion above, once we prove Claim 4.10.1, Theorem 4.5 will follow.

[5] We use the name Ext_3 because during the proof we will argue about the random variables obtained by the two-steps look-ahead extractor from Section 4.4, which uses two strong seeded extractors we denoted by Ext_1 and Ext_2.

Claim 4.10.1. *If $k \geq 11a$ and $\ell \geq 13h$, then*

$$(X_2''', X_1''', X_3''') \approx (U_h, X_1''', X_3''').$$

Before proving Claim 4.10.1, we present the high-level strategy of the proof, which consists of three steps.

- First, we show that in iterations that precede the "good" iteration (that is, the iteration in which the good row is given the B variable, which in our case is the second iteration) the assumption on the input is preserved. Namely, at the end of each such iteration, an output row that corresponds to a good input row is uniform, and the joint distribution of the rows is independent of Y.
- In the second step we show that after the good iteration was executed, the respective output row "gains its independence". That is, an output row that corresponds to a good input row is uniform even conditioned on all other output rows. Moreover, the joint distribution of the rows is independent of Y.
- In the third step we show that the independence of the good row is "preserved" throughout the remaining iterations. Namely, an output row that corresponds to a good input row remains uniform even conditioned on all other output rows, and moreover, the joint distribution of all output rows is independent of Y.

Proof of Claim 4.10.1. The proof will follow the three iterations of the algorithm. In the first iteration we give the "lead", namely the B variable, to the "wrong" row X_1. We show that nothing bad happens by letting X_1 have the lead, in the following sense: after this iteration, we have that the joint distribution (X_1', X_2', X_3') is independent of Y (more formally, this independence holds conditioned on any fixing of carefully chosen (X, Y)-history), and X_2' is close to uniform. So besides losing some entropy in Y and in X_2, and observing some error, we maintain the assumption we had initially about our input – the second row X_2' is uniform, and the joint distribution of the three rows is independent of Y. Thus, in some sense we can "skip" to the iteration in which we give the lead to the good row, which in our case is the second row. This easily generalizes to any number of rows that precede the good row.

Analyzing the first iteration. Recall that $A_2 = \mathsf{Ext}_1(Y, (X_2)|_s)$. Moreover, Y and $(X_2)|_s$ are independent, and $H_\infty(Y) = k \geq 11a$. Since Ext_1 is a strong seeded extractor for entropy $2a$, we have that

$$(A_2, (X_2)|_s) \approx (U_a, \cdot).$$ [6]

Note further that conditioned on any fixing of $(X_2)|_s$, the random variables A_2 and $X|_h$ are independent. Thus, we can apply Lemma 2.11 and conclude that

$$(A_2, X|_h) \approx (U_a, \cdot). \tag{4.15}$$

[6] Recall that our notation dictates that $(X, Z_1, \ldots, Z_r) \approx (Y, \cdot)$ is a shorthand for $(X, Z_1, \ldots, Z_r) \approx (Y, Z_1, \ldots, Z_r)$

Recall that $X_2' = \mathsf{Ext}_3(X_2, A_2)$. We apply Lemma 4.6 to the (X,Y)-history $X|_h$ with $P = A_2$, $M = X_2$ and the extractor Ext_3. The hypothesis of Lemma 4.6 is met since A_2 is a deterministic function of Y and $X|_h$. Moreover, since Ext_3 is an extractor for entropy $2h$, and since

$$\widetilde{H}_\infty\left(X_2 \mid (X|_h)\right) \geq \ell - 3h \geq 10h,$$

Equation (4.2) of Lemma 4.6 holds. Therefore, Lemma 4.6 together with Equation (4.15) imply that

$$(X_2', A_2, X|_h) \approx (U_h, \cdot).$$

Moreover, $A_2, X|_h$ is an (X,Y)-history.

We now apply Lemma 4.7 with $P = X_2'$, $J = (B_1, A_3)$ and the (X,Y)-history $A_2, X|_h$. Since X_2' is a deterministic function of X and A_2 and since B_1, A_3 are deterministic functions of Y and $X|_h$, Lemma 4.7 implies that $\mathcal{H}_1 = (B_1, A_2, A_3, X|_h)$ is an (X,Y)-history and that

$$(X_2', \mathcal{H}_1) \approx (U_h, \cdot). \tag{4.16}$$

Since each of B_1, A_2, A_3 consists of a bits and since Y is independent of $X|_h$, Lemma 2.3 and Lemma 2.6 imply that

$$\widetilde{H}_\infty\left(Y \mid \mathcal{H}_1\right) \geq \widetilde{H}_\infty\left(Y \mid (X|_h)\right) - 3a = H_\infty(Y) - 3a \geq 8a. \tag{4.17}$$

Similarly, conditioned on any fixing of $X|_h$, the random variables B_1, A_2, A_3 are deterministic functions of Y, whereas X_2 is a deterministic function of X. Hence, Lemma 2.6 implies that $\widetilde{H}_\infty\left(X_2 \mid \mathcal{H}_1\right) = \widetilde{H}_\infty\left(X_2 \mid (X|_h)\right)$. Since $X|_h$ consists of $3h$ bits, we have that

$$\widetilde{H}_\infty\left(X_2 \mid \mathcal{H}_1\right) \geq H_\infty(X_2) - 3h = 10h. \tag{4.18}$$

This concludes the first iteration. Note that after the first iteration X_2' is close to uniform (Equation (4.16)). Moreover, Y and X_2 still have (enough) entropy (Equation (4.17), Equation (4.18)).

Analyzing the second iteration. We reached the iteration in which we give the lead to the good row – X_2. We want to show that after this iteration, $(X_2'', X_1'', X_3'') \approx (U_h, X_1'', X_3'')$. Namely, the good row "gains its independence" in the iteration in which it takes the lead.

We continue from Equation (4.16) and apply Lemma 4.8 to the (X,Y)-history $\mathcal{H}_1$, with the $3 \times h$ matrix X' and the weak-source Y. Equation (4.5) of Lemma 4.8 holds by Equation (4.17). Since $h \geq 5s$, Equation (4.6) of Lemma 4.8 holds as well. Therefore, Lemma 4.8 together with Equation (4.16) imply that

$$\mathcal{H}_1' = (X', Z_2', A_1', A_2', A_3', (X')|_s, \mathcal{H}_1)$$

is an (X,Y)-history, and that

$$(B_2', \mathcal{H}_1') \approx (U_a, \cdot).$$

Furthermore, the third item of Lemma (4.8) together with Equation (4.17) imply that

$$\widetilde{H}_\infty\left(Y \mid \mathcal{H}_1'\right) \geq \widetilde{H}_\infty\left(Y \mid \mathcal{H}_1\right) - 3a \geq 5a. \tag{4.19}$$

The fourth item of Lemma 4.8, applied with $N = X_2$, together with Equation (4.18), implies that

$$\widetilde{H}_\infty\left(X_2 \mid \mathcal{H}_1'\right) \geq \widetilde{H}_\infty\left(X_2 \mid \mathcal{H}_1\right) - 3h \geq 7h. \tag{4.20}$$

We now apply Lemma 4.7 to the (X, Y)-history $\mathcal{H}_1'$ with $P = B_2'$ and $J = (X_1'', X_3'')$. Lemma 4.7 is applicable since B_2' is a deterministic function of Y, X_2', and the latter is contained in $\mathcal{H}_1'$. Moreover, X_1'', X_3'' are deterministic functions of X, A_1', A_3', and A_1', A_3' are contained in $\mathcal{H}_1'$. Thus, Lemma 4.7 implies that

$$(B_2', X_1'', X_3'', \mathcal{H}_1') \approx (U_a, \cdot), \tag{4.21}$$

and that $X_1'', X_3'', \mathcal{H}_1'$ is an (X, Y)-history.

Recall that $X_2'' = \mathsf{Ext}_3(X_2, B_2')$. We now apply Lemma 4.6 to the (X, Y)-history $X_1'', X_3'', \mathcal{H}_1'$ with $P = B_2'$, $M = X_2$ and the extractor Ext_3. The hypothesis of the lemma holds since B_2' is a deterministic function of Y and X_2', and the latter is contained in $\mathcal{H}_1'$. By Equation (4.20) and Lemma 2.3, it holds that

$$\widetilde{H}_\infty\left(X_2 \mid X_1'', X_3'', \mathcal{H}_1'\right) \geq \widetilde{H}_\infty\left(X_2 \mid \mathcal{H}_1'\right) - 2h \geq 5h. \tag{4.22}$$

Since Ext_3 is a strong seeded extractor for entropy $2h$, Equation (4.2) of Lemma 4.6 holds. Therefore, Lemma 4.6 together with Equation (4.21) imply that

$$(X_2'', \mathcal{H}_2) \approx (U_h, \cdot), \tag{4.23}$$

where $\mathcal{H}_2 = (B_2', X_1'', X_3'', \mathcal{H}_1')$ is an (X, Y)-history. In terms of entropy-loss,

$$\widetilde{H}_\infty\left(Y \mid \mathcal{H}_2\right) \geq \widetilde{H}_\infty\left(Y \mid X_1'', X_3'', \mathcal{H}_1'\right) - a = \widetilde{H}_\infty\left(Y \mid \mathcal{H}_1'\right) - a \geq 4a, \tag{4.24}$$

where the first inequality follows by Lemma 2.3 and the fact that $|B_2'| = a$. The second equality follows by Lemma 2.6 and the fact that conditioned on any fixing of $\mathcal{H}_1'$, the random variables X_1'', X_3'' are deterministic functions of X, and are thus independent of Y. The last inequality follows by Equation (4.19).

Similarly, since B_2' is independent of X_2 conditioned on any fixing of $\mathcal{H}_1'$, we have that

$$\widetilde{H}_\infty\left(X_2 \mid \mathcal{H}_2\right) = \widetilde{H}_\infty\left(X_2 \mid X_1'', X_3'', \mathcal{H}_1'\right) \geq 5h, \tag{4.25}$$

where the last inequality follows by Equation (4.22). Since X_1'', X_3'' are contained in $\mathcal{H}_2$, this proves what we wanted for this iteration. Namely, after the second iteration, in which the good row takes the lead, $(X_2'', X_1'', X_3'') \approx (U_h, X_1'', X_3'')$.

Analyzing the third iteration. We now show that the independence of the good row X_2'' is "preserved" throughout the following iteration, where again another row takes the lead. We continue from Equation (4.23). We note that conditioned on any fixing of $\mathcal{H}_2$, the random variables A_1'', B_3'' are deterministic functions of Y (as X_1'', X_3'' are contained in $\mathcal{H}_2$). On the other hand, conditioned on any fixing of $\mathcal{H}_2$, the random variable X_2'' is a deterministic function of X (as B_2' is contained in $\mathcal{H}_2$). Thus, by Lemma 4.7 applied to the (X,Y)-history $\mathcal{H}_2$ with $P = X_2''$ and $J = (A_1'', B_3'')$, it holds that

$$(X_2'', A_1'', B_3'', \mathcal{H}_2) \approx (U_h, \cdot).$$

Furthermore, $A_1'', B_3'', \mathcal{H}_2$ is an (X,Y)-history. By Lemma 2.3 and Equation (4.24), it holds that

$$\widetilde{H}_\infty(Y \mid A_1'', B_3'', \mathcal{H}_2) \geq \widetilde{H}_\infty(Y \mid \mathcal{H}_2) - 2a \geq 2a. \tag{4.26}$$

Recall that $A_2'' = \mathsf{Ext}_1(Y, (X_2'')|_s)$. We apply Lemma 4.6 to the (X,Y)-history $A_1'', B_3'', \mathcal{H}_2$ with $P = (X_2'')|_s$, $M = Y$ and the extractor Ext_1. Since Ext_1 is an extractor for entropy $2a$, Equation (4.2) of Lemma 4.6 holds by Equation (4.26). Furthermore, $(X_2'')|_s$ is a deterministic function of X and B_2', which is contained in $\mathcal{H}_2$. Thus, the hypothesis of Lemma 4.6 is met, and we get that

$$(A_2'', X_2'', A_1'', B_3'', \mathcal{H}_2) \approx (U_a, \cdot),$$

and $X_2'', A_1'', B_3'', \mathcal{H}_2$ is an (X,Y)-history. In terms of entropy-loss, since $|X_2''| = h$ and since A_1'', B_3'' are deterministic functions of Y conditioned on any fixing of X_1'', X_3'', which are contained in $\mathcal{H}_2$, Lemma 2.3 together with Equation (4.25) imply that

$$\widetilde{H}_\infty(X_2 \mid X_2'', A_1'', B_3'', \mathcal{H}_2) \geq \widetilde{H}_\infty(X_2 \mid A_1'', B_3'', \mathcal{H}_2) - h = \widetilde{H}_\infty(X_2 \mid \mathcal{H}_2) - h \geq 4h. \tag{4.27}$$

We now apply Lemma 4.7 to the (X,Y)-history $X_2'', A_1'', B_3'', \mathcal{H}_2$ with $P = A_2''$ and $J = X_1''', X_3'''$. Recall that $X_1''' = \mathsf{Ext}_3(X_1, A_1'')$ and $X_3''' = \mathsf{Ext}_3(X_3, B_3'')$. Thus, conditioned on any fixing of A_1'', B_3'', it holds that X_1''', X_3''' are deterministic functions of X, whereas A_2'' is a deterministic function of Y conditioned on any fixing of X_2''. Thus, Lemma 4.7 implies that

$$(A_2'', X_1''', X_3''', \mathcal{H}_2') \approx (U_a, \cdot),$$

where $\mathcal{H}_2' = (X_2'', A_1'', B_3'', \mathcal{H}_2)$ is an (X,Y)-history. In terms of entropy-loss, by Equation (4.27) and Lemma 2.3, we have that

$$\widetilde{H}_\infty(X_2 \mid X_1''', X_3''', \mathcal{H}_2') \geq \widetilde{H}_\infty(X_2 \mid X_2'', A_1'', B_3'', \mathcal{H}_2) - 2h \geq 2h. \tag{4.28}$$

Recall that $X_2''' = \mathsf{Ext}_3(X_2, A_2'')$. We apply Lemma 4.6 to the (X,Y)-history $X_1''', X_3''', \mathcal{H}_2'$, with $P = A_2''$, $M = X_2$ and the extractor Ext_3. Note that A_2'' is a deterministic function of Y and X_2'', which is contained in $\mathcal{H}_2'$. Equation (4.2) of Lemma 4.6 follows by Equation (4.28) and the fact that Ext_3 is an extractor for entropy $2h$. Lemma 4.6 then implies that

$$(X_2''', A_2'', X_1''', X_3''', \mathcal{H}_2') \approx (U_h, \cdot).$$

By Lemma 2.12 it follows that

$$(X_2''', X_1''', X_3''') \approx (U_h, \cdot),$$

which concludes the proof of the claim. □

As mentioned above, the proof of Theorem 4.5 readily follows by Claim 4.10.1. □

4.6.1 Merging r rows – an overview

Generalizing the proof of the three-rows merger presented above to $r > 3$ rows is straightforward. Instead of three iterations, we can apply the algorithm above for r iterations, where at the i^{th} iteration we give the lead to row i. Working out the parameters, one can show that this generalization works for $\ell = O(r^4 \cdot \log n)$ and $k = O(r^3 \cdot \log\log n)$. We now explain how one can improve this, and construct a merger for $\ell = \tilde{O}(r^2) \cdot \log n$ and $k = \tilde{O}(r) \cdot \log\log n$, as we obtain in the actual construction (Theorem 4.8).

For the purpose of constructing mergers with weak-seeds, this improvement, although desired, is not crucial, especially when r is small. This, for example, is the case in the construction of our three-source extractor. Thus, in these cases, the simpler merger depicted above is sufficient. However, for our construction of LCBs the somewhat more involved construction is necessary, and so in the rest of this section we give an informal overview of the actual construction.

Consider the complete graph on r vertices, where vertex $i \in [r]$ represents the i^{th} row of X. In the straightforward generalization of the three rows merger to r rows, we (implicitly) considered r cuts of this graph, where the i^{th} cut is $(\{i\}, [r] \setminus \{i\})$. The construction in Theorem 4.5 guarantees that if X_i is good then after the i^{th} iteration, row i is uniform even given all other rows (and remains as such throughout the following iterations). In the actual construction of our merger (and LCBs) we generalize this idea and guarantee that the following stronger property holds. For any cut (S, S^c) of $[r]$, the i^{th} row is independent of all rows with indices that are separated from i by the cut (S, S^c). Notice that when we used the cuts of the form $(\{i\}, [r] \setminus \{i\})$, we knew that at some iteration the good row $g \in [r]$ is separated from all other rows, and moreover, we knew on which side of the cut g will be (the side that contains the single vertex). By inspecting the construction above, one can see that the algorithm used this second piece of information. Indeed, in each iteration we gave the lead to the single row, namely, we gave the single row the B variable, and all other rows got the A variables.

When considering general cuts (S, S^c), we no longer know which side of the cut contains the good row g. Namely, to who should we give the B variables – to the rows in S or to the rows in S^c. We solve this problem by applying the construction used above twice, in a "flip-flop". Namely, we first give the B variables to the rows in S and the A variables to the rows in S^c, and then run one more round, giving the B variables to the rows in S^c and the A variables to the rows in S. We only then proceed to the next cut in the sequence.

Having the ability to use general cuts allows us to run for only $\log r$ iterations rather than for r iterations. Indeed, instead of choosing r (highly unbalanced) cuts, and run for

r iterations, we use $q = \log r$ (efficiently computable) cuts $S_1, \ldots, S_q$ with the following property. For any two distinct $i, j \in [r]$, there exists $v \in [q]$ such that the cut (S_v, S_v^c) separates i from j. By working with these cuts, the same independence guarantee holds when the algorithm terminates. Indeed, after the v^{th} iteration, row i is uniform and independent of all rows that were separated from i by at least one of the cuts $(S_1, S_1^c), \ldots, (S_v, S_v^c)$. By the property of $S_1, \ldots, S_q$ it follows that after all q iterations were executed, row i is uniform and independent of all other rows. Since we run for only $q = \log r$ iterations, as apposed to r iterations, we obtain a multiplicative saving of roughly $r/\log r$ in both k, ℓ, which yields the desired improvement.

Our construction of LCBs follows the same idea as the construction of the mergers described above. The only difference is that the analysis is done "locally" on t rows, rather than on r rows. The fact that we run for $\log r$ iterations introduces only logarithmic factors of r into k, ℓ, as apposed to polynomial factors.

4.7 Local Correlation Breakers

In this section we prove Theorem 4.1. We start by giving a more formal and complete statement of the theorem.

Theorem 4.6. *For all integers n, r, t and any $\varepsilon > 0$, there exists a $\mathrm{poly}(n, r, \log(1/\varepsilon))$-time computable t-LCB*

$$\mathsf{LCB}\colon \left(\{0,1\}^{\ell}\right)^r \times \{0,1\}^n \to (\{0,1\}^m)^r$$

for entropy k, with error ε, where

$$\begin{aligned} \ell &= \Theta\left(t^2 \cdot \log\left(\frac{nr}{\varepsilon}\right) \cdot \log r\right), \\ m &= \Omega\left(\frac{\ell}{t \cdot \log r}\right), \\ k &= \Omega\left(t \cdot \log(r) \cdot \log\left(\frac{r \cdot \log n}{\varepsilon}\right)\right). \end{aligned}$$

In fact, we prove the following stronger theorem which readily implies Theorem 4.6.

Theorem 4.7. *Let n, r, t be integers, and let $\varepsilon > 0$. Set*

$$\begin{aligned} h &= \Theta\left(t \cdot \log\left(\frac{nr}{\varepsilon}\right)\right), \\ \ell &= \Theta\left(ht \cdot \log r\right) = \Theta\left(t^2 \cdot \log\left(\frac{nr}{\varepsilon}\right) \cdot \log r\right). \end{aligned}$$

There exists a function $\mathsf{LCB}\colon \left(\{0,1\}^{\ell}\right)^r \times \{0,1\}^n \to \left(\{0,1\}^h\right)^r$ with the following property. Let X be an $r \times \ell$ somewhere-random source. Assume that X_g is a good row of X. Let

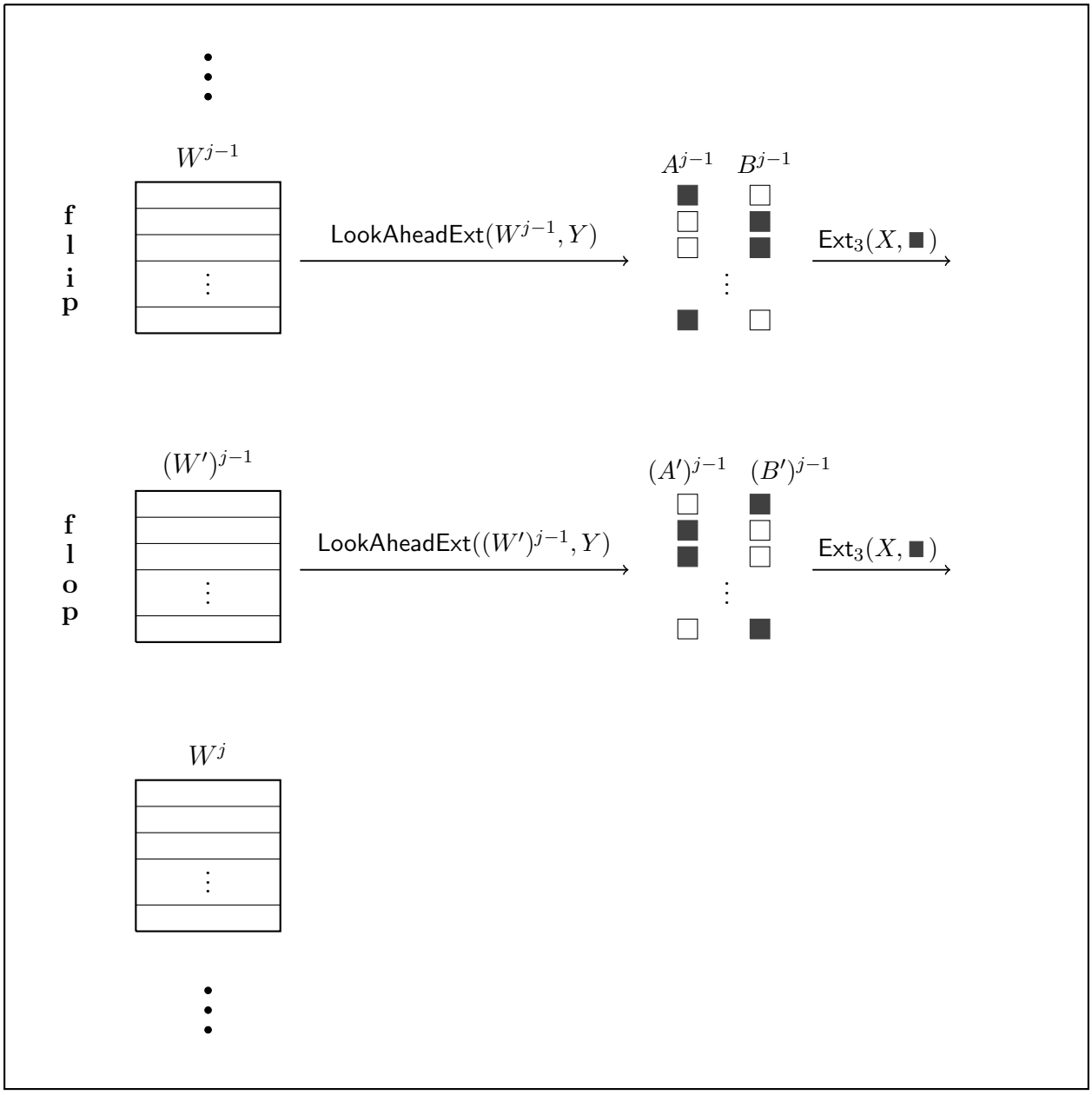

Figure 4.2: A schematic diagram of j^{th} iteration of LCB.

$\mathcal{I} = \{i_1, \ldots, i_{t-1}\} \subseteq [r] \setminus \{g\}$. Let Y be an (n,k)-weak-source that is independent of X, such that

$$k = \Omega\left(t \cdot \log(r) \cdot \log\left(\frac{r \cdot \log n}{\varepsilon}\right)\right). \tag{4.29}$$

Let $\bar{W} = \mathsf{LCB}(X,Y)$. Then, there exists an (X,Y)-history $\mathcal{H}$ that contains $\{\bar{W}_i \mid i \in \mathcal{I}\}$, such that the following holds:

- $(\bar{W}_g, \mathcal{H}) \approx_\varepsilon (U_h, \mathcal{H})$.
- $\widetilde{H}_\infty(X_g \mid \mathcal{H}) \geq 0.9 \cdot \ell$.
- $\widetilde{H}_\infty(Y \mid \mathcal{H}) \geq 0.9 \cdot k$. [7]
- *$\bar{W}_g$ is a deterministic function of X and $\mathcal{H}$.*

Furthermore, for any $i \in [r]$, $\bar{W}_i$ in computable in time $\mathrm{poly}(n, t, \log r, \log(1/\varepsilon))$.

Proof. Let $\varepsilon' = \varepsilon/(32r)$. We make use of the following building blocks for the construction of LCB.

- Let $\mathsf{LookAheadExt}\colon \{0,1\}^h \times \{0,1\}^n \to \{0,1\}^a \times \{0,1\}^a$ be the two-steps look-ahead extractor from Section 4.4, set with error ε' and $a = \Theta(\log(\ell/\varepsilon'))$. Note that this choice of a satisfies the condition $a \geq \Omega(\log(h/\varepsilon'))$ (since $\ell \geq h$), as required by the two-steps look-ahead extractor. Note further that, by the choice of h, it holds that $h = \Omega(t \cdot \log(n/\varepsilon'))$. During the proof we apply Lemma 4.8 to somewhere-random sources with t rows, for which this setting of h is sufficient.
- Let $\mathsf{Ext}_3\colon \{0,1\}^\ell \times \{0,1\}^a \to \{0,1\}^h$ be the strong seeded-extractor from Theorem 2.2 for entropy $2h$, set with error ε'. Note that a was chosen to be large enough so that a seed of length a is sufficient for extracting entropy from length ℓ sources, with error ε'.
- Let $S_1, \ldots, S_q \subseteq [r]$ with the following property. For any two distinct $i, j \in [r]$, there exists $v \in [q]$ such that $|\{i,j\} \cap S_v| = 1$. We think of $S_1, \ldots, S_q$ as a sequence of cuts of the complete graph on vertex set $[r]$. The property above states that for any two distinct vertices $i, j \in [r]$, at least one of the cuts in the sequence separates i from j, namely, either $i \in S_v$, $j \notin S_v$ or $j \in S_v$, $i \notin S_v$, for some $v \in [q]$. We note that such a sequence, with length $q = \lceil \log_2 r \rceil$, can be constructed efficiently in the sense that given $i \in [r]$ and $v \in [q]$, one can determine whether or not $i \in S_v$ in time $\mathrm{polylog}(r)$. This can be done, for example, by taking S_v to be all $i \in [r]$ such that the v^{th} bit in the binary expansion of i is 1. This specific sequence of cuts was used in [Li13] for the construction of his multi-source extractor, though any sequence with the above property will do.

[7] The constant 0.9 in the second and third items can be replaced by $1 - \delta$ for any (not necessarily constant) $\delta > 0$, by taking ℓ to be $1/\delta$ times the stated ℓ, and by taking k to be $\log(1/\delta)/\delta$ times the stated k.

The algorithm LCB iteratively computes a sequence $W^0, W^1, \ldots, W^q$ of $r \times h$ matrices as follows. First, we set $W^0 = X|_h$. As depict in Figure 4.2, for any $j \geq 1$, the matrix W^j is computed as follows, given W^{j-1}. For each row $i \in [r]$ of W^{j-1}, we apply the two-steps look-ahead extractor together with the weak-source Y to obtain

$$\left(A_i^{j-1}, B_i^{j-1}\right) = \mathsf{LookAheadExt}\left(W_i^{j-1}, Y\right).$$

We then define

$$C_i^{j-1} = \begin{cases} A_i^{j-1}, & i \in S_j, \\ B_i^{j-1}, & i \notin S_j. \end{cases}$$

Next, we compute

$$(W')_i^{j-1} = \mathsf{Ext}_3\left(X_i, C_i^{j-1}\right).$$

We apply the two-steps look-ahead extractor for the second time, as follows

$$\left((A')_i^{j-1}, (B')_i^{j-1}\right) = \mathsf{LookAheadExt}\left((W')_i^{j-1}, Y\right),$$

and define

$$(C')_i^{j-1} = \begin{cases} (B')_i^{j-1}, & i \in S_j, \\ (A')_i^{j-1}, & i \notin S_j. \end{cases}$$

Note that the roles of A, B in this application of the two-steps look-ahead extractor were flipped, compared to the previous application. Finally, the i^{th} row of W^j is defined by

$$W_i^j = \mathsf{Ext}_3\left(X_i, (C')_i^{j-1}\right).$$

The output of LCB is then defined by $\mathsf{LCB}(X, Y) = W^q$.

We now turn to the analysis, starting with the running-time. First, note that row W_i^j is a function only of the corresponding rows W_i^{j-1}, X_i and the weak-source Y. For computing each row $i \in [r]$, the algorithm runs for $q = O(\log r)$ iterations. In each iteration it checks which side of the current cut contains i, and performs a constant number of calls to various seeded extractors (Ext_3 and the two extractors $\mathsf{Ext}_1, \mathsf{Ext}_2$ within the two calls to $\mathsf{LookAheadExt}$). Thus, the running-time for computing each output row is $\mathrm{poly}(n, t, \log r, \log(1/\varepsilon))$, as claimed.

For $j = 1, \ldots, q$, define $I_j \subseteq \mathcal{I}$ by

$$I_j = \{i_v \in \mathcal{I} : \ |\{g, i_v\} \cap S_j| = 1\}.$$

That is, I_j contains all vertices in $\mathcal{I}$ that are separated from g by the cut S_j. We further define $I_0 = \emptyset$, and let $\mathcal{I}_j = \cup_{j'=0}^{j} I_{j'}$. Note that $\mathcal{I}_j$ is the set of vertices in $\mathcal{I}$ that are separated from g by at least one of the cuts $S_1, \ldots, S_j$. By the property of the sequence $S_1, \ldots, S_q$, we have that $\mathcal{I}_q = \mathcal{I}$. We prove the following claim by induction on j.

Claim 4.10.2. *There exists an (X, Y)-history $\mathcal{H}_j$ such that the following holds:*

- $\mathcal{H}_j$ *contains* $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$.
- $\mathcal{H}_j$ *contains* $\left\{(C')_i^{j-1} \mid i \in \mathcal{I} \cup \{g\}\right\}$ *for all* $j \geq 1$.
- $\left(W_g^j, \mathcal{H}_j\right) \approx_{\varepsilon_j} (U_h, \mathcal{H}_j)$, *where* $\varepsilon_0 = 0$ *and* $\varepsilon_j \leq 2\varepsilon_{j-1} + 16\varepsilon'$ *for all* $j \geq 1$.
- $\widetilde{H}_\infty(X_g \mid \mathcal{H}_j) \geq \ell - 5htj$.
- $\widetilde{H}_\infty(Y \mid \mathcal{H}_j) \geq k - 5atj$.

Proof of Claim 4.10.2. We prove the claim by induction on j. The claim readily follows for $j = 0$ with an empty (X, Y)-history $\mathcal{H}_0$. Consider $j \geq 1$ and assume the correctness of the claim for $j - 1$. By the induction hypothesis, we have that

$$\left(W_g^{j-1}, \mathcal{H}_{j-1}\right) \approx_{\varepsilon_{j-1}} (U_h, \cdot)\,.$$

Recall that

$$W_g^{j-1} = \begin{cases} (X_g)\,|_h, & j = 1; \\ \mathsf{Ext}_3\left(X_g, (C')_g^{j-2}\right), & j > 1. \end{cases}$$

If $j = 1$ then clearly W_g^{j-1} is a deterministic function of X. For $j > 1$, by the induction hypothesis, $\mathcal{H}_{j-1}$ contains $(C')_g^{j-2}$ and so W_g^{j-1} is a deterministic function of $X, \mathcal{H}_{j-1}$. Moreover, by the induction hypothesis $\{W_i^{j-1} \mid i \in \mathcal{I}_{j-1}\}$ are all contained in $\mathcal{H}_{j-1}$. Since C_i^{j-1} is a deterministic function of W_i^{j-1} and Y, we have that $\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\}$ are all deterministic functions of $\mathcal{H}_{j-1}$ and Y. Thus, we can apply Lemma 4.7 to the (X, Y)-history $\mathcal{H}_{j-1}$ with $P = W_g^{j-1}$ and $J = \{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\}$, and conclude that

$$\left(W_g^{j-1}, \left\{C_i^{j-1}\right\}_{i \in \mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right) \approx_{\varepsilon_{j-1}} (U_h, \cdot)\,, \tag{4.30}$$

and that $\left\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}, \mathcal{H}_{j-1}$ is an (X, Y)-history. In terms of entropy-loss, by Lemma 2.3 and by the induction hypothesis,

$$\begin{aligned} \widetilde{H}_\infty\left(Y \mid \left\{C_i^{j-1}\right\}_{i \in \mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right) &\geq \widetilde{H}_\infty\left(Y \mid \mathcal{H}_{j-1}\right) - |\mathcal{I}_{j-1}| \cdot a \\ &\geq k - 5atj + 4at + a, \end{aligned} \tag{4.31}$$

where we used the fact that $|\mathcal{I}_{j-1}| \leq |\mathcal{I}| \leq t - 1$ and that $|C_i^{j-1}| = a$ for all $i \in \mathcal{I}_{j-1}$. As for the entropy of X_g, we have that

$$\widetilde{H}_\infty\left(X_g \mid \left\{C_i^{j-1}\right\}_{i \in \mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right) = \widetilde{H}_\infty\left(X_g \mid \mathcal{H}_{j-1}\right) \geq \ell - 5ht(j-1), \tag{4.32}$$

where the first equality holds by Lemma 2.6 and by the induction hypothesis. Indeed, the induction hypothesis implies that all random variables $\left\{W_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ are contained in $\mathcal{H}_{j-1}$. Since C_i^{j-1} is a deterministic function of W_i^{j-1} and Y, it holds that conditioned on any fixing of $\mathcal{H}_{j-1}$, the random variables $\left\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ are deterministic functions of Y, and thus are independent of X_g. The second inequality follows by the induction hypothesis.

We proceed with the analysis by considering two cases, according to whether or not $g \in S_j$.

Case 1: $g \in \mathbf{S_j}$. Recall that $A_g^{j-1} = \mathsf{Ext}_1(Y, (W_g^{j-1})|_s)$. We apply Lemma 4.6 to the (X,Y)-history $\left\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}, \mathcal{H}_{j-1}$, with $P = \left(W_g^{j-1}\right)|_s$, $M = Y$ and the extractor Ext_1. The hypothesis of Lemma 4.6 is met since, as explained above, W_g^{j-1} is a deterministic function of X and $\mathcal{H}_{j-1}$. Furthermore, Equation (4.2) of Lemma 4.6 follows by Equation (4.31), the fact that Ext_1 is a strong seeded extractor for entropy $2a$ with error ε', and our assumption on k, namely, Equation (4.29). Thus, Lemma 4.6 together with Equation (4.30) imply that

$$\left(A_g^{j-1}, \mathcal{H}'_{j-1}\right) \approx_{\varepsilon_{j-1}+2\varepsilon'} (U_a, \cdot),$$

where

$$\mathcal{H}'_{j-1} = \left(W_g^{j-1}, \left\{C_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right)$$

is an (X,Y)-history. Note that we added W_g^{j-1} rather than $(W_g^{j-1})|_s$ to $\mathcal{H}'_{j-1}$. This can be done either by applying Lemma 4.7 or by considering the extractor Ext'_1 that takes W_g^{j-1} as the seed, ignore the length $h-s$ suffix of it and use $(W_g^{j-1})|_s$ as a seed for Ext_1. In terms of entropy-loss,

$$\widetilde{H}_\infty\left(X_g \mid \mathcal{H}'_{j-1}\right) \geq \widetilde{H}_\infty\left(X_g \mid \left\{C_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right) - \left|W_g^{j-1}\right| \geq \ell - 5htj + 5ht - h, \quad (4.33)$$

where the first inequality follows by Lemma 2.3, and the second inequality follows by Equation (4.32).

We now apply Lemma 4.7 to the (X,Y)-history $\mathcal{H}'_{j-1}$, with $P = A_g^{j-1}$ and

$$J = \left\{(W')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\} \cup \left\{W_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_{j-1}\right\}.$$

To see that this application of Lemma 4.7 is valid, note that A_g^{j-1} is a deterministic function of Y and W_g^{j-1}, and the latter is contained in $\mathcal{H}'_{j-1}$ – the history to which we apply the lemma. On the other hand, each of the random variables in $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ is a deterministic function of X and $\left\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$, all of which are contained in $\mathcal{H}'_{j-1}$. Moreover, as explained above, $\left\{W_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_{j-1}\right\}$ are all deterministic functions of X and $\mathcal{H}_{j-1}$ (this was shown for $i = g$ but can be easily shown to hold for all $i \in \mathcal{I} \cup \{g\}$). Thus, Lemma 4.7 implies that

$$\left(A_g^{j-1}, \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \left\{W_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_{j-1}}, \mathcal{H}'_{j-1}\right) \approx_{\varepsilon_{j-1}+2\varepsilon'} (U_a, \cdot),$$

and that $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}, \left\{W_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_{j-1}\right\}, \mathcal{H}'_{j-1}$ is an (X,Y)-history. In terms of entropy-loss, Equation (4.33) together with Lemma 2.3 imply that

$$\begin{aligned}\widetilde{H}_\infty\left(X_g \mid \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \left\{W_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_{j-1}}, \mathcal{H}'_{j-1}\right) &\geq \widetilde{H}_\infty\left(X_g \mid \mathcal{H}'_{j-1}\right) - |\mathcal{I}| \cdot h \\ &\geq \ell - 5htj + 4ht. \quad (4.34)\end{aligned}$$

Recall that $(W')_g^{j-1} = \mathsf{Ext}_3\left(X_g, C_g^{j-1}\right)$. Since $g \in S_j$ it holds that $C_g^{j-1} = A_g^{j-1}$. We apply Lemma 4.6 to the (X,Y)-history $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}, \left\{W_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_{j-1}\right\}$,

$\mathcal{H}'_{j-1}$, with $P = A_g^{j-1} = C_g^{j-1}$, $M = X_g$ and the extractor Ext_3. The hypothesis of Lemma 4.6 is met since A_g^{j-1} is a deterministic function of Y and W_g^{j-1}, and the latter in contained in the (X,Y)-history to which we apply the lemma. Furthermore, Equation (4.2) of Lemma 4.6 follows by Equation (4.34) and our hypothesis on k, namely, Equation (4.29), and since Ext_3 is a strong seeded extractor for entropy $2h$ with error ε'. Therefore, Lemma 4.6 implies that

$$\left((W')_g^{j-1}, \mathcal{H}''_{j-1}\right) \approx_{\varepsilon_{j-1}+4\varepsilon'} (U_h, \cdot)\,,$$

where

$$\mathcal{H}''_{j-1} = \left(A_g^{j-1}, \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \left\{W_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_{j-1}}, \mathcal{H}'_{j-1}\right).$$

In terms of entropy-loss,

$$\begin{aligned}
\widetilde{H}_\infty\left(Y \mid \mathcal{H}''_{j-1}\right) &\geq \widetilde{H}_\infty\left(Y \mid \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \left\{W_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_{j-1}}, \mathcal{H}'_{j-1}\right) - a \\
&= \widetilde{H}_\infty\left(Y \mid \mathcal{H}'_{j-1}\right) - a \\
&= \widetilde{H}_\infty\left(Y \mid \left\{C_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right) - a \\
&\geq k - 5atj + 4at, \qquad (4.35)
\end{aligned}$$

where the first inequality follows by Lemma 2.3, and since $|A_g^{j-1}| = a$. The second equality follows by Lemma 2.6, which is applicable since conditioned on any fixing of the random variables $\left\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$, all of which are all contained in $\mathcal{H}'_{j-1}$, the random variables $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ are deterministic functions of X, and in particular are independent of Y. Moreover, as explained above, conditioned on any fixing of $\mathcal{H}'_{j-1}$, the random variables $\left\{W_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_{j-1}\right\}$ are deterministic functions of X. The third equality follows by Lemma 2.6 as W_g^{j-1} is a deterministic function of X and $\mathcal{H}_{j-1}$. The last inequality follows by Equation (4.31).

We now apply Lemma 4.7 to the (X,Y)-history $\mathcal{H}''_{j-1}$, with $P = (W')_g^{j-1}$ and

$$J = \left\{(C')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\} \cup \left\{C_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_{j-1}\right\}.$$

This application of Lemma 4.7 is valid since $(W')_g^{j-1}$ is a deterministic function of X, C_g^{j-1}, and C_g^{j-1} is contained in the history to which we apply the lemma. This is because $C_g^{j-1} = A_g^{j-1}$ since $g \in S_j$. On the other hand, each random variable in $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ is a deterministic function of Y and $(W')_i^{j-1}$, and all of the random variables $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ are contained in the $\mathcal{H}''_{j-1}$. Furthermore, each of the random variables in $\left\{C_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_{j-1}\right\}$ is a deterministic function of Y and $\left\{W_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_{j-1}\right\}$, and the latter random variables are all contained in $\mathcal{H}''_{j-1}$. Thus, Lemma 4.7 implies that

$$\left((W')_g^{j-1}, \mathcal{H}'''_{j-1}\right) \approx_{\varepsilon_{j-1}+4\varepsilon'} (U_h, \cdot)\,, \qquad (4.36)$$

where

$$\mathcal{H}'''_{j-1} = \left(\left\{(C')_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \left\{C_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_{j-1}}, \mathcal{H}''_{j-1}\right)$$

is an (X,Y)-history. By Lemma 2.3, Equation (4.35) and the fact that $|\mathcal{I}| = t-1$, it holds that

$$\widetilde{H}_\infty\left(Y \mid \mathcal{H}'''_{j-1}\right) \geq \widetilde{H}_\infty\left(Y \mid \mathcal{H}''_{j-1}\right) - |\mathcal{I}| \cdot a \geq k - 5atj + 3at + a. \tag{4.37}$$

Note that all random variables $\left\{C_i^{j-1} \mid i \in \mathcal{I} \cup \{g\}\right\}$ are contained in $\mathcal{H}'''_{j-1}$. Indeed, $\left\{C_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_{j-1}\right\}$ are contained in $\mathcal{H}'''_{j-1}$ by definition. Moreover, the random variables $\left\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ are contained in $\mathcal{H}'_{j-1}$ which in turn is contained in $\mathcal{H}'''_{j-1}$. Finally, since $g \in S_j$, it holds that $C_g^{j-1} = A_g^{j-1}$, and A_g^{j-1} is contained in $\mathcal{H}''_{j-1}$, and so it is also contained in $\mathcal{H}'''_{j-1}$.

We apply Lemma 4.8 to the (X,Y)-history $\mathcal{H}'''_{j-1}$, with W (in the notation of Lemma 4.8) equals to rows $\mathcal{I}\cup\{g\}$ of $(W')^{j-1}$. The hypothesis of Lemma 4.8 is met since these rows of $(W')^{j-1}$ are deterministic functions of X and $\left\{C_i^{j-1} \mid i \in \mathcal{I} \cup \{g\}\right\}$ which, by the above, are contained in $\mathcal{H}'''_{j-1}$. Furthermore, Equation (4.5) of Lemma 4.8 follows by Equation (4.37), and Equation (4.6) follows by our choice of h. Lemma 4.8 together with Equation (4.36) imply that

$$\left((B')_g^{j-1}, \mathcal{H}''''_{j-1}\right) \approx_{2\varepsilon_{j-1}+14\varepsilon'} (U_a, \cdot),$$

where

$$\mathcal{H}''''_{j-1} = \left(\left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}\cup\{g\}}, (Z')_g^{j-1}, \left\{(A')_i^{j-1}\right\}_{i\in\mathcal{I}\cup\{g\}}, \left\{\left((W')_i^{j-1}\right)|_s\right\}_{i\in\mathcal{I}\cup\{g\}}, \mathcal{H}'''_{j-1}\right)$$

is an (X,Y)-history. Moreover, by the third item of Lemma 4.8 and Equation (4.37), we have that

$$\widetilde{H}_\infty\left(Y \mid \mathcal{H}''''_{j-1}\right) \geq \widetilde{H}_\infty\left(Y \mid \mathcal{H}'''_{j-1}\right) - |\mathcal{I}\cup\{g\}| \cdot a \geq k - 5atj + 2at + a. \tag{4.38}$$

As for the entropy-loss of X_g, it holds that

$$\begin{aligned}
\widetilde{H}_\infty\left(X_g \mid \mathcal{H}''''_{j-1}\right) &\geq \widetilde{H}_\infty\left(X_g \mid \mathcal{H}'''_{j-1}\right) - |\mathcal{I}\cup\{g\}| \cdot h \\
&= \widetilde{H}_\infty\left(X_g \mid \mathcal{H}'''_{j-1}\right) - th \\
&= \widetilde{H}_\infty\left(X_g \mid \mathcal{H}''_{j-1}\right) - th \\
&= \widetilde{H}_\infty\left(X_g \mid \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \left\{W_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_{j-1}}, \mathcal{H}'_{j-1}\right) - th \\
&\geq \widetilde{H}_\infty\left(X_g \mid \mathcal{H}'_{j-1}\right) - (2t-1)h \\
&\geq \ell - 5htj + 2ht + h,
\end{aligned} \tag{4.39}$$

where the first inequality follows by the fourth item of Lemma 4.8, with $N = X_g$. The second equality follows since $|\mathcal{I}\cup\{g\}| = t$. The third equality follows by Lemma 2.6 and the fact that conditioned on any fixing of $\mathcal{H}''_{j-1}$, all random variables $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$, $\left\{C_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_{j-1}\right\}$ are deterministic functions of Y. The fourth equality follows by Lemma 2.6 since A_g^{j-1} is a deterministic function of Y conditioned on any fixing of

$\mathcal{H}'_{j-1}$. The penultimate inequality follows by Lemma 2.3. The last inequality follows by Equation (4.34).

Recall that $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ are contained in $\mathcal{H}''''_{j-1}$, as these random variables are already contained in $\mathcal{H}'''_{j-1}$. However, due to the application of Lemma 4.8 above, we have that $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$ are all contained in $\mathcal{H}''''_{j-1}$. To see this, note that after the application of the two-steps look-ahead extractor, $\left\{(A')_i^{j-1} \mid i \in \mathcal{I} \cup \{g\}\right\}$ are all contained in $\mathcal{H}''''_{j-1}$. However, $(A')_i^{j-1} = (C')_i^{j-1}$ for $i \notin S_j$, which is equivalent to $i \in I_j$ as $g \in S_j$. Since $\mathcal{I}_j = \mathcal{I}_{j-1} \cup I_j$, it follows that $(C')_i^{j-1}$ is contained in $\mathcal{H}''''_{j-1}$ for all $i \in \mathcal{I}_j$, as claimed.

We apply Lemma 4.7 to the (X,Y)-history $\mathcal{H}''''_{j-1}$ with $P = (B')_g^{j-1}$ and $J = \left\{W_i^j \mid i \in \mathcal{I}_j\right\}$, and conclude that

$$\left((B')_g^{j-1}, \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \mathcal{H}''''_{j-1}\right) \approx_{2\varepsilon_{j-1}+14\varepsilon'} (U_a, \cdot)\,.$$

This application of Lemma 4.7 is valid since $(B')_g^{j-1}$ is a deterministic function of Y and $(W')_g^{j-1}$, which is contained in $\mathcal{H}''''_{j-1}$. Furthermore, the random variables $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$ are deterministic functions of X and $\left\{(C')_i^j \mid i \in \mathcal{I}_j\right\}$ which, by the above, are also contained in $\mathcal{H}''''_{j-1}$. By Equation (4.39) and Lemma 2.3, it holds that

$$\widetilde{H}_\infty\left(X_g \mid \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \mathcal{H}''''_{j-1}\right) \geq \widetilde{H}_\infty\left(X_g \mid \mathcal{H}''''_{j-1}\right) - (t-1)h \geq \ell - 5htj + ht + 2h. \quad (4.40)$$

We apply Lemma 4.6 to the (X,Y)-history $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}, \mathcal{H}''''_{j-1}$ with $P = (B')_g^{j-1}$, $M = X_g$ and the extractor Ext_3. The hypothesis of Lemma 4.6 is met since $(B')_g^{j-1}$ is a deterministic function of Y and $(W')_g^{j-1}$, which is contained $\mathcal{H}''''_{j-1}$. Moreover, Equation (4.2) of Lemma 4.6 follows by Equation (4.40). Since $W_i^j = \mathsf{Ext}_3\left(X_i, (C')_i^{j-1}\right)$, and since $(C')_g^{j-1} = (B')_g^{j-1}$ (as $g \in S_j$), Lemma 4.6 implies that

$$\left(W_g^j, (B')_g^{j-1}, \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \mathcal{H}''''_{j-1}\right) \approx_{2\varepsilon_{j-1}+16\varepsilon'} (U_h, \cdot)\,.$$

Finally, we apply Lemma 4.7 to the (X,Y)-history $(B')_g^{j-1}$, $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$, $\mathcal{H}''''_{j-1}$, with $P = W_g^j$ and $J = \left\{(C')_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$. This application of Lemma 4.7 is valid since W_g^j is a deterministic function of X and $(C')_g^{j-1}$, which is contained in the history to which we apply the lemma, since $(C')_g^{j-1} = (B')_g^{j-1}$ (recall that $g \in S_j$). Moreover, the random variables $\left\{(C')_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$ are deterministic functions of Y and $\left\{(W')_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$, all of which are contained in $\mathcal{H}''''_{j-1}$. Thus, Lemma 4.7 implies that

$$\left(W_g^j, \mathcal{H}_j\right) \approx_{2\varepsilon_{j-1}+16\varepsilon'} (U_h, \cdot)\,,$$

where

$$\mathcal{H}_j = \left(\left\{(C')_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_j}, (B')_g^{j-1}, \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \mathcal{H}''''_{j-1}\right)$$

is an (X,Y)-history. This proves the third item of the claim. Note that $\mathcal{H}_j$ contains $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$, which proves the first item of the claim.

As for the second item, as proved above, $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$ are contained in $\mathcal{H}''''_{j-1}$ and therefore also contained in $\mathcal{H}_j$. Moreover, by definition, $\left\{(C')_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$ are contained in $\mathcal{H}_j$, and so all random variables $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}\right\}$ are contained in $\mathcal{H}_j$. Finally, since $g \in S_j$ it holds that $(C')_g^{j-1} = (B')_g^{j-1}$, and the latter is contained in $\mathcal{H}_j$. To summarize, all random variables $\left\{(C')_i^{j-1} \mid i \in \mathcal{I} \cup \{g\}\right\}$ are contained in $\mathcal{H}_j$, and so the second item of the claim follows.

The fourth item of the claim follows by Equation (4.40), Lemma 2.6 and the fact that $(B')_g^{j-1}$ and $\left\{(C')_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$ are all deterministic functions of Y and $\left\{(W')_i^{j-1} \mid i \in \mathcal{I} \cup \{g\}\right\}$. As for the fifth item,

$$\begin{aligned}\widetilde{H}_\infty\left(Y \mid \mathcal{H}_j\right) &\geq \widetilde{H}_\infty\left(Y \mid \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \mathcal{H}''''_{j-1}\right) - \left(|\mathcal{I} \setminus \mathcal{I}_j| + 1\right) \cdot a \\ &= \widetilde{H}_\infty\left(Y \mid \mathcal{H}'''_{j-1}\right) - \left(|\mathcal{I} \setminus \mathcal{I}_j| + 1\right) \cdot a \\ &\geq k - 5atj + at + a,\end{aligned}$$

where the first inequality follows by Lemma 2.3. The second equality follows by Lemma 2.6, which is applicable as conditioned on any fixing of $\mathcal{H}''''_{j-1}$, the random variables $\{W_i^j \mid i \in \mathcal{I}_j\}$ are deterministic functions of X. Indeed, W_i^j is a deterministic function of X and $(C')_i^{j-1}$, and as shown above, all random variables $\{(C')_i^{j-1} \mid i \in \mathcal{I}_j\}$ are contained in $\mathcal{H}''''_{j-1}$. The last inequality follows by Equation (4.38).

Case 2: $\mathbf{g \notin S_j}$. We continue from Equation (4.30), and apply Lemma 4.8 to the (X,Y)-history $\left\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}, \mathcal{H}_{j-1}$, and rows $\mathcal{I} \cup \{g\}$ of W^{j-1}. We note that the hypothesis of Lemma 4.8 is met. Indeed, as shown above, for any $i \in \mathcal{I} \cup \{g\}$, W_i^{j-1} is a deterministic function of X and $\mathcal{H}_{j-1}$. Moreover, Equation (4.5) of Lemma 4.8 follows by Equation (4.31), and Equation (4.6) follows by our choice of h. Therefore, by Lemma 4.8 and Equation (4.30) we have that

$$\left(B_g^{j-1}, \mathcal{H}'_{j-1}\right) \approx_{2\varepsilon_{j-1}+6\varepsilon'} \left(U_a, \cdot\right),$$

where

$$\mathcal{H}'_{j-1} = \left(\left\{W_i^{j-1}\right\}_{i\in\mathcal{I}\cup\{g\}}, Z_g^{j-1}, \left\{A_i^{j-1}\right\}_{i\in\mathcal{I}\cup\{g\}}, \left\{\left(W_i^{j-1}\right)|_s\right\}_{i\in\mathcal{I}\cup\{g\}}, \left\{C_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right)$$

is an (X,Y)-history. Furthermore, by the third item of Lemma 4.8 and by Equation (4.31) it holds that

$$\widetilde{H}_\infty\left(Y \mid \mathcal{H}'_{j-1}\right) \geq \widetilde{H}_\infty\left(Y \mid \left\{C_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right) - |\mathcal{I} \cup \{g\}| \cdot a \geq k - 5atj + 3at + a. \quad (4.41)$$

The fourth item of Lemma 4.8, with $N = X_G$, together with Equation (4.32) imply that

$$\widetilde{H}_\infty\left(X_g \mid \mathcal{H}'_{j-1}\right) \geq \widetilde{H}_\infty\left(X_g \mid \left\{C_i^{j-1}\right\}_{i\in\mathcal{I}_{j-1}}, \mathcal{H}_{j-1}\right) - |\mathcal{I} \cup \{g\}| \cdot h \geq \ell - 5htj + 4ht. \quad (4.42)$$

Note that $\left\{C_i^{j-1} \mid i \in \mathcal{I}_{j-1}\right\}$ are all contained in $\mathcal{H}'_{j-1}$ even prior to the application of the two-steps look-ahead extractor. But in fact, after this application, all random

variables in $\left\{C_i^{j-1} \mid i \in \mathcal{I}_j\right\}$ are contianed in $\mathcal{H}'_{j-1}$. To see this, recall that $A_i^{j-1} = C_i^{j-1}$ for all $i \in S_j$. Furthermore, $i \in S_j \iff i \in I_j$ as $g \notin S_j$. Since $\mathcal{I}_j = \mathcal{I}_{j-1} \cup I_j$, and since $\{A_i^{j-1} \mid i \in \mathcal{I} \cup \{g\}\}$ are all contained in $\mathcal{H}'_{j-1}$, the claimed assertion follows. Namely, for all $i \in \mathcal{I}_j$, the random variable C_i^{j-1} is contained in $\mathcal{H}'_{j-1}$.

We now apply Lemma 4.7 to the (X,Y)-history $\mathcal{H}'_{j-1}$, with $P = B_g^{j-1}$ and $J = \left\{(W')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$. Lemma 4.7 is applicable as B_g^{j-1} is a deterministic function of Y and W_g^{j-1}, which is contained in $\mathcal{H}'_{j-1}$. Moreover, by the above, for each $i \in \mathcal{I}_j$, the random variable C_i^{j-1} is contained in $\mathcal{H}'_{j-1}$, and so $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$ are all deterministic functions of X and $\mathcal{H}'_{j-1}$. Thus, Lemma 4.7 implies that

$$\left(B_g^{j-1}, \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \mathcal{H}'_{j-1}\right) \approx_{2\varepsilon_{j-1}+6\varepsilon'} (U_a, \cdot)\,.$$

In terms of entropy-loss, Lemma 2.3 together with Equation (4.42) imply that

$$\widetilde{H}_\infty\left(X_g \mid \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \mathcal{H}'_{j-1}\right) \geq \widetilde{H}_\infty\left(X_g \mid \mathcal{H}'_{j-1}\right) - |\mathcal{I}_j| \cdot h \geq \ell - 5htj + 3ht + h. \quad (4.43)$$

We now apply Lemma 4.6 to the (X,Y)-history $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$, $\mathcal{H}'_{j-1}$, with $P = B_g^{j-1}$, $M = X_g$ and the extractor Ext_3. Recall that $(W')_g^{j-1} = \mathsf{Ext}_3\left(X_g, C_g^{j-1}\right)$. Since $g \notin S_j$, it follows that $C_g^{j-1} = B_g^{j-1}$. Thus, Lemma 4.6 implies that

$$\left((W')_g^{j-1}, B_g^{j-1}, \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \mathcal{H}'_{j-1}\right) \approx_{2\varepsilon_{j-1}+8\varepsilon'} (U_h, \cdot)\,.$$

Lemma 4.6 is applicable since B_g^{j-1} is a deterministic function of Y and W_g^{j-1}, which is contained in $\mathcal{H}'_{j-1}$. Furthermore, Equation (4.2) of Lemma 4.6 follows by Equation (4.43). In terms of entropy-loss, we have that

$$\begin{aligned}\widetilde{H}_\infty\left(Y \mid B_g^{j-1}, \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \mathcal{H}'_{j-1}\right) &\geq \widetilde{H}_\infty\left(Y \mid \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \mathcal{H}'_{j-1}\right) - a \\ &= \widetilde{H}_\infty\left(Y \mid \mathcal{H}'_{j-1}\right) - a \\ &\geq k - 5atj + 3at, \end{aligned} \quad (4.44)$$

where the first inequality follows by Lemma 2.3 and the fact that $\left|B_g^{j-1}\right| = a$. The second equality follows by Lemma 2.6 and the fact that conditioned on any fixing of $\mathcal{H}'_{j-1}$, each random variable in $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$ is a deterministic function of X. Indeed, as shown above, $\left\{C_i^{j-1} \mid i \in \mathcal{I}_j\right\}$ are all contained in $\mathcal{H}'_{j-1}$. The last inequality follows by Equation (4.41).

We now apply Lemma 4.7 to the (X,Y)-history B_g^{j-1}, $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$, $\mathcal{H}'_{j-1}$, with $P = (W')_g^{j-1}$ and

$$J = \left\{(C')_i^{j-1} \mid i \in \mathcal{I}_j\right\} \cup \left\{C_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}.$$

Lemma 4.7 is applicable since $(W')_g^{j-1}$ is a deterministic function of X and $C_g^{j-1} = B_g^{j-1}$, which is contained in $\mathcal{H}'_{j-1}$. On the other hand, each random variable in $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$ is a deterministic function of Y and $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$, and the latter are contained in the

history to which we apply the lemma. Similarly, every random variable in $\left\{C_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$ is a deterministic function of Y and $\left\{W_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$, and the latter are contained in $\mathcal{H}'_{j-1}$. We conclude that

$$\left((W')_g^{j-1}, \mathcal{H}''_{j-1}\right) \approx_{2\varepsilon_{j-1}+8\varepsilon'} (U_h, \cdot),$$

where

$$\mathcal{H}''_{j-1} = \left(\left\{(C')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \left\{C_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_j}, B_g^{j-1}, \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \mathcal{H}'_{j-1}\right).$$

In terms of entropy, Equation (4.44) together with Lemma 2.3 imply that

$$\begin{aligned}\widetilde{H}_\infty\left(Y \mid \mathcal{H}''_{j-1}\right) &\geq \widetilde{H}_\infty\left(Y \mid B_g^{j-1}, \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \mathcal{H}'_{j-1}\right) - |\mathcal{I}| \cdot a \\ &\geq k - 5atj + 2at + a. \end{aligned} \tag{4.45}$$

Recall that $(A')_g^{j-1} = \mathsf{Ext}_1(Y, ((W')_g^{j-1})|_s)$. We apply Lemma 4.6 to the (X,Y)-history $\mathcal{H}''_{j-1}$ with $P = ((W')_g^{j-1})|_s$, $M = Y$ and the extractor Ext_1. Lemma 4.6 is applicable since $(W')_g^{j-1}$ is a deterministic function of X and $C_g^{j-1} = B_g^{j-1}$, which is contained in $\mathcal{H}''_{j-1}$. Furthermore, Equation (4.2) of Lemma 4.6 holds by Equation (4.45). Thus, Lemma 4.6 implies that

$$\left((A')_g^{j-1}, (W')_g^{j-1}, \mathcal{H}''_{j-1}\right) \approx_{2\varepsilon_{j-1}+10\varepsilon'} (U_a, \cdot).$$

In terms of entropy-loss, we have that

$$\begin{aligned}\widetilde{H}_\infty\left(X_g \mid (W')_g^{j-1}, \mathcal{H}''_{j-1}\right) &\geq \widetilde{H}_\infty\left(X_g \mid \mathcal{H}''_{j-1}\right) - h \\ &= \widetilde{H}_\infty\left(X_g \mid \left\{(W')_i^{j-1}\right\}_{i\in\mathcal{I}_j}, \mathcal{H}'_{j-1}\right) - h \\ &\geq \ell - 5htj + 3ht, \end{aligned} \tag{4.46}$$

where the first inequality follows by Lemma 2.3 and the fact that $\left|(W')_g^{j-1}\right| = h$. The second equality follows by Lemma 2.6 which is applicable as conditioned on any fixing of $\mathcal{H}'_{j-1}$, all random variables $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$, $\left\{C_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$ and B_g^{j-1} are deterministic functions of Y. The last inequality follows by Equation (4.43).

Next we apply Lemma 4.7 to the (X,Y)-history $(W')_g^{j-1}, \mathcal{H}''_{j-1}$, with $P = (A')_g^{j-1}$ and

$$J = \left\{W_i^j \mid i \in \mathcal{I}_j\right\} \cup \left\{(W')_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$$

to conclude that

$$\left((A')_g^{j-1}, \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \left\{(W')_i^{j-1}\right\}_{i\in\{g\}\cup\mathcal{I}\setminus\mathcal{I}_j}, \mathcal{H}''_{j-1}\right) \approx_{2\varepsilon_{j-1}+10\varepsilon'} (U_a, \cdot).$$

This application of Lemma 4.7 is valid since $(A')_g^{j-1}$ is a deterministic function of Y and $(W')_g^{j-1}$, which is contained in the history to which we apply the lemma. On the other hand, $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$ are all deterministic functions of X and $\left\{(C')_i^j \mid i \in \mathcal{I}_j\right\}$, all of which are contained in $\mathcal{H}''_{j-1}$. Furthermore, all random variables $\left\{(W')_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$

are deterministic functions of X and $\left\{C_i^{j-1} \mid i \in \mathcal{I} \setminus \mathcal{I}_j\right\}$, and these random variables are contained in $\mathcal{H}''_{j-1}$. As for the entropy-loss, Lemma 2.3 and Equation (4.46) imply that

$$\begin{aligned}\widetilde{H}_\infty\left(X_g \mid \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \left\{(W')_i^{j-1}\right\}_{i\in\{g\}\cup\mathcal{I}\setminus\mathcal{I}_j}, \mathcal{H}''_{j-1}\right) &\geq \widetilde{H}_\infty\left(X_g \mid (W')_g^{j-1}, \mathcal{H}''_{j-1}\right) - |\mathcal{I}|\cdot h \\ &\geq \ell - 5htj + 2ht + h. \end{aligned} \tag{4.47}$$

We apply Lemma 4.6 to the (X,Y)-history $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$, $\left\{(W')_i^{j-1} \mid i \in \{g\}\cup\mathcal{I}\setminus\mathcal{I}_j\right\}$, $\mathcal{H}''_{j-1}$, with $P = (A')_g^{j-1}$, $M = X_g$ and the extractor Ext_3. Recall that $W_g^j = \mathsf{Ext}_3\left(X_g, (C')_g^{j-1}\right)$. Since $g \notin S_j$, we have that $(C')_g^{j-1} = (A')_g^{j-1}$. Thus, Lemma 4.6 implies that

$$\left(W_g^j, (A')_g^{j-1}, \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \left\{(W')_i^{j-1}\right\}_{i\in\{g\}\cup\mathcal{I}\setminus\mathcal{I}_j}, \mathcal{H}''_{j-1}\right) \approx_{2\varepsilon_{j-1}+12\varepsilon'} (U_h, \cdot)\,.$$

We note that Lemma 4.6 is applicable since $(A')_g^{j-1}$ is a deterministic function of Y and $(W')_g^{j-1}$, which is contained in the history to which we apply the lemma. Furthermore, Equation (4.2) of Lemma 4.6 follows by Equation (4.47).

We apply Lemma 4.7 to the (X,Y)-history $(A')_g^{j-1}$, $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$, $\left\{(W')_i^{j-1} \mid i \in \{g\}\cup\mathcal{I}\setminus\mathcal{I}_j\right\}$, $\mathcal{H}''_{j-1}$, with $P = W_g^j$ and $J = \left\{(C')_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_j\right\}$. Lemma 4.7 is applicable since W_g^j is a deterministic function of X and $(C')_g^{j-1}$. Recall that $(C')_g^{j-1} = (A')_g^{j-1}$ as $g \notin S_j$, and so $(C')_g^{j-1}$ is contained in the history to which we apply the lemma. Moreover, all random variables $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_j\right\}$ are deterministic functions of Y and $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_j\right\}$ and these random variables are contained in the history to which we apply the lemma. Therefore, Lemma 4.7 implies that

$$\left(W_g^j, \mathcal{H}_j\right) \approx_{2\varepsilon_{j-1}+12\varepsilon'} (U_h, \cdot)\,,$$

where

$$\mathcal{H}_j = \left(\left\{(C')_i^{j-1}\right\}_{i\in\mathcal{I}\setminus\mathcal{I}_j}, (A')_g^{j-1}, \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \left\{(W')_i^{j-1}\right\}_{i\in\{g\}\cup\mathcal{I}\setminus\mathcal{I}_j}, \mathcal{H}''_{j-1}\right).$$

This proves the third item of the claim. The first item of the claim holds since $\mathcal{H}_j$ contains $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$. To see that the second item follows, recall that $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}_j\right\}$ are contained in $\mathcal{H}''_{j-1}$. By the last application of Lemma 4.7, the random variables $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_j\right\}$ are also contained in $\mathcal{H}_j$. Finally, since $g \notin S_j$ we have that $(C')_g^{j-1} = (A')_g^{j-1}$, which is also contained in $\mathcal{H}_j$. Thus, $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}\cup\{g\}\right\}$ are all contained in $\mathcal{H}_j$, and the second item of the claim holds.

The fourth item follows by Equation (4.47), Lemma 2.6 and the fact that $(A')_g^{j-1}$ is a deterministic function of Y conditioned on the fixing of $(W')_g^{j-1}$, and $\left\{(C')_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_j\right\}$ are deterministic functions of Y conditioned on the fixing of $\left\{(W')_i^{j-1} \mid i \in \mathcal{I}\setminus\mathcal{I}_j\right\}$. The fifth item follows since

$$\begin{aligned}\widetilde{H}_\infty\left(Y \mid \mathcal{H}_j\right) &\geq \widetilde{H}_\infty\left(Y \mid \left\{W_i^j\right\}_{i\in\mathcal{I}_j}, \left\{(W')_i^{j-1}\right\}_{i\in\{g\}\cup\mathcal{I}\setminus\mathcal{I}_j}, \mathcal{H}''_{j-1}\right) - (1+|\mathcal{I}\setminus\mathcal{I}_j|)\cdot a \\ &= \widetilde{H}_\infty\left(Y \mid \mathcal{H}''_{j-1}\right) - (1+|\mathcal{I}\setminus\mathcal{I}_j|)\cdot a \\ &\geq k - 5atj + at + a,\end{aligned}$$

where the first inequality follows by Lemma 2.3. The second equality follows by Lemma 2.6 and the fact that conditioned on any fixing of $\mathcal{H}''_{j-1}$, the random variables $\left\{W_i^j \mid i \in \mathcal{I}_j\right\}$, $\left\{(W')_i^{j-1} \mid i \in \{g\} \cup \mathcal{I} \setminus \mathcal{I}_j\right\}$ are deterministic functions of X. The third inequality follows by Equation (4.45).

This concludes the proof for the case $g \notin S_j$. The proof of the claim then follows. □

By Claim 4.10.2 applied with $j = q$, we have that $\mathcal{H} \triangleq \mathcal{H}_q$ contains $\left\{W_i^j \mid i \in \mathcal{I}\right\}$ as $\mathcal{I} = \mathcal{I}_q$. Furthermore,

- $\left(W_g^q, \mathcal{H}\right) \approx_\varepsilon (U_h, \mathcal{H})$. This follows since

$$\varepsilon_q \leq (2^q - 1) \cdot 16 \cdot \varepsilon' \leq 32r \cdot \varepsilon' \leq \varepsilon,$$

  where the second inequality follows since $q = \lceil \log_2 r \rceil$, and the last inequality holds by our choice of ε'.

- $\widetilde{H}_\infty(X_g \mid \mathcal{H}) \geq \ell - 5htq \geq 0.9\ell$.

- $\widetilde{H}_\infty(Y \mid \mathcal{H}) \geq k - 5atq \geq 0.9k$.

Since $\bar{W} = W^q$, the first three items of the theorem follows. As for the fourth item, recall that $\bar{W}_g = W_g^q = \mathsf{Ext}_3\left(X_g, (C')_g^{q-1}\right)$. The second item of the claim states that $(C')_g^{q-1}$ is contained in $\mathcal{H}_q$, and so W_g^q is a deterministic function of X and $\mathcal{H}$, as stated. This concludes the proof of the theorem. □

4.8 Mergers with Weak-Seeds

In this section we prove Theorem 4.3 using Theorem 4.7. We start by giving a formal definition of mergers with weak-seeds using the definition of somewhere-random sources. We further define strong mergers with weak-seeds. We then give a complete and formal restatement of Theorem 4.3 and present its proof.

Definition 4.11 (Mergers with weak-seeds)**.** *A function*

$$\mathsf{Merg}\colon \left(\{0,1\}^\ell\right)^r \times \{0,1\}^n \to \{0,1\}^m$$

is called a merger with weak-seeds *for entropy k, with error ε, if the following holds. For any $r \times \ell$ somewhere-random source X and an independent (n,k)-weak-source Y, it holds that*

$$\mathsf{Merg}(X, Y) \approx_\varepsilon U_m.$$

We say that Merg *is* strong *if*

$$(\mathsf{Merg}(X, Y), Y) \approx_\varepsilon (U_m, Y).$$

Theorem 4.8. *For all integers n, r and for any $\varepsilon > 0$, there exists a* $\mathrm{poly}(n, r, \log(1/\varepsilon))$*-time computable strong merger with weak-seeds for entropy k, with error ε,*

$$\mathsf{Merg}\colon \left(\{0,1\}^{\ell}\right)^{r} \times \{0,1\}^{n} \to \{0,1\}^{m},$$

with

$$\begin{aligned}
\ell &= \Theta\left(r^2 \cdot \log(r) \cdot \log\left(\frac{nr}{\varepsilon}\right)\right),\\
k &= \Omega\left(r \cdot \log(r) \cdot \log\left(\frac{r \cdot \log n}{\varepsilon}\right)\right),\\
m &= \ell/(2r).
\end{aligned}$$

Before proving the theorem, we remark that if one is willing to output $\Omega(\ell/(r \log r))$ bits rather than $\Omega(\ell/r)$, one can construct a merger with weak-seeds using the LCB from Theorem 4.7 in a slightly simpler way. Indeed, one can compute $W = \mathsf{LCB}(X, Y)$ with $t = r$ and then output $\mathsf{Merg}(X, Y) = \oplus_{i=1}^{r} W_i$. That is, one can skip the extra "round" that has the purpose of increasing the output length.

Proof of Theorem 4.8. We first describe the construction of Merg, and then turn to the analysis. To this end, we need the following building blocks.

- Let $\mathsf{LCB}\colon \left(\{0,1\}^{\ell}\right)^{r} \times \{0,1\}^{n} \to \left(\{0,1\}^{h}\right)^{r}$ be the t-LCB from Theorem 4.7, set with $t = r$ and error ε. Note that the hypothesis of Theorem 4.7 is met by our choice of ℓ, k. Furthermore, by Theorem 4.7, $h = \Theta(r \cdot \log(nr/\varepsilon))$.
- Set $s = \Theta(\log(\ell/\varepsilon)) = \Theta(\log(r \cdot \log(n)/\varepsilon))$. Let $\mathsf{Ext}_4\colon \{0,1\}^{n} \times \{0,1\}^{h} \to \{0,1\}^{s}$ be the strong seeded extractor from Theorem 2.2 for entropy $2s$, with error ε. Note that $h = \Omega(\log(n/\varepsilon))$, and so a seed of length h suffices for Theorem 2.2.
- Let $\mathsf{Ext}_5\colon \{0,1\}^{\ell} \times \{0,1\}^{s} \to \{0,1\}^{m}$ be the extractor from Theorem 2.2 for entropy $2m$, set to extract with error ε. Note that s was chosen so that a seed of length s suffices for Theorem 2.2.

The function Merg is defined as follows. First, we compute the $r \times h$ matrix $W = \mathsf{LCB}(X, Y)$. For each $i \in [r]$, we compute

$$\begin{aligned}
Z_i &= \mathsf{Ext}_4(Y, W_i),\\
T_i &= \mathsf{Ext}_5(X_i, Z_i).
\end{aligned}$$

The output of $\mathsf{Merg}(X, Y)$ is then defined to be

$$\mathsf{Merg}(X, Y) = \bigoplus_{i=1}^{r} T_i\,.$$

We now turn to the analysis. Let $g \in [r]$ be such that X_g is uniformly distributed. By Theorem 4.7, applied with $\mathcal{I} = [r] \setminus \{g\}$, there exists an (X, Y)-history $\mathcal{H}$ that contains $\{W_i \mid i \in [r] \setminus \{g\}\}$, such that the following holds:

- $(W_g, \mathcal{H}) \approx_\varepsilon (U_h, \mathcal{H})$.
- $\widetilde{H}_\infty(X_g \mid \mathcal{H}) \geq 0.9 \cdot \ell$.
- $\widetilde{H}_\infty(Y \mid \mathcal{H}) \geq 0.9 \cdot k$.
- W_g is a deterministic function of X and $\mathcal{H}$.

We apply Lemma 4.7 to the (X, Y)-history $\mathcal{H}$ with $P = W_g$ and $J = \{Z_i \mid i \in [r] \setminus \{g\}\}$. Lemma 4.7 is applicable since W_g is a deterministic function of X and $\mathcal{H}$. Moreover, since $Z_i = \mathsf{Ext}_4(Y, W_i)$ and since $\{W_i \mid i \in [r] \setminus \{g\}\}$ are all contained in $\mathcal{H}$, the random variables $\{Z_i \mid i \in [r] \setminus \{g\}\}$ are deterministic functions of Y and $\mathcal{H}$. Therefore, Lemma 4.7 implies that

$$\left(W_g, \{Z_i\}_{i \in [r] \setminus \{g\}}, \mathcal{H}\right) \approx_\varepsilon (U_h, \cdot).$$

In terms of entropy, by Lemma 2.3, we have that

$$\widetilde{H}_\infty\left(Y \mid \{Z_i\}_{i \in [r] \setminus \{g\}}, \mathcal{H}\right) \geq 0.9k - (r-1)s \geq 2s + \log(1/\varepsilon). \tag{4.48}$$

We apply Lemma 4.6 to the (X, Y)-history $\{Z_i \mid i \in [r] \setminus \{g\}\}, \mathcal{H}$ with $P = W_g$, $M = Y$ and the extractor Ext_4. Lemma 4.6 is applicable since W_g is a deterministic function of X and $\mathcal{H}$. Furthermore, Equation (4.2) of Lemma 4.6 follows by Equation (4.48). Since $Z_g = \mathsf{Ext}_4(Y, W_g)$, we have that

$$\left(Z_g, W_g, \{Z_i\}_{i \in [r] \setminus \{g\}}, \mathcal{H}\right) \approx_{3\varepsilon} (U_s, \cdot).$$

In terms of entropy,

$$\begin{aligned}
\widetilde{H}_\infty\left(X_g \mid W_g, \{Z_i\}_{i \in [r] \setminus \{g\}}, \mathcal{H}\right) &\geq \widetilde{H}_\infty\left(X_g \mid \{Z_i\}_{i \in [r] \setminus \{g\}}, \mathcal{H}\right) - h \\
&= \widetilde{H}_\infty(X_g \mid \mathcal{H}) - h \\
&\geq 0.9\ell - h,
\end{aligned} \tag{4.49}$$

where the first inequality follows by Lemma 2.3 and the fact that $|W_g| = h$. The second equality follows by Lemma 2.6 and since conditioned on the fixing of $\mathcal{H}$, the random variables $\{Z_i \mid i \in [r] \setminus \{g\}\}$ are all deterministic functions of Y.

Next we apply Lemma 4.7 to the (X, Y)-history W_g, $\{Z_i \mid i \in [r] \setminus \{g\}\}$, $\mathcal{H}$, with $P = Z_g$ and $J = \{T_i \mid i \in [r] \setminus \{g\}\}$. Lemma 4.7 is applicable since Z_g is a deterministic function of Y and W_g, and the latter is contained in the history to which we apply the lemma. On the other hand, $\{T_i \mid i \in [r] \setminus \{g\}\}$ are all deterministic functions of X and $\{Z_i \mid i \in [r] \setminus \{g\}\}$, all of which are contained in the history to which we apply the lemma. Lemma 4.7 implies that

$$\left(Z_g, \{T_i\}_{i \in [r] \setminus \{g\}}, W_g, \{Z_i\}_{i \in [r] \setminus \{g\}}, \mathcal{H}\right) \approx_{3\varepsilon} (U_s, \cdot).$$

Equation (4.49) together with Lemma 2.3 imply that

$$\widetilde{H}_\infty\left(X_g \mid \{T_i\}_{i\in[r]\setminus\{g\}}, W_g, \{Z_i\}_{i\in[r]\setminus\{g\}}, \mathcal{H}\right) \geq 0.9\ell - h - (r-1)m \geq 2m + \log(1/\varepsilon). \quad (4.50)$$

Recall that $T_g = \mathsf{Ext}_5(X_g, Z_g)$. We apply Lemma 4.6 to the (X,Y)-history $\{T_i \mid i \in [r]\setminus\{g\}\}$, W_g, $\{Z_i \mid i \in [r]\setminus\{g\}\}$, $\mathcal{H}$, with $P = Z_g$ and $M = X_g$. The application of Lemma 4.6 is valid since Z_g is a deterministic function of Y and W_g, and the latter is contained in the history to which we apply the lemma. Furthermore, Equation (4.2) of Lemma 4.6 holds by Equation (4.50). Hence, Lemma 4.6 implies that

$$\left(T_g, Z_g, \{T_i\}_{i\in[r]\setminus\{g\}}, W_g, \{Z_i\}_{i\in[r]\setminus\{g\}}, \mathcal{H}\right) \approx_{5\varepsilon} (U_m, \cdot).$$

We apply Lemma 4.7 to the (X,Y)-history Z_g, $\{T_i \mid i \in [r]\setminus\{g\}\}$, W_g, $\{Z_i \mid i \in [r]\setminus\{g\}\}$, $\mathcal{H}$ with $P = T_g$ and $J = Y$. This application of Lemma 4.7 is valid since T_g is a deterministic function of X and Z_g, and the latter is contained in the history to which we apply the lemma. Thus,

$$\left(T_g, Y, Z_g, \{T_i\}_{i\in[r]\setminus\{g\}}, W_g, \{Z_i\}_{i\in[r]\setminus\{g\}}, \mathcal{H}\right) \approx_{5\varepsilon} (U_m, \cdot).$$

Lemma 2.12 then implies that

$$\left(T_g, \{T_i\}_{i\in[r]\setminus\{g\}}, Y\right) \approx_{5\varepsilon} (U_m, \cdot).$$

Since $\mathsf{Merg}(X,Y) = \oplus_{i=1}^{r} T_i$, it holds that

$$\left(\mathsf{Merg}\,(X,Y), \{T_i\}_{i\in[r]\setminus\{g\}}, Y\right) \approx_{5\varepsilon} (U_m, \cdot).$$

By applying Lemma 2.12 again, one get that

$$(\mathsf{Merg}\,(X,Y), Y) \approx_{5\varepsilon} (U_m, \cdot).$$

Note further that the error can be reduced from 5ε to ε without affecting the theorem's hypothesis. This concludes the proof of the theorem. □

4.9 Three-Source Extractors with a Double-Logarithmic Entropy Source

In this section we prove Theorem 4.2. We give a formal restatement of the theorem here that accounts for the dependence in the error ε, as well as the strongness properties of the extractor.

Theorem 4.9. *There exist universal constants $0 < \alpha < 1 < c$ such that the following holds. For any integer n, $\varepsilon > 0$ and for any $\delta > \Omega((\log(n/\varepsilon)/n)^{\alpha})$, there exists a $\mathrm{poly}(n, \log(1/\varepsilon))$-time computable three-source extractor $\mathsf{3Ext}\colon (\{0,1\}^n)^3 \to \{0,1\}^m$, with error ε, that is strong in $\{1,3\}$ and in $\{2,3\}$, for entropies*

$$\begin{aligned} k_1 &= \delta n, \\ k_2 &= \Omega\left((1/\delta)^{3c} \cdot \log\left(n/\varepsilon\right)\right), \\ k_3 &= \Omega\left((1/\delta)^{2c} \cdot \log\left(\frac{\log n}{\varepsilon}\right)\right). \end{aligned}$$

The number of output bits is $m = \Omega\left((1/\delta)^c \cdot \log(n/\varepsilon)\right)$.

For the proof of Theorem 4.9, we make use of the following two-source extractor of Raz [Raz05].

Theorem 4.10 ([Raz05])**.** *For all integers n_1, n_2, b_1, b_2, such that*

$$\begin{aligned} n_1 &\geq 6\log n_1 + 2\log n_2, \\ b_1 &\geq 0.6n_1 + 3\log n_1 + \log n_2, \\ b_2 &\geq 5\log n_1, \\ m &\leq \min\left(n_1/80, b_2/400\right) - 1, \end{aligned}$$

there exists an efficiently-computable function $\mathsf{Raz}\colon \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^m$ with the following property. For any (n_1, b_1)-weak-source X, and an independent (n_2, b_2)-weak-source Y,

$$\begin{aligned} (\mathsf{Raz}(X,Y), X) &\approx_{\varepsilon} (U_m, X), \\ (\mathsf{Raz}(X,Y), Y) &\approx_{\varepsilon} (U_m, Y), \end{aligned}$$

where $\varepsilon = 2^{-1.5m}$.

We also make use of the following construction of a somewhere-condenser.

Theorem 4.11 ([Raz05, BKS+05, Zuc07])**.** *There exist universal constants $c_1, c_2 > 0$ such that the following holds. For every $\delta > 0$, there exists an efficiently-computable function $\mathsf{Cond}\colon \{0,1\}^n \to (\{0,1\}^{\ell})^r$, where the output is interpreted as an $r \times \ell$ matrix, with $r = \Theta((1/\delta)^{c_1})$ rows and $\ell = \Theta(n \cdot \delta^{c_2})$ columns. If X is an $(n, \delta n)$-weak-source, then $\mathsf{Cond}(X)$ is $2^{-\Omega(\delta^2 n)}$-close to a convex combination of distributions, each of which has some row with min-entropy rate 0.9.*

Proof of Theorem 4.9. We start by describing the construction of $\mathsf{3Ext}$, and then turn to the analysis. For the construction of $\mathsf{3Ext}$ we need the following building blocks:

- Let $\mathsf{Cond}\colon \{0,1\}^n \to \left(\{0,1\}^{\ell}\right)^r$ be the somewhere-condenser from Theorem 4.11. By Theorem 4.11, $r = \Theta((1/\delta)^{c_1})$ and $\ell = \Theta(n \cdot \delta^{c_2})$, where $c_1, c_2 > 1$ are the universal constants from Theorem 4.11. We set $c = c_1$.

- Let $\mathsf{Raz}\colon \{0,1\}^{\ell} \times \{0,1\}^{n} \to \{0,1\}^{t}$ be the extractor from Theorem 4.10, set to extract $t = \Theta(r^2 \cdot \log(r) \cdot \log(nr/\varepsilon))$ bits.
- Let $\mathsf{Merg}\colon (\{0,1\}^{t})^{r} \to \{0,1\}^{m}$ be the merger with weak-seeds from Theorem 4.8, set with error ε and output length $m = t/(2r) = \Theta(r \cdot \log(r) \cdot \log(nr/\varepsilon))$.

Given these building blocks, the construction of $\mathsf{3Ext}$ is as follows. Let X_1, X_2, X_3 be n-bit sources with entropies k_1, k_2, k_3, respectively. We first compute $\mathsf{Cond}(X_1)$, which is an $r \times \ell$ matrix R. Secondly, for each $i \in [r]$, we compute $S_i = \mathsf{Raz}(R_i, X_2)$. We stack $S_1, \ldots, S_r$ in an $r \times t$ matrix S. The output is then $\mathsf{3Ext}(X_1, X_2, X_3) = \mathsf{Merg}(S, X_3)$.

We now turn to the analysis. We prove that the extractor is strong in $\{2,3\}$. Since Raz is strong in both of its sources, a similar argument can be used to show that the extractor is also strong in $\{1,3\}$. By Theorem 4.11, since X_1 is an $(n, \delta n)$-weak-source, the matrix R is $2^{-\Omega(\delta^2 n)}$-close to a convex combination of distributions, each of which has some row with min-entropy rate 0.9. Therefore, we may assume that R is $2^{-\Omega(\delta^2 n)}$-close to a random variable R', such that there exists $g \in [r]$ where R'_g has min-entropy rate 0.9. Note that by taking the universal constant α to be smaller than $1/2$, we get that $2^{-\Omega(\delta^2 n)} \leq \varepsilon$.

One can easily verify that the hypothesis of Theorem 4.10 is met assuming

$$\delta \geq \Omega\left(\left(\frac{\log(n/\varepsilon)}{n}\right)^{1/(3c_1+c_2)}\right),$$

which holds by taking the universal constant $\alpha = 1/(3c_1 + c_2)$ (note that $\alpha \leq 1/2$). By Theorem 4.10, and since $2^{-1.5t} \leq \varepsilon$,

$$\big(\mathsf{Raz}(R'_g, X_2), X_2\big) \approx_{\varepsilon} (U_t, X_2).$$

Since (R_g, R'_g) and X_2 are independent and since $\mathsf{SD}(R, R') \leq \varepsilon$, Lemma 2.8 (applied to the random function $f(Z) = (\mathsf{Raz}(Z_g, X_2), X_2)$, where X_2 is the internal randomness of f) implies that

$$(S_g, X_2) = (\mathsf{Raz}(R_g, X_2), X_2) \approx_{2\varepsilon} (U_t, X_2).$$

Thus, by Markov's inequality, with probability at least $1 - \sqrt{\varepsilon}$ over the fixing $X_2 = x_2$, it holds that $(S_g \mid X_2 = x_2)$ is $2\sqrt{\varepsilon}$-close to uniform. We condition on the event $X_2 = x_2$ for such x_2. Lemma 2.15 then implies that $S \approx_{2\sqrt{\varepsilon}} S'$, where S' is a somewhere-random source.

We apply the merger from Theorem 4.8 to S' and the weak-source X_3. By the choice of t, and since $k_3 \geq \Omega(r \cdot \log(r) \cdot \log(r \log(n)/\varepsilon))$, the hypothesis of Theorem 4.8 is met, and so Theorem 4.8 implies that

$$(\mathsf{Merg}(S', X_3), X_3) \approx_{\varepsilon} (U_m, X_3)\,. \tag{4.51}$$

Since $S \approx_{2\sqrt{\varepsilon}} S'$, and since (S, S') and X_3 are independent, Lemma 2.8 (applied to the random function $f(Z) = (\mathsf{Merg}(Z, X_3), X_3)$, where X_3 is the internal randomness of f), implies that

$$(\mathsf{Merg}(S, X_3), X_3) \approx_{2\sqrt{\varepsilon}} (\mathsf{Merg}(S', X_3), X_3)\,, \tag{4.52}$$

and so by Equation (4.51) and Equation (4.52) we have that

$$(\mathsf{Merg}(S, X_3), X_3) \approx_{3\sqrt{\varepsilon}} (U_m, X_3)\,.$$

By Markov's inequality, with probability at least $1-\varepsilon^{1/4}$ over the further fixing of $X_3 = x_3$, it holds that $\mathsf{Merg}(S, x_3)$ is $3\varepsilon^{1/4}$-close to uniform. Thus, we have that

$$(\mathsf{3Ext}(X_1, X_2, X_3), X_2, X_3) = (\mathsf{Merg}(S, X_3), X_2, X_3) \approx_{O(\varepsilon^{1/4})} (U_m, X_2, X_3)\,.$$

Note that the error can be reduced from $O(\varepsilon^{1/4})$ to ε without affecting the theorem statement. This concludes the proof of the theorem.

□

4.10 Two-Source Non-Malleable Extractors

In this section we construct a two-source t-non-malleable extractor using our LCBs from Theorem 4.7. We start by giving a formal definition of two-source t-non-malleable extractors and a formal restatement of Theorem 4.4.

Definition 4.12 (Two-source t-non-malleable extractors)**.** *A function* $f\colon (\{0,1\}^n)^2 \times \{0,1\}^d \to \{0,1\}^m$ *is called a* two-source t-non-malleable extractor *for entropies* k_1, k_2*, with error* ε*, if for any* (n, k_1)*-weak-source* X *and an independent* (n, k_2)*-weak-source* Y*, the following holds. For any* t *functions* $A_1, \ldots, A_t\colon \{0,1\}^d \to \{0,1\}^d$*, where each* A_i *has no fixed points (that is, for all* $i \in [t]$ *and* $s \in \{0,1\}^d$*,* $A_i(s) \neq s$*), it holds that*

$$\left(f(X, Y, S), S, \{f(X, Y, A_i(S))\}_{i=1}^t\right) \approx_\varepsilon (U_m, \cdot)\,.$$

We say that f *is* strong in the first-source *if*

$$\left(f(X, Y, S), X, S, \{f(X, Y, A_i(S))\}_{i=1}^t\right) \approx_\varepsilon (U_m, \cdot)\,.$$

Similarly, we say that f *is* strong in the second source *if*

$$\left(f(X, Y, S), Y, S, \{f(X, Y, A_i(S))\}_{i=1}^t\right) \approx_\varepsilon (U_m, \cdot)\,.$$

Theorem 4.12. *For all integers* n, t *and for any* $\varepsilon > 0$*, there exists a* $\mathrm{poly}(n, t, \log(1/\varepsilon))$*-time computable two-source* t*-non-malleable extractor*

$$\mathsf{2NMExt}\colon (\{0,1\}^n)^2 \times \{0,1\}^d \to \{0,1\}^h,$$

with $h = \Theta(t \cdot \log(n/\varepsilon))$ *output bits and error* ε*, for entropies*

$$k_1 = \Omega\left(t^2 \cdot \log^2(n/\varepsilon)\right),$$
$$k_2 = \Omega\left(t \cdot \log^2(n/\varepsilon)\right),$$

with seed length $d = O(\log(n/\varepsilon))$*. Moreover,* 2NMExt *is strong in the second source.*

For the proof of Theorem 4.12 we introduce and make use of the following type of random sources.

Definition 4.13 (α-where random sources)**.** *Let $\alpha \in [0,1]$. A random variable X in the form of an $r \times \ell$ matrix is called an* α-where random source*, if α-fraction of the rows of X are good. That is, if there exists $S \subseteq [r]$ with $|S| \geq \alpha r$, such that for every $i \in S$ it holds that X_i is uniformly distributed.*

For the proof of Theorem 4.12 we make use of the following result due to Zuckerman [Zuc97].

Lemma 4.14. *Let $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a seeded extractor for entropy k with error ε. Let $B \subseteq \{0,1\}^m$ be a set with size $|B| = \beta \cdot 2^m$. Define*

$$\mathsf{OverHit}_{\mathsf{Ext}}(B) = \left\{ x \in \{0,1\}^n \;\middle|\; \Pr[\mathsf{Ext}(x, U_d) \in B] > \beta + \varepsilon \right\}.$$

Then, $|\mathsf{OverHit}_{\mathsf{Ext}}(B)| \leq 2^k$.

We now turn to prove Theorem 4.12.

Proof of Theorem 4.12. We first describe the construction of $\mathsf{2NMExt}$ and then turn to the analysis. For the construction we make use of the following building blocks.

- Let $\mathsf{Ext}_1\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^D$ be the strong seeded extractor from Theorem 2.2 set with error ε. By Theorem 2.2, Ext_1 is an extractor for entropy $2D$ with $d = O(\log(n/\varepsilon))$. We denote $r = 2^d = \mathrm{poly}(n/\varepsilon)$ and identify $[r]$ with $\{0,1\}^d$. In particular, we think of elements of $[r]$ as seeds for Ext_1.
- Set $\varepsilon' = \varepsilon^2/r$ and $\ell = O(t^2 \cdot \log(nr/\varepsilon) \cdot \log r) = O(t^2 \cdot \log^2(n/\varepsilon))$ and let $\mathsf{Ext}_2\colon \{0,1\}^n \times \{0,1\}^D \to \{0,1\}^\ell$ be the strong seeded extractor from Theorem 2.2 set with error ε'. By Theorem 2.2, Ext_2 is an extractor for entropy 2ℓ with $D = O(\log(n/\varepsilon')) = O(\log(n/\varepsilon))$.
- Let $\mathsf{LCB}\colon (\{0,1\}^\ell)^r \times \{0,1\}^n \to (\{0,1\}^m)^r$ be the $(t+1)$-LCB from Theorem 4.7 set with error ε. By Theorem 4.7, $m = O(t \cdot \log(n/\varepsilon))$.

We now describe the construction of $\mathsf{2NMExt}$. First, we define the $r \times D$ matrix $\bar{Y}$ as follows. For $s \in \{0,1\}^d$, row s of $\bar{Y}$ is defined by $\mathsf{Ext}_1(Y, s)$. Next, we define the $r \times \ell$ matrix $\bar{X}$ as follows. For $s \in \{0,1\}^d$, row s of $\bar{X}$ is defined by $\bar{X}_s = \mathsf{Ext}_2(X, \bar{Y}_s)$. We define the $r \times m$ matrix Z by $Z = \mathsf{LCB}(\bar{X}, Y)$. Finally, we define $\mathsf{2NMExt}(X, Y, S) = Z_S$.

We turn to the analysis, starting with the running-time. We note that given S, one can compute row S of $\bar{Y}$, $\bar{X}$ and Z without resorting to computing other rows of these matrices, and do so in time $\mathrm{poly}(n, \log(1/\varepsilon))$ (even though the number of rows is polynomial in $1/\varepsilon$). Thus, the running-time for computing $\mathsf{2NMExt}$ on inputs (X, Y, S) is $\mathrm{poly}(n, \log(1/\varepsilon))$.

Now, let

$$B = \left\{ y \in \{0,1\}^n \;\middle|\; (\bar{X} \mid Y = y) \text{ is not } \varepsilon\text{-close to a } (1-2\varepsilon)\text{-where random source} \right\}.$$

Claim 4.14.1. $|B| \leq 2^{2D}$.

Proof of Claim 4.14.1. Since Ext_2 is a strong seeded extractor for entropy 2ℓ with error ε', and since $H_\infty(X) \geq 2\ell$, it holds that

$$(\mathsf{Ext}_2(X,S),S) \approx_{\varepsilon'} (U_\ell,S)\,,$$

where S is uniformly distributed over $\{0,1\}^D$ and is independent of X. Thus, by Markov's inequality, with probability at least ε over $s \sim U_D$, it holds that $\mathsf{Ext}_2(X,s)$ is $(\varepsilon'/\varepsilon)$-close to uniform. We define

$$\mathsf{BadSeeds}(X) = \left\{s \in \{0,1\}^D \mid \mathsf{SD}(\mathsf{Ext}_2(X,s),U_m) > \varepsilon'/\varepsilon\right\}.$$

By the above we have that $|\mathsf{BadSeeds}(X)| \leq \varepsilon \cdot 2^D$. Let

$$B' = \left\{y \in \{0,1\}^n \;\middle|\; \Pr\left[\mathsf{Ext}_1(y,U_d) \in \mathsf{BadSeeds}(X)\right] \geq 2\varepsilon\right\}.$$

The proof of the claim will follow by showing that $B \subseteq B'$ and that $|B'| \leq 2^{2D}$. Since Ext_1 is an extractor with error ε and since $|\mathsf{BadSeeds}(X)| \leq \varepsilon \cdot 2^D$, the second assertion, namely $|B'| \leq 2^{2D}$, follows by Lemma 4.14. As for the first assertion, we will show that $y \notin B' \implies y \notin B$. If $y \notin B'$ then conditioned on $Y = y$, at least $1-2\varepsilon$ fraction of the rows of $\bar{Y}$ are not contained in $\mathsf{BadSeeds}(X)$. Thus, conditioned on $Y = y$, at least $1-2\varepsilon$ fraction of $\bar{X}$'s rows are ε'/ε-close to uniform. Thus, by Lemma 2.15 applied $(1-2\varepsilon)r$ times, $\bar{X}$ is $r \cdot \varepsilon'/\varepsilon$-close to a $(1-2\varepsilon)$-where random source. Thus, $y \notin B$. The second assertion and the claim then follows by our choice of ε'. □

By Claim 4.14.1 and since $H_\infty(Y) \geq 2D + \log(1/\varepsilon)$ it follows that

$$\Pr[Y \in B] = \sum_{y \in B} \Pr[Y = b] \leq 2^{2D} \cdot 2^{-H_\infty(Y)} \leq \varepsilon.$$

Therefore, there exists a random variable Y' such that $\Pr[Y' \in B] = 0$ and $Y \approx_\varepsilon Y'$. Moreover, $H_\infty(Y') \geq H_\infty(Y) - 1$. We continue with the analysis as if we were given a sample from Y' rather than from Y, and then aggregate the ε statistical distance between Y and Y' to the total error. For the sake of readability we abuse notation and denote Y' by Y. We continue by proving the following claim.

Claim 4.14.2. *For any $\delta > 0$ and $J \subseteq [r]$, $|J| \leq t+1$, it holds that*

$$\Pr_v\left[H_\infty(Y \mid \bar{Y}_J = v) < H_\infty(Y) - (t+1)D - \log(1/\delta)\right] \leq \delta.$$

Proof. As each row of $\bar{Y}$ consists of D bits, Lemma 2.3 implies that $\widetilde{H}_\infty(Y \mid \bar{Y}_J) \geq H_\infty(Y) - (t+1)D$. Thus, by Lemma 2.4, except with probability δ over the fixing $\bar{Y}_J = v$ it holds that $H_\infty(Y \mid \bar{Y}_J = v) \geq H_\infty(Y) - (t+1)D - \log(1/\delta)$. □

For $g \in \{0,1\}^d$, let $I_g = \{A_1(g), \ldots, A_t(g)\}$. Since the A_i's have no fixed points, we have that $g \notin I_g$. Let $J_g = I_g \cup \{g\}$. By the above, conditioned on any fixing $Y = y$, $\bar{X}$ is ε-close to a $(1-2\varepsilon)$-where random source which we denote by $\bar{X}^y$. Thus, for $1-2\varepsilon$ fraction of $g \in \{0,1\}^d$ it holds that $\bar{X}^y_g$ is uniform. Moreover, by Claim 4.14.2 we have that with probability at least $1-\delta$ over the fixing of $\bar{Y}_{J_g}$, the min-entropy of Y is at least $H_\infty(Y) - (t+1)D - \log(1/\delta)$.

We would like to apply Theorem 4.7 to $\bar{X}$ and Y with rows g, I_g. However, the theorem is not directly applicable since $\bar{X}$ and Y are not necessarily independent. Nevertheless, we make the observation that for the proof of Theorem 4.7 to go through, it is enough that the relevant rows of $\bar{X}$ (which in our case are $J_g = I_g \cup \{g\}$) and Y are independent. This indeed holds when conditioned on the fixing of $\bar{Y}_{J_g}$.

By the above, we have that with probability $1-2\varepsilon$ over $g \sim \{0,1\}^d$ it holds that $\bar{X}^y_g$ is uniform and that for any g, even conditioned on the fixing of $\bar{Y}_{J_g}$, Y still has min-entropy $H_\infty(Y) - (t+1)D - \log(1/\delta)$ except with probability δ. By setting $\delta = \varepsilon/r$ and by our choice of parameters, we have that with probability $1-O(\varepsilon)$ over the fixing of Y, it holds that with probability $1-2\varepsilon$ over $g \sim \{0,1\}^d$, Z_g is $O(\varepsilon)$-close to uniform even conditioned on $\{Z_i \mid i \in I_g\}$. This concludes the proof of the theorem. □

Chapter 5

Zero-Fixing Extractors for Sub-Logarithmic Entropy

5.1 Bit-Fixing Sources

In Chapter 2 we introduced the notion of an (n, k)-weak-source of randomness. One important and well-studied subclass of weak-sources are bit-fixing sources.

Definition 5.1 (Bit-fixing sources)**.** *Let n, k be integers such that $n \geq k$. A random variable X on n bits is called an* (n, k)-bit-fixing source*, if there exists $S \subseteq [n]$ with size $|S| = k$, such that $X|_S$ is uniformly distributed, and each X_i with $i \notin S$ is fixed.*

The problem of extracting randomness from bit-fixing sources was initiated in the works of [Vaz85, BBR85, CGH+85], motivated by applications to fault-tolerance, cryptography and communication complexity. More recently, bit-fixing extractors have found applications to formulae lower bounds [KRT13], and for compression algorithms for "easy" Boolean functions [CKK+13].

The early works on bit-fixing extractors were concentrated on positive and negative results for extracting a truly uniform string. In [CGH+85], it was observed that one can efficiently extract a uniform bit even from $(n, 1)$-bit-fixing sources, simply by XOR-ing all the input bits. In a sharp contrast, it was shown that extracting two jointly uniform bits cannot be done even from $(n, n/3 - 1)$-bit-fixing sources. Given this state of affairs, early works dealt with what we call "the high-entropy regime". Using a relation to error correcting codes, Chor *et al.* [CGH+85] showed how to efficiently extract roughly $n - t \cdot \log_2(n/t)$ truly uniform output bits from $(n, n - t)$-bit-fixing sources, with $t = o(n)$. The authors complemented this result by an almost matching upper bound of $n - (t/2) \cdot \log_2(n/t)$ on the number of truly uniform output bits one can extract. In the same paper, some results were obtained also for (n, k)-bit-fixing sources, where k is slightly below $n/2$. Further lower bounds for this regime of parameters were obtained by Friedman [Fri92].

These negative results naturally led to study the relaxation, where the output of

the extractor is only required to be close to uniform, in statistical distance.[1] A simple probabilistic argument can be used to show that, computational aspects aside, one can extract $m = k - 2\log(1/\varepsilon) - O(1)$ bits that are ε-close to uniform, from any (n,k)-bit-fixing source, as long as $k \geq \log(n) + 2\log(1/\varepsilon) + O(1)$. For simplicity, in this chapter that covers the paper [CS15], we think of ε as a small constant. Thus, in particular, by allowing for some small constant error $\varepsilon > 0$, one can extract almost all the entropy k from any (n,k)-bit-fixing source, even for k as low as $\log(n) + O(1)$. We call the range $\log n \leq k \leq o(n)$, "the low-entropy regime".

The probabilistic argument mentioned above only yields an existential proof, whereas efficiently computable extractors are far more desired. Kamp and Zuckerman [KZ06] gave the first explicit construction of an (n,k)-bit-fixing extractor, with $k = o(n)$. More precisely, for any constant $\gamma > 0$, an explicit $(n, n^{1/2+\gamma})$-bit-fixing extractor was given, with $\Omega(n^{2\gamma})$ output bits. In a subsequent work, Gabizon, Raz and Shaltiel [GRS06] obtained an explicit $(n, \log^c n)$-bit-fixing extractor, where $c > 1$ is some universal constant. Moreover, the latter extractor outputs $(1-o(1))$-fraction of the entropy, thus getting very close to the parameters of the non-explicit construction obtained by the probabilistic method. Using different techniques, Rao [Rao09b] obtained a bit-fixing extractor with improved dependence on the error ε.

For a vast majority of randomness extraction problems, such as the problem of constructing two-source extractors and affine extractors, a naïve probabilistic argument yields (non-explicit) extractors with essentially optimal parameters. Interestingly, this is not the case for bit-fixing extractors. The first evidence for that comes from the observation mentioned above. Namely, the XOR function is an extractor for $(n,1)$-bit-fixing sources. A result of Kamp and Zuckerman [KZ06] shows that this is not an isolated incident, and in fact, for *any* $k \geq 1$ there is an (explicit and simple) extractor for (n,k)-bit-fixing sources, that outputs $0.5 \cdot \log_2(k) - O(\log\log k)$ random bits that are close to uniform. This result was later improved and simplified by Reshef and Vadhan [RV13], who showed how to output $0.5 \cdot (\log k - \log\log(1/\varepsilon))$ bits. On the other hand, one can show that, with high probability, a random function with a single output bit is constant on some bit-fixing source with entropy, say, $\log(n)/10$. Thus, in this setting, *structured* functions outperform *random* functions, in the sense that the former can extract a logarithmic amount of the entropy from bit-fixing sources with arbitrarily low entropy, whereas the latter are constant, with high probability, on some $(n, \log(n)/10)$-bit-fixing source.

Reshef and Vadhan [RV13] considered k that is sub-logarithmic in n – a regime we call the "very low entropy regime". In [RV13] it is shown that any extractor that is computable by a space-bounded streaming algorithm can output only $O(\log k)$ bits in this regime.

[1] Friedman [Fri92] studied other notions of closeness. Although different measures are of interest, when analyzing extractors, the gold standard measure of closeness between distributions is statistical distance. In this paper we follow the convention, and measure the error of an extractor by the statistical distance of its output to the uniform distribution.

5.2 Our Contribution

Our first result states that when the entropy k is small enough compared to n, one cannot extract more than $0.5 \cdot \log_2(k) + O(1)$ bits from an (n, k)-bit-fixing source, information theoretically. That is, for small enough k, the computational assumption on the extractor imposed in [RV13] can be removed. Note that this negative result is tight as implied by the constructions of [KZ06, RV13].

In fact, the following impossibility result holds also for what we call *zero-fixing sources*. A random variable X is an (n, k)-zero-fixing source if it is an (n, k)-bit-fixing source, where all the fixed bits are set to zero. More formally,

Definition 5.2 (Zero-fixing sources)**.** *Let n, k be integers such that $n \geq k$. A random variable X on n bits is called an* (n, k)-zero-fixing source*, if there exists $S \subseteq [n]$ with size $|S| = k$, such that $X|_S$ is uniformly distributed, and each X_i with $i \notin S$ is fixed to zero.*

Zero-fixing sources are natural as they model bit-fixing sources in which the fixed bits are set to some default value rather than to an arbitrary value. To state the result, we introduce the following notation. The function $\mathsf{Tower}\colon \mathbb{N} \to \mathbb{N}$ is defined as follows: $\mathsf{Tower}(0) = 1$, and for an integer $n \geq 1$, $\mathsf{Tower}(n) = 2^{\mathsf{Tower}(n-1)}$.

Theorem 5.1. *For any integers n, k such that $\mathsf{Tower}(k^{3/2}) < n$, the following holds. Let $\mathsf{Ext}\colon \{0,1\}^n \to \{0,1\}^m$ be an (n, k)-zero-fixing extractor with error ε. If $m > 0.5 \cdot \log_2(k) + O(1)$, then $\varepsilon \geq 0.99$.*

Since the impossibility result stated in Theorem 5.1 holds for zero-fixing sources, it is natural to try and complement it with feasibility results. Using a naïve probabilistic argument, one can prove the existence of an (n, k)-zero-fixing extractor, for any $k \geq \log\log n + \log\log\log n + O(1)$, with $m = k - O(1)$ output bits, where we treat the error ε as constant, for simplicity. Our second result is an almost matching explicit construction.

Theorem 5.2. *For any constant $\mu > 0$, and $n, k \in \mathbb{N}$, such that $k \geq (\log\log n)^{2+\mu}$, there exists an efficiently computable function*

$$\mathsf{ZeroBFExt}\colon \{0,1\}^n \to \{0,1\}^m,$$

where $m = \Omega(k)$, with the following property. For any (n, k)-zero-fixing source X, it holds that $\mathsf{ZeroBFExt}(X)$ is $(2^{-k^{\Omega(1)}} + (k \log n)^{-\Omega(1)})$-close to uniform.

We remark that the techniques used in [GRS06, Rao09b] for the constructions of bit-fixing extractors seem to work only for $k \geq \operatorname{poly}\log n$, even for zero-fixing sources, and new ideas are required so to exploit the extra structure of zero-fixing sources in order to extract $\Omega(k)$ bits from such sources with sub-logarithmic entropy.

As mentioned, Reshef and Vadhan [RV13] proved that for $k = o(\log n)$, any space-bounded streaming algorithm can extract at most $O(\log k)$ bits. The authors left open the problem of whether or not one can extract $\Omega(k)$ bits for $k = o(\log n)$. Theorem 5.1 shows that this is impossible for k which is very small compared to n. Nevertheless, in

the following theorem we answer the open problem of [RV13] positively and show that one can extract $k - O(1)$ bits even when $k = O(\log \log n)$. For simplicity, we state here the theorem for a constant error ε.

Theorem 5.3. *For any integers n, k, and constant $\varepsilon > 0$, such that $k > \log \log n + 2 \log \log \log n + O_\varepsilon(1)$, there exists a function*

$$\mathsf{QuasiBFExt}\colon \{0,1\}^n \to \{0,1\}^m,$$

where $m = k - O_\varepsilon(1)$, with the following property. Let X be an (n,k)-bit-fixing source. Then, $\mathsf{QuasiBFExt}(X)$ is ε-close to uniform. The running-time of evaluating $\mathsf{QuasiBFExt}$ is $n^{O_\varepsilon((\log \log n)^2)}$.

On top of the semi-explicit construction in Theorem 5.3, in [CS15] we give a simpler existential proof for an extractor $\mathsf{QuasiBFExt}$, with parameters as in Theorem 5.3, based on the Lóvasz Local Lemma.

5.3 Proofs Overview

In this section we give an overview for the proofs of Theorem 5.1 and Theorem 5.2. The full proofs of these theorems and of Theorem 5.3 can be found in [CS15]. For the sake of clarity, in this section we allow ourselves to be informal and somewhat imprecise.

5.3.1 Proof overview for Theorem 5.1

To give an overview for the proof of Theorem 5.1, we start by considering a related problem. Instead of proving an upper bound on the number of output bits of an (n,k)-zero-fixing extractor, we prove an upper bound for zero-error dispersers. Generally speaking, a *zero-error disperser* for a class of sources is a function that obtains all outputs, even when restricted to any source in the class. More concretely, an *(n,k)-zero-fixing zero-error disperser* is a function $\mathsf{ZeroErrDisp}\colon \{0,1\}^n \to \{0,1\}^m$, such that for any (n,k)-zero-fixing source X, it holds that $\mathsf{supp}(\mathsf{ZeroErrDisp}(X)) = \{0,1\}^m$. We show that for any such zero-error disperser, if k is small enough compared to n, then $m \leq \log_2(k+1)$. More specifically, we prove that for any integers n, k such that $\mathsf{Tower}(k^2) < n$ and $m = \lfloor \log_2(k+1) \rfloor + 1$, for any function $f\colon \{0,1\}^n \to \{0,1\}^m$, there exists an (n,k)-zero-fixing source, restricted to which f is a symmetric function, i.e., f depends only on the input's weight. In particular, f does not obtain all possible outputs.[2] This implies that if $f\colon \{0,1\}^n \to \{0,1\}^m$ is a (n,k)-zero-fixing zero-error dispersers and $\mathsf{Tower}(k^2) < n$, then $m \leq \log_2(k+1)$.

Given $f\colon \{0,1\}^n \to \{0,1\}^m$, we construct the required source X in a level-by-level fashion, as follows. Trivially, f is symmetric on any $(n,1)$-zero-fixing source, regardless of the value of m. Next, we find an $(n,2)$-zero-fixing source on which f is symmetric.

[2] If $m > \lfloor \log_2(k+1) \rfloor + 1$, then the same result can be obtained by restricting the output to the first $\lfloor \log_2(k+1) \rfloor + 1$ output bits.

By the pigeonhole principle, there exists a set of indices $I_1 \subseteq [n]$, with size $|I_1| \geq n/2^m$, such that $f(e_i) = f(e_j)$ for all $i, j \in I_1$. Here, for an index $i \in [n]$, we denote by e_i the unit vector with 1 at the i'th coordinate. If $n > 2^m$, then $|I_1| \geq 2$, and so there exist two distinct $i, j \in I_1$. Thus, f restricted to the $(n, 2)$-zero-fixing source $\{0, e_i, e_j, e_i + e_j\}$ is symmetric.

We take a further step, and find an $(n, 3)$-zero-fixing source on which f is symmetric. We restrict ourselves to the index set I_1 above, and consider the complete graph with vertex set I_1, where for every two distinct vertices $i, j \in I_1$, the edge connecting them is colored by the color $f(e_i + e_j)$, where we think of $\{0,1\}^m$ as representing 2^m colors. By the multi-color variant of Ramsey theorem, there exists a set $I_2 \subseteq I_1$, of size

$$|I_2| \geq \log(|I_1|)/\mathrm{poly}(2^m),$$

such that the complete graph induced by I_2 is monochromatic. Therefore, if $n > 2^{2^{O(m)}} = 2^{\mathrm{poly}(k)}$, then $|I_2| \geq 3$, and so there exist distinct $i_1, i_2, i_3 \in I_2$ such that

$$f(e_{i_1}) = f(e_{i_2}) = f(e_{i_3}),$$
$$f(e_{i_1}+e_{i_2}) = f(e_{i_1} + e_{i_3}) = f(e_{i_2} + e_{i_3}).$$

Thus, f is symmetric on the $(n, 3)$-zero-fixing source spanned by $\{e_{i_1}, e_{i_2}, e_{i_3}\}$.

To construct an $(n, 4)$-zero-fixing source on which f is symmetric, we consider the complete 3-uniform hypergraph on vertex set I_2 as above, where an edge $\{i_1, i_2, i_3\}$ is colored by $f(e_{i_1} + e_{i_2} + e_{i_3})$. Applying the multi-color Ramsey theorem for hypergraphs [ER52], we obtain a subset of the vertices $I_3 \subseteq I_2$, with size

$$|I_3| \geq \log\log(|I_2|)/\mathrm{poly}(2^m),$$

such that the induced complete hypergraph by the vertex set I_3 is monochromatic. Therefore, if $\log\log\log n \geq \mathrm{poly}(k)$, then $|I_3| \geq 4$, and thus there are distinct coordinates $i_1, i_2, i_3, i_4 \in I_3$ such that f is symmetric on the $(n, 4)$-zero-fixing source spanned by $\{e_{i_1}, e_{i_2}, e_{i_3}, e_{i_4}\}$.

We continue this way, and find an (n, k)-zero-fixing source on which f is symmetric, by applying similar Ramsey-type arguments on r-uniform complete hypergraphs, with 2^m colors, for $r = 4, 5, \ldots, k-1$. A calculation shows that as long as $\mathsf{Tower}(k^2) < n$, such a source can be found.

To obtain the negative result for (n, k)-bit-fixing extractors, we follow a similar argument. The only difference is that in this case, it is enough to find an (n, k)-bit-fixing source X, such that f is symmetric restricted only to the $O(\sqrt{k})$ middle levels of X. Since most of the weight of X sits in these levels, an (n, k)-bit-fixing extractor cannot be symmetric restricted to these middle levels, regardless of the values obtained by the extractor in the remaining points of X.

5.3.2 Proof overview for Theorem 5.2

Informally speaking, the advantage one should exploit when given a sample from an (n, k)-zero-fixing source X, as apposed to a sample from a more general bit-fixing source,

is that "1 hits randomness". More formally, if $X_i = 1$, then we can be certain that $i \in S$, where $S \subset [n]$ is the set of indices for which $X|_S$ is uniform. How should we exploit this advantage?

A natural attempt would be the following. Consider all (random) indices $1 \leq i_1 < i_2 < \cdots < i_W \leq n$, such that $X_{i_1} = \cdots = X_{i_W} = 1$. Note that W, the Hamming weight of the sample, is a random variable concentrated around $k/2$. Let $M = i_{W/2}$ be the median of these random indices. One can show that, with high probability with respect to the value of M, both the prefix $(X_1, X_2, \ldots, X_M)$ and the suffix $(X_{M+1}, X_{M+2}, \ldots, X_n)$ have entropy roughly $k/2$. Intuitively, this is because the "hidden" random bits, namely bits in coordinates $i \in S$ such that $X_i = 0$, must be somewhat intertwined with the "observed" random bits – bits in coordinates $i \in S$ for which $X_i = 1$. In particular, except with probability $2^{-\Omega(k)}$ over the value of M, both the prefix and the suffix have entropy at least $0.49k$. Thus, by appending these prefix and suffix with zeros, one can get two n bit sources $X_{\mathsf{left}}, X_{\mathsf{right}}$, each having entropy at least $0.49k$.

We observe that conditioned on the value of the median M, the random variables X_{left} and X_{right} preserve the zero-fixing structure. Unfortunately, however, $X_{\mathsf{left}}, X_{\mathsf{right}}$ are *dependent*. In this proof overview, we rather continue with the description of the zero-fixing extractor as if $X_{\mathsf{left}}, X_{\mathsf{right}}$ were independent, and deal with the dependencies later on.

After obtaining X_{left} and X_{right}, we apply the lossless-condenser of Rao [Rao09b] on each of these random variables. This is an efficiently computable function $\mathsf{Cond}\colon \{0,1\}^n \to \{0,1\}^{k \log n}$, that is one-to-one when restricted to any (n,k)-bit-fixing source. We compute $Y_{\mathsf{left}} = \mathsf{Cond}(X_{\mathsf{left}})$ and $Y_{\mathsf{right}} = \mathsf{Cond}(X_{\mathsf{right}})$ to obtain two $(k \log n, 0.49k)$-weak sources. Note that the one-to-one guarantee implies that no entropy is lost during the condensing, and so the entropy of $Y_{\mathsf{left}}, Y_{\mathsf{right}}$ equals the entropy of $X_{\mathsf{left}}, X_{\mathsf{right}}$, respectively.

At this point, for simplicity, assume we have an explicit optimal two-source extractor

$$\mathsf{TwoSourceExt}\colon \{0,1\}^{k \log n} \times \{0,1\}^{k \log n} \to \{0,1\}^m$$

to our disposal. The output of our zero-fixing extractor is then $\mathsf{TwoSourceExt}(Y_{\mathsf{left}}, Y_{\mathsf{right}})$. Working out the parameters, one can see that an optimal two-source extractor would yield an (n,k)-zero-fixing extractor for $k > \log\log n + O(\log\log\log n)$, error $2^{-\Omega(k)}$ and output length, say, $0.9k$.

Constructing two-source extractors for even sub-linear entropy, let alone for logarithmic entropy, as used in the last step, is a major open problem in pseudorandomness. Even for our short input length $k \log n = \tilde{O}(\log n)$, no $\mathrm{poly}(n)$-time construction is known. In this proof overview however, we choose to rely on such an assumption for the sake of clarity. In the real construction, we apply the split-in-the-median process above, recursively, to obtain c weak-sources, for any desired constant c. In a recent breakthrough, Li [Li13] gave an explicit construction of a multi-source extractor, that extracts a constant fraction of the entropy, from a constant number of weak-sources with poly-logarithmic entropy. In the actual construction, instead of using a two-source extractor, we use the extractor of Li with the appropriate constant c.

Working around the dependencies. So far we ignored the dependencies between X_{left} and X_{right}, even though their condensed images are given as inputs to a two-source extractor, and the latter expects its inputs to be independent. As we now explain, the dependencies between X_{left} and X_{right} can be worked around.

The crucial observation is the following: conditioned on the fixing of the Hamming weight W of the sample X, and conditioned on any fixing of the median M, the random variables $X_{\mathsf{left}}, X_{\mathsf{right}}$ are independent! To see this, fix $W = w$. Then, conditioned on the event $M = m$, the value of the prefix $X_1, \ldots, X_m$ gives no information whatsoever about the suffix. More precisely, conditioned on any fixing of the prefix $X_1, \ldots, X_m$, the suffix is distributed uniformly at random over all $n - m$ bit strings, with zeros outside $S \cap \{m+1, \ldots, n\}$, and exactly $w/2$ ones in $S \cap \{m+1, \ldots, n\}$.

This observation motivates the following definition. We say that a random variable X is an (n, k, w)-*fixed-weight source*, if there exists $S \subseteq [n]$, with size $|S| = k$, such that a sample $x \sim X$ is obtained as follows. First, one samples a string $x' \in \{0,1\}^k$ of weight w, uniformly at random from all $\binom{k}{w}$ such strings, and then sets $X|_S = x'$, and $X_i = 0$ for all $i \notin S$. It is easy to see that any (n, k)-zero-fixing source is $2^{-\Omega(k)}$-close to a convex combination of (n, k, w)-fixed-weight sources, with w ranges over $k/3, \ldots, 2k/3$. Therefore, any extractor for (n, k, w)-fixed-weight sources, for such values of w, is also an extractor for (n, k)-zero-fixing sources.

We now reanalyze the algorithm described above. Since an (n, k)-zero-fixing source is $2^{-\Omega(k)}$-close to a convex combination of (n, k, w)-fixed-weight sources, with $k/3 \leq w \leq 2k/3$, we may assume, for the analysis sake, that the input is sampled from an (n, k, w)-fixed-weight source for some *fixed* $k/3 \leq w \leq 2k/3$. Fix also the median M to some value $m \in [n]$. Note that X_{left} is an $(n, k_{\mathsf{left}}(m), w/2)$-fixed-weight source[3], and X_{right} is an $(n, k_{\mathsf{right}}(m), w/2)$-fixed-weight source, with $k_{\mathsf{left}}(m)$ and $k_{\mathsf{right}}(m)$ being deterministic functions of m, satisfying $k_{\mathsf{left}}(m) + k_{\mathsf{right}}(m) = k$. Moreover, by the discussion above, we have that conditioned on the fixing $M = m$, the two random variables X_{left}, X_{right} are independent.

To summarize, conditioned on any fixing $M = m$, the two random variables $X_{\mathsf{left}}, X_{\mathsf{right}}$ are independent and preserve their fixed-weight structure. We further note that, with probability $1 - 2^{-\Omega(k)}$ over the value of M, it holds that $k_{\mathsf{left}}, k_{\mathsf{right}} \geq 0.49k$.

Recall that at this point we apply Rao's lossless-condenser on both X_{left} and X_{right}, to obtain shorter random variables Y_{left}, Y_{right}. Rao's condenser is one-to-one when restricted to bit-fixing sources. Since X_{left} and X_{right} are fixed-weight sources, they are in particular contained in some (n, k)-bit-fixing sources, and so the random variables $Y_{\mathsf{left}}, Y_{\mathsf{right}}$ have the same entropy as $X_{\mathsf{left}}, X_{\mathsf{right}}$, respectively.

It is worth mentioning that Rao's condenser Cond is linear, and as a result, if X_{left} were a bit-fixing source, then the resulting $Y_{\mathsf{left}} = \mathsf{Cond}(X_{\mathsf{left}})$ would have been an affine source. This property was crucial for Rao's construction of bit-fixing extractors. Since we wanted to maintain independence between $X_{\mathsf{left}}, X_{\mathsf{right}}$, in our case these random variables are no longer bit-fixing sources, but rather fixed-weight sources. Thus, the resulting Y_{left},

[3]To be more precise, X_{left} is not an $(n, k_{\mathsf{left}}(m), w/2)$-fixed-weight source per se, as its m^{th} bit is constantly 1. Ignoring this bit would make X_{left} a fixed-weight source.

Y_{right} are not affine sources, but only weak sources, with min-entropy $\log_2(\binom{0.49k}{w/2}) = \Omega(k)$. This is good enough for our needs, as in the next step we use a two-source extractor, and do not rely on the affine-ness.

Lastly, we apply a two-source extractor on the condensed random variables $Y_{\mathsf{left}}, Y_{\mathsf{right}}$, which is a valid application, as these sources are independent, and with probability $1 - 2^{-\Omega(k)}$, both have entropy $\Omega(k)$.

5.4 An Impossibility Result

For the proof of Theorem 5.1 we use the following notation. Let n be an integer and let $I \subseteq [n]$. We denote by $\{0,1\}^I$ the set of binary strings of length $|I|$ indexed by the elements of I. Using this notation for a string $x \in \{0,1\}^I$, we let $x^I \in \{0,1\}^n$ be the n bit string y such that $y|_I = x$, and $y_i = 0$ for all $i \notin I$, i.e., x^I is the extension of x to an n bit string with zeros outside the coordinate set I. Also, for a function $f\colon \{0,1\}^n \to \{0,1\}^m$ and a subset $I \subseteq [n]$, we define the function $f_I\colon \{0,1\}^I \to \{0,1\}^m$ as the restriction of f obtained by fixing the coordinates outside I to zeros. That is f_I is defined as $f_I(x) = f(x^I)$

We will need the following classical result of Erdős and Rado [ER52] on Ramsey numbers of multicolored hypergraphs .

Theorem 5.4 ([ER52], Theorem 1)**.** *Let G be the complete r-uniform hypergraph with vertex set $[n]$. Assume that each edge in G is colored by some color from $[c]$. Then, there exists a subset $I \subseteq [n]$ of size $|I| \geq c^{-O(1)} \cdot \log^{(r-1)}(n)$, such that the induced complete hypergraph by I is monochromatic.*[4]

The following corollary readily follows by Theorem 5.4. Indeed, the corollary is simply a rephrasing of the theorem in a slightly different language.

Corollary 5.5. *For any function $f\colon \{0,1\}^n \to \{0,1\}^m$ and integer $r \in [n]$, there exists a set of indices $I \subseteq [n]$, with size $|I| \geq 2^{-O(m)} \cdot \log^{(r-1)}(n)$, such that f_I is constant on the r^{th} level of $\{0,1\}^I$, i.e., $f_I(x) = f_I(y)$ for all $x, y \in \{0,1\}^I$ with $|x| = |y| = r$.*

Proof. Consider the complete r-uniform hypergraph on vertex set $[n]$, where each hyperedge $S \subseteq [n]$ of size $|S| = r$ is colored with $f(1_S)$, where 1_S is the characteristic function of the set S, i.e., $(1_S)_i = 1$ if and only if $i \in S$. By Theorem 5.4, there exists a subset of the vertices $I \subseteq [n]$, of size $2^{-O(m)} \cdot \log^{(r-1)}(n)$, such that the complete hypergraph induced by the vertex set I is monochromatic. By construction, this implies that for any $x, y \in \{0,1\}^I$ with $|x| = |y| = r$, it holds that $f(x^I) = f(y^I)$. Therefore, the function $f_I : \{0,1\}^I \to \{0,1\}^m$ is as desired. □

[4] We remark that Theorem 1 in [ER52] is stated somewhat differently. The theorem, as stated in the original paper, asserts that any large enough complete r-uniform hypergraph, with hyperedges colored by c colors, contains a monochromatic complete r-uniform hypergraph on N vertices. By large enough we mean that the number of vertices is some (tower) function that depends on r, c and N. For our purposes, however, it will be more convenient to apply the theorem as we state it.

Before proving Theorem 5.1, we prove an analogous theorem for zero-error dispersers. We do so as the proof is slightly cleaner. Moreover, we consider this to be a natural impossibility result by itself.

Theorem 5.6. *Let* $\mathsf{ZeroErrDisp}\colon \{0,1\}^n \to \{0,1\}^m$ *be an* (n,k)*-zero-fixing zero-error disperser. If* $n > \mathsf{Tower}(k^2)$*, then* $m \leq \log(k+1)$*.*

Proof. Clearly, it is enough to prove that if $m = \lfloor \log(k+1) \rfloor + 1$ then there exists an (n,k)-zero-fixing source on which $\mathsf{ZeroErrDisp}$ obtains at most $k+1$ distinct values, and thus reach a contradiction. (If m is larger, then we will obtain a contradiction by restricting the output to the first $m = \lceil \log(k+1) \rceil + 1$ bits.)

We define a sequence $[n] = I_0 \supseteq I_1 \supseteq I_2 \supseteq \ldots \supseteq I_k$, such that for each $i = 1, \ldots, k$ the restricted function f_{I_i} is symmetric on the levels $0, \ldots, i$ of the restricted hypercube $\{0,1\}^{I_i}$. Then, we shall claim that if $n > \mathsf{Tower}(k^2)$ then $|I_k| \geq k$. By taking $I \subseteq I_k$ to be any subset of size k we obtain a zero-fixing source spanned by the coordinates of I, such that the restriction of f to this zero-fixing source is a symmetric function, and in particular obtains at most $k+1$ values. By the argument above, this implies that $m \leq \log(k+1)$.

We define the subsets I_i iteratively by applying Corollary 5.5 for each $i = 1, \ldots, k$ with the function $f_{i-1} : \{0,1\}^{I_{i-1}} \to \{0,1\}^m$ and with $r = i$. Letting $n_{i-1} = |I_{i-1}|$, by Corollary 5.5 we obtain a subset $I_i \subseteq I_{i-1}$ of size $n_i = |I_i| \geq 2^{-O(m)} \cdot \log^{(i-1)}(n_{i-1})$. One can show, e.g., by induction on $i = 1, \ldots, k-1$, that $n_i \geq 2^{-O(m)} \cdot \log^{(s_i)}(n)$, where $s_i = \sum_{j=1}^{i}(j-1) = i(i-1)/2$. In particular, this implies that $n_k \geq 2^{-O(m)} \cdot \log^{(k^2/2)}(n)$, and so if $n > \mathsf{Tower}(k^2)$ then $|I_k| = n_k \geq k$. This completes the proof of the theorem. □

We now turn to prove Theorem 5.1.

Proof of Theorem 5.1. The proof outline is similar to the proof of Theorem 5.6, The only difference, when considering extractors rather than zero-error dispersers, is that it is enough to find an (n,k)-zero-fixing source such that f restricted to this source is symmetric only in the middle $O(\sqrt{k})$ levels, and not on all points of X. More precisely, let $\mathsf{bottom} = k/2 - c \cdot \sqrt{k \cdot \log(1/\delta)}$ and $\mathsf{top} = k/2 + c \cdot \sqrt{k \cdot \log(1/\delta)}$, where c is a universal constant such that

$$\sum_{i=\mathsf{bottom}}^{\mathsf{top}} \binom{k}{i} \geq \left(1 - \frac{\delta}{2}\right) \cdot 2^k.$$

One can show that such a constant c exists using a Chernoff bound. Let $I_{\mathsf{bottom}-1} = [n]$ and define a sequence $I_{\mathsf{bottom}-1} \supseteq I_{\mathsf{bottom}} \supseteq I_{\mathsf{bottom}+1} \supseteq \ldots \supseteq I_{\mathsf{top}}$, such that for each $i = \mathsf{bottom}, \ldots, \mathsf{top}$, the function f_{I_i} is symmetric on the levels $\mathsf{bottom}, \ldots, i$ of the restricted hypercube $\{0,1\}^{I_i}$. For each $i = \mathsf{bottom}, \ldots, \mathsf{top}$, given I_{i-1} we apply Corollary 5.5 with $f = f_{I_{i-1}}$ and $r = i$ to obtain $I_i \subseteq I_{i-1}$, such that the restriction f_{I_i} of $f_{I_{i-1}}$ is symmetric on level i, as well as on levels $\mathsf{bottom}, \ldots, i-1$, as f_{I_i} is a restriction of $f_{I_{i-1}}$.

By construction, it follows that if $n_{\mathsf{top}} \geq k$, then there exists an (n,k)-zero-fixing source X, such that f is symmetric restricted to the levels $\mathsf{bottom}, \ldots, \mathsf{top}$ of X. By our

choice of bottom and top, with probability $1-\delta/2$ over a uniformly random $x \sim X$, it holds that x has Hamming weight in $[\mathsf{bottom}, \mathsf{top}]$. Let C be the set of outputs obtained by f restricted to these levels of X. Note that $|C| \leq |\mathsf{top} - \mathsf{bottom} + 1| = O(\sqrt{k \cdot \log(1/\delta)})$. Thus, by considering the event $f(X) \in C$, we see that the statistical distance between the output of f on X, and the uniform distribution on m bits, is at least $1-\delta/2-|C|/2^m$, which is at least $1-\delta$, whenever $m \geq 0.5\log_2(k) + O(\log(1/\delta))$.

By Corollary 5.5, for every $i = \mathsf{bottom}, \ldots, \mathsf{top}$, we have that $|I_i| \geq 2^{-O(m)} \cdot \log^{(i-1)}(|I_{i-1}|)$ which is larger than $2^{-O(m)} \cdot \log^{(k)}(|I_{i-1}|)$. Therefore, if $n > \mathsf{Tower}(O(k^{3/2} \cdot \sqrt{\log(1/\delta)}))$, then $|I_{\mathsf{top}}| \geq k$, as required. □

5.5 Explicit Zero-Fixing Extractors for Double Logarithmic Entropy

In this section we prove Theorem 5.2. We repeat the statement of the theorem here for the readers convenience.

Theorem 5.7. *For any constant $\mu > 0$, and $n, k \in \mathbb{N}$, such that $k \geq (\log\log n)^{2+\mu}$, there exists an efficiently computable function*

$$\mathsf{ZeroBFExt}\colon \{0,1\}^n \to \{0,1\}^m,$$

where $m = \Omega(k)$, with the following property. For any (n,k)-zero-fixing source X, it holds that $\mathsf{ZeroBFExt}(X)$ is $(2^{-k^{\Omega(1)}} + (k\log n)^{-\Omega(1)})$-close to uniform.

We start by proving the following lemma that, informally speaking, shows how to efficiently split one fixed-weight source to two independent fixed-weight sources, each with half the weight and roughly half the entropy of the original source.

Lemma 5.3. *For every integer n, there exists an $O(n)$-time computable function*

$$\mathsf{SplitInMedian}\colon \{0,1\}^n \to (\{0,1\}^n)^2,$$

with the following property. Let X be an (n,k,w)-fixed-weight source, with $k/10 \leq w \leq 9k/10$. Denote the two n bit outputs of $\mathsf{SplitInMedian}(X)$ by X_{left} and X_{right}. Then, there exists a random variable M, and deterministic functions $k_{\mathsf{left}}, k_{\mathsf{right}}$ of M, such that conditioned on any fixing $M = m$, the following holds:

- *The random variables $X_{\mathsf{left}}, X_{\mathsf{right}}$ are independent.*
- *X_{left} is an $(n, k_{\mathsf{left}}, w/2-1)$-fixed-weight source.*
- *X_{right} is an $(n, k_{\mathsf{right}}, w/2)$-fixed-weight source, where $k_{\mathsf{left}} + k_{\mathsf{right}} + 1 = k$.*

Furthermore, for any $\varepsilon > 0$, it holds that

$$\Pr_{m\sim M}\left[\left|\frac{k_{\mathsf{left}}}{k} - \frac{1}{2}\right| \geq \varepsilon\right] \leq 2^{-\Omega(\varepsilon^2 \cdot k)}.$$

Proof. We first describe the algorithm for computing $\mathsf{SplitInMedian}(X)$, and then turn to the analysis. Let $1 \leq i_1 < i_2 < \cdots < i_w \leq n$ be the (random) indices such that $X_{i_1} = X_{i_2} = \cdots = X_{i_w} = 1$, and set $M = i_{w/2}$ to be the median coordinate. We define the n bit strings X_{left} and X_{right} as follows:

$$(X_{\mathsf{left}})_i = \begin{cases} X_i, & 1 \leq i < M; \\ 0, & M \leq i \leq n. \end{cases}$$

$$(X_{\mathsf{right}})_i = \begin{cases} 0, & 1 \leq i \leq M; \\ X_i, & M < i \leq n. \end{cases}$$

The output of $\mathsf{SplitInMedian}(X)$ is then $(X_{\mathsf{left}}, X_{\mathsf{right}})$. Clearly, the running-time of the algorithm is $O(n)$, as computing M and constructing $X_{\mathsf{left}}, X_{\mathsf{right}}$ can be carried out in linear-time.

Let $S \subseteq [n]$, with $|S| = k$, be the set of indices associated with X. That is, $X_i = 0$ for all $i \notin S$, and $X|_S$ is uniformly distributed over all k bit strings with Hamming weight w. Conditioned on the event $M = m$, it holds that conditioned on any fixing of the prefix $X_1, \ldots, X_m$, the suffix $X_{m+1}, \ldots, X_n$ is sampled uniformly at random from all $n - m$ bit strings, with Hamming weight $w/2$, and zeros outside $S \cap \{m+1, \ldots, n\}$. Since X_{left} and X_{right} are deterministic functions of these prefix and suffix, respectively, we have that conditioned on any fixing of M, the random variables X_{left} and X_{right} are independent.

As for the second and third items, we note that, for any value m, conditioned on the event $M = m$, the prefix $X_1, \ldots, X_{m-1}$ is a fixed-weight source. Indeed, $X_1, \ldots, X_{m-1}$ has the same distribution as sampling a vector $X' \in \{0,1\}^{k_{\mathsf{left}}}$, where $k_{\mathsf{left}} = |S \cap [m-1]|$, uniformly at random out of all such vectors with Hamming weight $w/2 - 1$, and setting $X|_{S \cap [m-1]} = X'$, and $X_i = 0$ for all $i \in [m-1] \setminus S$. Since X_{left} is obtained by concatenating zeros to the prefix $X_1, \ldots, X_{m-1}$, we have that X_{left} is an $(n, k_{\mathsf{left}}, w/2 - 1)$-fixed-weight source. A similar argument shows that X_{right} is an $(n, k_{\mathsf{right}}, w/2)$-fixed-weight source, where $k_{\mathsf{right}} = |S \cap \{m+1, \ldots, n\}|$. In particular, it holds that $k_{\mathsf{left}} + k_{\mathsf{right}} + 1 = k$.

For the furthermore part of the lemma, we want to bound the probability that k_{left} deviates from $k/2$, where the probability is taken with respect to the random variable M. Recall that k_{left} is a deterministic function of M, given by $k_{\mathsf{left}} = |S \cap [M-1]|$. Thus,

$$\begin{aligned} \Pr_{m \sim M}\left[k_{\mathsf{left}} \leq \left(\frac{1}{2} - \varepsilon\right) \cdot k\right] &= \Pr_{m \sim M}\left[|S \cap [m-1]| \leq \left(\frac{1}{2} - \varepsilon\right) \cdot |S|\right] \\ &= \binom{k}{w}^{-1} \cdot \sum_{t=w/2}^{(1/2-\varepsilon)\cdot k} \binom{t}{w/2}\binom{k-t}{w/2} \\ &\leq 2^{-\Omega(\varepsilon^2 \cdot k)}, \end{aligned}$$

where the last inequality follows by applying Stirling's approximation (or alternatively, by approximating the binomial distribution by a uniform distribution, and applying a

Chernoff bound), and our assumption that $k/10 \leq w \leq 9k/10$. By symmetry, the same upper bound holds for $\Pr\left[k_{\mathsf{left}} \geq \left(\frac{1}{2}+\varepsilon\right)\cdot k\right]$, which concludes the proof. □

For the proof of Theorem 5.7, we need to split the source to more than 2 independent sources. The following corollary accomplishes that, based on Lemma 5.3, and a recursive argument.

Corollary 5.8. *For any integers n, c, where c is a power of 2, there exists an $O(cn)$-time computable function*

$$\mathsf{Splitter}\colon \{0,1\}^n \to \left(\{0,1\}^n\right)^c,$$

with the following property. Let X be an (n,k,w)-fixed-weight source, with $k/3 \leq w \leq 2k/3$. Let $(Y_1,\ldots,Y_c) = \mathsf{Splitter}(X)$, with $Y_i \in \{0,1\}^n$ for all $i \in [c]$. Then, there exist random variables $M_1,\ldots,M_{c-1}$, and deterministic functions $k_1,\ldots,k_{c-1}$ of them, such that conditioned on any fixing $(M_1,\ldots,M_{c-1}) = (m_1,\ldots,m_{c-1})$, the following holds:

- *The random variables $Y_1,\ldots,Y_c$ are independent.*
- *For every $i \in [c]$, the random variable Y_i is an (n,k_i,w_i)-fixed-weight source, with $w_i \in [w/c-1, w/c]$, and $k_1+\cdots+k_c = k-c+1$.*

Furthermore, except with probability $c\cdot 2^{-\Omega(k/(c\cdot\log^2 c))}$ over the fixings of $(M_1,\ldots,M_{c-1})$, it holds that for all $i \in [c]$, $k_i \geq 0.9k/c$.

Proof. Let $d = \log_2 c$. Consider a depth d binary tree T with c leaves. With each node v of T, we associate a random variable X_v, defined recursively with respect to the depth, as follows. Let r be the root of T. We define $X_r = X$. Let v be a node in T, that is not a leaf, for which X_v was already defined. Denote by $\mathsf{leftChild}(v), \mathsf{rightChild}(v)$ the left and right children of v in T, respectively. Let $((X_v)_{\mathsf{left}}, (X_v)_{\mathsf{right}}) = \mathsf{SplitInMedian}(X_v)$. We associate the random variable $(X_v)_{\mathsf{left}}$ with the vertex $\mathsf{leftChild}(v)$, and the random variable $(X_v)_{\mathsf{right}}$ with the vertex $\mathsf{rightChild}(v)$. Namely, $X_{\mathsf{leftChild}(v)} = (X_v)_{\mathsf{left}}$ and $X_{\mathsf{rightChild}(v)} = (X_v)_{\mathsf{right}}$. Let M_v be the random variable M, in the notation of Lemma 5.3, with respect to the application of $\mathsf{SplitInMedian}$ to X_v. Let $\ell_1,\ldots,\ell_c$ be the c leaves of T. The output of $\mathsf{Splitter}$ on input X is defined by

$$\mathsf{Splitter}(X) = (X_{\ell_1},\ldots,X_{\ell_c}).$$

We now turn to the analysis. First, clearly, $\mathsf{Splitter}$ is computable in time $O(cn)$, as it involves $c-1$ applications of $\mathsf{SplitInMedian}$. Let $h \in \{0,1,\ldots,d-1\}$, and let V_h be the set of nodes of T with depth h. Let $\varepsilon = 1/(20d)$. We prove the following, by induction on h. Conditioned on any fixing of the random variables $\{M_v \mid v \in V_0 \cup V_1 \cup \cdots \cup V_{h-1}\}$, the following holds:

- The random variables $\{X_v \mid v \in V_h\}$ are independent.
- For any $v \in V_h$, the random variable X_v is an (n,k_v,w_v)-fixed-weight source, with $w_v \in [w/2^h, w/2^h - 1]$, and where k_v is a deterministic function of the random variables $\{M_u\}_{u\in P_v}$, where P_v is the nodes on the path from the root r to v in T, not including v. Moreover, $\sum_{v\in V_h} k_v = k-h$.

Furthermore, except with probability $\delta_h = 2^h \cdot 2^{-\Omega(k/(c\cdot\log^2 c))}$ over the fixings of $\{M_v \mid v \in V_0 \cup V_1 \cup \cdots \cup V_{h-1}\}$, it holds

$$\forall v \in V_h \quad \left(\frac{1}{2} - \varepsilon\right)^h \leq \frac{k_v}{k} \leq \left(\frac{1}{2} + \varepsilon\right)^h .$$

These claims clearly hold for $h = 0$. We now prove that the claims hold for $h \geq 1$, assuming they hold for $1, \ldots, h-1$. By the induction hypothesis, conditioned on any fixings of $\{M_v \mid v \in V_0 \cup V_1 \cup \cdots \cup V_{h-2}\}$, the random variables $\{X_v \mid v \in V_{h-1}\}$ are independent. Since $\{X_v \mid v \in V_h\} = \{X_{\mathsf{leftChild}(v)}, X_{\mathsf{rightChild}(v)} \mid v \in V_{h-1}\}$, the independence of the random variables $\{X_v \mid v \in V_h\}$, conditioned on the further fixings of $\{M_v \mid v \in V_{h-1}\}$, follows by Lemma 5.3. The second item readily follows by the induction hypothesis and Lemma 5.3.

As for the furthermore part, by the induction hypothesis, except with probability δ_{h-1} over the fixings of $\{M_v \mid v \in V_0 \cup V_1 \cup \cdots \cup V_{h-2}\}$, it holds that

$$\forall v \in V_{h-1} \quad \left(\frac{1}{2} - \varepsilon\right)^{h-1} \leq \frac{k_v}{k} \leq \left(\frac{1}{2} + \varepsilon\right)^{h-1} .$$

One can easily verify that conditioned on this event, by our choice of ε, the hypothesis of Lemma 5.3 is met when computing $\mathsf{SplitInMedian}(X_v)$, namely,

$$\frac{k_v}{10} \leq w_v \leq \frac{9k_v}{10} .$$

Thus, by the union bound, except with probability

$$\delta_{h-1} + \sum_{v \in V_{h-1}} 2^{-\Omega(\varepsilon^2 \cdot k_v)} , \tag{5.1}$$

it holds that for all $v \in V_h$

$$\frac{k_v}{k} = \frac{k_v}{k_{\mathsf{parent}(v)}} \cdot \frac{k_{\mathsf{parent}(v)}}{k} \leq \left(\frac{1}{2} + \varepsilon\right) \cdot \left(\frac{1}{2} + \varepsilon\right)^{h-1} = \left(\frac{1}{2} + \varepsilon\right)^h ,$$

where $\mathsf{parent}(v)$ is the parent of v in the tree T. Similarly,

$$\frac{k_v}{k} = \frac{k_v}{k_{\mathsf{parent}(v)}} \cdot \frac{k_{\mathsf{parent}(v)}}{k} \geq \left(\frac{1}{2} - \varepsilon\right) \cdot \left(\frac{1}{2} - \varepsilon\right)^{h-1} = \left(\frac{1}{2} - \varepsilon\right)^h .$$

Since $|V_{h-1}| = 2^{h-1}$, $\varepsilon = 1/(20d) = O(1/\log c)$ and $k_v = \Omega(k/2^h) \geq \Omega(k/c)$, the error expression in Equation (5.1) is bounded above by

$$\delta_{h-1} + 2^{h-1} \cdot 2^{-\Omega(k/(c\cdot\log^2 c))} \leq \delta_h .$$

This concludes the inductive proof. The proof of the corollary follows by plugging $h = d$. □

For the proof of Theorem 5.7 we also need the following claim, which states that an (n, k)-zero-fixing source is close to a convex combination of fixed-weight sources, with weight roughly $k/2$.

Claim 5.3.1. *Let X be an (n, k)-zero-fixing source. Then, X is $2^{-\Omega(k)}$-close to a convex combination of (n, k, w)-fixed-weight sources, with $k/3 \leq w \leq 2k/3$.*

For the proof of the claim we make use of the following well-known fact.

Fact 5.4. *For any integer n, and $0 < \alpha < 1/2$, it holds that*

$$\sum_{k=0}^{\lfloor \alpha n \rfloor} \binom{n}{k} \leq 2^{H(\alpha)\cdot n},$$

where $H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ is the binary entropy function.

Proof of Claim 5.3.1. We first note that X can be written as the convex combination

$$X = \sum_{w=0}^{k} \lambda_w X_w,$$

where $\lambda_w = \binom{k}{w} \cdot 2^{-k}$, and X_w is an (n, k, w)-fixed-weight source. To see this, let $S \subseteq [n]$, $|S| = k$, be the set that is associated with the source X. Namely, $X|_S$ is uniformly distributed, whereas $X|_{S^c}$ is fixed to 0. Sampling $x \sim X$ can be done in two steps. In the first step, one samples a weight W according to a binomial distribution $\mathrm{Bin}(k, 1/2)$. Namely, for any $0 \leq w \leq k$, $\Pr[W = w] = \binom{k}{w} \cdot 2^{-k}$. In the second step, one samples a string $x' \in \{0,1\}^k$ uniformly at random among all strings with Hamming weight w. Lastly, we set $X|_S = x'$, and $x_i = 0$ for all $i \notin S$. It is easy to verify that this two-steps procedure yields the same distribution as sampling from the (n, k)-zero-fixing source X. Note that the sampling done in the second step, conditioned on the event $W = w$, is from an (n, k, w)-fixed-weight source, which we denote by X_w.

By Fact 5.4, we have that

$$\sum_{w=k/3}^{2k/3} \lambda_w \geq 1 - 2 \cdot 2^{(H(1/3)-1)\cdot k} = 1 - 2^{-\Omega(k)}.$$

This concludes the proof, as it shows that X is $2^{-\Omega(k)}$-close to the convex combination

$$X = \sum_{w=k/3}^{2k/3} \lambda_w X_w.$$

□

For the proof of Theorem 5.7 we also make use of the following lossless condenser due to Rao [Rao09b].

Theorem 5.9 ([Rao09b]). *For all integers n, k, there exists an efficiently computable linear transformation* $\mathsf{Cond} : \{0,1\}^n \to \{0,1\}^{k \log n}$, *such that for any (n,k)-bit-fixing source X it holds that* Cond *restricted to X is one-to-one.*

We are now ready to prove Theorem 5.7.

Proof of Theorem 5.7. We first describe the construction of $\mathsf{ZeroBFExt}$ and then turn to the analysis. For the construction of $\mathsf{ZeroBFExt}$ we need the following building blocks:

- Let $\mathsf{Li}\colon \left(\{0,1\}^{k \log n}\right)^c \to \{0,1\}^\ell$ be the multi-source extractor from Theorem 2.4, set to extract $\ell = \Omega(k)$ bits from c independent $(k \log n, k)$-weak-sources, with $k \geq O(\log^{2+\mu}(k \log n))$. By Theorem 2.4, it suffices to take $c = O(1/\mu)$.
- With c as above, let $\mathsf{Splitter}\colon \{0,1\}^n \to (\{0,1\}^n)^c$ be the function from Corollary 5.8.
- Let $\mathsf{Cond}\colon \{0,1\}^n \to \{0,1\}^{k \log n}$ be the lossless-condenser of Rao from Theorem 5.9.

With these building blocks, we compute $\mathsf{ZeroBFExt}(X)$ as follows. We first compute $(Y_1, \ldots, Y_c) = \mathsf{Splitter}(X)$. Secondly, for each $i \in [c]$, we compute $Z_i = \mathsf{Cond}(Y_i)$. The output is then $\mathsf{ZeroBFExt}(X) = \mathsf{Li}(Z_1, \ldots, Z_c)$.

We now turn to the analysis. By Claim 5.3.1, X is $2^{-\Omega(k)}$-close to a convex combination of (n,k,w)-weight-fixing sources $\{X_w\}_{w=k/3}^{2k/3}$. Therefore, $\mathsf{Splitter}(X)$ is $2^{-\Omega(k)}$-close to a convex combination of the random variables $\{\mathsf{Splitter}(X_w)\}_{w=k/3}^{2k/3}$. We denote $((Y_w)_1, \ldots, (Y_w)_c) = \mathsf{Splitter}(X_w)$. Fix such w. By Corollary 5.8, conditioned on some carefully chosen random variables, $(Y_w)_1, \ldots, (Y_w)_c$ are independent random variables. Moreover, except with probability $2^{-\Omega(k)}$ with respect to the conditioning, it holds that for all $i \in [c]$, $(Y_w)_i$ is an $(n, k', w/c)$-weight-fixing source, with $k' \geq 0.9k/c$. Since

$$\binom{k'}{w/c} \geq \left(\frac{k'}{w/c}\right)^{w/c} \geq \left(\frac{0.9k}{w}\right)^{w/c} \geq \left(\frac{0.9}{2/3}\right)^{w/c} = 2^{\Omega(k)},$$

we have that $H_\infty((Y_w)_i) = \Omega(k)$ for all $i \in [c]$, except with probability $2^{-\Omega(k)}$.

Recall that $Z_i = \mathsf{Cond}(Y_i)$. With the notation above, we have that Z_i is $2^{-\Omega(k)}$-close to a convex combination of $(Z_w)_i = \mathsf{Cond}((Y_w)_i)$, where $k/3 \leq w \leq 2k/3$. Since $(Y_w)_i$ is contained in some (n,k)-bit-fixing source, Theorem 5.9 guarantees that Cond restricted to the support of $(Y_w)_i$ is one-to-one, and so $H_\infty((Z_w)_i) = H_\infty((Y_w)_i) = \Omega(k)$. Thus, except with probability $2^{-\Omega(k)}$, we have that for all $i \in [c]$, $(Z_w)_i$ is a $(k \log n, \Omega(k))$-weak source. This implies that $\mathsf{ZeroBFExt}(X_w) = \mathsf{Li}((Z_w)_1, \ldots, (Z_w)_c)$ is $(2^{-k^{\Omega(1)}} + (k \log n)^{-\Omega(1)})$-close to uniform, which completes the proof of the theorem as $\mathsf{Li}(X)$ is $2^{-\Omega(k)}$-close to a convex combination of $\{\mathsf{Li}(X_w)\}_{w=k/3}^{2k/3}$.

As for the running-time. Computing $Y_1, \ldots, Y_c$ by applying $\mathsf{Splitter}$ to X is done in time $O(n)$. Applying Rao's condenser to each Y_i can be done in $\mathrm{poly}(n)$-time. Finally, Li's extractor runs in time $\mathrm{poly}(k \log n) = o(n)$. □

A comment regarding the error. As stated, the extractor ZeroBFExt in Theorem 5.2 has an error of $2^{-k^{\Omega(1)}} + (k \log n)^{-\Omega(1)}$. This error is induced by the error of Li's multi-source extractor. Indeed, the error contributed by the other parts of the construction of ZeroBFExt is only $2^{-\Omega(k)}$. The error of Li's extractor, when applied to (n, k)-weak sources, is stated to be $n^{-\Omega(1)} + 2^{-k^{\Omega(1)}}$. However, by inspection, one can see that Li's extractor has an error of $(\delta n)^{O(1)} + 2^{-k^{\Omega(1)}}$, for any desired parameter $\delta > 0$. The running-time of the extractor is $\text{poly}(n/\delta)$. Clearly, when one is interested in $\text{poly}(n)$ running-time, then one must take $\delta \geq 1/\text{poly}(n)$. However, in our case, the inputs to Li's extractor have length $O(k \log n)$. Thus, we can set δ to be such that the total error in our application of Li's extractor is $2^{-k^{\Omega(1)}}$, and the running-time of that application would then be $\text{poly}(2^k \cdot \log n)$, which is $o(n)$ for the parameters of interest, namely for $k = o(\log n)$. To summarize, the error in Theorem 5.7 can be reduced to $2^{-k^{\Omega(1)}}$. We choose to state Theorem 5.2 as we did so to be able to use Li's extractor in a black-box fashion.

5.6 Bit-Fixing Extractors for Double-Logarithmic Entropy

In this section we prove Theorem 5.3. We restate the theorem here, allowing also for non-constant error ε.

Theorem 5.10. *For any integers n, k, and $\varepsilon > 0$, such that*

$$k > \log(\log(n)/\varepsilon^2) + 2\log\log(\log(n)/\varepsilon) + O(1),$$

there exists a function

$$\mathsf{QuasiBFExt}\colon \{0,1\}^n \to \{0,1\}^m,$$

where $m = k - 2\log(1/\varepsilon) - O(1)$, with the following property. Let X be an (n, k)-bit-fixing source. Then, $\mathsf{QuasiBFExt}(X)$ is ε-close to uniform. The running time of evaluating QuasiBFExt *is $n^{O(\log^2(\frac{\log n}{\varepsilon}))}$.*

Before proving Theorem 5.3, we sketch two proofs for the existence of (n, k)-bit-fixing extractors, with double-logarithmic entropy. Our first proof relies on the Lóvasz local lemma.

Lemma 5.5 (Lóvasz local lemma [EL75, Spe77])**.** *Let $E_1, \ldots, E_k$ be events in a probability space, such that each event occurs with probability at most p, and such that each event is independent of all but at most d events. If $ep(d+1) \leq 1$,* [5] *then*

$$\Pr\left[\bigcap_{i=1}^{k} \bar{E}_i\right] > 0.$$

[5] Here e is the base of the natural logarithm.

Existential proof-sketch based on the Lóvasz Local Lemma. Let $f\colon \{0,1\}^n \to \{0,1\}^m$ be a random function. For any (n,k)-bit-fixing source X, let E_X be the event $\mathsf{SD}(f(X),U_m) > \varepsilon$ (here the randomness is taken over f). Fix an (n,k)-bit-fixing source X. By taking the union bound over all 2^{2^m} test functions, Chernoff bound implies that

$$\Pr_f[E_X] \leq 2^{2^m} \cdot 2^{-\Omega(2^k \cdot \varepsilon^2)} = p.$$

Consider any two bit-fixing sources X, Y. We note that if there exists a coordinate $i \in [n]$, in which both X and Y are fixed, then in order for X and Y to be dependent, it must hold that $X_i = Y_i$. Indeed, if $X_i \neq Y_i$ then $X \cap Y = \emptyset$. Thus, for any (n,k)-bit-fixing source X, there are at most $d = \binom{n}{k} \cdot 2^k$ bit-fixing sources Y such that E_X depends on E_Y. One can easily verify that by taking

$$\begin{aligned} k &= \log\log(n) + 2\log(1/\varepsilon) + \log(\log\log(n) + 2\log(1/\varepsilon)) + O(1), \\ m &= k - 2\log(1/\varepsilon) - O(1), \end{aligned}$$

the hypothesis of Lemma 5.5 is met. Thus, even for k as above, there exists an (n,k)-bit-fixing extractor, with error ε, that outputs $m = k - 2\log(1/\varepsilon) - O(1)$.

Existential proof-sketch based on Rao's linear lossless-condenser. Theorem 5.9 states that there exists a linear function $\mathsf{Cond}\colon \{0,1\}^n \to \{0,1\}^{k\log n}$, such that for any (n,k)-bit-fixing source X, the mapping Cond, restricted to X, is one-to-one. Since Cond is linear, this implies that $\mathsf{Cond}(X)$ is a $(k\log n, k)$-affine source. At this point, one can use a simple probabilistic argument to show the existence of (n,k)-affine extractors, with error ε, that outputs $m = k - 2\log(1/\varepsilon) - O(1)$ bits, as long as $k \geq \log(n) + 2\log(1/\varepsilon) + O(1)$. By applying the latter (implicit) extractor to the affine source $\mathsf{Cond}(X)$, we obtain (n,k)-bit-fixing extractors with parameters as in the proof-sketch based on the Lóvasz local lemma.

We note that by iterating over all $(2^M)^{2^N}$ functions $f\colon \{0,1\}^N \to \{0,1\}^M$, and checking each of them against any of the possible $\binom{2^N}{K+1} \cdot 2^{2^M}$ pairs of an (N,K)-affine source and a test function, one can find an (N,K)-affine extractor, with $K = \log(N) + \log\log(N) + O(1)$ and $M = K - O(1)$ output bits, in time $2^{O(2^N \cdot \log N)}$. After the application of Cond in the proof-sketch above, we only need $(k\log n, k)$-affine extractors, with $k = \log\log n + \log\log\log n + O(1)$. Namely, we can set $N = k\log n$, $K = k$ and $M = m$. Thus, the proof-sketch above, together with this brute-force search for affine extractors, yields a construction of an (n,k)-bit-fixing extractors, in time $2^{n^{O(\log\log n)}}$.

The proof of Theorem 5.3 follows the same argument as the second proof-sketch. The improvement in running-time, from the $2^{n^{O(\log\log n)}}$-time algorithm described above to the stated $n^{O((\log\log n)^2)}$, is obtained by using a more efficient construction of essentially-optimal affine extractors, as capture by the following lemma.

Lemma 5.6. *For every integer n and $\varepsilon > 0$, there exists an affine extractor*

$$\mathsf{AExt}\colon \{0,1\}^n \to \{0,1\}^m,$$

for (n,k)-affine sources, with $k = \log(n/\varepsilon^2) + \log\log(n/\varepsilon^2) + O(1)$, and any $m \leq k - 2\log(1/\varepsilon) - O(1)$. The running-time of evaluating AExt *at a given point $x \in \{0,1\}^n$ is $2^{2^m} \cdot 2^{O(n \cdot \log(n/\varepsilon))}$.*

The proof of Lemma 5.6 makes use of sample spaces that are almost k-wise independent, introduced by Naor and Naor [NN93].

Definition 5.7 (Almost k-wise independence)**.** *Let n, k be integers such that $k \leq n$, and let $\delta > 0$. A random variable X over n bit strings is called* (n,k,δ)-independent*, if for any $S \subseteq [n]$, with $|S| \leq k$, the marginal distribution $X|_S$ is δ-close to uniform, in statistical distance.*

We use the following explicit construction of Alon *et al.* [AGHP92].

Theorem 5.11 ([AGHP92])**.** *For all $\delta > 0$ and integers n, k, there exists an explicit construction of an (n,k,δ)-independent sample space, with size $(k\log(n)/\varepsilon)^{2+o(1)}$.*

Proof of Lemma 5.6. For an integer k and $\delta > 0$ which will be determined later, let $Z \in \{0,1\}^{2^n \cdot m}$ be a sample from a $(2^n \cdot m, 2^k \cdot m, \delta)$-independent sample space. We index a bit of the sample Z by a pair composed of $x \in \{0,1\}^n$ and $i \in [m]$, and denote the respective random bit by $Z_{x,i}$. Define the (random) function $\mathsf{AExt}\colon \{0,1\}^n \to \{0,1\}^m$ by $\mathsf{AExt}(x) = (Z_{x,1}, \ldots, Z_{x,m})$.

Let $U \subseteq \{0,1\}^n$ be an affine subspace of dimension k, and let $f\colon \{0,1\}^m \to \{0,1\}$ be an arbitrary function, which we think of as a "test" function, or a distinguisher. Since AExt restricted to U is a function of $2^k \cdot m$ bits of Z, it holds that $\{\mathsf{AExt}(u)\}_{u \in U}$ are 2^k random variables over $\{0,1\}^m$ that are δ-close to uniform. Thus, the random variable (with randomness coming from Z)

$$\mathop{\mathbf{E}}_{u \sim U}\left[f(\mathsf{AExt}(u))\right] = \mathop{\mathbf{E}}_{u \sim U}\left[f(Z_{u,1}, \ldots, Z_{u,m})\right]$$

is δ-close, in statistical distance, to the random variable $\mathbf{E}_{u \sim U}\left[f(R_{u,1}, \ldots, R_{u,m})\right]$, where $\{R_{u,i}\}_{u \in U, i \in [m]}$ are $2^k \cdot m$ uniformly distributed and independent random bits. Now, by the Chernoff bound,

$$\Pr_R\left[\left|\mathop{\mathbf{E}}_{u \sim U}\left[f(R_{u,1}, \ldots, R_{u,m})\right] - \mathop{\mathbf{E}}_{x \sim \{0,1\}^m}\left[f(x)\right]\right| > \varepsilon\right] \leq 2^{-\Omega(\varepsilon^2 \cdot 2^k)}.$$

Thus,

$$\Pr_Z\left[\left|\mathop{\mathbf{E}}_{u \sim U}\left[f(\mathsf{AExt}(u))\right] - \mathop{\mathbf{E}}_{x \sim \{0,1\}^m}\left[f(x)\right]\right| > \varepsilon\right] \leq 2^{-\Omega(\varepsilon^2 \cdot 2^k)} + \delta\,.$$

By the union bound taken over all affine subspaces U of dimension k and functions $f\colon \{0,1\}^m \to \{0,1\}$, we get that as long as

$$\binom{2^n}{k+1} \cdot 2^{2^m} \cdot \left(2^{-\Omega(\varepsilon^2 \cdot 2^k)} + \delta\right) < 1,$$

there exists a point in the sample space for Z that induces an (n, k)-affine extractor AExt with error ε. By taking $\delta = 2^{-\Omega(\varepsilon^2 \cdot 2^k)}$, one can verify that the equation above holds as long as

$$\begin{aligned} k &\geq \log(n/\varepsilon^2) + \log\log(n/\varepsilon^2) + O(1), \\ m &\leq k - 2\log(1/\varepsilon) - O(1). \end{aligned}$$

We use the construction of a $(2^n \cdot m, 2^k \cdot m, \delta)$-independent sample space from Theorem 5.11. One can verify that the sample space size is $2^{O(n \cdot \log(n/\varepsilon))}$. One can then go over each point in the sample space and check whether the point induces an (n, k)-affine extractor with error ε. By the choice of parameters, such a point exists. Each point from the sample space should be compared against $\binom{2^n}{k+1} \cdot 2^{2^m}$ pairs of an affine subspace and a test function $f \colon \{0,1\}^m \to \{0,1\}$. Checking each fixed point in the sample space can be done in time $2^{2^m} \cdot 2^{O(n \cdot \log(n/\varepsilon))}$. Hence, the total running-time is $2^{2^m} \cdot 2^{O(n \cdot \log(n/\varepsilon))}$, as stated. □

Proof of Theorem 5.10. The construction of $\mathsf{QuasiBFExt}$ is very simple, and is defined by

$$\mathsf{QuasiBFExt}(x) = \mathsf{AExt}(\mathsf{Cond}(x)),$$

for all $x \in \{0,1\}^n$, where Cond is Rao's linear lossless-condenser from Theorem 5.9. As for the analysis, let X be an (n, k)-bit-fixing source. By Theorem 5.9, $Y = \mathsf{Cond}(X)$ is a $(k \log n, k)$-affine source. Therefore, Lemma 5.6 implies that $\mathsf{AExt}(Y)$ is ε-close to uniform. It is straightforward to verify that the running-time and number of output bits of $\mathsf{QuasiBFExt}$ is as claimed. □

Chapter 6

Efficient Multiparty Protocols via Log-Depth Threshold Formulae

6.1 Secure Multiparty Computation

Secure multiparty computation (MPC) enables a set of parties to jointly accomplish some distributed computational task, while maintaining the secrecy of the inputs and the correctness of the outputs in the presence of coalitions of dishonest parties. Originating from the seminal works of [Yao82b, GMW87, BGW88, CCD88], secure MPC has been the subject of an enormous body of work.

Despite this body of work, MPC protocols remain quite complicated and their security is difficult to prove. In the paper [CDI+13] that is covered in this chapter, we propose a new general approach to the construction of efficient multiparty protocols in the presence of an honest majority. Our approach enables us to obtain conceptually simple derivations of known feasibility results (or slightly weaker variants of such results), and also to obtain new results.

Our approach is inspired by, and builds on, the "player emulation" technique of Hirt and Maurer [HM00], who obtain secure MPC protocols by reducing the construction of an n-party protocol to the task of constructing a protocol π for a constant (e.g., three or four) number of parties. The motivation of [HM00] was to obtain n-party protocols that are secure with respect to general (non-threshold) adversary structures. A disadvantage of their n-party protocols is that their complexity grows exponentially with n. This seems inevitable when considering arbitrary adversary structures.

Our motivation is very different: We would like to use the atomic protocol π for constructing *efficient* n-party protocols in the traditional MPC setting of *threshold* adversary structures. Since π only involves a small number of parties, its design may employ simpler techniques that do not scale well with the number of corrupted parties. Thus, our goal is to simplify the design of efficient n-party protocols by reducing it to the design of a simpler atomic protocol π.

To make the approach of [HM00] scale with the number of parties, we introduce a new

complexity-theoretic primitive: a logarithmic-depth formula[1] which is composed only of constant-size threshold gates and computes an n-input threshold function. The problem of constructing such formulae is closely related to a classical problem in complexity theory. In this work we also make a contribution to this complexity-theoretic problem, which may be of independent interest.

In addition to providing conceptually simple protocols, our approach is very general and can be applied in a variety of settings and models. In contrast to most traditional MPC protocols, it is not tied to some underlying algebraic structure. We demonstrate this generality by obtaining new results on MPC over black-box groups and other algebraic structures, improving on previous results from the literature.

Before proceeding to describe the details of our approach, we note that the goal of designing MPC protocols whose complexity grows (only) polynomially with the number of parties also has relevance to *two-party* cryptography. Indeed, there are general techniques for applying MPC protocols with security in the presence of an honest majority (where the number of parties grows with the security parameter) towards two-party tasks such as zero-knowledge proofs and secure two-party computation [IKOS09, IPS08].

6.2 Our Approach

In the following, for simplicity, we consider the case of perfect security against a *passive* adversary. In this setting, parties are honest but curious. That is, they follow the protocol but may attempt to learn secret information based on what they see. We note that, in contrast to the norm, the extension of this approach to the case of an active adversary is relatively straightforward.

We first give an overview of the player emulation technique of Hirt and Maurer [HM00] and then proceed to describe how we overcome the exponential blow-up incurred by [HM00] in the case of threshold adversary structures. Recall that security of MPC protocols is defined by comparing a real protocol to an ideal protocol, in which, in addition to the parties involved in the computation, there is a trusted party. A protocol is deemed secure if for every adversary in the real protocol controlling a subset of the parties, there is an equivalent adversary controlling the same subset in the ideal protocol.

The technique from [HM00] is to reduce the design of n-party protocols to the design of protocols that support only 3 parties (the minimal number of parties for perfect security in the passive security model).

We proceed to present an informal description of the reduction. Indeed, suppose that the 3-party case has been solved. That is, for every computational task involving three parties there exists a secure protocol that securely implements this task when at most one of the parties is passively corrupted.[2] We describe how to use this protocol to securely

[1] A formula is a circuit with fan-out 1. A logarithmic-depth formula (more precisely, infinite family of formulas) is one whose depth is $O(\log n)$, where n is the number of inputs. Throughout this paper we consider only *monotone* formulas without negations or constants.

[2] Since we deal with *perfect* security, the size of the secure protocol depends only on the size of the original protocol. In particular, any constant size protocol can be implemented securely in constant size.

implement computational tasks using a larger number of parties.

Consider n parties that wish to securely accomplish some joint computational task. It is best to think of this task as being specified by an ideal protocol π_0 which involves, in addition to the n parties, a trusted party τ. The ideal protocol is secure (by definition) even if the adversary controls any subset of the parties that does not contain τ.

Consider a new protocol π_1 that involves the n original parties but where we replace the trusted party τ with three new virtual parties v_1, v_2, v_3. Since in π_0, the trusted party τ is just involved in a computational task, we can use the given 3-party protocol to simulate τ using v_1, v_2, v_3. When is the new protocol π_1 secure? Since π_0 was only insecure whenever the adversary controlled τ and since the 3-party protocol is secure as long as the adversary controls at most one of the virtual parties, π_1 is secure as long as the adversary does not control two or more of the virtual parties.

We continue this process by designing a new protocol π_2 in which the virtual party v_1 is itself simulated by three new virtual parties w_1, w_2, w_3. Since π_1 is only insecure whenever the adversary controls more than one of v_1, v_2, v_3 and since the protocol for emulating v_1 is secure when at most one of w_1, w_2, w_3 is controlled by the adversary, π_2 is secure as long as the adversary does not control either v_2 and v_3 or one of v_2, v_3 and two or more of w_1, w_2, w_3.

We continue in this process simulating virtual parties by more virtual parties. The sets of corrupted parties against which the resulting protocol is secure can be described by looking at a formula composed of 3-input majority gates which we denote by Maj_3. Each wire represents a virtual party. The protocol π_1 can be represented by a simple formula F_1 consisting of a single Maj_3 gate where the three input wires correspond to the virtual parties v_1, v_2, v_3 and the output wire corresponds to τ. We assign to each input wire corresponding to an honest party a value of 0 and a value of 1 to those corresponding to dishonest parties. It can be easily verified that the protocol is secure whenever the formula F_1 evaluates to 0.

Similarly, the protocol π_2 can be represented by a formula F_2 which is constructed from F_1 by connecting the input wire corresponding to v_1 with an additional Maj_3 gate with three new input wires (corresponding to w_1, w_2, w_3). It is easy to verify that the new protocol is secure whenever the formula evaluates to 0.

Suppose that we continue on like this but instead of arbitrarily choosing which virtual party to simulate, we choose it according to some formula F, composed only of Maj_3 gates.[3] Once we reach the input layer of the formula, we associate each input variable to a real party and every remaining virtual party is simulated by the real party associated with the corresponding input wire.

As above, the protocol is secure against every set T of parties on which the formula F evaluates to 0. (Here and in the following we associate a set T with its characteristic vector χ_T.) Thus, to obtain a protocol that is secure for a particular adversary structure, it suffices to provide a formula that evaluates to 0 on all sets in the structure. Since, in contrast to [HM00], our goal is merely to obtain security in the presence of an honest

[3] Actually, [HM00] do not present their construction in the terminology of Maj_3 formulae; we use this presentation since it is more intuitive and is better suited for our purposes.

majority, we need only to construct a formula that computes the majority function (using only Maj_3 gates and no constants).

Such a formula was implicitly constructed by Hirt and Maurer [HM00] for general Q_2 functions[4] and in particular for majority. Unfortunately, the formula of [HM00] has linear depth. This yields a protocol whose complexity grows exponentially with the number of parties, since when traversing the formula we increased the complexity of the protocol by a constant multiplicative factor (corresponding to the number of operations in the 3-party protocol) at every layer.

To overcome the exponential blowup, we replace the formula of [HM00] by a *logarithmic-depth* formula (which computes the majority function using only Maj_3 gates). Using the formula-based protocol described above, the logarithmic depth results in an efficient protocol, namely one whose complexity only grows polynomially with the number of parties. In Section 6.3 we describe the construction of a good "approximation" of such a formula as well as exact constructions under standard complexity-theoretic assumptions.

This approach is indeed very general and can be used in different models of secure MPC. For example, it can be used to obtain both passive security as outlined above and active security by using an underlying 4-party protocol that is secure against one active party and a log-depth threshold formula composed of two-out-of-four threshold gates (denoted by Th_2^4) which we also construct (see Section 6.3).

In fact, this reduction gives us a "cookbook" for designing secure multiparty protocols. The first step is to design a protocol for a constant number of parties that is secure against one dishonest party and the second step is to use a logarithmic-depth threshold from thresholds formula to obtain an efficient multiparty protocol that is secure against a constant fraction of corrupted parties.

We demonstrate the generality of this approach by deriving protocols in both passive and active settings and in different MPC models which differ in the type of underlying algebraic structure, including models for which no protocols were known. We also obtain conceptually simple protocols for classical problems in distributed computing such as broadcast protocols.

Simplified feasibility results. The classical results of Ben-Or et al. [BGW88] and Chaum et al. [CCD88] allow n parties to evaluate an arbitrary function, using secure point-to-point channels, with perfect security against $t < n/2$ passively corrupted parties or $t < n/3$ actively corrupted parties. We can derive conceptually simpler variants of these results by applying our approach with π being a 3-party or 4-party instance of the simple MPC protocol of Maurer [Mau06]. On the one hand our results are slightly weaker because they either need the threshold t to be slightly sub-optimal or alternatively require (standard) complexity theoretic assumptions to construct an appropriate formula for implementing the protocol. It is instructive to note that the complexity of Maurer's protocol grows exponentially with the number of parties. Our approach makes this a

[4] A monotone function $f : \{0,1\}^n \to \{0,1\}$ is said to be of type Q_d if $f(x_1) = f(x_2) = \ldots = f(x_d) = 0$ implies that $x_1 \vee x_2 \vee \ldots \vee x_d \neq 1^n$.

non-issue, as we only use the protocol from [Mau06] with a constant number of parties.[5]

MPC over blackbox algebraic structures. There has been a considerable amount of work on implementing MPC protocols for computations over different algebraic structures such as fields, rings, and groups. Algebraic computations arise in many application scenarios. While it is possible in principle to emulate each algebraic operation by a sequence of boolean operations, this is inefficient both in theory and in practice. In particular, the communication complexity of the resulting protocols grows with the computational complexity of the algebraic operations rather than just with the bit-length of the inputs and outputs. This overhead can be avoided by designing protocols which make a *black-box* (i.e., oracle) use of the underlying structure. The advantage of such protocols is that their communication complexity and the number of algebraic operations they employ are independent of the complexity of the structure.

MPC over rings and k-linear maps. The work of Cramer *et al.* [CFIK03] shows how to efficiently implement secure MPC over blackbox *rings*. We obtain a simpler derivation of such a protocol by noting that the simple protocol of Maurer [Mau06] directly generalizes to work over a blackbox ring. As before, one could not apply this protocol directly because its complexity is exponential in the number of parties. We show how to use a similar approach for obtaining the first blackbox feasibility results for MPC over *k-linear maps*.

MPC over groups. The problem of MPC over blackbox *groups* was introduced by Desmedt *et al.* [DPSW07] and further studied in [SYT08, DPS+12b, DPS12a]. [6]

To apply our approach in the group model, we need to specify the atomic protocol π that we use. For the case of passive security, we directly construct a simple 3-party protocol that has security against one corrupted party. This protocol is loosely based on a protocol by Feige *et al.* [FKN94] and considerably simplifies the 3-party instance of a general result from [DPS+12b].

In the active security model, we rely on the recent work of [DPS12a] who obtain the first MPC protocols with active security in the group model. The complexity of the protocol of [DPS12a] grows exponentially with the number of parties. However, we only need to employ the [DPS12a] protocol for four parties and so we do not suffer the exponential blowup. Thus, we settle the main problem left open in [DPS12a] by applying our technique to an instance of their results.

[5] While in the present work we apply our approach only to perfectly secure protocols, one could apply a similar technique to derive the result of Rabin and Ben-Or [RBO89], namely a statistically secure protocol which tolerates $t < n/2$ actively corrupted parties.

[6] Interestingly, group-based MPC with low security threshold was implicitly used in the recent work of Miles and Viola on leakage-resilient circuits [MV13]. It seems likely that efficient group-based MPC protocols with near-optimal security threshold, such as those obtained in our work, can be useful in this context.

We also obtain the first *two-party* MPC protocols over blackbox groups. In the passive corruption model, we combine a group product randomization technique due to Kilian [Kil88] with a "subset sum" based statistical secret sharing of group elements. We then get security against active corruptions by combining this two-party protocol with our efficient n-party protocol for the active model via the IPS compiler [IPS08].

Broadcast. Broadcast is one of the most basic problems in distributed computing. Recall that in a broadcast protocol a broadcaster wants to send a message to all other parties. A broadcast protocol should end with all parties holding the same value, even if some of the parties, possibly including the broadcaster, behave adversarially. Obtaining efficient broadcast protocols is a highly nontrivial task [PSL80, Dol82, GM98]. Our generic approach for MPC protocols can be used to directly construct simple broadcast protocols for $t < n/3$ corrupted parties. We also get a simplified proof of a result of Fitzi and Maurer [FM00], showing that an ideal primitive allowing broadcast for 3 parties (so-called 2-cast) implies broadcast with $t < n/2$ corrupted parties. Our proof technique also yields broadcast for the more general case of Q_2 adversaries which was previously an open problem.

6.3 Threshold Formulae from Threshold Gates

Motivated by the above applications to MPC, we consider the problem of constructing a logarithmic-depth threshold formula from threshold gates. Before discussing the general problem, we first discuss the special case of constructing a logarithmic depth formula composed of Maj_3 gates that computes the majority function. Note that this is exactly the type of formula required in the setting of *passive* MPC security.

6.3.1 Majority from majorities.

A closely related problem was considered by Valiant [Val84] who proved the existence of a logarithmic-depth monotone formula that computes the majority function where the formula uses And and Or gates, both of fan-in 2. As noted independently by several authors [Mil92, GM96, Zwi96, Gol11b], a slight modification of Valiant's argument shows the existence of a logarithmic-depth formula composed of Maj_3 gates that computes the majority function.

Valiant's proof is based on the probabilistic method and is non-constructive. Namely, the proof only assures us of the existence of a formula with the above properties, but does not hint on how to find it efficiently. Motivated by the applications presented in Section 6.2, we ask whether Valiant's proof can be derandomized using only Maj_3 gates and no constants.[7] We raise the following conjecture:

[7] We cannot allow the use of the constant 0, as this would correspond to assuming parties to be incorruptible. The use of the constant 1 alone is not helpful in our context.

Conjecture 6.1 (Majority from Majorities)**.** *There exists an algorithm A that given an odd integer n as input, runs in* $\mathrm{poly}(n)$*-time and generates a formula F on n inputs, with the following properties:*

- *F consists only of* Maj_3 *gates and no constants.*
- $\mathsf{depth}(F) = O(\log n)$.
- *F computes the majority function on n inputs.*

A derandomization for Valiant's proof for formulas over And and Or gates follows from the seminal paper of Ajtai, Komlós and Szemerédi [AKS83], though the latter does not seem to imply a derandomization in the context of Maj_3 gates, where constants are not allowed.[8]

In this work we make a significant progress towards proving Conjecture 6.1. In particular, we prove that relaxed variants of the conjecture hold. In addition, we show that the conjecture follows from standard complexity assumptions, namely, $\mathsf{E} \triangleq \mathsf{DTIME}(2^{O(n)})$ does not have $2^{\varepsilon n}$-size circuits for some constant $\varepsilon > 0$. Note that the latter follows from the existence of exponentially hard one-way functions.[9]

6.3.2 Threshold formulae from threshold gates.

Motivated by applications to the active MPC setting, and being a natural complexity-theoretic problem on its own, we initiate the study of a generalization of the majority from majorities problem, which we call *the threshold from thresholds problem*.

For integers $2 \leq j \leq k$, define the threshold function $\mathsf{Th}^k_j : \{0,1\}^k \to \{0,1\}$ as follows. $\mathsf{Th}^k_j(x) = 1$ if and only if the Hamming weight of x is at least j. Note that $\mathsf{Maj}_3 = \mathsf{Th}^3_2$.

Unlike the majority from majorities problem, it is not a priori clear what threshold function, if any, can be computed by a log-depth formula composed only of Th^k_j gates, even if no explicit construction is required. We make significant progress also on this question. Roughly speaking, we provide an explicit construction of a logarithmic depth formula composed solely of Th^k_j gates, that well approximates $\mathsf{Th}^n_{n/m}$, where $m = \frac{k-1}{j-1}$. For further details, see Section 6.4.3.

6.4 Our Results

We first describe the applications of our approach in cryptography and distributed computing, and then proceed to the complexity-theoretic results.

[8] Note that And and Or gates can be implemented using Maj_3 gates and constants.

[9] We find it curious that *perfectly secure* MPC results are based on the existence of (sufficiently strong) one-way functions.

6.4.1 Cryptographic results

We start by stating known results that we re-derive using our approach, and later state our new results.

In the passive Ring-MPC model, we get the following results.

- If the majority from majorities conjecture (Conjecture 6.1) holds then we obtain an explicit MPC protocol that has optimal security in the passive model. That is, it is secure as long as at most a $\frac{1}{2} - \Omega(\frac{1}{n})$ fraction of the n parties (more precisely, $t < n/2$) are passively corrupted. As noted above and stated formally in Theorem 6.2, Conjecture 6.1 follows from widely-believed conjectures in complexity theory and cryptography.

- An unconditional explicit and close to optimal protocol in the passive model in which the fraction of dishonest parties is at most $\frac{1}{2} - 2^{-O(\sqrt{\log n})}$ out of the n parties (in contrast to the optimal threshold of $\frac{1}{2} - \Omega(\frac{1}{n})$).

- A randomized construction of an optimal protocol in the passive model. By randomized construction we mean that the protocol is constructed by a randomized algorithm which may fail with negligible (undetectable) probability, but otherwise outputs the description of a perfect protocol.

We obtain the following result in the *active* Ring-MPC model.

- An explicit but non-optimal protocol that is secure against any *active* adversary that controls at most a $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of the n parties (in contrast to the optimal bound of $\frac{1}{3} - \Omega(\frac{1}{n})$).

Next we state our new results in the blackbox group model, introduced by Desmedt *et al.* [DPSW07, DPS+12b]. In this model the function computed by the protocol is specified by an arithmetic circuit over a (possibly non-Abelian) group, and the parties are restricted to making blackbox access to the group. (This includes oracle access to the group operation, taking inverses, and sampling random group elements.) In particular, the number of group operations performed by the protocol should not depend on the structure of the group or the complexity of implementing a group operation using, say, a Boolean circuit.

- **Group-MPC, passive:** The best explicit protocol of [DPS+12b] offers perfect security against a $\frac{1}{n^\varepsilon}$ fraction of passively corrupted parties, for any constant $\varepsilon > 0$, where n is the total number of parties. We improve upon the latter by constructing an explicit protocol that has perfect security against an (almost optimal) $\frac{1}{2} - 2^{-O(\sqrt{\log n})}$ fraction of passively corrupted parties. Alternatively, we get an optimal bound of $\frac{1}{2} - \Omega(\frac{1}{n})$ assuming the majority from majorities conjecture, via a non-uniform construction, or under standard derandomization or cryptographic assumptions. Lastly, we also obtain a protocol with an optimal bound of $\frac{1}{2} - \Omega(\frac{1}{n})$ with a running time that is only quasi-polynomial in the number of parties .

- **Group-MPC, active:** In a recent work, Desmedt *et al.* [DPS12a] constructed a secure MPC protocol in the group model with security against an *active* adversary. However, their result only gives a protocol whose complexity depends exponentially on the number of parties, regardless of the corruption threshold. We construct an *efficient* secure MPC protocol in the group model where an active adversary can control (an almost optimal) $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of the n parties.

- **Secure two-party computation over groups:** We construct the first secure *two-party* protocols over blackbox groups. Our protocols offer statistical security against active corruptions (assuming an oblivious transfer oracle) and rely on the afforementioned n-party protocols over black-box groups.

Finally, our protocols for the Ring-MPC model described above can be generalized to yield the following new result for MPC over k-linear maps.

- **MPC over k-linear maps:** We show that, for any constant k and any basis B of k-linear maps over finite Abelian groups, there are efficient MPC protocols for computing circuits over B which only make blackbox access to functions in B and group operations. This generalizes previous results for MPC over blackbox rings [CFIK03], which follow from the case $k = 2$, and can potentially be useful in cryptographic applications that involve complex bilinear or k-linear maps. These protocols are perfectly secure against a $\frac{1}{k} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of passively corrupted parties or a $\frac{1}{k+1} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of actively corrupted parties.

6.4.2 Distributed computing results

Broadcast. It is well known that broadcast can be implemented over point-to-point channels if and only if less than a third of the parties are actively corrupted [PSL80, Dol82] or, more generally, if and only if no three of the subsets the adversary may corrupt cover the entire set of parties [HM00, FM98], a so called Q_3-adversary.

In this work we show that a trivial broadcast protocol for 4 parties where one is actively corrupted easily implies the result of [FM98] using existing constructions of (super-logarithmic depth) formulae. Substituting instead our own logarithmic depth formula constructions implies a simple polynomial-time broadcast protocol for less than $n(\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}}))$ corrupted parties.

Broadcast from 2-cast. In [FM00], Fitzi and Maurer identify a minimal primitive that allows to improve the $\frac{n}{3}$ corruption threshold: if we are given the ability to broadcast among any subset of 3 parties for free, a so-called *2-cast* primitive, then broadcast becomes possible when less than $\frac{n}{2}$ parties are corrupted. It is natural to ask whether 2-cast also implies broadcast secure against general Q_2-adversaries (where no two corruptible subsets cover the entire set of parties). This problem was previously open.

We apply our approach to construct broadcast protocols based on a 2-cast primitive. Together with existing constructions of (super-logarithmic depth) formulae composed of

$\mathsf{majority}_3$-gates, this immediately implies a construction of broadcast from 2-cast for every Q_2-adversary, resolving the above problem. Substituting instead our logarithmic-depth formula constructions, we get a simplified derivation of polynomial-time protocols for the case of an honest majority considered in [FM00]. We do not know if the formula based approach also implies the results in [CFF+05], which consider generalizations of the 2-cast primitive.

6.4.3 Complexity-theoretic results

In this section we describe our results on constructing threshold formulae from threshold gates. For the special case of computing majority from Maj_3 gates we obtain stronger results which we state first.

Majority from majorities

Our first complexity-theoretic result shows that given a small promise on the bias of the input (defined as the difference between the normalized Hamming weight and 1/2), Conjecture 6.1 holds.

Theorem 6.1. *There exists an algorithm A that given an odd integer n as input, runs in $\mathrm{poly}(n)$-time and computes a formula F on n inputs, with the following properties:*

- *F consists only of Maj_3 gates and no constants.*
- $\mathsf{depth}(F) = O(\log n)$.
- $\forall x \in \{0,1\}^n$ *such that* $\mathrm{bias}(x) \geq 2^{-O(\sqrt{\log n})}$ *it holds that* $F(x) = \mathsf{majority}(x)$.

We note that the proof of Theorem 6.1 also gives a construction of a formula that computes the majority function *exactly* (i.e., without a promise on the bias) but with depth that is only poly-logarithmic (rather than logarithmic). Our second result shows that under standard complexity hardness assumptions, Conjecture 6.1 holds.

Theorem 6.2. *If there exists an $\varepsilon > 0$ such that $\mathsf{E} \triangleq \mathsf{DTIME}(2^{O(n)})$ does not have $2^{\varepsilon n}$-size circuits then Conjecture 6.1 holds. In particular, if there exist exponentially hard one-way functions then Conjecture 6.1 holds.*[10]

In fact, the proof of Theorem 6.2 explicitly presents an algorithm for constructing a formula as in Conjecture 6.1 given the truth table of any function in E, on a suitable number of inputs, that cannot be computed by $2^{\varepsilon n}$-size circuits. Moreover, the assumption made in Theoerem 6.2 can be relaxed.

[10] A one-way function f is *exponentially hard* if there exists an $\varepsilon > 0$ such that every family of $2^{\varepsilon n}$-size circuits can invert f with only $2^{-\varepsilon n}$ probability. If there exists such a function f, then the language $\mathcal{L}_f$ is in E but does not have $2^{\varepsilon n}$-size circuits, where $\mathcal{L}_f = \{(y, x', 1^n) : y \text{ has a preimage of length } n \text{ under } f \text{ which starts with } x'\}$.

Thresholds formulae from threshold gates

Lemma 6.2. *There exists an algorithm A that given $t, j, k \in \mathbb{N}$ as input, where j, k are constants in t such that $j \geq 2$ and $k \geq 2j-1$,*[11] *runs in $\exp(t)$-time and generates a formula F with the following properties:*

- *F has $mt+1$ inputs, where $m = \left\lfloor \frac{k-1}{j-1} \right\rfloor$.*
- *F consists only of Th_j^k gates and no constants.*
- *$\mathsf{depth}(F) = O(t)$.*
- *$\forall x \in \{0,1\}^{mt+1}$ it holds that $F(x) = \mathsf{Th}_{t+1}^{mt+1}(x)$.*

Lemma 6.2 generalizes results of [AR63, HM00, BIW10], who proved it for particular values of j and k, and uses a similar technique. We note that the depth of the formula generated in Lemma 6.2 is linear, which is too large for our applications. Nevertheless, the following theorem, which uses Lemma 6.2 as a building block, shows that a formula with *logarithmic depth* can be generated efficiently assuming a sufficient "bias" on the input.

Theorem 6.3. *There exists an algorithm A that given $n, j, k \in \mathbb{N}$ as input, where j, k are constants in n such that $j \geq 2$ and $k \geq 2j-1$, runs in $\mathrm{poly}(n)$-time and generates a formula F on n inputs, with the following properties:*

- *F consists only of Th_j^k gates and no constants.*
- *$\mathsf{depth}(F) = O(\log n)$.*
- *$\forall x \in \{0,1\}^n$ with normalized Hamming weight at least $\frac{1}{m} + \Omega(\frac{1}{\sqrt{\log n}})$, it holds that $F(x) = 1$, where $m = \left\lfloor \frac{k-1}{j-1} \right\rfloor$.*
- *$\forall x \in \{0,1\}^n$ with normalized Hamming weight at most $\frac{1}{m} - \Omega(\frac{1}{\sqrt{\log n}})$, it holds that $F(x) = 0$.*

Note that Theorem 6.1 is not a special case of Theorem 6.3 (with $j = 2, k = 3$) as the required promise on the bias in Theorem 6.1 is exponentially smaller than that in Theorem 6.3.

We do not know whether an analog of Conjecture 6.1 is plausible for the threshold from thresholds problem, even without the time-efficiency requirement. Theorem 6.3 might serve as evidence for the affirmative. However, the probabilistic argument used in the majority from majorities problem (see, e.g., [Gol11b]) breaks for this more general case. We consider this to be an interesting open problem for future research.

[11] Throughout the paper we assume, without loss of generality, that $k \geq 2j-1$. The complementary case can be reduced to this one by using Th_{k-j+1}^k gates and interpreting 0 as 1 and vice versa.

6.5 Proof Overview of Complexity-Theoretic Results

In this section we give an overview of our complexity-theoretic constructions. For simplicity, we start by giving an overview of our construction of a logarithmic-depth formula composed of Maj_3 gates, and no constants, that computes the majority function for inputs with constant bias. That is, we informally describe an efficient algorithm that given n, ε as inputs, where $\varepsilon > 0$ is constant in n, outputs a logarithmic-depth formula with n inputs which computes the majority function correctly on inputs with bias at least ε. It is not hard to see that it is enough to construct a logarithmic-depth *circuit*, since such a circuit can be efficiently converted to an equivalent logarithmic-depth formula.

To this end, we design an algorithm called $\mathsf{ShrinkerGenerator}$ that given n, ε as inputs, generates a *constant-depth* circuit $\mathsf{Shrinker}$ with n inputs and $\frac{n}{2}$ outputs, composed of Maj_3 gates and no constants, such that

$$\forall x \in \{0,1\}^n \quad \mathrm{bias}(x) \geq \varepsilon \implies \mathrm{bias}(\mathsf{Shrinker}(x)) \geq \varepsilon.$$

Thus, $\mathsf{Shrinker}$ shrinks the number of variables to half while maintaining the bias, assuming the input has a sufficiently large bias. By repeatedly calling $\mathsf{ShrinkerGenerator}$ on inputs $n, \frac{n}{2}, \frac{n}{4}, \ldots, 2$ (with the same ε) and concatenating the resulting circuits, one gets a logarithmic-depth circuit that computes the majority function assuming the input has large enough bias.

A key object we use in the design of $\mathsf{ShrinkerGenerator}$ is a Boolean sampler. Roughly speaking, a Boolean sampler is a randomized algorithm which on input $x \in \{0,1\}^n$ approximates the Hamming weight of x by reading only a small number of the bits of x. More precisely, a (d, ε, δ)-Boolean sampler is a randomized algorithm that on input $x \in \{0,1\}^n$ with normalized Hamming weight ω, samples at most d bits of x, and outputs $\beta \in [0,1]$ such that $\Pr[|\omega - \beta| \geq \varepsilon] \leq \delta$.

We will use a special type of samplers which take their samples in a non-adaptive fashion, and their output is simply the average of the sampled bits. For any $\varepsilon, \delta > 0$ there exist efficient (d, ε, δ)-Boolean samplers, with $d = O(\varepsilon^{-2} \cdot \delta^{-1})$, that on inputs of length n use only $\log n$ random bits.

Because such a sampler is non-adaptive and simply outputs the average of the sampled bits, it can be represented as a bipartite graph $G = (L, R, E)$, with $|L| = |R| = n$. For an input $x \in \{0,1\}^n$, the i'th vertex in L is labeled with the i'th bit of x. Each vertex in R represents one of the possible $\log n$ bit random strings used by the sampler. Each right vertex r is connected to the d left-vertices that are sampled by the algorithm when r is used as the random string.

The algorithm $\mathsf{ShrinkerGenerator}$ on inputs n, ε starts by constructing a graph G that represents a $(d, \frac{\varepsilon}{2}, \frac{1}{8})$-Boolean sampler, with $d = \mathrm{poly}(\frac{1}{\varepsilon}) = O(1)$. It then arbitrarily chooses half of the right vertices in G and discards the rest. This gives a bipartite graph $G' = (L', R', E')$ with $|L'| = n$, $|R'| = \frac{n}{2}$ and constant right-degree d. The circuit $\mathsf{Shrinker}$ that the algorithm $\mathsf{ShrinkerGenerator}$ outputs is given by placing a circuit that computes the majority function on d inputs for every right vertex. The inputs of this majority circuit are the neighbors of the respective right vertex. Note that as d is constant, a

constant-depth circuit that computes the majority function on d inputs can be found in constant time.

As for the correctness of the construction, assume now that $x \in \{0,1\}^n$ has some constant bias ε and, without loss of generality, assume that the bias is towards 1 (i.e., $\mathrm{wt}(x) \geq (\frac{1}{2}+\varepsilon)n$). Then, by the guarantee of the sampler, for all but $\frac{1}{8}$ of the right vertices in the original graph G, the fraction of neighbors with label 1 of a right vertex is at least $\frac{1}{2}+\varepsilon-\frac{\varepsilon}{2} > \frac{1}{2}$. Thus, all but $\frac{1}{8}$ of the (constant-size) majority circuits located in R output 1. Hence, the fraction of majority circuits that output 0 in R' is at most $\frac{n/8}{n/2} = \frac{1}{4} \leq \frac{1}{2}-\varepsilon$, as desired.

6.5.1 Supporting sub-constant bias

For sub-constant ε, the sampler technique described above is wasteful, as it requires us to use a sequence of $O(\log n)$ layers with fan-in $O(\varepsilon^{-2})$. For sub-constant ε, this results in a circuit with a super-logarithmic depth. However, we observe that one layer of fan-in $O(\varepsilon^{-2})$ circuits is enough to amplify the bias from ε to 0.4 (rather than just keep the bias at ε). This reduces us to the constant bias case, which can be solved as above with an additional $O(\log n)$-depth.

Thus, in order to obtain an $O(\log n)$-depth circuit on n inputs, that computes majority correctly for inputs with bias at least ε, it is enough to construct an $O(\log n)$-depth circuit with $O(\varepsilon^{-2})$ inputs that computes majority correctly on all inputs.

Using a naive brute-force algorithm, one can efficiently find an optimal-depth circuit on roughly $\log n$ inputs that computes majority. By plugging this circuit into the above scheme, one immediately gets an $O(\log n)$-depth circuit that computes majority on n inputs with bias roughly $\varepsilon = \Omega(\frac{1}{\sqrt{\log n}})$.

We improve on this by using an additional derandomization idea. Specifically, we construct an $O(\log n)$-depth circuit on $2^{O(\sqrt{\log n})}$ inputs, that computes majority (under no assumption on the bias). Thus, we obtain an explicit construction of a circuit that computes majority assuming the bias is at least $\varepsilon = 2^{-O(\sqrt{\log n})}$.

We first describe a randomized construction of an $O(\log m)$-depth circuit on m inputs for majority, where m is set, in hindsight, to $2^{O(\sqrt{\log n})}$. Our construction only uses $O(\log^2 m)$ random bits (compared to $\mathrm{poly}(m)$ random bits used in Valiant's construction). We then show how to derandomize this construction.

Our randomized construction works as follows. Consider an input $x \in \{0,1\}^m$ with bias ε. Suppose that we sample uniformly and independently at random 3 bits of x and compute their majority. It is shown in [Gol11b] that the majority's bias is at least 1.2ε (as long as ε is not too large).

Thus, by placing m majority gates of fan-in 3, and selecting their inputs from x uniformly and independently at random, the output of the m majority gates will have bias of at least 1.1ε with overwhelming probability. By composing $O(\log(1/\varepsilon))$ such layers, we can amplify the bias to a constant. Note that this construction uses $O(m \cdot \log m \cdot \log(1/\varepsilon))$ random bits.

To save on the number of random bits used (which is essential for the derandomization

step), instead of sampling the inputs of each one of the m gates uniformly at random, we choose them in each layer using a 6-wise independent hash function. While 3-wise independence suffices for the expectation of the bias to be as before, the 6-wise independence guarantees that the outputs of the majority gates in each layer are pairwise independent. Using tail inequalities we show that, with probability $1 - o(1)$, the bias increases in each layer as before.

By composing $O(\log(1/\varepsilon))$ such layers, each of which requires $O(\log m)$ random bits, we obtain a circuit as desired. The total number of random bits used is $O(\log(m) \cdot \log(1/\varepsilon))$, which is bounded by $O(\log^2 m)$. We derandomize the construction by placing all $2^{O(\log^2 m)}$ majority circuits that can be output by the randomized construction and taking the majority vote of these circuits.

Since we have a guarantee that almost all (a $1 - o(1)$ fraction) of the circuits correctly compute majority, it is enough to compute the majority vote at the end using a circuit with $2^{O(\log^2 m)}$ inputs that works for, say, constant bias. Such a circuit, with depth $O(\log^2 m)$, can be constructed in time $2^{O(\log^2 m)}$ by the constant-bias scheme described earlier.

As we set $m = 2^{O(\sqrt{\log n})}$, we get a $\mathrm{poly}(n)$-time uniform construction of an $O(\log n)$-depth circuit on $2^{O(\sqrt{\log n})}$ inputs that computes majority correctly on all inputs. This circuit is then used in the scheme described above.

Threshold formulae from thresholds gates. The scheme described above works also in the more general setting of threshold from thresholds. Indeed, in [CDI+13] we present the scheme in the general setting. To apply the scheme in the thresholds setting, one needs to construct a small circuit that computes the required threshold formula, to be used by ShrinkerGenerator. We accomplish this by extending results of [AR63, HM00, BIW10].

6.6 Preliminaries for the Complexity Theoretic Results

Let $x \in \{0,1\}^n$. We denote the Hamming weight of x by $\mathrm{wt}(x)$. The normalized Hamming weight is denote by $\mathsf{relwt}(x)$, that is, $\mathsf{relwt}(x) = \mathrm{wt}(x)/n$. We further denote the bias of x by $\mathrm{bias}(x) = |\mathsf{relwt}(x) - 1/2|$. Let $G = (V, E)$ be an undirected graph. The degree of a vertex v is denoted by $d(v)$. For a set $S \subseteq V$ define $d_S(v) = |\{s \in S : sv \in E\}|$.

Circuits and Formulae. Let C be a circuit on n inputs that outputs m bits. For $x \in \{0,1\}^n$, we denote by $C(x) \in \{0,1\}^m$ the output of C when fed with x as input. Let C_1 be a circuit on n inputs that outputs m bits. Let C_2 be a circuit on m inputs that outputs r bits. We denote by $C_2 \circ C_1$ the circuit on n inputs and r outputs that is composed of C_1 and C_2, where the m outputs of C_1 are wired to the m inputs of C_2. Clearly, $C_2 \circ C_1(x) = C_2(C_1(x))$.

The size of a circuit C, denoted by $\mathsf{size}(C)$, is the number of gates in the circuit. The depth, denoted by $\mathsf{depth}(C)$, is the largest number of gates from an input to an output

in C.

In this paper we focus on logarithmic-depth formulae composed of constant fan-in threshold gates. Note that such formulae always have polynomial size.

Observation 6.3. *A logarithmic-depth* circuit *composed of constant fan-in gates can be efficiently transformed to a logarithmic-depth* formula *composed of constant fan-in gates, that computes the same function.*

Indeed, one can work his way bottom-up and duplicate every gate with fan-out $k > 1$, together with its sub-formula, to k copies. As the duplication does not change the depth of a gate, the resulting formula has logarithmic depth (and so, due to the constant fan-in, a polynomial size). Thus, in all of our theorems we are satisfied with constructing logarithmic-depth circuits composed of constant fan-in gates.

For integers $0 \leq j \leq k$ define the threshold function $\mathsf{Th}_j^k : \{0,1\}^k \to \{0,1\}$ as follows. $\mathsf{Th}_j^k(x) = 1$ if and only if $\mathrm{wt}(x) \geq j$. Define $\mathsf{majority}_{2t+1} = \mathsf{Th}_{t+1}^{2t+1}$. That is, $\mathsf{majority}_{2t+1}$ computes the majority function on $2t+1$ inputs. When the number of variables is clear from the context we omit it from the subscript and write $\mathsf{majority}$.

From bipartite graphs to circuits. The following notation will be useful for us. Let $G = (L, R, E)$ be a bipartite graph with right-degree $mt + 1$. Define the circuit $C_{G,m,t}$ as follows. $C_{G,m,t}$ has $|L|$ inputs and $|R|$ outputs. With every vertex $r \in R$ associate a threshold circuit Th_{t+1}^{mt+1} in $C_{G,m,t}$. The inputs to this circuit are the $mt+1$ neighbors of r in G. The outputs of $C_{G,m,t}$ are the output of the $|R|$ threshold circuits, one for each vertex in R. When m, t are clear from the context we omit them from the subscript and write C_G.

Samplers. Samplers are key pseudorandom objects we use in our constructions. In this paper we only use a special kind of samplers, known in the literature as *non-adaptive averaging Boolean samplers*. For brevity we call them samplers. For more details we refer the reader to a survey by Goldreich [Gol11a]. It will be useful for us to define samplers in terms of bipartite graphs.

Definition 6.4 (Samplers)**.** *An* $(n, d, \varepsilon, \delta)$-sampler *is a bipartite graph* $\mathsf{Sampler} = (L, R, E)$ *with the following properties:*

- $|L| = |R| = n$.
- $\mathsf{Sampler}$ *has right-degree* d.
- *For every* $S \subseteq L$*, it holds that for at least* $1 - \delta$ *fraction of the vertices* $r \in R$*,*

$$\left| \frac{d_S(r)}{d(r)} - \frac{|S|}{n} \right| \leq \varepsilon.$$

Theorem 6.4 ([KPS85, Gol11a])**.** *For every* $n \in \mathbb{N}$ *and for every* $\varepsilon = \varepsilon(n) > 0$*,* $\delta = \delta(n) > 0$*, there exists an* $(n, d, \varepsilon, \delta)$*-sampler* $\mathsf{Sampler}$*, with* $d = O(\varepsilon^{-2} \cdot \delta^{-1})$*. Moreover,* $\mathsf{Sampler}$ *can be constructed in time* $\mathrm{poly}(n, \varepsilon^{-1}, \delta^{-1})$*.*

6.7 Threshold Formulae from Threshold Gates

In this section we prove Theorem 6.3. Recall that we assume without loss of generality that $j \geq 2$ and $k \geq 2j-1$. We start by proving Lemma 6.2.

6.7.1 Proof of Lemma 6.2

To prove Lemma 6.2 we recall a couple of definitions and prove helpful lemmas.

Definition 6.5 (Q_m functions)**.** *Let $m \in \mathbb{N}$. A function $f : \{0,1\}^n \to \{0,1\}$ is a Q_m* function *if for every $x^{(1)}, \ldots, x^{(m)} \in f^{-1}(0)$ there exists an $h \in [n]$ such that $x_h^{(1)} = \cdots = x_h^{(m)} = 0$.*

Lemma 6.6. *Let $j, k \in \mathbb{N}$. Every function that can be computed by a formula composed only of Th_j^k gates and no constants is a Q_m function, where $m = \left\lfloor \frac{k-1}{j-1} \right\rfloor$.*

Proof. We prove the lemma by induction on the depth of the formula. Let F be a depth 0 formula composed of Th_j^k gates and without constants. Then, F computes the function $F(x) = x_h$ for some index h. Clearly this is a Q_m function.

Let F be a formula composed of Th_j^k gates without constants, where $\mathsf{depth}(F) > 0$. Let G be the output gate of F. Let $G_1, \ldots, G_k$ be the gates that have their output wired to the inputs of G. By induction, for every $i \in [k]$, the function computed by the sub formula of F with output gate G_i is a Q_m function.

Let x be an input rejected by F. Since G is a Th_j^k gate, at most $j-1$ of the G_i's accept one of the m inputs. Thus, for every m inputs rejected by F, at most $m(j-1) \leq k-1$ of the gates G_i's accept (at least) one of the inputs. Hence, there exists an $\ell \in [k]$ such that G_ℓ rejects all of those m inputs. The proof follows by applying the induction hypothesis on the sub-circuit with output gate G_ℓ. □

Lemma 6.7. *There exists an algorithm A that given constants $j, k \in \mathbb{N}$, where $m = \left\lfloor \frac{k-1}{j-1} \right\rfloor$, and the truth table of a Q_m function $f : \{0,1\}^n \to \{0,1\}$ as input, runs in $\exp(n)$-time and generates a formula F on n inputs, with the following properties:*

- *F consists only of Th_j^k gates and no constants.*
- *$\mathsf{depth}(F) = O(n)$.*
- *$\forall x \in f^{-1}(0)$ it holds that $F(x) = 0$.*

Proof. The algorithm A is recursive. The base case for the recursion is $|f^{-1}(0)| \leq m$. In this case, since f is a Q_m function, there exists an $h \in [n]$ such that for every $x \in f^{-1}(0)$ it holds that $x_h = 0$. The algorithm A can find such h in $\exp(n)$-time and return the formula that on input x outputs x_h.

Assume now that $|f^{-1}(0)| \geq m+1$. Partition the set $f^{-1}(0)$ into k sets

$$f^{-1}(0) = A_0 \cup \cdots \cup A_{k-1}$$

of size $\lceil |f^{-1}(0)|/k \rceil$ or $\lfloor |f^{-1}(0)|/k \rfloor$ each, such that $|A_0| \geq |A_1| \geq \cdots \geq |A_{k-1}|$. Let $t \leq k-1$ be the maximum integer such that $A_t \neq \emptyset$. Note that $t \geq m+1$ because if there is an empty set among $A_0, \ldots, A_t$ then every non-empty set in the partition has size exactly 1, but their union has size $|f^{-1}(0)| \geq m+1$.

Let $G = (L, R, E)$ be a bipartite graph with $L = \{0, 1, \ldots, k-1\}$, $R = \{0, 1, \ldots, t\}$. A vertex $r \in R$ is connected by edge to the vertices $\{(j-1)r, (j-1)r+1, \ldots, (j-1)r+j-2\} \subseteq L$, where addition is modulo k. By construction, the degree of every right vertex is $j-1$. Moreover, the degree of every left vertex is at least 1 since $(j-1)t + j - 2 \geq k-1$. Indeed,

$$(j-1)t + j - 2 \geq (j-1)(m+1) + j - 2 \geq (j-1)\left(\left\lfloor \frac{k-1}{j-1} \right\rfloor + 1\right) + j - 2 \geq k - 1.$$

For $\ell = 0, 1, \ldots, k-1$ define the function $f_\ell : \{0,1\}^n \to \{0,1\}$ to be such that

$$f_\ell^{-1}(0) = f^{-1}(0) \setminus \left(\bigcup_{r : \ell r \in E} A_r \right).$$

For every $\ell = 0, 1, \ldots, k-1$, the algorithm A makes a recursive call on inputs j, k and the truth table of f_ℓ. Denote by $F_0, \ldots, F_{k-1}$ the formulae generated by these recursive calls. Define F to be the formula where the outputs of $F_0, \ldots, F_{k-1}$ are wired to the inputs of a Th_j^k gate, which is the output gate of F. Note that the recursion will eventually get to the base case as the degree of every left vertex in G is at least 1, and so, for every ℓ, $|f_\ell^{-1}(0)|$ is strictly smaller than $|f^{-1}(0)|$.

Let $x \in f^{-1}(0)$. We now show that $F(x) = 0$. Let $r \in \{0, 1, \ldots, t\}$ be the unique integer such that $x \in A_r$. As the degree of r in G is $j-1$, there are exactly $j-1$ functions among $f_0, \ldots, f_{k-1}$ that accept x. Thus, by induction, there are at most $j-1$ formulae among $F_0, \ldots, F_{k-1}$ that accept x. Hence, $F(x) = 0$.

We now analyze the depth and running time of the algorithm. Fix $\ell \in \{0, 1, \ldots, k-1\}$. If $|f^{-1}(0)| \geq 2k$ it holds that

$$\begin{aligned} |f_\ell^{-1}(0)| &\leq |f^{-1}(0)| - (j-1) \cdot \left\lfloor \frac{|f^{-1}(0)|}{k} \right\rfloor \\ &\leq \left(1 - \frac{j-1}{k}\right) \cdot |f^{-1}(0)| + j - 1 \\ &\leq \left(1 - \frac{1}{2k}\right) |f^{-1}(0)|. \end{aligned}$$

That is, $f_\ell^{-1}(0)$ has size which is a constant fraction, strictly smaller than 1, of the size of $f^{-1}(0)$, as long as $|f^{-1}(0)| \geq 2k$. Together with the fact that $|f^{-1}(0)| \leq 2^n$, this implies that the depth of the recursion is $O(n) + 2k = O(n)$. We note that the depth of F is exactly the recursion's depth, and so $\mathsf{depth}(F) = O(n)$.

For a function $g : \{0,1\}^n \to \{0,1\}$, let $\mathsf{rt}(g)$ be the running time of A when given as input j, k and the truth table of g. Then,

$$\mathsf{rt}(f) = \sum_{i=0}^{k-1} \mathsf{rt}(f_i) + \exp(n),$$

where the exponential time in n is due to the computation of $A_0, \ldots, A_{k-1}$ from $g^{-1}(0)$. Solving the recursion yields $\mathsf{rt}(f) = \exp(n)$. □

Proof of Lemma 6.2. We first note that Th_{t+1}^{mt+1} is a Q_m function. Let F be the formula generated by the algorithm in Lemma 6.7 on input j, k and the truth table of the function Th_{t+1}^{mt+1}. By Lemma 6.7, every $x \in \{0,1\}^{mt+1}$ such that $\mathrm{wt}(x) \leq t$ is rejected by F.

We now show that every $x \in \{0,1\}^{mt+1}$ such that $\mathrm{wt}(x) \geq t+1$ is accepted by F. Assume for contradiction that there exists $x^{(1)} \in \{0,1\}^{mt+1}$ such that $\mathrm{wt}(x^{(1)}) \geq t+1$ but $x^{(1)}$ is rejected by F. Then, there exist $x^{(2)}, \ldots, x^{(m)} \in \{0,1\}^{mt+1}$, each of weight at most t, such that for every $h \in [mt+1]$ there exists an $i \in [m]$ such that $x_h^{(i)} = 1$. On the other hand, since F is composed of Th_j^k gates, by Lemma 6.6, the function computed by F is a Q_m function. This contradicts the existence of such $x^{(1)}$. Thus, $F(x) = \mathsf{Th}_{t+1}^{mt+1}(x)$ for every $x \in \{0,1\}^{mt+1}$.

The depth of F as well as the running time of A follows by Lemma 6.7. □

6.7.2 Proof of Theorem 6.3

In this section we prove Theorem 6.3. A key step in proving Theorem 6.3 is the following lemma.

Lemma 6.8. *Suppose that there exists an algorithm A' that given $t, j, k \in \mathbb{N}$ as input, where j, k are constants in t, runs in $T'(t)$-time and generates a circuit C' on $mt+1$ inputs, where $m = \left\lfloor \frac{k-1}{j-1} \right\rfloor$, with the following properties:*

- *C' consists only of Th_j^k gates and no constants.*
- *$\mathsf{size}(C') = S'(t)$.*
- *$\mathsf{depth}(C') = D'(t)$.*
- *$\forall x \in \{0,1\}^{mt+1}$, $C'(x) = \mathsf{Th}_{t+1}^{mt+1}(x)$.*

Then, there exists an algorithm A that given n, t, j, k as input, where j, k are constants in n, runs in $\mathrm{poly}(n, T'(O(t)))$-time and generates a circuit C on n inputs, with the following properties:

- *C consists only of Th_j^k gates and no constants.*
- *$\mathsf{size}(C) = O\left(n \cdot S'(O(t))\right)$.*
- *$\mathsf{depth}(C) = D'(O(t)) + O(\log n)$.*

- $\forall x \in \{0,1\}^n$ *such that* $\mathsf{relwt}(x) \geq \frac{1}{m} + \frac{1}{\sqrt{t}}$ *it holds that* $C(x) = 1$.
- $\forall x \in \{0,1\}^n$ *such that* $\mathsf{relwt}(x) \leq \frac{1}{m} - \frac{1}{\sqrt{t}}$ *it holds that* $C(x) = 0$.

To prove Lemma 6.8 we need the following key lemma. Informally, Lemma 6.9 shows how to construct a circuit that shrinks the number of variables to half while maintaining a promise on the weight.

Lemma 6.9. *There exists an algorithm* ShrinkerGenerator *that given* $n \in \mathbb{N}$ *as input as well as constants* $\varepsilon > 0$ *and* $j, k \in \mathbb{N}$ *such that* $0 < \varepsilon \leq \frac{1}{2m}$, *where* $m = \left\lfloor \frac{k-1}{j-1} \right\rfloor$, ShrinkerGenerator *runs in* $\mathrm{poly}(n)$*-time and generates a circuit* Shrinker *with the following properties:*

- Shrinker *has* n *inputs and* $\frac{n}{2}$ *outputs.*
- Shrinker *consists only of* Th^k_j *gates and no constants.*
- $\mathsf{size}(\mathsf{Shrinker}) = O(n)$.
- $\mathsf{depth}(\mathsf{Shrinker}) = O(1)$.
- $\forall x \in \{0,1\}^n$ *such that* $\mathsf{relwt}(x) \geq \frac{1}{m} + \varepsilon$ *it holds that* $\mathsf{relwt}(\mathsf{Shrinker}(x)) \geq \frac{1}{m} + \varepsilon$.
- $\forall x \in \{0,1\}^n$ *such that* $\mathsf{relwt}(x) \leq \frac{1}{m} - \varepsilon$ *it holds that* $\mathsf{relwt}(\mathsf{Shrinker}(x)) \leq \frac{1}{m} - \varepsilon$.

Proof. Let $\mathsf{Sampler} = (L, R, E)$ be an $(n, d, \frac{\varepsilon}{2}, \frac{1}{4m})$-sampler. By Theorem 6.4, Sampler can be constructed in $\mathrm{poly}(n, \varepsilon^{-1}, m) = \mathrm{poly}(n)$-time with $d = O(\varepsilon^{-2}m) = O(1)$. Consider the circuit C_{Sampler}. Let $t \in \mathbb{N}$ be such that $d = mt + 1$.[12] Since $d = O(1)$, by Lemma 6.2, the circuit C_d, used in C_{Sampler}, that computes the function Th^{mt+1}_{t+1} can be generated in $O(1)$-time. Clearly, the size and depth of C_d are constants.

The circuit Shrinker is defined to be the circuit C_{Sampler}, where we arbitrarily choose half of the outputs and discard the rest, together with the respective copies of C_d. By construction, Shrinker has n inputs and $\frac{n}{2}$ outputs. It consists of Th^k_j gates and no constants. Clearly, $\mathsf{size}(\mathsf{Shrinker}) = \frac{n}{2} \cdot \mathsf{size}(C_d) = O(n)$ and $\mathsf{depth}(\mathsf{Shrinker}) = \mathsf{depth}(C_d) = O(1)$.

Let $x \in \{0,1\}^n$ be such that $\mathsf{relwt}(x) \geq \frac{1}{m} + \varepsilon$. Define the set $S_x = \{i \in [n] : x_i = 1\}$. Note that $|S_x| \geq (\frac{1}{m} + \varepsilon)n$. By identifying L with $[n]$, we get by the definition of Sampler, that for at least $1 - \frac{1}{4m}$ fraction of the vertices $r \in R$ it holds that

$$\frac{d_{S_x}(r)}{d(r)} \geq \frac{|S_x|}{n} - \frac{\varepsilon}{2} \geq \frac{1}{m} + \frac{\varepsilon}{2}.$$

Thus, the copy of C_d in C_{Sampler} that is associated with such r gets an input with weight at least

$$\left(\frac{1}{m} + \frac{\varepsilon}{2}\right) \cdot (mt + 1) > t.$$

[12] For simplicity of presentation we assume $d \equiv 1 \pmod{m}$. This assumption can be met easily.

Thus, the output of C_d that is associated with such r is 1, and so $\mathsf{relwt}(C_{\mathsf{Sampler}}(x)) \geq 1 - \frac{1}{4m}$. Hence, for any subset of size $\frac{n}{2}$ of the outputs of C_{Sampler}, at least $1 - \frac{1}{2m}$ fraction of them are equal to 1. That is,

$$\mathsf{relwt}(\mathsf{Shrinker}(x)) \geq 1 - \frac{1}{2m} \geq \frac{1}{m} + \varepsilon,$$

where the last inequality follows since $m \geq 2$ and $\varepsilon \leq \frac{1}{2m}$.

Let $x \in \{0,1\}^n$ be such that $\mathsf{relwt}(x) \leq \frac{1}{m} - \varepsilon$. In this case $|S_x| \leq (\frac{1}{m} - \varepsilon)n$. By the definition of $\mathsf{Sampler}$, we have that for at least $1 - \frac{1}{4m}$ fraction of the vertices $r \in R$ it holds that

$$\frac{d_{S_x}(r)}{d(r)} \leq \frac{|S_x|}{n} + \frac{\varepsilon}{2} \leq \frac{1}{m} - \frac{\varepsilon}{2}.$$

Thus, the copy of C_d in C_{Sampler} that is associated with such r gets an input with weight at most

$$\left(\frac{1}{m} - \frac{\varepsilon}{2}\right) \cdot (mt + 1) < t + 1.$$

Thus, the output of C_d that is associated with such r is 0, and so $\mathsf{relwt}(C_{\mathsf{Sampler}}(x)) \leq \frac{1}{4m}$. Hence, for any subset of size $\frac{n}{2}$ of the outputs of C_{Sampler}, at most $\frac{1}{2m}$ fraction of them are equal to 1. That is,

$$\mathsf{relwt}(\mathsf{Shrinker}(x)) \leq \frac{1}{2m} \leq \frac{1}{m} - \varepsilon,$$

where the last inequality follows since $\varepsilon \leq \frac{1}{2m}$. □

Proof of Lemma 6.8. Let $\mathsf{Sampler} = (L, R, E)$ be an $(n, d, \frac{1}{2\sqrt{t}}, \frac{1}{2m})$-sampler. By Theorem 6.4, $\mathsf{Sampler}$ can be constructed in $\mathrm{poly}(n, t, m) = \mathrm{poly}(n, t)$-time with $d = O((2\sqrt{t})^2 \cdot (2m)) = O(t)$. Consider the circuit C_{Sampler}. Let t' be an integer such that $d = mt' + 1$. Note that $t' = O(t)$. By calling A' on input t', j, k, the algorithm A can generate the circuit C' on d inputs, used in C_{Sampler} in $T'(t') = T'(O(t))$-time. Moreover, $\mathsf{size}(C') = S'(t') = S'(O(t))$ and $\mathsf{depth}(C') = D'(t') = D'(O(t))$.

For an integer ℓ, let $\mathsf{Shrinker}_\ell$ be the circuit generated by $\mathsf{ShrinkerGenerator}$ on input $\ell, \frac{1}{2m}, j, k$ (see Lemma 6.9). The algorithm A generates $\mathsf{Shrinker}_\ell$ for $\ell = \frac{n}{2^i}$, where $i = 0, 1, \ldots, \log_2(n) - 1$. The output of A is the circuit

$$C = \mathsf{Shrinker}_2 \circ \cdots \circ \mathsf{Shrinker}_{\frac{n}{4}} \circ \mathsf{Shrinker}_{\frac{n}{2}} \circ \mathsf{Shrinker}_n \circ C_{\mathsf{Sampler}}.$$

Let $x \in \{0,1\}^n$ be such that $\mathsf{relwt}(x) \geq \frac{1}{m} + \frac{1}{\sqrt{t}}$. Define $S_x = \{i \in [n] : x_i = 1\}$. Note that $|S_x| \geq (\frac{1}{m} + \frac{1}{\sqrt{t}}) \cdot n$. By identifying L with $[n]$, and by the definition of $\mathsf{Sampler}$, for at least $1 - \frac{1}{2m}$ fraction of the vertices $r \in R$ it holds that

$$\frac{d_{S_x}(r)}{d(r)} \geq \frac{|S_x|}{n} - \frac{1}{2\sqrt{t}} \geq \frac{1}{m} + \frac{1}{2\sqrt{t}}.$$

Thus, the copy of C' in C_{Sampler} that is associated with such r gets an input with weight at least

$$\left(\frac{1}{m}+\frac{1}{2\sqrt{t}}\right)(mt'+1) > t'.$$

Thus, the output of C' that is associated with such r is 1. Hence,

$$\mathsf{relwt}(C_{\mathsf{Sampler}}(x)) \geq 1-\frac{1}{2m} \geq \frac{1}{m}+\frac{1}{2m}, \tag{6.1}$$

where the last inequality follows since $m \geq 2$. Define $x^{(1)} = \mathsf{Shrinker}_n\left(C_{\mathsf{Sampler}}(x)\right) \in \{0,1\}^{n/2}$. By the definition of $\mathsf{Shrinker}_n$ and by Equation (6.1), we have that $\mathsf{relwt}(x^{(1)}) \geq \frac{1}{m}+\frac{1}{2m}$. Similarly, $x^{(2)} \triangleq \mathsf{Shrinker}_{n/2}(x^{(1)}) \in \{0,1\}^{n/4}$ has weight at least $\frac{1}{m}+\frac{1}{2m}$. Continuing this way we get that $C(x) \in \{0,1\}$ has weight at least $\frac{1}{m}+\frac{1}{2m}$. As $C(x)$ is a single bit it follows that $C(x)=1$.

Let $x \in \{0,1\}^n$ be such that $\mathsf{relwt}(x) \leq \frac{1}{m}-\frac{1}{\sqrt{t}}$. In this case $|S_x| \leq (\frac{1}{m}-\frac{1}{\sqrt{t}})\cdot n$. By the definition of $\mathsf{Sampler}$, for at least $1-\frac{1}{2m}$ fraction of the vertices $r \in R$ it holds that

$$\frac{d_{S_x}(r)}{d(r)} \leq \frac{|S_x|}{n}+\frac{1}{2\sqrt{t}} \leq \frac{1}{m}-\frac{1}{2\sqrt{t}}.$$

Thus, the copy of C' in C_{Sampler} that is associated with such r gets an input with weight at most

$$\left(\frac{1}{m}-\frac{1}{2\sqrt{t}}\right)(mt'+1) < t'+1.$$

Thus, the output of C' that is associated with such r is 0. Hence,

$$\mathsf{relwt}(C_{\mathsf{Sampler}}(x)) \leq \frac{1}{2m} = \frac{1}{m}-\frac{1}{2m}. \tag{6.2}$$

Define $x^{(1)} \triangleq \mathsf{Shrinker}_n\left(C_{\mathsf{Sampler}}(x)\right) \in \{0,1\}^{n/2}$. By the definition of $\mathsf{Shrinker}_n$ and by Equation (6.2) we have that $\mathsf{relwt}(x^{(1)}) \leq \frac{1}{m}-\frac{1}{2m}$. Similarly, $x^{(2)} \triangleq \mathsf{Shrinker}_{\frac{n}{2}}(x^{(1)}) \in \{0,1\}^{n/4}$ has weight at most $\frac{1}{m}-\frac{1}{2m}$. Continuing this way we get that $C(x) \in \{0,1\}$ has weight at most $\frac{1}{m}-\frac{1}{2m}$. Since $C(x)$ is a single bit, it follows that $C(x)=0$.

We now analyze the size and depth of C.

$$\begin{aligned}
\mathsf{size}(C) &= \mathsf{size}\left(C_{\mathsf{Sampler}}\right) + \sum_{i=0}^{\log n} \mathsf{size}\left(\mathsf{Shrinker}_{\frac{n}{2^i}}\right) \\
&= n \cdot S'(O(t)) + O\left(\sum_{i=0}^{\log n} \frac{n}{2^i}\right) \\
&= O\left(n \cdot S'(O(t))\right),
\end{aligned}$$

and

$$\mathsf{depth}(C) = \mathsf{depth}\left(C_{\mathsf{Sampler}}\right) + \sum_{i=0}^{\log n} \mathsf{depth}\left(\mathsf{Shrinker}_{\frac{n}{2^i}}\right) = D'(O(t)) + O(\log n).$$

□

Theorem 6.3 readily follows by Lemma 6.2 and Lemma 6.8.

Proof of Theorem 6.3. By calling the algorithm from Lemma 6.2 on input $t = \log n, j, k$, the algorithm A can generate in $\exp(t) = \text{poly}(n)$-time an $O(t) = O(\log n)$-depth formula on $mt + 1 = O(\log n)$ inputs, composed of Th_j^k gates with no constants, that computes the function Th_{t+1}^{mt+1}. By Lemma 6.8 this implies that in time $\text{poly}(n, \exp(t)) = \text{poly}(n)$, A can generate a circuit C on n inputs that is composed of Th_j^k gates, with the following property. For $x \in \{0,1\}^n$ such that $\mathsf{relwt}(x) \geq \frac{1}{m} + \Omega(\frac{1}{\sqrt{\log n}})$, $C(x) = 1$, and for $x \in \{0,1\}^n$ such that $\mathsf{relwt}(x) \leq \frac{1}{m} - \Omega(\frac{1}{\sqrt{\log n}})$, $C(x) = 0$. Moreover $\mathsf{depth}(C) = O(t + \log n) = O(\log n)$.

By Observation 6.3, the circuit C can be transformed in $\text{poly}(n)$-time to a formula with the same depth, that computes the same function. □

6.8 Majority Formulae from Majority Gates

In this section we prove Theorem 6.1 and Theorem 6.2.

6.8.1 Proof of Theorem 6.1

Theorem 6.1 readily follows by Lemma 6.8 together with the following lemma.

Lemma 6.10. *There exists an algorithm A that given an integer n as input, runs in $\text{poly}(n)$-time and computes a circuit C on $m \triangleq 2^{\sqrt{\log n}}$ inputs, with the following properties:*

- *C consists only of Maj_3 gates and no constants.*
- *$\mathsf{size}(C) = \text{poly}(n)$.*
- *$\mathsf{depth}(C) = O(\log n)$.*
- *$\forall x \in \{0,1\}^m$ it holds that $C(x) = \mathsf{majority}(x)$.*

To prove Lemma 6.10 we need the following definitions and results on bounded independence distributions. The following well known lemma gives a concentration bound for a sequence of pairwise independent random variables (see, e.g., [Vad11]).

Lemma 6.11. *Let $X_1, \ldots, X_m \in_R [0,1]$ be a sequence of pairwise independent random variables. Let*

$$X = \frac{X_1 + \cdots + X_m}{m},$$

and let $\mu = \mathbb{E}[X]$. Then,

$$\Pr\left[|X - \mu| \geq \varepsilon\right] \leq \frac{1}{m\varepsilon^2}.$$

Definition 6.12 (k-Wise Independent Hash Functions)**.** *Let $n, m, k \in \mathbb{N}$. A family of functions $\mathcal{H} = \{h : [n] \to [m]\}$ is called* k-wise independent *if for every distinct $x_1, \ldots, x_k \in [n]$, the random variables $h(x_1), \ldots, h(x_k)$, where h is sampled uniformly from $\mathcal{H}$, are independent and uniformly distributed in $[m]$.*

Theorem 6.5. *For every $n, m, k \in \mathbb{N}$ there exists an explicit construction of a k-wise independent family of hash functions $\mathcal{H} = \{h : [n] \to [m]\}$, with size $|\mathcal{H}| = O(\max\{n, m\}^k)$.*

A proof for Theorem 6.5 can be found in, e.g., [Vad11]. To prove Lemma 6.10 we also need the following fact proved by Goldreich [Gol11b] in his exposition of Valiant's proof. The fact roughly states that the output of a Maj_3 gate, applied to three uniformly sampled entries of a (biased) string x, has a higher bias than the bias of x.

Fact 6.13. *Let n be an odd integer. Let $x \in \{0,1\}^n$ and let $\varepsilon = \mathrm{bias}(x)$. Define*

$$\varepsilon' \triangleq \Pr_{i,j,k \sim [n]} [\mathsf{Maj}_3(x_i, x_j, x_k) = \mathsf{majority}(x)] - \frac{1}{2}.$$

Then, $\varepsilon' = 1.5\varepsilon - 2\varepsilon^3$. In particular, if $\varepsilon \leq 0.4$ then $\varepsilon' > 1.15\varepsilon$. Moreover, for every $\varepsilon \in [0, 1/2]$ it holds that $\varepsilon' \geq \varepsilon$.

The following notation will be useful for us. Let $m \in \mathbb{N}$. For a function $f : [3m] \to [m]$, define the bipartite graph $G_f = (L, R, E)$, with $|L| = |R| = m$ as follows. Associate L and R with $[m]$, and connect every vertex $r \in R$ with the vertices $f(3r-2), f(3r-1)$ and $f(3r)$ in L. Note that the right-degree of G_f is 3. We also define $C_f \triangleq C_{G_f}$.

Lemma 6.14. *Let $m \in \mathbb{N}$, and let $\mathcal{H} = \{h : [3m] \to [m]\}$ be the 6-wise independent family of hash functions from Theorem 6.5. Then, for every $x \in \{0,1\}^m$ such that $\mathrm{bias}(x) \geq \frac{1}{m^{1/4}}$:*

$$\Pr_{h \sim \mathcal{H}} \left[\mathrm{bias}\left(C_h(x)\right) \leq \min\left\{1.1 \cdot \mathrm{bias}(x), 0.4\right\} \right] = O\left(\frac{1}{\sqrt{m}}\right).$$

Proof. Fix $x \in \{0,1\}^m$. Since $\mathcal{H}$ is a 6-wise independent family of hash functions, it is in particular 3-wise independent. As the majority gates in C_h have 3 inputs each, we can apply Fact 6.13 and conclude that

$$\mathbb{E}_{h \sim \mathcal{H}} \left[\mathrm{bias}\left(C_h(x)\right) \right] \geq \min\left\{1.15 \cdot \mathrm{bias}(x), 0.4\right\}.$$

By the 6-wise independence of $\mathcal{H}$, the outputs of the majority gates in the circuit C_h, where h is sampled uniformly from $\mathcal{H}$, are pairwise independent. Thus, by Lemma 6.11 and since $\mathrm{bias}(x) \geq \frac{1}{m^{1/4}}$,

$$\Pr_{h \sim \mathcal{H}} \left[\mathrm{bias}\left(C_h(x)\right) < \min\left\{1.1 \cdot \mathrm{bias}(x), 0.4\right\} \right] \leq \frac{1}{m \cdot (0.05 \cdot \mathrm{bias}(x))^2} = O\left(\frac{1}{\sqrt{m}}\right).$$

□

Proof of Lemma 6.10. We first describe the circuit C and then prove its correctness.

As a first step, the circuit C replicates each one of its m inputs m^3 times to obtain a total of $m' = m^4$ wires. Note that the bias of these wires is at least $\frac{1}{m} = \frac{1}{(m')^{1/4}}$.

Let $\mathcal{H} = \{h : [3m'] \to [m']\}$ be the 6-wise independent family of hash functions from Theorem 6.5. Recall that $\mathcal{H}$ has size $O((m')^6) = \text{poly}(m)$, and can be constructed in $\text{poly}(m)$-time. Let $h_1, \ldots, h_\ell$ be functions sampled from $\mathcal{H}$ uniformly and independently at random, where ℓ is the smallest integer that satisfies the inequality $1.1^\ell \geq 0.4m$.

Let C' be the circuit generated by the algorithm from Theorem 6.3 on input $m', j = 2, k = 3$. The circuit C' computes the majority function correctly on inputs with bias at least $o(1)$ and thus certainly when the bias is at least 0.4. Moreover, C' has $\text{poly}(m)$-size, $O(\log m)$-depth, and can be constructed in $\text{poly}(m)$-time. For $h_1, \ldots, h_\ell \in \mathcal{H}$, define the circuit

$$C_{h_1,\ldots,h_\ell} \triangleq C' \circ C_{h_\ell} \circ \cdots \circ C_{h_1}.$$

Let C'' be the circuit generated by the algorithm in Theorem 6.3 on input $|\mathcal{H}|^\ell, j = 2, k = 3$. Since $|\mathcal{H}|^\ell = m^{O(\log m)} = \text{poly}(n)$, the circuit C'' has $\text{poly}(n)$-size, $O(\log n)$-depth, and can be constructed in $\text{poly}(n)$-time. C'' computes the majority function correctly on inputs with bias at least 0.1 (and, in fact, even for bias at least $o(1)$).

After the replication step, the m' replicated wires are wired as inputs to the circuit $C_{h_1,\ldots,h_\ell}$ for every ℓ-tuple $h_1, \ldots, h_\ell \in \mathcal{H}$. Wire the output of each such circuit to an input of C''. The output of C is defined to be the output of C''. By the above, C can be constructed in $\text{poly}(n)$-time. This implies that $\mathsf{size}(C) = \text{poly}(n)$. Since

$$\mathsf{depth}(C_{h_1,\ldots,h_\ell}) = \mathsf{depth}(C') + \sum_{i=1}^{\ell} \mathsf{depth}(C_{h_i}) = O(\log m) + \ell \cdot O(1) = O(\log m)$$

for every $h_1, \ldots, h_\ell \in \mathcal{H}$, and since $\mathsf{depth}(C'') = O(\log n)$, we have that $\mathsf{depth}(C) = O(\log n)$.

We now show that C computes the majority function. Let $x \in \{0,1\}^m$ and let $x' \in \{0,1\}^{m'}$ be the resulted replicated string. By Lemma 6.14, and by applying a union bound,

$$\Pr_{h_1,\ldots,h_\ell \sim \mathcal{H}} [\text{bias}\left(C_{h_\ell} \circ \cdots \circ C_{h_1}(x')\right) < 0.4] < \frac{\ell}{\sqrt{m'}} = O\left(\frac{\log m}{m^2}\right) < 0.1.$$

Thus, by the definition of C',

$$\Pr_{h_1,\ldots,h_\ell \sim \mathcal{H}} [C_{h_1,\ldots,h_\ell}(x') = \mathsf{majority}(x)] \geq 0.9.$$

That is, at least a 0.9 fraction of the circuits $C_{h_1,\ldots,h_\ell}$ output $\mathsf{majority}(x)$. This implies that C'', and thus C, outputs $\mathsf{majority}(x)$.

□

Proof of Theorem 6.1. By Lemma 6.10, a circuit of $\text{poly}(n)$-size and $O(\log n)$-depth, that computes the majority function on $2^{\sqrt{\log n}}$ inputs can be generated in $\text{poly}(n)$-time. By

applying Lemma 6.8 we obtain, in poly(n)-time, an $O(\log n)$-depth circuit that computes the majority under the assumed promise on the bias. The proof then follows by Observation 6.3. □

6.8.2 Proof of Theorem 6.2

In this section we prove Theorem 6.2. We first recall a few definitions and results.

Definition 6.15. *Let $n, s \in \mathbb{N}$. A distribution $\mathcal{R}$ over $\{0,1\}^n$ is* s-pseudorandom *if for every circuit C on n inputs with size at most s, it holds that*

$$|\Pr[C(\mathcal{R}) = 1] - \Pr[C(U_n) = 1]| < 0.1.$$

Definition 6.16. *Let $S : \mathbb{N} \to \mathbb{N}$ be a time-constructible function [13]. A function $\mathsf{PRG} : \{0,1\}^* \to \{0,1\}^*$ is an* S*-pseudorandom generator [14] if*

- $\forall r \in \{0,1\}^*$, $|\mathsf{PRG}(r)| = S(|r|)$.
- *There exists an algorithm that, given $r \in \{0,1\}^*$, runs in time $2^{O(|r|)}$ and outputs $\mathsf{PRG}(r)$.*
- $\forall s \in \mathbb{N}$, *the distribution $\mathsf{PRG}(U_s)$ is $(S(s))^5$-pseudorandom.*

In a long line of research initiated by Nisan and Wigderson [NW94] (with some of the ideas, in the context of cryptography, dating back to Yao, Blum and Micali [Yao82c, BM84]), it has been shown that pseudorandom generators can be constructed under hardness assumptions. We state a more recent theorem by Umans [Uma03].

Theorem 6.6 ([Uma03])**.** *Let $S : \mathbb{N} \to \mathbb{N}$. Given the truth table of a function $f : \{0,1\}^s \to \{0,1\}$ that cannot be computed by a circuit with size at most $S(s)$, there exists an $S(s)^{\Omega(1)}$-pseudorandom generator.*

We also make use of the following theorem.

Theorem 6.7 ([Val84], see also [Gol11b])**.** *There exists a constant c_1 such that the following holds. For every $n \in \mathbb{N}$ there exists a family of formulae $\mathcal{F}_n = \left\{F_w^{(n)} : w \in \{0,1\}^{n^{c_1}}\right\}$, each on n inputs, such that for every $w \in \{0,1\}^{n^{c_1}}$,*

- $F_w^{(n)}$ *consists only of* Maj_3 *gates and no constants.*
- $\mathsf{size}(F_w^{(n)}) = O(n^{c_1})$.
- $\mathsf{depth}(F_w^{(n)}) = O(\log n)$,

[13] A function $S : \mathbb{N} \to \mathbb{N}$ is *time constructible* if $\forall s \in \mathbb{N}$ it holds that $S(s) \geq s$, and there exists a Turing machine that on input 1^s outputs $1^{S(s)}$ in $O(S(s))$-time.

[14] Note that this definition refers to pseudorandom generators in the context of computational complexity rather than cryptography.

and

$$\Pr_{w\sim\{0,1\}^{n^{c_1}}}\left[\forall x\in\{0,1\}^n \;\; F_w^{(n)}(x)=\mathsf{majority}(x)\right]>1-2^{-n}.$$

Moreover, given $n\in\mathbb{N}, w\in\{0,1\}^{n^{c_1}}$, the formula $F_w^{(n)}$ can be constructed in $O(n^{2c_1})$-time.

Proof of Theorem 6.2. By Theorem 6.6, the assumption of Theorem 6.2 implies the existence of a $2^{s/c_2}$-pseudorandom generator PRG, for some constant $c_2>1$. Set $s=c_1c_2\log_2 n$, where c_1 is the constant from Theorem 6.7. Note that the PRG outputs n^{c_1} bits.

We first describe the circuit C and then turn to prove its correctness. Let C' be the circuit generated by the algorithm from Theorem 6.3 on input $n^{c_1c_2}, j=2, k=3$. The circuit C' computes the majority correctly on inputs with bias at least 0.1 (and, in fact, $o(1)$). Moreover, C' has $O(\log n)$-depth, and can be generated in $\mathrm{poly}(n)$-time.

Define the circuit C on n inputs as follows. The n inputs of C are wired to the inputs of the formula $F_w^{(n)}$ for every $w\in\{\mathsf{PRG}(r):r\in\{0,1\}^s\}$. The output of each such formula is wired to an input of C'. The output of C is defined to be the output of C'.

For every $r\in\{0,1\}^s$, the string $w=\mathsf{PRG}(r)$ can be computed in $\exp(s)=\mathrm{poly}(n)$-time. By Theorem 6.7, given $n\in\mathbb{N}$ and $w\in\{0,1\}^{n^{c_1}}$, the formula $F_w^{(n)}$ can be generated in $\mathrm{poly}(n)$-time. Since there are $2^s=n^{c_1c_2}$ such formulae, and since C' can be generated in $\mathrm{poly}(n)$-time, the circuit C can be generated in $\mathrm{poly}(n)$-time. By Theorem 6.3 and Theorem 6.7, $\mathsf{depth}(C')=O(\log n)$ and for every $w\in\{0,1\}^{n^{c_1}}$, $\mathsf{depth}(F_w^{(n)})=O(\log n)$. Hence, $\mathsf{depth}(C)=O(\log n)$, as stated.

We now prove that C computes the majority function. Assume, for contradiction, that there exists $x_0\in\{0,1\}^n$ such that $C(x_0)\neq\mathsf{majority}(x_0)$. Then, by the construction of C,

$$\Pr_{r\sim\{0,1\}^s}\left[F_{\mathsf{PRG}(r)}^{(n)}(x_0)=\mathsf{majority}(x_0)\right]\leq 0.6. \tag{6.3}$$

By Theorem 6.7, given $n\in\mathbb{N}, w\in\{0,1\}^{n^{c_1}}$, the formula $F_w^{(n)}$ can be constructed in $O(n^{2c_1})$-time. Therefore, there exists a circuit $C_{\mathrm{universal}}$ of size $O(n^{4c_1})$ that has $n^{c_1}+n$ inputs, such that for every $w\in\{0,1\}^{n^{c_1}}, x\in\{0,1\}^n$ it holds that $C_{\mathrm{universal}}(w,x)=F_w^{(n)}(x)$.[15] Consider the circuit C_0 on n^{c_1} inputs defined by hard wiring x_0 as x to $C_{\mathrm{universal}}$. Clearly, $C_0(w)=C_{\mathrm{universal}}(w,x_0)$ for every $w\in\{0,1\}^{n^{c_1}}$. By Equation (6.3),

$$\Pr_{r\sim\{0,1\}^s}\left[C_0(\mathsf{PRG}(r))=\mathsf{majority}(x_0)\right]\leq 0.6.$$

On the other hand, by Theorem 6.7,

$$\Pr_{w\sim\{0,1\}^{n^{c_1}}}\left[C_0(w)=\mathsf{majority}(x_0)\right]\geq 1-o(1).$$

This yields a contradiction as C_0, which has size $O(n^{4c_1})<(n^{c_1}+n)^5$, distinguishes with probability $0.4-o(1)>0.1$ a random string from a random output of PRG. □

[15] The quadratic overhead is due to the simulation of an algorithm by a family of circuits; see, e.g., Theorem 6.6 in [AB09].

Weakening the Assumption in Theorem 6.2. One can show that the assumption in Theorem 6.2 (i.e., the existence of an $\varepsilon > 0$ such that $\mathsf{E} \triangleq \mathsf{DTIME}(2^{O(n)})$ does not have $2^{\varepsilon n}$-size circuits) can be relaxed to the following assumption: there exists a pseudorandom generator with seed length $O(\log n)$ for read once branching programs of length n and width $O(n)$.[16] The latter assumption is implied by the assumption that appears in Theorem 6.2.

The state of the art pseudorandom generator for read once branching programs has seed length $O(\log^2 n)$ [Nis92, INW94]. We stress that this construction is *unconditional* (i.e., no computational assumptions are necessary). We also note that it is possible to use this pseudorandom generator to give an alternative proof for Lemma 6.10, and any advancement in the construction of pseudorandom generators for read once branching programs is a step forward in resolving Conjecture 6.1. We omit the details in this version of the paper.

6.9 From Threshold Formulae to Broadcast

In this section, we show a simple way to construct broadcast protocols for n parties from protocols for a constant number of parties based on threshold and also more general formulae. Following the notation of [HM00], we differentiate here between a *processor* which is the entity doing the actual computation and communication and the *player* which is the entity that receives input and produces output. A *party* is the player together with the associated processor.

We will assume throughout that processors can communicate via synchronous secure point-to-point channels. The results and protocols in this section could also be phrased in the more general MPC framework we present later, but we have chosen not to do this, to make the section more self-contained and easier to read. We begin with some basic definitions:

Definition 6.17. *In a* Byzantine Agreement *(BA) protocol, each player P_i has input x_i and output y_i. All honest players must terminate and output the same value, and if $x_i = x$ for all honest P_i then it must be the case that $y_i = x$ for all honest P_i.*

In a Broadcast protocol *some designated player P starts from input x. All honest players must terminate and output the same value y, and if P is honest, it must be the case that $x = y$.*

A 2-cast protocol *is a broadcast protocol for three players.*

It is well known and easy to see that BA implies Broadcast: the broadcaster sends his message to all processors and then we do BA. Conversely, if less than $\frac{n}{2}$ processors are corrupt, then Broadcast implies BA: each processor broadcasts his input and then each processor takes majority decision to get the output. Note that in case the inputs are not bits but come from a larger set, it may happen that that no value is in majority (if the

[16] We do not give here a formal definition for read once branching programs. For more information the reader is referred to [AB09], Chapter 21, Section 6.

honest processors do not agree on the input to start with). In this case, we adopt in the following the convention that processors output $\perp$.

In the following we want to consider more general adversaries than those that are limited simply by the number of processors they can corrupt. For this we need some definitions:

Definition 6.18. *An* adversary structure *is a family Γ of subsets of the processors such that $A \in \Gamma$, $B \subseteq A$ implies $B \in \Gamma$. An adversary structure is Q_2 (resp., Q_3) if it is the case that no two (resp., no three) subsets in Γ cover the entire processor set. A Q_2-adversary is an adversary that may only corrupt sets of processor belonging to a Q_2 adversary structure. A Q_3-adversary is defined in the same way.*

As an example, an adversary who can corrupt less than $\frac{n}{2}$ of the processors is a special case of a Q_2-adversary.

Assume now we are given a monotone formula F on n inputs. If we let each input variable correspond to a processor, an input string corresponds to a subset of the processors in a natural way, by considering it as the characteristic vector of the subset. The family of subsets that are rejected by F under this correspondence forms an adversary structure because F is monotone. It is not hard to see by induction on the height of F that if it is composed of Th_2^3 gates, the corresponding adversary structure is Q_2, in fact the same arguments shows that any formula composed of Th_j^k gates where $j - 1 < k/2$ has a corresponding Q_2 adversary structure (a stronger statement follows by Lemma 6.6). Moreover, for Th_2^4 gates we have a stronger condition, namely they lead to Q_3 adversary structures. Conversely, it follows from [HM00] that for every Q_2 (Q_3) adversary structure Γ, there exists a formula composed of Th_2^3 (Th_2^4) gates that rejects exactly Γ. The formulas obtained this way are not necessarily logarithmic depth, in fact they are very large even for threshold functions.

6.9.1 Virtual processors and ormulas

Consider a formula F composed of Th_j^k gates with n inputs. We assume the formula is laid out as a k-ary tree with the output gate on top. As described in the introduction, we assign a virtual processor to the output wire of each gate, and a real processor to each input wire of F. A virtual processor is emulated by the k processors that "sit below him" in the formula.

A virtual processor is defined to be honest if at most $j - 1$ of his k emulators are corrupt. An honest virtual processor *holds a value* if all honest processors emulating him agree on that value.

Now assume for a moment that we are given BA for free as a primitive for any group of k processors emulating a virtual processor, where the BA is guaranteed to work if that virtual processor is honest. This immediately implies the following recursively defined protocol for sending a message x from a (virtual or real) processor S to another (virtual or real) processor R:

Message Transfer Protocol (MTP)

1. If S, R are real, send x from S to R.

2. If S is real and R is virtual, S sends x to each processor emulating R using MTP. The emulators do BA to agree on what was received.

3. If S is virtual and R is real, processors emulating S send x to R using MTP, and R takes majority decision to decide what was received.

4. Otherwise (both processors are virtual) each processor emulating S sends x to each processor emulating R using MTP. Each emulator takes majority decision on what he receives, and finally the emulators do BA to agree on what was received.

We will only use this protocol in cases where $j-1 < k/2$. This means that for an honest virtual processor, a majority of his emulators are honest. This and a straightforward inductive argument implies:

Lemma 6.19. *Consider a formula F composed of Th_j^k gates as above, where $j-1 < k/2$. Assume that we are given BA as a primitive for all groups of k processors emulating a virtual processor, where the BA is guaranteed to work if that virtual processor is honest. If S and R are both honest and run the Message Transfer Protocol, R will end up holding x. Even if S is not honest, an honest R always ends up holding some message.*

Under the same assumption as in the above lemma, we can build the following simple protocol for broadcasting a bit, based on formula F composed of Th_j^k gates:

Broadcast Protocol based on F.

1. The broadcaster sends his bit to the virtual processor sitting on top of the formula using MTP.

2. The virtual processor sitting on top of the formula now holds a bit. He sends it to each real processor using MTP.

Theorem 6.8. *Consider a formula F composed of Th_j^k gates as above, where $j-1 < k/2$. Assume that we are given BA as a primitive for all groups of k processors emulating a virtual processor, where the BA is guaranteed to work if that virtual processor is honest. If the set of corrupt real processors is rejected by F, then the Broadcast Protocol based on F is correct.*

Proof. Since F is built from Th_j^k gates where $j-1 < k/2$, the family of subsets it rejects must be Q_2 as noted above. Hence the set of honest real processors is accepted by F. This means that the virtual processor sitting on top of F is honest. The theorem now follows immediately from Lemma 6.19. □

6.9.2 Broadcast for less than $n/3$ corrupted parties.

As a warm-up we show how to get known results for broadcast in a simple way from the formula based approach.

Corollary 6.9. *The formula based approach implies a broadcast protocol for n parties secure if less than $n(\frac{1}{3} - \frac{1}{\sqrt{\log n}})$ parties are corrupted, and more generally a broadcast protocol secure against any Q_3-adversary.*

Proof. As noted above, there exists a formula composed of Th_2^4 gates that rejects exactly the given adversary structure. This formula satisfies the assumptions in Theorem 6.8, so all we have to do is to argue that we do BA for 4 processors where at most 1 is corrupt. But this is easy, we can get broadcast and hence BA as follows: the broadcaster P sends his message to all processors using the Message Transfer Protocol, and then each receiver sends to all receivers what he got from P. Finally, the receivers take majority decision to get the output. This construction implies that the BA protocol and the MTP are recursively defined in terms of each other, but this is not a problem, as each recursive call is always to the next lower level, so the recursion eventually "bottoms out". This implies the Q_3-result, and using the formulas constructed in Theorem 6.3, we get a polynomial time broadcast protocol for the case where at most t processors are corrupted and $t < n(\frac{1}{3} - \frac{1}{\sqrt{\log n}})$. □

This result also follows from the results in [HM00], albeit in a more complicated way. Also, Fitzi and Maurer [FM98] show a broadcast protocol for Q_3-adversaries that is polynomial time in n given access to an oracle that decides membership in the adversary structure.

6.9.3 From 2-cast to broadcast

It was shown by Fitzi and Maurer [FM00] that given 2-cast as a primitive for any group of 3 processors, one can get perfectly secure broadcast for n processors assuming less than $\frac{n}{2}$ are corrupted.

Here, we give an alternative and simple proof of this result, and moreover the same proof trivially extends to show that broadcast is possible in a more general case of a Q_2-adversary. This generalization was previously open.

It is of course easy to do Byzantine agreement (BA) among 3 processors where at least two are honest, if 2-cast is given. So an obvious approach is to do something similar as in the previous corollary: take a formula composed of Th_2^3 gates rejecting exactly the given Q_2 structure and apply Theorem 6.8. However, to satisfy the assumptions in that result, we need BA, or equivalently 2-cast, among any group of 3 processors, *even virtual processors*. But we are only given 2-cast among real processors.

We therefore need to construct a 2-cast protocol π_{2c} that works for virtual processors, i.e., it is a protocol for 9 processors split in 3 committees of 3 processors each, such that each committee emulates a virtual processor and at least 2 of the 3 virtual processors are honest. There will be a sending virtual processor S and two receivers R_0, R_1. π_{2c} may

use 2-cast among any 3-subset of the 9 emulating processors, and must ensure that 1) if S is honest and holds a message (a bit b), then the honest receiver(s) receive b; and 2) if S is not honest, then R_0 and R_1 (who must then be honest) hold the same bit after the protocol[17]. As a first step, we construct a 2-cast protocol π'_{2c}, where the sender is not the entire committee S, but one of the processors in S, here denoted by P_S.

Protocol π'_{2c}

1. P_S 2-casts his bit b to every pair of processors, where one processor is in R_0 and one is in R_1.

2. Each processor in R_0 or R_1 considers all received bits. If the same bit b was received in all 2-casts, store as temporary result b. Else store $\perp$.

3. Each committee R_i $(i = 0, 1)$ does BA using the temporary results as inputs. The result will be a bit b_i or $\perp$. The result is reported to the other committee using the message transfer protocol described above.

4. Each processor in R_i computes an output as follows: if your own committee reported b_i, output b_i. If your own committee reported $\perp$ and R_{1-i} reported b_{1-i}, output b_{1-i}. Otherwise (both committees reported $\perp$) output 0.

Lemma 6.20. *Protocol π'_{2c} implements correctly 2-cast from P_S to R_0 and R_1.*

Proof. If the sender P_S is honest and sends b, then if the committee R_i is honest, then the (at least 2) honest processors in R_i will only see b, therefore R_i will report b and honest processors from R_i will output b as they should no matter what happens in R_{1-i}.

If P_S is corrupt, we can assume that both R_0 and R_1 are honest and we must show that they output the same bit. Clearly, if at least one of R_0 and R_1 report $\perp$ or they both report the same bit b, then R_0 and R_1 output the same bit. We now show that the only remaining case where they report different bits cannot occur: assume without loss of generality that R_0 reports 0. In order for this to happen, at least one of the honest processors in R_0 must have 0 as temporary result. But this means (by the 2-casts we used in the first step) that all honest processors in R_1 see at least one 0. Therefore none of them can have 1 as temporary result, and R_1 therefore cannot report 1. □

To get the protocol π_{2c}, we simply run π'_{2c} 3 times with each processor in S playing the role of P_S, and processors in R_0, R_1 take majority decision on the bit to output. The properties we wanted for π_{2c} now follow immediately from the above lemma.

This and Theorem 6.8 now immediately implies

Corollary 6.10. *Given 2-cast among any group of 3 parties, there exists a broadcast protocol for n parties secure against any Q_2-adversary.*

[17] Note that we cannot rely on the result by Fitzi and Maurer to get such a protocol, since we do not have honest majority: it may be the case that only 4 of the 9 processors are honest.

Note that our results on construction of formulas for the majority function (Theorems 6.2, 6.1) together with the construction behind the corollary immediately gives polynomial time (in n) broadcast protocols for at most t corrupted processors. We can use a randomized construction or a construction conditioned on a complexity assumption to get the optimal result $t < \frac{n}{2}$, matching the result from [FM00]. Or we can use the (unconditional) explicit construction that gives us $t < n(\frac{1}{2} - 2^{-O(\sqrt{\log n})})$.

6.10 The Multiparty Computation Framework

In Section 6.10.1 we give a general overview of the player emulation technique and in Section 6.10.2 we give a formal definition of the MPC framework that we will use (based on the [HM00] MPC framework).

6.10.1 The player emulation technique

The formulas constructed in Section 6.7 and Section 6.8 will be used to construct efficient multiparty protocols via the "player emulation" technique from [HM00]. Variants of this technique, also referred to as player *virtualization* or *simulation*, were used for different purposes in several other works (e.g. [Bra87, Cha89, HIKN08, DIK+08, IPS08, LRM10, LOP11]). While implementing player emulation in the passive security model is quite straightforward, in the active security model it requires more care. In the following we give more details on the implementation of this technique.

Recall that in a single player emulation step, the role of a party τ participating in a protocol Π is replaced by a secure protocol π which involves a small set of parties $v_1, \ldots, v_k$, along with the parties of Π. We will typically let $k = 3$ (resp., $k = 4$) in the case of security against a passive (resp., active) adversary, and let π be a protocol which remains secure as long as at most one of the emulating parties v_i is corrupted. Furthermore, the total computational complexity of all parties in π (which is typically cast in some algebraic computation model) is only bigger by a constant factor than that of the emulated party τ in Π. As explained in the Introduction, a logarithmic-depth threshold formula defines a sequence of such player emulation steps which result in transforming an atomic protocol π for a constant number of parties into an efficient n-party protocol which tolerates an optimal or near-optimal fraction of corrupted parties.

The application of the player emulation technique in [HM00] is formulated in a specialized framework for secure MPC and is restricted to the protocol compiler of BGW [BGW88].[18] However, the technique is quite insensitive to many of these details and can be applied with other protocols and notions of security from the literature.

A conceptually simple way for implementing a player emulation step is by viewing the role of τ in Π as a *reactive* ideal functionality, which interacts with the parties in Π (receiving incoming messages as inputs and delivering outgoing messages as outputs), and

[18] In Section 6.10.2 we use a slightly stronger variant of the model from [HM00] which supports player emulation without any further requirements.

maintains a state information during this interaction. The protocol π emulating τ then needs to realize the corresponding functionality using the emulating parties v_i instead of τ. Note that protocol π does not only involve the players emulating τ. It also specifies how players communicating with τ should translate their messages into whatever format π uses.[19]

The protocol π can satisfy any composable notion of security that applies to reactive functionalities, namely one which ensures that π can be securely used as a substitute for τ in an arbitrary execution environment if at most a single v_i is corrupted. The protocols π we use in this work all satisfy the standard notion of UC-security from [Can01], which suffices for this purpose.[20]

Alternatively, it is possible to implement a player emulation step by only relying on protocols for secure function evaluation which satisfy the standard definitions of standalone security [Can00, Gol04]. The idea is to first ensure that only a single message is sent in each round of Π, and then implement a round in which τ interacts with party P by a protocol involving P and the emulating parties v_i. The functionality realized by such a protocol is determined by the choice of a concrete (robust) secret sharing scheme which is used to distribute the state of τ between the emulating parties.

6.10.2 The [HM00] MPC framework

We consider n players that wish to perform some computational task together. We focus only on perfect security and, as usual in this context, we assume secure point-to-point communication channels between every two processors. We introduce an abstract framework in which the specific operations that can be performed by the individual processors remains undefined. This allows us to obtain generic results that are applied in subsequent sections to concrete multiparty computation models that are instantiations of our abstract framework. Our definitions are based on [HM00], except that we make an additional "locality" requirement which is implicitly used in [HM00].

As in [HM00], we distinguish between a player and the corresponding processor. A *player* is the entity that receives input and produces output and the *processor* is in charge of executing operations and communicating. A *party* is a player together with the associated processor.

We assume a global variable space $\mathcal{X}$. A variable $x \in \mathcal{X}$ can take value from some fixed finite set of values $\mathcal{V}$. We associate with each variable a set of processors that know the value of the variable. We assume that variables are only used in a write-once manner

[19] Alternatively, if the communication channels are modeled as an ideal functionality, one can extend the definition of this functionality so it will do the translation, and then in a final step implement the translation. This leads in some cases to a slightly simpler protocol in the end.

[20] In particular, all these protocols are perfectly secure with a straight-line black-box simulator, which was shown in [KLR10] to imply UC-security in the case of secure function evaluation. We note that while standard UC-security is cast in an asynchronous network model and does not guarantee output delivery, it can be extended to capture synchronous protocols which guarantee output delivery (cf. [CDN12, Chapter 4]).

and therefore, transmitting a message between two processors simply entails of adding a variable that is in the sending processor's view to the receiving processor's view.

A *multiparty computation model* (or *MPC model*) defines the set of values $\mathcal{V}$ that may be assigned to variables and the different operators (and respective operands) that the processors may use. The MPC model may also specify that some of these variables have predetermined values and are treated as constants. Additionally the MPC model specifies the power of the adversary (i.e., whether it is passive or active, see Section 6.10.2). All the following definitions are with respect to some fixed MPC model $\mathcal{M}$ which will always be clear from the context.

A *protocol* π among a set P of processors, that involves variables from a variable space $\mathcal{X}$, is a sequence of statements. Each statement may be of the following forms:

1. An input statement $input(p_i, x)$ instructs the processor $p_i \in P$ to read a value from its input tape and to assign the value to its local variable $x \in \mathcal{X}$.

2. A transmit statement $transmit(p_1, p_2, x)$ instructs the processor $p_1 \in P$ to send the value of the variable $x \in \mathcal{X}$ (that is in its view) to the processor $p_2 \in P$. This simply means that x is added to the view of p_2.

3. An output statement $output(p, x)$ instructs the processor $p \in P$ to output the value of the variable $x \in \mathcal{X}$ to its associated processor. We define by $\mathsf{output}(p)$ the list of all values that p has output throughout the execution of the protocol and by $\mathsf{output}_i(p)$ the output of p before the execution of the i-th statement of the protocol.

4. A computation statement $comp(p, op, X, x)$ instructs the processor p to perform the atomic operation op on the variables $X \subseteq \mathcal{X}$ (that are in its view) and to assign the result to the variable $x \in \mathcal{X}$. This may include assigning a random value. The specification of the operation as well as its operands are a part of the MPC model $\mathcal{M}$.

A *multiparty computation specification* (or simply called *specification*) formally specifies the task to be accomplished by the processors. Intuitively, a specification specifies the cooperation in an ideal environment which includes, in addition to the n processors, a trusted processor. Formally, a specification is a pair (π, τ) consisting of a protocol π among a set P_0 of processors, and the name of a virtual processor $\tau \in P_0$. The protocol π of the specification is also called the ideal protocol. We assume that τ is never involved in *input* and *output* statements.

As an example, consider a specification described by a protocol in which all the parties first read their input, send their inputs to the trusted processor τ who computes some function of their input and transmits a result of this computation to each processor. Each processor outputs its own result. We note that this type of specification is known as secure function evaluation.

A *multiparty protocol generator* G for the set P_G of processors is a polynomial-time algorithm that takes as input a multiparty computation specification (π_0, τ) involving processors from a set P_0 and returns a protocol π for the processors $(P_0 \backslash \{\tau\}) \cup P_G$. A

statement index function for a specification (π_0, τ) and protocol π is a strictly monotone function f,

$$f : \{1, \ldots, |\pi_0| + 1\} \to \{1, \ldots, |\pi| + 1\}$$

where $f(1) = 1$ and $f(|\pi_0|+1) = |\pi|+1$. We say that a protocol generator is *local* if each statement α in π_0 is mapped to a sequence of statements in π such that the mapping does not depend on variables that do not appear in the statement α. All protocol generators considered in this work are local.

Intuitively, a protocol generator G simulates the virtual trusted processor τ by a multiparty computation protocol among the processors in P_G. Each statement of the ideal protocol π_0 is expanded into a sequence of statements, and all these sequences are concatenated to the resulting protocol π. The statement index function f specifies how statements in π_0 are expanded into subprotocol in π. Thus, each index i maps a statement in π_0 to the index $f(i)$ of the first statement in the corresponding subprotocol in π.

We consider different levels of explicitness of the protocol generator. If the protocol generator can be implemented by a polynomial-time Turing Machine, we say that it is *explicit*. If it can be implemented by a probabilistic polynomial-time Turing Machine that only fails with probability that is exponentially vanishing in the number of players then we say that it is a *randomized construction*. We will be mainly interested in protocol generators that only have *blackbox* access to an algebraic structure $\mathcal{V}$. This can be modeled by assuming that each element in $\mathcal{V}$ is given some adversarially chosen identifier.

For an integer $k \in \mathbb{N}$, a k-processor protocol generator is a protocol generator that supports specification involving at most k processors other than the trusted processor τ.

Security notions for multiparty protocols

Let π be a protocol for the set P of processors. Our framework supports adversaries that may be either active or passive (whether the adversary is passive or active is defined as part of the MPC model).

A *passive* adversary A for the protocol π that corrupts the processors Z_A is a (probabilistic) strategy. After each statement of the protocol π, the passive adversary A may read the variables in the views of the corrupted processors Z_A and can extend its own current view based on these values. It may also do arbitrary computation over its own view. The adversary is computationally unbounded.

An *active adversary* A for the protocol π is a passive adversary that may, in addition, take complete control over the corrupted processors Z_A.

Let A be an adversary and (π_0, τ) be a specification for the set of processors P_0. We say that the protocol π *A-securely computes the specification* (π_0, τ) if there exists a statement index function $f : \{1, \ldots, |\pi_0| + 1\} \to \{1, \ldots, |\pi| + 1\}$ and an adversary A_0 for the ideal protocol π_0 with $Z_{A_0} = Z_A \cap (P_0 \backslash \{\tau\})$ that satisfy the following property. For all inputs and for every $i = 1, \ldots, \pi_0 + 1$, the joint distribution of A_0's view with the output of $\mathsf{output}_i(p)$ for all non-corrupted processors $p \in (P_0 \backslash \{\tau\} \backslash Z_{A_0})$ before the i-th statement of the ideal protocol π_0 (with the adversary A_0 present) is equal to the joint distribution of A's view and the views $v_{f(i)}(p)$ of all non-corrupted processors $p \in (P_0 \backslash \{\tau\} \backslash Z_{A_0})$ before

the $f(i)$-th statement of the real protocol π (with the adversary A present). Moreover, the complexity of A_0 must be polynomial in the complexity of A.

The adversary A_0 can be thought of as a kind of simulator for A.

A *structure* $\mathcal{Z} \subseteq 2^P$ is a monotone set of subsets of the processors P where by monotone we mean that it is closed to taking subsets. For a structure $\mathcal{Z}$ and a specification (π_0, τ), we say that a protocol π *$\mathcal{Z}$-securely computes* (π_0, τ) if for every adversary A such that $\mathcal{Z}_A \in \mathcal{Z}$, it holds that π A-securely computes (π_0, τ).

A protocol generator G for the set of processors P is A-secure if, for every specification, the protocol that results by applying G to this specification A-securely computes the specification. For a structure $\mathcal{Z} \subseteq 2^P$, a protocol generator G for the set P of processors is $\mathcal{Z}$-secure if, for every adversary A such that $\mathcal{Z}_A \cap P \in \mathcal{Z}$, the protocol generator is A-secure. (See [HM00] for further details.)

Remapping and simulating processors

Let P and P' be sets of processors. A processor mapping $\sigma : P \to P'$ is a surjective function from P onto P'. If π is a protocol and $\sigma : P \to P'$ is a processor mapping then the mapped protocol $\sigma(\pi)$ is the same protocol, where each statement involving a processor in $p \in P$ is replaced by the processor $\sigma(p)$.

For a specification (π_0, τ) with $\tau \notin P$, the mapped specification $\sigma(\pi_0, \tau)$ is defined as $(\sigma(\pi_0), \tau)$.

The inverse processor mapping σ^{-1} of a processor mapping is defined by

$$\sigma^{-1} : P' \to 2^P$$

where $\sigma^{-1}(p') = \{p \in P : \sigma(p) = p'\}$. For a set of processors P, we define $\sigma(P) = \cup_{p \in P}\{\sigma(p)\}$ and $\sigma^{-1}(P) = \cup_{p \in P}\sigma^{-1}(p)$.

For a structure $\mathcal{Z}$ for the set P of processors and a processor mapping $\sigma : P \to P'$, the mapped structure is $\sigma(\mathcal{Z}) = \{Z \subseteq P' : \sigma^{-1}(Z) \in \mathcal{Z}\}$.

Lemma 6.21 (Processor Remapping Lemma, [HM00])**.** *Given a protocol π for the set P of processors that $\mathcal{Z}$-securely computes the specification (π, τ), and some processor mapping σ, then $\sigma(\pi)$ is a protocol for the set $\sigma(P)$ of processors that $\sigma(\mathcal{Z})$-securely computes the specification $\sigma(\pi_0, \tau)$.*

Consider a multiparty protocol π among the set P of processors and a protocol generator G for the set P_G of processors. To simulate a virtual processor $p \in P$ in π applying the protocol generator G means to consider this processor p as a trusted processor and to have this processor simulated by a subprotocol among the processors in P_G, according to G. More precisely, the specification (π, p) is used as input for the protocol generator G. We note that only processors that do not have *input* and *output* statements are simulated.

The following theorem of [HM00] show that a processor in an MPC protocol can be simulated by other processors.[21]

[21] The actual theorem in [HM00] restricts the simulating protocol generators to [BGW88] protocol generators. However, as stated in [HM00, Footnote 16], the only property of the [BGW88] protocol generators that they use is that they are *local* (see Section 6.10).

Theorem 6.11 (Processor Simulation Theorem, adapted from [HM00]). *Let π be a protocol in the MPC-model $\mathcal{M}$ among the set P of processors that $\mathcal{Z}$-securely computes a specification (π_0, τ), and let $G_1, \ldots, G_k$ be $\mathcal{Z}_1, \ldots, \mathcal{Z}_k$-secure local protocol generators for the processor sets $P_1, \ldots, P_k$, respectively. Assume that in π the k processors $p_{r_1}, \ldots, p_{r_k} \in P$ (which have no input or output statements) are simultaneously simulated by subprotocols applying the protocol generators $G_1, \ldots, G_k$ respectively. Then the resulting multiparty protocol π^* (also in the MPC-model $\mathcal{M}$) is for the set P^* of processors and $\mathcal{Z}^*$-securely computes the specification (π_0, τ), where*

$$P^* = (P \backslash R) \cup \bigcup_{i=1}^{k} P_i,$$

$$\mathcal{Z}^* = \left\{ Z \subseteq P^* : \left(\left(Z \cap (P \backslash R) \right) \cup \left\{ p_{r_i} \in R : Z \cap P_i \notin \mathcal{Z}_i \right\} \right) \in \mathcal{Z} \right\},$$

and $R = \{p_{r_1}, \ldots, p_{r_k}\}$ is the set of replaced processors.

6.11 From Threshold Formulae to Secure Multiparty Computation

In this section we show how to use logarithmic depth threshold formulae to obtain an efficient generic reduction from multiparty MPC protocols to MPC protocols for a constant number of parties. Before proceeding to the statement and proof of Lemma 6.22 which captures our approach, we recall some standard notations that will be used in this section.

Notation. For a set $I \subseteq [n]$, we denote by 1_I the n-bit indicator vector $(b_1, \ldots, b_n) \in \{0,1\}^n$ where $b_i = 1$ if $i \in I$ and $b_i = 0$ otherwise. Let F be a depth d formula. We say that the output wire of F has depth 0. For any other wire, we say that it has depth i if it is an input to a gate whose output wire has depth $i-1$. For a wire w, that is not an input wire, we define by $\mathsf{children}(w)$ the input wires of the gate of which w is an output wire. For a subset of *input* wires S, we define $F(S)$ we be the value obtained by evaluating F when the input wires of S have value 1 and the other input wires have value 0.

Lemma 6.22. *Let $j < k < n$ be integers and suppose that F is a formula on n inputs, which uses only Th_j^k gates and uses no constants. If there exists an explicit and local k-processor protocol generator for the MPC model $\mathcal{M}$ that is secure against the structure $\{Z \subseteq \{p_1, \ldots, p_k\} : |Z| < j\}$ then there exists an explicit $\mathcal{Z}_F$-secure n-processor protocol generator also in the MPC model $\mathcal{M}$ for a set $P = \{p_1, \ldots, p_n\}$ of n processors where,*

$$\mathcal{Z}_F = \left\{ \{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\} : k \leq n \text{ and } F(1_{\{i_1, \ldots, i_k\}}) = 0 \right\}.$$

Given a specification (π, τ), the protocol generator outputs a protocol π' that $\mathcal{Z}_F$-securely computes (π, τ) such that the $|\pi'| = c^{\mathsf{depth}(F)} \cdot |\pi|$ for a suitable constant c.

Proof. Let G_k be the k-processor protocol generator (that is secure against adversaries that control less than j processors). Given a specification (π, τ) for the processor set $\{p_1, \ldots, p_n, \tau\}$, our protocol generator computes a sequence of protocols $\pi_0, \ldots, \pi_d, \pi_{d+1}$ and finally outputs the protocol π_{d+1}. Each protocol π_i involves a set of processors P_i (which includes $p_1, \ldots, p_n$ but may include additional processors) and $\mathcal{Z}_i$-securely computes (π, τ), for a specific adversary structure $\mathcal{Z}_i$. Note that the protocol generator only outputs the final protocol π_{d+1}, which only involves the processors $P \triangleq \{p_1, \ldots, p_n\}$. The main part of the proof will be to show that π_{d+1} is indeed secure with respect to the adversary structure $\mathcal{Z}_F$. In other words, that $\mathcal{Z}_{d+1} = \mathcal{Z}_F$.

The protocols $\pi_1, \ldots, \pi_d$ are constructed recursively, whereas π_0 is described directly and π_{d+1} is a simple transformation of π_d. For $i \in \{0, \ldots, d\}$, the protocol π_i involves the processors $p_1, \ldots, p_n$ (which we call real processors) as well as additional processors (which we call virtual processors). Each protocol π_i is constructed by simulating the virtual processors in π_{i-1} by other virtual processors. Only in the very last step (i.e., the $d+1$-th protocol), all the remaining virtual processors are simulated by real processors.

We associate each wire of F with a (virtual) processor. For a wire w, we abuse notation and use w to denote both the wire and the associated processor. It will be clear from the context whether we think of w as a wire or as the associated processor.

We denote by W_i the set of wires that are either of depth (exactly) i or are *input wires* and are of depth no larger than i. Note that W_0 contains only the output wire of F whereas W_d includes all of the wires of F.

We proceed to describe the protocols $\pi_0, \ldots, \pi_d$. For every $i \in \{0, \ldots, d\}$, the protocol π_i will involve the (virtual) processors associated with wires in W_i as well as the (real) processors P. That is, $P_i = P \cup W_i$.

We first describe the protocol π_0. As noted above, π_0 involves the processors $P_0 = \{p_1, \ldots, p_n, w_0\}$ where w_0 is the processor associated with the output wire (i.e., the single wire of depth 0). The protocol π_0 is simply the protocol π where τ (the trusted processor) is replaced with w_0. Formally, let σ be the processor mapping that maps w_0 to τ. Then $\pi_0 = \sigma(\pi)$. Let $\mathcal{Z}_0 = \{Z \subseteq P_0 \colon Z \subseteq P\}$. Since w_0 acts as a trusted processor in $\mathcal{Z}_0$, the protocol π_0 trivially $\mathcal{Z}_0$-privately computes the specification (π, τ).

For $i \in \{1, \ldots, d\}$, we define π_i recursively, based on π_{i-1}. Let $R_{i-1} \subseteq W_{i-1}$ be the subset of wires of depth $i-1$ that are not input wires. The protocol π_i is constructed using Theorem 6.11 where the processors R_{i-1} are (simultaneously) replaced by their children. Specifically, every processor $w \in R_{i-1}$ is replaced by its own children $\mathsf{children}(w) \subseteq W_i$ using the k-processor local protocol generator G_k. By Theorem 6.11, the resulting protocol π_i, $\mathcal{Z}_i$-privately computes the specification (π, τ) for

$$\mathcal{Z}_i \triangleq \left\{ Z \subseteq P_i : \left(Z \cap (P_{i-1} \backslash R_{i-1})\right) \cup \left\{ w \in R_{i-1} : \left| Z \cap \mathsf{children}(w) \right| \geq j \right\} \in \mathcal{Z}_{i-1} \right\}.$$

The final protocol π_{d+1} is obtained by renaming every virtual processor w (corresponding to an input wire w) by a real processor. More specifically, if w is connected to the i-th input variable, then w is simulated by p_i. Formally, let ϕ be a processor mapping

that maps every input wire w that is connected to the i-th input variable to p_i. We define $\pi_{d+1} = \phi(\pi_d)$.

To prove that $\mathcal{Z}_{d+1} = \mathcal{Z}_F$, we introduce the notion of a suffix of a formula. We denote by F_i the i-deep suffix of F. That is, F_i is the formula that contains all the wires in $\cup_{j \leq i} W_j$. Note that the gates at depth i are not included. Also, note that the wires W_i are included in F_i but are not the output of any gate (in F_i). The wires W_i are used as the input wires of F_i. It is easy to see that F_0 is a trivial formula taking 1 input to 1 output. We also point out that F_d is not (necessarily) equal to F since different input wires of F_d may be wired to the same input *variable* of F.

Claim 6.22.1. *For every $i \in \{0, \ldots, d\}$ it holds that $Z \in \mathcal{Z}_i$ if and only if $F_i(Z \backslash P) = 0$ where F_i is the i-deep suffix of F.*

Proof. We prove the claim by induction. For $i = 0$, by the definition of $\mathcal{Z}_0$, it holds that $Z \in \mathcal{Z}_0$ if and only if $Z \subseteq P$. Note that the formula F_0 is just the trivial formula composed of a single wire that is both the input and output wire. Hence, $Z \in \mathcal{Z}_0$ if and only if $F_0(Z \backslash P) = 0$.

Let $i \in \{1, \ldots, d\}$ and suppose that the claim holds for $i - 1$. Let $Z \in P_i$ and let

$$Z' \triangleq \left(Z \cap (P_{i-1} \backslash R_{i-1})\right) \cup \left\{w \in R_{i-1} : \left|Z \cap \mathsf{children}(w)\right| \geq j\right\}. \tag{6.4}$$

By the definition of $\mathcal{Z}_i$, it holds that $Z \in \mathcal{Z}_i$ if and only if $Z' \in \mathcal{Z}_{i-1}$. By the inductive hypothesis, $Z' \in \mathcal{Z}_{i-1}$ if and only if $F_{i-1}(Z' \backslash P) = 0$. Thus, it suffices to show that $F_i(Z \backslash P) = F_{i-1}(Z' \backslash P)$.

Consider the evaluation of the formula F_i on input $Z \backslash P$. That is, the evaluation of F_i when the only input wires of F_i that have value 1 are those in $Z \backslash P$. Consider the values of the wires W_{i-1} during the evaluation of the formula. We show that the wires of W_{i-1} that have value 1 are exactly those in Z'. Fix a wire $w \in W_{i-1}$. We separate into two cases:

1. Suppose that $w \notin R_{i-1}$ (i.e., w is an input wire).
   ($\Rightarrow$) If w has a value of 1 then $w \in Z$ (since only input wires in Z have a value of 1). Thus, by definition of Z' and since $w \notin R_{i-1}$, it holds that $w \in Z'$.

   ($\Leftarrow$) On the other hand, if $w \in Z'$, then since $w \notin R_{i-1}$, it holds that $w \in Z$ and therefore has value 1.

2. Suppose that $w \in R_{i-1}$.
   ($\Rightarrow$) If w has value 1 then, since w is the output of a Th^k_j gate, at least j of its children have value 1 and in particular at least j of w's children are in Z. Thus, by definition of Z', it holds that $w \in Z'$.

   ($\Leftarrow$) On the other hand if $w \in Z'$ then, by the definition of Z', at least j of w's children are in Z. The children of w are input wires and therefore have value 1. Since w's value is computed as the threshold of at least j of its children, w has value 1.

Thus, $F_i(Z \backslash P) = F_{i-1}(Z' \backslash P)$ and the claim follows. □

By Claim 6.22.1, $Z \in \mathcal{Z}_d$ if and only if $F_d(Z \backslash P) = 0$. By Lemma 6.21,

$$\mathcal{Z}_{d+1} = \phi(\mathcal{Z}_d) = \left\{ Z \subseteq P : \phi^{-1}(Z) \in \mathcal{Z}_d \right\}.$$

Claim 6.22.2. *For $Z \subseteq P$ it holds that $Z \in \mathcal{Z}_{d+1}$ if and only if $F(Z) = 0$.*

Proof. ($\Rightarrow$) Suppose that $Z \in \mathcal{Z}_{d+1}$, then $Z \subseteq P$ and $\phi^{-1}(Z) \in \mathcal{Z}_d$. In particular $F_d(\phi^{-1}(Z) \backslash P) = 0$. Now consider the evaluation of F on input 1_Z. An input wire w that is connected to the i-th input variable has value 1 if and only if $p_i \in Z$. Thus, the input wires correspond exactly to $\phi^{-1}(Z) \backslash P$. Since the formula proceeds by evaluating F_d on the input wires we have $F(1_Z) = 0$.

($\Leftarrow$) If $F(Z) = 0$, then in particular $F_d(\phi^{-1}(Z)) = 0$. Hence, $\phi^{-1}(Z) \in \mathcal{Z}_d$ and so $Z \in \mathcal{Z}_{d+1}$. □

We conclude that the protocol π_{d+1} $\mathcal{Z}_F$-securely computes the specification (π, τ). Observe that $|\pi_0| = |\pi|$ and that the Processor Simulation Theorem (Theorem 6.11) replaces every statement in π_{i-1} by a constant number of statements in π_i. Therefore, $|\pi_d| = c^{\mathsf{depth}(F)}|\pi|$ for some suitable constant c. The protocol π_{d+1} has the same length as π_d since it is constructed by replacing every statement in π_d with a single statement. Therefore, the protocol generator outputs a protocol of length $c^{\mathsf{depth}(F)}|\pi|$. □

6.12 Secure MPC over Blackbox Rings

In this section we consider multiparty computation over any commutative ring $(R, +, *)$. The processors are only given blackbox access to the ring. This model of secure computation was first considered by Cramer *et al.*[CFIK03] and it generalizes the classical model of [BGW88], where the computation is over a finite field. We note that the classical results of [BGW88] for passive and active security do not extend to this model as they require field operations[22]. We present a protocol generator based on the threshold formula to MPC approach introduced in Section 6.11. We stress that our protocols use the ring in an entirely blackbox manner, are fairly simple and do not rely on any nontrivial facts from algebra (as in [CFIK03]).

Formally, we define an MPC model which we call the *Ring-MPC model* as follows. Every variable $x \in \mathcal{X}$ may take values in the ring R. We allow processors to compute addition and multiplication over the ring. That is, protocols in the Ring-MPC model support the operator $+$ (resp., $*$), which takes two operands and returns their sum (resp., product). Additionally, the processors can sample a random ring element and have access to the constant -1 (i.e., the additive inverse of the multiplicative neutral element of the ring R).

[22] Specifically, the ability to find multiplicative inverses is used by Shamir's secret sharing scheme [Sha79].

6.12.1 The passive model

Our approach to constructing secure multiparty protocol generators is to use Lemma 6.22 to reduce the problem to that of constructing protocols for a constant number of processors. For the latter, we use Maurer's [Mau06] simple and elegant protocol generator. We note that Maurer's protocol works over any ring and uses the ring in a blackbox manner. A downside of Maurer's protocol generator is that it produces protocols of length exponential in the threshold of tolerated adversaries. However, this does not concern us since we only need to use the base protocol for a constant number of processors.

In fact, instead of using Maurer's protocol generator we could also use the classical [BGW88] protocol. We choose to use Maurer's protocol generators because (1) they are significantly simpler than [BGW88] and combined with our approach yield a fairly simple and straightforward construction of multiparty protocol generators and (2) they can be implemented over any ring and not just a field. We stress that we improve upon Maurer's protocol generators in that we produce protocols of length polynomial, rather than exponential, in the desired threshold of corrupted processors.

The main caveat of our approach is that it is either (1) based on an unproven conjecture (the majority from majorities conjecture - Conjecture 6.1) or (2) uses a randomized construction or (3) supports a non-optimal threshold of corrupt processors.

As noted above, we only require the following special cases of Maurer's protocol generator:

Theorem 6.12 ([Mau06])**.** *There exists an explicit three processor local protocol generator in the Ring-MPC model that is secure against a single* passive *adversary.*

Using Lemma 6.22 combined with Theorem 6.12 and our majority formulae constructions (see Section 6.8) we obtain Theorems 6.13, 6.14 and 6.15.

Theorem 6.13. *If the majority from majorities conjecture (Conjecture 6.1) holds then there exists an explicit protocol generator in the Ring-MPC model that is secure against a passive adversary that controls any $t < \frac{n}{2}$ of the n processors.*

Given a protocol for n processors that involves t ring operations, the protocol generator outputs a protocol involving $t \cdot \mathrm{poly}(n)$ ring operations.[23]

Proof. The majority from majorities conjecture implies that there exists an algorithm that on input n outputs a logarithmic depth formula F_n composed of Maj_3 gates (i.e., Th_2^3 gates) that computes majority. The running time of the algorithm is polynomial in n.

Given a specification (π, τ) that involves n processors, the protocol generator constructs F_n and then applies Lemma 6.22 while using Maurer's 3-processor local protocol generator of Theorem 6.12. □

[23] Note that the number of communicated ring elements is always upper bounded by the amount of computation.

Note that by Theorem 6.2, the conjecture used in Theorem 6.13 can be replaced by the assumption that E does not have $2^{\varepsilon n}$- size circuits for some $\varepsilon > 0$, or even more specifically on the existence of sub-exponentially hard one-way functions.

As an additional result, using our construction of formulae that compute majority given sufficient bias (Theorem 6.1), we obtain the following result which guarantees close to optimal security.

Theorem 6.14. *There exists an explicit protocol generator in the passive Ring-MPC model that is secure against any adversary that controls a $\frac{1}{2} - 2^{-O(\sqrt{\log n})}$ fraction of the n processors.*

Given a protocol for n processors that involves t ring operations, the protocol generator outputs a protocol involving $t \cdot \mathrm{poly}(n)$ ring operations.

Proof. By Theorem 6.1, there exists an algorithm that on input n outputs a logarithmic depth formula F_n composed of Maj_3 such that given any string of relative Hamming weight less than $\frac{1}{2} - 2^{-O(\sqrt{\log n})}$ as input, the formula F_n outputs 0. The running time of A is polynomial in n.

Given a specification (π, τ) that involves n processors, the protocol generator constructs F_n by running A and then applies Lemma 6.22 while using Maurer's 3-processor local protocol generator of Theorem 6.12. □

Alternatively, the logarithmic depth majority formulae can be obtained using the randomized construction of [Val84] (see also [Gol11b]). This results in the following randomized construction.

Theorem 6.15. *There exists a randomized construction of a protocol generator in the passive Ring-MPC model that is secure against an adversary that controls $t < \frac{n}{2}$ of the n processors.*

Given a protocol for n processors that involves t ring operations, the protocol generator outputs a protocol involving $t \cdot \mathrm{poly}(n)$ ring operations.

Proof. By Theorem 6.7, there exists a randomized algorithm A that on input n, other than with exponentially small probability, outputs a logarithmic depth formula F_n composed of Maj_3 gates that computes majority. The running time of A is polynomial in n. As the advice string for our protocol generator we use the random coins of A.

Given a specification (π, τ) that involves n processors, the protocol generator generates F_n by running A on the random coins specified by its advice string and then applies Lemma 6.22 while using Maurer's local 3-processor protocol generator of Theorem 6.12. □

6.12.2 The active model

In this section we show a protocol generator in the Ring-MPC model that is secure against an active adversary. To do so we shall once again use Lemma 6.22, only this time we reduce the n-processor problem to a four processor problem.

In order to solve the four processor case we once again use Maurer's [Mau06] actively secure protocol. Indeed, Maurer gives a simple and elegant protocol that is secure against any adversary that *actively* controls at most one processor.

Unfortunately, Maurer's four processor active protocol requires operations not supported by the blackbox ring model. Specifically, the protocol requires the ability to test for equality and to choose a majority between three values (where a tie is not possible).

To support Maurer's protocol we could potentially add these operations to our model. However, in order to use Lemma 6.22, we would have to be able to simulate such operations done by a virtual processor using other processors (which does not seem easy).

Instead, we take a different approach. We slightly extend our model by allowing *global* variables. That is, variables that exist in the view of all processors (both real and virtual). Every processor may give a global variable a value (for simplicity we assume that the value of each global variable is only set once) and read the values of global variables.

In addition, we allow the parties to do arbitrary computation over global variables. This computation may use oracle access to the underlying ring but only based on the (adversarially chosen) identifiers of ring elements. In particular the number of oracle queries may not depend on the underlying ring. We call this the Active Ring-MPC model.

Note that the Processor Simulation Theorem (Theorem 6.11) and Lemma 6.22 can be extended to this model. This can be done since an operation over global variables by a virtual processor can be simulated by having each simulating processor do the exact same computation. Note that since the variables are global, each simulating processor has access to these variables.

It is worthwhile to point out that a protocol in the Active Ring-MPC model can be easily transformed to work in a more standard model in which (1) we allow broadcasts and (2) allow additional operations such as testing equality and taking majority. However, it will be useful for us to present our protocols in the Active Ring-MPC and they can later be adapted to other models.

We give a sketch of the steps required to adapt Maurer's four processor protocol to the Active Ring-MPC model:

1. **Handling Equality:** equality is used in Maurer's protocol only in the consistency check of the underlying verifiable secret sharing (VSS) protocol.

   In the four processor VSS protocol, a dealer sharing a secret $s \in R$ sends shares s_1, s_2, s_3, s_4 to processors p_1, p_2, p_3, p_4 such that processor i receives all shares but s_i. Then, for every $i \in \{1, 2, 3, 4\}$, every processor except the i-th processor check that their received value s_i is the same. This is done by three pairwise equality tests.

   Suppose that processors j and k holding the respective shares $s_i^{(j)}$ and $s_i^{(k)}$ (which are supposed to be equal to s_i) want to verify that their share are equal. In Maurer's protocol this is done by simply sending each other the shares and broadcasting a complaint if they differ.

Instead of testing equality, both the j-th and k-th processor publish the difference $s_i^{(j)} - s_i^{(k)}$ between their two shares as a global variable. We argue that this does not extend the view of the adversary. Indeed, if the adversary controls either processor i, j or k then its view is not extended since it already knows both $s_i^{(j)}$ and $s_i^{(k)}$. If the adversary controls the fourth processor then its view is also not extended since it will always see the constant 0 (recall that since we deal with a four processor protocol there is only one processor controlled by the adversary).

Suppose that one of these global variables $v_{i,j,k}$ corresponding to a share s_i was set to a non-zero value. Then, as in Maurer's protocol, the dealer publishes s_i as a global value which is used by all parties as the correct value of s_i. Note that the latter operation can be implemented solely using global variables and computation over these global variables.

2. **Handling Majority:** A majority between three value $a, b, c \in R$ (where there is no possibility of a tie) is used in Maurer's VSS reconstruction phase. In this protocol each processor computes the majority of three values that were broadcast by three processors. We replace these broadcasts by publishing global variables and therefore the majority operation is an operation done over global variables which we allow in our model.

Using this transformation, we can state Maurer's active protocol in terms of a four-processor protocol generator:

Theorem 6.16 ([Mau06])**.** *There exists an explicit four processor local protocol generator in the Active Ring-MPC model that is secure against an active adversary that controls a single processor.*

Using Theorem 6.16 combined with Lemma 6.22 and our construction of a logarithmic depth threshold formula from Th_2^4 gates (Theorem 6.3), we obtain the following result in the active model.

Theorem 6.17. *There exists an explicit protocol generator in the Active Ring-MPC model that is secure against an active adversary that controls at most a $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of the n processors.*

Given a protocol for n processors that involves t ring operations, the protocol generator outputs a protocol involving $t \cdot \mathrm{poly}(n)$ ring operations.

Proof. By Theorem 6.3, there exists an algorithm that on input n outputs a logarithmic depth formula F_n composed of Th_2^4 gates that outputs 0 for any input of relative Hamming weight $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ or less.

Given a specification (π, τ) that involves n processors, the protocol generator generates F_n and then applies Lemma 6.22 while using Maurer's 4-processor local protocol generator of Theorem 6.16. □

6.12.3 MPC over k-linear maps

In this section we describe an extension of Maurer's protocol generator which supports blackbox computation over an arbitrary basis of k-linear maps. For simplicity, we restrict the attention here to k-linear maps over Abelian groups (rather than vector spaces or modules, over which they are usually defined). The computational model is defined by a set of finite Abelian groups $G_1, \ldots, G_m$ written in additive notation and a basis B of k-linear maps over these groups, where a k-linear map is a function $L : G_{i_1} \times \cdots \times G_{i_k} :\to G_{i_0}$ with the following property. If all of the input variables but the j-th are held constant, then the resulting function $L' : G_{i_j} \to G_{i_0}$ satisfies $L'(g + g') = L'(g) + L'(g')$ for each $g, g' \in G_{i_j}$. Note that k'-linear maps for $k' < k$ (including addition in a single group G_i) are special cases of k-linear maps. A blackbox computation over B can have inputs and variables taken from any of the groups G_i and may combine them using an arbitrary sequence of k-linear maps from B as well as individual group operations.

Our main observation is that by using a simple generalization of Maurer's protocol, $n = k + 1$ processors (resp., $n = k + 2$ processors) suffice for evaluating an arithmetic circuit over B with security against a single passively (resp., actively) corrupted processor. We sketch the approach for the passive case.

As in [Mau06], we represent each group element $g \in G$ by $k + 1$ additive shares $g_1, \ldots, g_{k+1}$ such that processor i holds all shares *except* g_i. Now, suppose we are given k group elements $g^{(1)}, \ldots, g^{(k)}$ represented in this way, so that $g^{(i)} = \sum_{j=1}^{k+1} g_j^{(i)}$. To locally compute additive shares of $L(g^{(1)}, \ldots, g^{(k)})$, we write

$$L(g^{(1)}, \ldots, g^{(k)}) = \sum_{a \in [k+1]^k} L\left(g_1^{(a_1)}, \ldots, g_k^{(a_k)}\right).$$

Noting that the value of each of the $(k + 1)^k$ terms is known to at least one of the $k + 1$ processors (namely, any processor whose index does not occur in the tuple a corresponding to the term), we can assign each term to a processor who can evaluate it. Letting each processor sum the values of the term assigned to it, we get an additive representation of $L(g^{(1)}, \ldots, g^{(k)})$. To continue the computation, each share of the output needs to be re-shared as in the protocol of [Mau06].

This approach, combined with Theorem 6.3, yields the following theorem.

Theorem 6.18. *For any constant $k \geq 2$, there exists an explicit protocol generator in the model of MPC over k-linear maps that is secure against a passive (resp., active) adversary controlling at most a $\frac{1}{k} - \Omega(\frac{1}{\sqrt{\log n}})$ (resp., $\frac{1}{k+1} - \Omega(\frac{1}{\sqrt{\log n}})$) fraction of the n processors.*

Given a protocol for n processors that involves t group operations, the protocol generator outputs a protocol involving $t \cdot \mathrm{poly}(n)$ group operations.

6.13 Secure MPC over Groups

In this section we consider a different instantiation of the abstract MPC framework introduced in Section 6.10, where the computation is done over a finite group while only

making a black-box access to the group. In particular, the number of group elements communicated during the protocol does not depend on the computational complexity of the group operation. This model was introduced by Desmedt *et al.* [DPSW07] and further studied in [SYT08, DPS+12b, DPS12a].

Let $(G, *)$ be a fixed finite group written in multiplicative notation. We do not assume that the group is Abelian (and think of it as being non-Abelian). We instantiate the framework with variables taking values in G. We allow the processors to compute the following operations:

- The group operator $*$. That is, given $g_1, g_2 \in G$ a processor can compute the value $g_1 * g_2$. Note that since the group is (usually) non-Abelian, the order of operands is important.
- The operator *invert* which given an operand $g \in G$ returns the inverse g^{-1} of g.
- The operator *rand* that takes no operands and returns a uniformly distributed group element.

We call this model the *Group-MPC model*. For brevity, we will sometimes write $g_1 g_2$ instead of $g_1 * g_2$.

6.13.1 The passive model

In this section, we directly present a simple 3-processor protocol generator (in the Group-MPC model) that has passive security against a single adversary. The protocol that we present is loosely based on a protocol of Feige, Killian and Naor [FKN94] and simplifies a previous protocol from [DPS+12b].

Let $x \in G$. Recall that a 2-out-of-2 secret sharing of x is the following (random) process: select $x_1 \in_R G$ and set $x_2 = x_1^{-1} x$ such that $x = x_1 x_2$). We call (x_1, x_2) a sharing of x. Note that since the group may be non-Abelian, x_1 and x_2 play different roles and are called the left and right shares respectively.

Our protocol consists of three processors p_1, p_2, p_3. The protocol generator will maintain the invariant that every variable held in the ideal protocol (i.e., the specification) by τ is secret shared by p_1 and p_2 such that p_1 holds the left share and p_2 holds the right share. In fact, by "computing a sharing z" we mean that p_1 computes z_1 and p_2 computes z_2 such that $z = z_1 z_2$ and z_1 is uniformly distributed in G.

Before presenting the 3-processor protocol generator, we present two useful sub-protocols that will be used by the protocol generator. The first protocol is useful for multiplying two sharings. It is described and proved in Claim 6.22.3. The second protocol transforms a sharing (a, b) into a random sharing of ba. It is described and proved in Claim 6.22.4.

Claim 6.22.3 (Share Multiplication Protocol)**.** *Let $\{p_1, p_2, p_3\}$ be a set of three processors in the Group-MPC model such that p_1 gets as input $a_1, a_2 \in G$, p_2 gets as input $b_1, b_2 \in G$ and p_3 has no input. Let $z = a_1 b_1 a_2 b_2$. Then there exists a protocol for computing a*

sharing of z such that the view of each processor is statistically independent of the input of the other processors.

Proof. Consider the following protocol:

1. p_1 selects at random $r_0, r_1, r_2, r_3 \in_R G$ and sends r_0, r_1, r_2, r_3 to p_2.
2. p_1 computes $a'_1 = r_0^{-1} a_1 r_1$ and $a'_2 = r_2^{-1} a_2 r_3$ and sends a'_1 and a'_2 to p_3.
3. p_2 computes $b'_1 = r_1^{-1} b_1 r_2$ and $b'_2 = r_3^{-1} b_2$ and sends b'_1 and b'_2 to p_3.
4. p_3 selects $r' \in_R G$ and sends $u_1 = a'_1 b'_1 a'_2 b'_2 r'$ to p_1 and $u_2 = r'^{-1}$ to p_2.
5. The share of p_1 is $z_1 = r_0 u_1$ and the share of p_2 is $z_2 = u_2$.

Note that (z_1, z_2) is indeed a sharing of z since z_2 is uniformly distributed in G and

$$z_1 z_2 = r_0 u_1 u_2 = r_0 a'_1 b'_1 a'_2 b'_2 = (r_0 r_0^{-1}) a_1 (r_1 r_1^{-1}) b_1 (r_2 r_2^{-1}) a_2 (r_3 r_3^{-1}) b_2 = a_1 b_1 a_2 b_2 = z.$$

The view of p_1 consists only of $a_1, a_2, r_0, r_1, r_2, r_3, z_1$. Since $z_1 = r_0 a_1 b_1 a_2 b_2 r'$, and since r' is not known to p_1, the view is statistically independent from b_1, b_2.

The view of p_2 consists only of $b_1, b_2, r_0, r_1, r_2, r_3, z_2$ which is statistically independent from a_1, a_2 (since $z_2 = r'^{-1}$).

The view of p_3 consists only of $r_0^{-1} a_1 r_1, r_1^{-1} b_1 r_2, r_2^{-1} a_2 r_3, r_3^{-1} b_2$ which is statistically independent of a_1, b_1, a_2, b_2. □

Claim 6.22.4 (Share Inversion Protocol)**.** *Let $\{p_1, p_2, p_3\}$ be a set of three processors in the Group-MPC model such that p_1 gets as input $a \in G$, p_2 gets as input $b \in G$ and p_3 has no input. Then there exists a protocol for computing a sharing of the inverted product ba such that the view of each processor is statistically independent of the input of the other processors.*

Proof. Consider the following protocol:

1. p_1 samples uniformly at random $r_1 \in_R G$, computes $y_1 = a r_1$ and sends r_1 to p_2 and y_1 to p_3.
2. p_2 samples uniformly at random $r_2 \in_R G$, computes $y_2 = r_2 b$ and sends r_2 to p_1 and y_2 to p_3.
3. p_3 samples uniformly at random $s \in_R G$, computes $w_1 = y_2 s$ and $w_2 = s^{-1} y_1$. It sends w_1 to p_1 and w_2 to p_2.
4. The share of p_1 is $z_1 = r_2^{-1} w_1$ and the share of p_2 is $z_2 = w_2 r_1^{-1}$.

Note that:

$$z_1 z_2 = r_2^{-1} w_1 w_2 r_1^{-1} = r_2^{-1} y_2 s s^{-1} y_1 r_1^{-1} = r_2^{-1} r_2 b a r_1 r_1^{-1} = ba$$

as required.

The view of p_1 consists of a, r_1, r_2, w_1. Since $w_1 = r_2 bs$ and s is unknown to p_1, its view is statistically independent of b.

Similarly, the view of p_2 consists of b, r_1, r_2, w_2. Since $w_2 = sar_1$ and s is unknown to p_2, its view is statistically independent of a.

Finally, the view of p_3 consists of $ar_1, r_2 b$. Since r_1 and r_2 are unknown to p_3, its view is statistically independent of a, b. □

Using Claims 6.22.3 and 6.22.4 we are ready to describe our protocol generator and prove its security against a single adversary.

Lemma 6.23. *There exists a 3-processor local protocol generator in the Group-MPC model that has passive security against a single adversary.*

Proof. Let (π, τ) be a specification for the set of processors $\{p_1, p_2, p_3\}$. Given (π, τ) as input, the protocol generator outputs a protocol π' by replacing each statement of π that involves τ by a sub-protocol. Statements that do not involve τ are mapped to π' as-is. An invariant that we maintain that in π', is that if τ has access to some variable x in π then p_1 and p_2 have access to a sharing of x in π'.

1. Every statement of the form $transmit(p_i, \tau, z)$ for $i \in \{1, 2, 3\}$ is mapped to the following sub-protocol. The processor p_i selects at random $z_1 \in_R G$ and sets $z_2 = z_1^{-1} z$ (such that $z = z_1 z_2$). It sends z_1 to p_1 and z_2 to p_2. Note that this process maintains our invariant that p_1 and p_2 hold a sharing of z.

2. Every statement of the form $transmit(\tau, p_i, z)$ for $i \in \{1, 2, 3\}$ is mapped to the following sub-protocol. Since z is in the view of τ, by our invariant, p_1 and p_2 have a sharing (z_1, z_2) of z. Processor p_1 sends z_1 to p_i and processor p_2 sends z_2 also to p_i. Processor p_i then computes $z = z_1 z_2$.

3. Every statement of the form $comp(\tau, *, z_1, z_2, z)$ is mapped to the following sub-protocol. Since z_1 is known to τ, by our invariant, p_1 and p_2 have a sharing (a_1, b_1) of z_1 and a sharing (a_2, b_2) of z_2. To compute a sharing of $z = z_1 * z_2 = a_1 b_1 a_2 b_2$, p_1 and p_2 simply run the share multiplication protocol (Claim 6.22.3).

4. Every statement of the form $comp(\tau, inverse, z, z')$ is mapped to the following sub-protocol. Since z is known to τ, by our invariant, p_1 and p_2 have a sharing (z_1, z_2) of z. They both invert their shares and run the inversion protocol (Claim 6.22.4) on the inverted shares. At the end of the protocol p_1 has u_1 and p_2 has u_2 such that $u_1 u_2 = z_2^{-1} z_1^{-1} = (z_1 z_2)^{-1} = z^{-1}$.

5. Every statement of the form $comp(\tau, random, z)$ is mapped to the following sub-protocol. Processor p_1 samples a random element r_1 and p_2 samples a random element r_2. They set $z = (r_1, r_2)$. That is (r_1, r_2) is a sharing of z.

We proceed to show that π' securely computes the specification (π, τ) with respect to an adversary that sees the view of a single processor.

Fix an adversary A' that sees the view of a single processor $p^* \in \{p_1, p_2, p_3\}$ in π' and fix inputs to the three processors. We consider the statement index function that maps each statement in π to the corresponding sub-protocol (described above) in π'. We will show an adversary A such that the joint distribution of its view together with the output of the uncorrupted processors in π is identical to the view of A' together with the output non-corrupted processors in π'.

We proceed to describe what the adversary A does for the i-th statement of π and argue why its view together with the output of the uncorrupted processors in π remains identically distributed to that of A' together with the uncorrupted processors in π'. For every $i \in \{1, \ldots, |\pi| + 1\}$, the adversary A performs the following steps for the i-th statement of π:

1. If the statement (1) does not involve τ or (2) it is of the form $transmit(p_i, \tau, z)$ or (3) is of form $transmit(\tau, p_i, z)$ for $p_i \neq p^*$ then A runs the corresponding sub-protocol in π' and performs the same steps that A' would perform. Since the views of neither A' is not extended and that output of all non-corrupt processors is either unchanged or changed similarly (in case of an *output* statement), the joint distribution of the view of A and the output of the uncorrupted processors in π' remains identical to that of A' and the uncorrupted processors in π.

2. If the statement is of the form $comp(\tau, *, z_1, z_2, z)$ and $p^* \in \{p_1, p_2\}$ then (by Claim 6.22.3, the view of A' is extended by a random share of z. Thus, A extends its view by simply selecting a uniformly distributed group element. If $p^* = p_3$ then it simply performs the same steps that A' would perform.

3. If the statement is of the form $comp(\tau, inverse, z_1, z)$ and $p^* \in \{p_1, p_2\}$ then the view of A' is extended by an independent random sharing of z^{-1} (by Claim6.22.4). Thus, A extends its own view by a uniformly distributed group element. If $p^* = p_3$ then it simply performs the same steps that A' would perform.

4. If the statement is of the form $comp(\tau, rand, z)$ then A simulates the corresponding sub-protocol in π'. The view of A' is (possibly extended) by a single share of the random element z. The adversary A simulates this by choosing a random group element in G.

5. If the statement is of the form $transmit(\tau, p^*, z)$ then A simulates the corresponding sub-protocol that recovers z from shares held by p_1 and p_2. It then performs the same steps as A' performs with respect to the above simulation. The view of both A and A' are extended by z.

Thus, after every statement, the joint distribution of the view of A together with the output of all uncorrupted processors in π is identically distributed to that of A' together with the output of the uncorrupted processors in π'. □

Using Lemma 6.23 together with Lemma 6.22 we obtain the following two results:

Theorem 6.19. *There exists a protocol generator in the passive Group-MPC model that is secure against an adversary that controls a $\frac{1}{2} - 2^{-O(\sqrt{\log n})}$ fraction of the n processors.*

Given a protocol for n processors that involves t group operations, the protocol generator outputs a protocol involving $t \cdot \mathrm{poly}(n)$ group operations.

Proof. By Theorem 6.1, there exists an algorithm that on input n outputs an $O(\log n)$ depth formula F_n composed of Maj_3 gates such that for every input of normalized weight less than $\frac{1}{2} - 2^{-O(\sqrt{\log n})}$, the formula F_n outputs 0. The theorem follows by an application of Lemma 6.22 based on the 3-processor local protocol generator of Lemma 6.23. □

Theorem 6.20. *If the majority from majorities conjecture (Conjecture 6.1) holds then there exists a protocol generator in the Group-MPC model that has passive security against an adversary that controls at most $t < \frac{n}{2}$ processors.*

Given a protocol for n processors that involves t group operations, the protocol generator outputs a protocol involving $t \cdot \mathrm{poly}(n)$ group operations.

Proof. The majority from majorities conjecture implies the existence of an algorithm that on input n outputs a logarithmic depth formula that uses only Maj_3 gates and computes majority. The theorem follows from an application of Lemma 6.22 based on the 3-processor local protocol generator of Lemma 6.23. □

Theorem 6.21. *There exists a randomized construction of a protocol generator in the passive Group-MPC model that is secure against an adversary that controls $t < \frac{n}{2}$ processors.*

Given a protocol for n processors that involves t group operations, the protocol generator outputs a protocol involving $t \cdot \mathrm{poly}(n)$ group operations.

Proof. By Theorem 6.7, there exists a randomized polynomial-time algorithm that on input n, other than with exponentially vanishing probability, outputs an $O(\log n)$ depth formula composed of Maj_3 gates that computes majority. The theorem follows by an application of Lemma 6.22 based on the 3-processor local protocol generator of Lemma 6.23. □

Theorem 6.22. *There exists a protocol generator in the passive Group-MPC model that is secure against an adversary that controls $t < \frac{n}{2}$ processors.*

Given a protocol for n processors that involves t group operations, the protocol generator outputs a protocol involving $t \cdot n^{O(\log n)}$ group operations.

Proof. By Lemma 6.10, there exists a (deterministic) polynomial-time algorithm that on input n, runs in time $n^{O(\log n)}$, and outputs an $O(\log^2 n)$ depth formula (of size $n^{O(\log n)}$) composed of Maj_3 gates that computes majority. The theorem follows by an application of Lemma 6.22 based on the 3-processor local protocol generator of Lemma 6.23. □

6.13.2 The active model

In a recent work, Desmedt *et al.* [DPS12a], gave a protocol (or in our terminology, a protocol generator) in the Group-MPC model which is *actively* secure against any Q_3 adversary. However, the complexity of their protocol generator is quadratic in the number of maximal sets in the adversary structure, which in the case of thresholds, is exponential in the number of processors. In this section we address an open problem stated by [DPS12a] by showing a protocol generator in the Group-MPC that has active security against threshold adversaries where the complexity of the protocol is polynomial in the number of processors. Our protocol is *actively* secure against any adversary that controls at most a $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ fraction of the n processors (in contrast to the optimal threshold of $\frac{1}{3} - \Omega(\frac{1}{n})$).

We basically use the same processor simulation approach formalized in Section 6.11 and used in previous sections. Recall that in order to use this approach we must rely on a protocol-generator for a constant number of processors. For this we simply use an instantiation of the protocol of Desmedt *et al.* [DPS12a] for four processors that is secure against an adversary that *actively* control one processor.

Unfortunately, as in the case of Maurer's four processor active protocol in the Ring-MPC model (see Section 6.12.2), the [DPS+12b] protocol uses operations that are not supported by our model. Specifically equality testing and computing majority.

As in Section 6.12.2, we extend the Group-MPC model by allowing *global* variables and allowing arbitrary computation over global variables. This computation may use oracle access to the underlying group but only based on the (adversarially chosen) identifiers of group elements. In particular, the number of oracle queries may not depend on the underlying ring. We call this the Active Group-MPC model.

Note that a protocol in the Active Group-MPC model can be easily transformed to work in a more standard model in which (1) we allow broadcasts and (2) allow additional operations such as testing equality and taking majority vote.

Below, we give a sketch of the steps required to adapt the four processor [DPS+12b] protocol to the Active Group-MPC model:

1. **Equality type 1:** The first type of equality test that is used in the [DPS+12b] protocol in the consistency check of the underlying VSS and consistency checks in the NodeMult protocol. The problem and its solution are almost identical to that encountered in Section 6.12.2.

   In this type of equality, a dealer send secret shares of some value and the receiving parties check the pairwise consistency of their shares. Specifically, suppose that processor j and k are sent respective shares $s_i^{(j)}$ and $s_i^{(k)}$ (which are supposed to be equal to a share s_i) and want to verify that their share are equal. In Maurer's protocol this is done by simply sending each other the shares and broadcasting a complaint if they differ.

   Instead of testing equality, both the j-th and k-th processor publish the ratio $s_i^{(j)} * (s_i^{(k)})^{-1}$ between their two shares as a global variable. We argue that this does not

extend the view of the adversary. Indeed, if the adversary controls either processor i, j or k then its view is not extended since it already knows both $s_i^{(j)}$ and $s_i^{(k)}$. If the adversary controls the fourth processor then its view is also not extended since it will always see the neutral element of G (recall that since we deal with a four processor protocol there is only one processor controlled by the adversary).

Suppose that one of these global variables $v_{i,j,k}$ corresponding to a share s_i was set to a value not-equal to the neutral element. Then, as in the [DPS12a] protocol, the dealer publishes s_i as a global value which is used by all parties as the correct value of s_i. Note that the latter operation can be implemented solely using global variables and computation over these global variables.

2. **Equality type 2:** The second type of equality operation occurs in the second step of the NodeMult protocol. However, in this step after a processor tests whether $a = b$, if $a \neq b$ it just broadcasts the value $a * b^{-1}$. Thus, instead of testing $a = b$ we just set $a * b^{-1}$ as a global variable and use it as specified.

3. **Equality type 3:** The third type of equality operations occurs is on global variables. Since this computation is done over global variables we do not need to change it (recall that we allow arbitrary computation over global variables).

4. **Majority:** A majority between three values $a, b, c \in R$ (where there is no possibility of a tie) is used in the [DPS12a] protocol only when a processor computes the majority of three values that were broadcast by three processors. We replace these broadcasts by publishing global variables and therefore the majority operation is an operation done over global variables which we allow in our model.

Thus, we have the following theorem:

Theorem 6.23 ([DPS12a])**.** *There exists an explicit four processor local protocol generator in the Active Group-MPC model that is secure against an active adversary that controls a single processor.*

Using Theorem 6.16 combined with Lemma 6.22 and our construction of a logarithmic depth threshold formula from Th_2^4 gates (Theorem 6.3), we obtain the following result in the active model.

Theorem 6.24. *There exists an explicit protocol generator in the Active Group-MPC model that is secure against an active adversary that controls at most* $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ *of the* n *processors.*

Given a protocol for n *processors that involves* t *group operations, the protocol generator outputs a protocol involving* $t \cdot \mathrm{poly}(n)$ *group operations.*

Proof. By Theorem 6.3, there exists an algorithm that on input n outputs a logarithmic depth formula F_n composed of Th_2^4 gates that outputs 0 for any input of normalized Hamming weight $\frac{1}{3} - \Omega(\frac{1}{\sqrt{\log n}})$ or less.

Given a specification (π, τ) that involves n processors, the protocol generator generates F_n and then applies Lemma 6.22 while using the [DPS12a] four processor local protocol generator of Theorem 6.23. □

6.13.3 Two-party protocols

In this section we establish the first feasibility results for secure *two-party* computation over black-box groups. The result we get for the active corruption model illustrates the usefulness of the "threshold from threshold" technique even in the context of two-party cryptography.

Instead of basing our protocols on concrete cryptographic assumptions, we follow the convention of allowing the parties access to an ideal oblivious transfer oracle. Recall that an oblivious transfer (OT) [Rab81, EGL85] oracle computes a two-party function that allows a receiver to obtain one out of two strings held by a sender, without revealing to the sender which of the two strings it chose. We refer to secure computation in the presence of an ideal OT oracle as secure computation in the OT-Hybrid model. The advantage of working in the OT-Hybrid model is that it enables unconditional security. Using composition theorems for secure computation, the ideal OT oracle can be replaced by (a black-box access to) any secure OT implementation.

Our two-party protocols are *statistically* secure in the OT-Hybrid model. That is, the parties are given a security parameter 1^k and we require that for every adversary, there exists an adversary in the ideal world whose view is $\exp(-k)$-close to that of the real world adversary. The running time of the two parties may depend polynomially on k and polylogarithmically on (an upper bound on) the group size. Since the latter may be inferred from the length of the identifiers of group elements in the standard generic group model, this convention does not violate the "black-box" aspect of the model (and in particular does not allow the protocol to learn the group structure).

In Section 6.13.3 we show a secure a two party passive protocol and in Section 6.13.3 we show, using the compiler of Ishai, Prabhakaran and Sahai [IPS08], how to transform the passive protocol together with the unconditionally secure active protocol of Section 6.13.2 into a two party actively secure protocol.

A two-party passively secure protocol

We denote the two parties by sender and receiver. The secure evaluation of a general circuit over a group reduces to securely computing an iterated group product of the form $c = a_1 b_1 a_2 b_2 \cdots a_m b_m$ where a_i are the sender's inputs, b_i are the receiver's inputs, and the receiver gets the output c. (Indeed, this suffices for generating random shares of the product xy given shares of x and shares of y.)

We will show a protocol for computing such an iterated group product in the OT-Hybrid model where the view of each party can be simulated, up to $2^{-\Omega(k)}$ statistical distance, given its input and output.

Before proceeding to the protocol we state and prove a lemma that will be useful in the analysis of our protocol. The lemma generalizes a lemma of Impagliazzo and Naor [IN96]

on the subset sum problem. Recall that the *statistical distance* between two distributions $\mathcal{D}_1, \mathcal{D}_2$ with support S is defined as $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2) \triangleq \max_{A \subseteq S} |\Pr[\mathcal{D}_1 \in S] - \Pr[\mathcal{D}_2 \in S]|$.

Lemma 6.24. *Let G be a finite group, let $k > 0$ be a security parameter and let $s = 4k + \log(|G|)$. Then, except for a 2^{-k} fraction of $r_1, \ldots, r_s \in G$ it holds that:*

$$\mathsf{SD}(r_1^{w_1} \cdots r_s^{w_s}, g) < 2^{-k},$$

where $w_1, \ldots, w_s$ are uniformly distributed bits and g is uniformly distributed in G.

Proof. Consider the universal hash function family $\{h_{r_1,\ldots,r_s} : \{0,1\}^s \to G\}_{r_1,\ldots,r_s \in G}$ defined as $h_{r_1,\ldots,r_s}(w) = \prod_{i=1}^{s} r_i^{w_i}$. Then:

$$\mathop{\mathbf{E}}_{r_1,\ldots,r_s \in G}\left[\mathsf{SD}_{\substack{w \in \{0,1\}^s \\ g \in G}}(h_{r_1,\ldots,r_s}(w), g)\right] = \mathsf{SD}_{\substack{g, r_1,\ldots,r_s \in G \\ w \in \{0,1\}^s}}\Big((r_1, \ldots, r_s, h_{r_1,\ldots,r_s}(w)), (r_1, \ldots, r_s, g)\Big)$$

which, by the Leftover Hash Lemma[24] [HILL99], is at most $\sqrt{\frac{|G|}{2^s}} = 2^{-2k}$. The lemma follows by an application of Markov's inequality. □

Lemma 6.24 is used to get a statistical secret sharing of group elements in the following way. The receiver publishes $s = 4k + \log|G|$ public random group elements $r_1, ..., r_s$. Now, for any $w \in \{0,1\}^s$ we can represent a group element g using (w, g') where $g' = g * \prod r_i^{w_i}$. By the lemma, if we pick w at random then, except with 2^{-k} probability over the choice of r_i, the element g' is 2^{-k}-close to uniform even when conditioned on the r_i. This means that (w, g') form a statistical 2-out-of-2 secret sharing of g given public randomness $(r_1, \ldots, r_s)$.

Using the above method for secret sharing group elements, the iterated group product protocol proceeds as follows:

1. The receiver picks $r_1, ..., r_s \in G$ uniformly at random and sends $r_1, \ldots, r_s$ to the sender.

2. Using these r_i, the receiver splits each of his inputs b_i into (w_i, b_i') and send b_i' to the sender. By Lemma 6.24, the view of the sender can be simulated up to $2^{-\Omega(k)}$ statistical distance without knowing b_i.

3. The sender can now write c as an iterated group product in which some slots include his own inputs a_i, some slots include the values b_i', and the rest include one of two options: either 1 or an element r_j^{-1}. The choice between the two options is determined by the bit w_i known to the receiver.

[24] A simplified version of the Leftover Hash Lemma states that if h is selected at random from a universal hash function family from $\mathcal{X}$ to $\mathcal{Y}$ then the distribution $(h, h(\mathcal{X}))$ and $(h, \mathcal{Y})$ are at most $\sqrt{\frac{|\mathcal{Y}|}{|\mathcal{X}|}}$-close (see, e.g., [Gol08, Appendix D]).

4. The sender randomizes this product using Kilian's group product randomization technique [Kil88]. That is, we first write the product as $g_1(c_1) * \cdots * g_n(c_n)$ where each g_i is some function from $\{0,1\}$ to G known to the sender and c_i is a choice bit known to the receiver. The sender then picks random group elements $q_1, ..., q_{n-1}$ and lets $g'_1(c_1) = g_1(c_1)q_1, g'_2(c_2) = q_1^{-1}g_2(c_2)q_2, \ldots, g'_n(c_n) = q_{n-1}^{-1}g_n(c_n)$.

5. The sender and receiver invoke the OT oracle n times, delivering to the receiver the values $g'_i(c_i)$ where c_i are the receiver's choice bits. The n received group elements contain no more information than $z = \prod g_i(c_i)$.

The above protocol implies the following theorem.

Theorem 6.25. *Let f be a two-party functionality defined by a circuit of size s over a group G. Then f can be realized in the OT-Hybrid model by a protocol which has statistical security against one passively corrupted party. The protocol requires a total of $O((k + \log |G|)s)$ group operations for achieving 2^{-k} simulation error.*

A two party actively secure protocol

To get active security in the two-party model, we use the general compiler of [IPS08]. This compiler yields a 2-party protocol for a function f with statistical security against active corruption in the OT-hybrid model by making a *blackbox* use of the following two ingredients:

1. an "outer protocol" for f which employs k auxiliary parties (servers) in addition to the two parties (clients) holding the inputs; this protocol should be perfectly or statistically secure against active corruption provided that only some constant fraction the servers can be corrupted; and

2. an "inner protocol" for implementing a reactive two-party functionality ("inner functionality") corresponding to the local computation of each server, in which the server's state is secret-shared between the two clients. In contrast to the outer protocol, this protocol only needs to be secure against *passive* corruption. The inner protocol can be implemented in the OT-Hybrid model.

We instantiate the outer protocol with the efficient protocol from Theorem 6.24 and the inner protocol with the two-party protocol from Theorem 6.25. The details of this combination are analogous to those used for secure two-party computation over black-box rings in [IPS09]. The crucial observation is that when the parties in the outer protocol only make a blackbox use of a group G, then the functionality realized by the inner protocol is also cast in the blackbox model and so parties in the inner protocol can make a blackbox use of G. We refer the reader to [IPS09] for more details.

The result for the active two-party model is summarized by the following theorem.

Theorem 6.26. *Let f be a two-party functionality defined by a circuit of size s over a group G. Then f can be realized in the OT-Hybrid model by a protocol which has*

statistical security against one actively corrupted party. The protocol requires a total of $\mathrm{poly}(k) \cdot \log |G| \cdot s$ *group operations for achieving* 2^{-k} *simulation error.*

Chapter 7

On Rigid Matrices and U-Polynomials

7.1 Matrix Rigidity

Motivated by the problem of proving lower bounds for linear circuits, [Val77] introduced the notion of *matrix rigidity*. Let A be an $m \times n$ matrix over a finite field $\mathbb{F}$. We consider the linear mapping $x \mapsto Ax$, and ask how hard it is to compute by linear circuits. A linear circuit is a circuit on n inputs and m outputs, that is composed of some subset of the gates $\{G_{a,b}\}_{a,b\in\mathbb{F}}$, where $G_{a,b}$ is a fan-in two gate that computes the function $G_{a,b}\colon \mathbb{F}\times\mathbb{F}\to\mathbb{F}$ defined by $G_{a,b}(x,y) = ax + by$. The *size* of a circuit is the number of gates it contains. The *depth* of a circuit is the number of gates in the longest path from an input to an output. In this chapter, which covers a joint work with Alon [AC13], we will focus on $\mathbb{F} = \mathbb{F}_2$. Note that in this case the only allowed gate is the Parity gate.

A simple counting argument shows that most linear mappings with $m = \Theta(n)$ have size $\Omega(n^2/\log n)$. Nevertheless, currently there is no explicit linear mapping we know of, that has size $\omega(n)$. In fact, even after more than three decades of study, there is no known linear mapping that cannot be computed by a circuit with linear size and logarithmic depth simultaneously. [Val77] suggested a route for resolving the latter problem by giving a sufficient condition for a matrix A that ensures the matrix cannot be computed by linear-sized linear circuits with logarithmic depth. The property suggested by Valiant essentially requires that the rank of A is robust against alterations to a small number of entries. There are a few variants of this notion. For more information, we refer the reader to a recent survey by [Lok09].

Definition 7.1 (Matrix Rigidity)**.** *Let A be an $m \times n$ matrix over $\mathbb{F}_2$. A is called* (k,d)-rigid *if for every $m \times n$ matrix R with rank at most k, the matrix $A - R$ contains a row with at least d non-zero entries.*

The above definition states that a matrix A is (k,d)-rigid if one cannot decrease the rank of A to k by altering less than d entries in each row of A. The following theorem, due to Valiant, has motivated the study of matrix rigidity.

Theorem 7.1 ([Val77])**.** *Let A be an $m \times n$ matrix over $\mathbb{F}_2$, where $m = O(n)$. If A is $(\Omega(n), n^{\Omega(1)})$-rigid, then any linear circuit with depth $O(\log n)$ that computes A, has size $\Omega(n \cdot \log\log n)$.*

[APY09] presented the problem of constructing rigid matrices in an equivalent, yet conceptually different way. To describe it, we need the following standard definition of distance between a point and a set.

Definition 7.2. *For $x \in \mathbb{F}_2^n$ and $U \subseteq \mathbb{F}_2^n$, define the* Hamming distance of x from U *by $\mathsf{dist}_{\mathsf{H}}(x, U) = \min_{u \in U} |x + u|$, where $|v|$ denotes the Hamming weight of the vector v.*

Definition 7.3 (Rigid Sets)**.** *A set $S \subseteq \mathbb{F}_2^n$ is called* (n, k, d)-rigid *if for every subspace $U \subseteq \mathbb{F}_2^n$ of dimension k, it holds that $\max_{s \in S} \mathsf{dist}_{\mathsf{H}}(s, U) \geq d$.*

It is an easy exercise to show that an (n, k, d)-rigid set S with size m induces a (k, d)-rigid matrix with size $m \times n$, and vice versa. We will also discuss the following stronger variant of rigid sets.

Definition 7.4 (Strong Rigid Sets)**.** *A set $S \subseteq \mathbb{F}_2^n$ is called* strong (n, k, d)-rigid *if for any dimension k subspace $U \subseteq \mathbb{F}_2^n$, it holds that $\mathbb{E}_{s \sim S}[\mathsf{dist}_{\mathsf{H}}(s, U)] \geq d$.*

Clearly, every strong rigid set is a rigid set with the same parameters. For implications to complexity theory using Valiant's Theorem (7.1), one needs to construct an $(n, \Omega(n), n^{\Omega(1)})$-rigid set with size $O(n)$. Thus, historically, the study of matrix rigidity focused on the tradeoff between k and d while fixing $m = O(n)$ [Fri93, Lok95, SSS97, KR98]. Given that after more than three decades of research we seem to be far from achieving a tradeoff between k, d that would suffice for establishing Theorem 7.1, [APY09] initiated the study of the tradeoff between m and d while fixing $k = n/2$. In this setting one no longer insists on $m = O(n)$, but aims at getting m as small as possible as a function of d. The end goal would be to achieve $m = O(n) + d^c$, for any constant c, which would suffice for applications to lower bounds via Theorem 7.1. Unfortunately, the construction of [APY09] yields $m = n \cdot 2^{O(d)}$.

Before presenting the contribution of this chapter, that covers [AC13], we mention here a completely different approach for constructing rigid matrices. [Dvi10] related the rigidity of a matrix with the local correctable properties of the linear code induced by the matrix. More specifically, the author showed that if the generating matrix of any locally decodable code is not rigid, then there exists a locally correctable code with rate close to one. Hence, proving that locally correctable codes with such parameters do not exist will give rise to explicit construction of rigid matrices.

7.2 Our Contribution

In a joint work with Alon [AC13], on which this chapter is based on, we suggest a new approach for constructing rigid sets (or equivalently, rigid matrices). Throughout the chapter we let $\rho \in (0, 1)$ be a parameter. The parameter ρ does not need to be constant,

though it is best to think of ρ as a constant that is close to 1. Central to our approach are polynomials with a special structure, which we call *U-polynomials*.

Definition 7.5 (U-Polynomials)**.** *For a subspace $U \subset \mathbb{F}_2^n$, define the mapping $p_U \colon \mathbb{F}_2^n \to \mathbb{R}$ as follows* [1]

$$p_U(x) = \frac{1}{\mathsf{W}_\rho(U)} \cdot \sum_{u \in U} \rho^{|u|} \cdot (-1)^{<u,x>},$$

where $\mathsf{W}_\rho(U) = \sum_{u \in U} \rho^{|u|}$ is the weight enumerator *of U with parameter ρ. The mapping p_U is called the* U-polynomial.

We emphasize that the mapping p_U is indeed a polynomial if one chooses to represent it using the the domain $\{\pm 1\}$ rather than $\{0,1\}$. More precisely, one can identify the function p_U defined above with the polynomial $p'_U \colon \{\pm 1\}^n \to \mathbb{R}$ given by

$$p'_U(x) = \frac{1}{\mathsf{W}_\rho(U)} \cdot \sum_{u \in U} \rho^{|u|} \cdot \prod_{i|u_i=1} x_i,$$

for the same $U \subset \mathbb{F}_2^n$. Namely, the respective polynomial has a monomial for each element in U, with coefficient that depends only on the weight of the element.

Let $\mathcal{P}_k$ be the class of all U-polynomials p_U, where $U \subset \mathbb{F}_2^n$ has dimension k. One can show that for any subspace U and for any $x \in \mathbb{F}_2^n$, $0 < p_U(x) \leq 1$ [2], where equality to 1 holds if and only if $x \in U^\perp$. Our first main theorem shows that $p_{U^\perp}(x)$ is related to the Hamming distance of x from U.

Theorem 7.2. *Let $U \subset \mathbb{F}_2^n$ be a subspace. Then, for any $\rho \in (0,1)$, and any $x \in \mathbb{F}_2^n$,*

$$\mathsf{dist}_{\mathsf{H}}(x, U) \geq \left(\log \frac{1+\rho}{1-\rho}\right)^{-1} \cdot \log \frac{1}{p_{U^\perp,\rho}(x)}.$$

By Theorem 7.2, the problem of explicitly constructing an (n,k,d)-rigid set is reduced to that of explicitly constructing a set S such that for every $U \subset \mathbb{F}_2^n$ with dimension $n-k$, there exists $s \in S$ such that $p_U(s) \leq 2^{-\Omega(d)}$. We informally refer to such sets as *hitting sets for $\mathcal{P}_{n-k}$*, as for values of k of interest (say, $k = \alpha n$ for a constant $\alpha \in (0,1)$), p_U evaluated on a random point is exponentially small in n, with high probability.

Similarly, by Theorem 7.2, the problem of explicitly constructing a *strong* (n,k,d)-rigid set is reduced to the problem of explicitly constructing a set S such that for every $U \subset \mathbb{F}_2^n$ of dimension $n-k$, for at least, say, half of the elements $s \in S$ it holds that $p_U(s) \leq 2^{-\Omega(d)}$. If $\mathcal{A}$ is an algorithm that given n, k, d as inputs, constructs such a set S, then we informally refer to $\mathcal{A}$ as a *pseudorandom generator for $\mathcal{P}_{n-k}$*.

[1] For the sake of readability, we suppress ρ in the notation when it is clear from context.

[2] The upper bound is trivial, while the lower bound is implicit in the proof of Theorem 7.2.

Small-bias sets as rigid sets. Unfortunately, we are unable to use the reduction above to obtain improved rigid sets. We hope that this reduction will be used as a starting point for future constructions of rigid sets. In this work we make use of Theorem 7.2 to show that small-bias sets (see Section 2) are strong rigid sets. However, their size is larger than desired.

Theorem 7.3. *Let n, d be such that $d \leq c \cdot n$ for some suitable constant $0 < c < 1$. Let $S \subset \mathbb{F}_2^n$ be an $\mathsf{exp}(-d)$-biased set. Then S is an $(n, n/2, d)$-strong rigid set.*

Theorem 7.3 is implicit in the work of [AS10], who studied the related remote point problem. In [AC13] we give three proofs for the fact that any small-bias set is a rigid set, with related parameters. One of our proofs is an easy corollary of Theorem 7.2.

Rigid sets from unbalanced expanders. We further show how to construct rigid sets from unbalanced expanders (see Section 2.5 for a formal definition of unbalanced expanders). Specifically, we prove the following theorem.

Theorem 7.4. *Let $G = (L, R, E)$ be a $(k_{max}, 2/3)$-bipartite expander with $L = [m]$, $R = [n]$ and left-degree $4d$. For every $\ell \in L$ define a vector $c_\ell \in \mathbb{F}_2^n$ as follows: for $i \in [n]$,*

$$(c_\ell)_i = \begin{cases} 1, & \ell i \in E; \\ 0, & \textit{otherwise.} \end{cases}$$

If

$$\sum_{i=0}^{k_{max}/2} \binom{m}{i} > 2^k,$$

then the set $C = \{c_\ell : \ell \in L\}$ is (n, k, d)-rigid.

The proof of Theorem 7.4 applies a different argument than any of the proofs for Theorem 7.3. In particular, it does not use the reduction to the problem of constructing hitting sets for U-polynomials. Moreover, it is interesting to note that the two rigid sets constructed in Theorem 7.3 and Theorem 7.4 have a different structure. Indeed, a typical element in a small-bias set $S \subseteq \mathbb{F}_2^n$ has weight roughly $n/2$. On the other hand, every element in the construction that is based on unbalanced expanders has weight at most $4d$. Nevertheless, plugging the unbalanced expander that is obtained by the probabilistic method [3] yields an (n, k, d)-rigid set with size $n \cdot \mathsf{exp}(d \cdot k/n)$. For $k = n/2$ this coincides with the size of the rigid set obtained by small-bias sets. Furthermore, we give a reduction from the construction of (n, k, d)-strong rigid sets to the construction of $(n, n/2, d)$-strong rigid sets. This reduction together with the construction from Theorem 7.3 yields the exact same size as the construction based on unbalanced expanders.

[3] The state of the art explicit construction for unbalanced expanders due to [GUV09] falls short of achieving the parameters of the probabilistic construction. This in turn gives a rigid set with a somewhat larger size.

7.2.1 Basic Facts in Fourier analysis

In this section we cover the required tools needed from Fourier analysis. We refer the reader to a recent book of [O'D] for a comprehensive treatment.

Consider all functions of the form $f : \mathbb{F}_2^n \to \mathbb{R}$. These form a vector space $\mathcal{F}$ over $\mathbb{F}_2$, where addition is conducted in a point-wise manner, that is, for every $f, g \in \mathcal{F}$, the function $f + g$ is defined by $(f + g)(x) = f(x) + g(x)$. For every $\alpha \in \mathbb{F}_2^n$, $\chi_\alpha : \mathbb{F}_2^n \to \mathbb{R}$ is defined by $\chi_\alpha(x) = (-1)^{<\alpha,x>}$. It is easy to see that $\{\chi_\alpha : \alpha \in \mathbb{F}_2^n\}$ is a basis for $\mathcal{F}$. This basis is called the *Fourier basis* for $\mathcal{F}$. Define an inner product over $\mathcal{F}$ as follows: for every $f, g \in \mathcal{F}$, $< f, g >= 2^{-n} \cdot \sum_{x \in \mathbb{F}_2^n} f(x)g(x)$. It is easy to see that $< \chi_\alpha, \chi_\beta > = 1$ if $\alpha = \beta$, and 0 otherwise. Under the above inner product, the Fourier basis is an orthonormal basis. Thus, every $f \in \mathcal{F}$ can be expanded according to the Fourier basis as follows $f = \sum_{\alpha \in \mathbb{F}_2^n} \widehat{f}(\alpha)\chi_\alpha$, where $\widehat{f}(\alpha) = < f, \chi_\alpha >$ is called *the Fourier coefficient* of f on point α.

Let $0 \leq \varepsilon \leq 1$. The *noise operator* $T_\varepsilon : \mathcal{F} \to \mathcal{F}$ is defined by

$$T_\varepsilon(f)(x) = \sum_{y \in \mathbb{F}_2^n} \left(\frac{1-\varepsilon}{2}\right)^{|y|} \cdot \left(\frac{1+\varepsilon}{2}\right)^{n-|y|} f(x+y)\,.$$

Fact 7.6. *For every $f \in \mathcal{F}$, $0 \leq \varepsilon \leq 1$ and $\alpha \in \mathbb{F}_2^n$,*

$$\widehat{T_\varepsilon(f)}(\alpha) = \varepsilon^{|\alpha|} \cdot \widehat{f}(\alpha).$$

7.3 U-Polynomials

This section studies U-polynomials. We start by proving Theorem 7.2. The main intuition behind the proof of Theorem 7.2 is to work with "scalar fields"[4] rather than with "distances". Recall that a scalar field associates a scalar, which in our case is some non-negative real number, to each point in the space. We now elaborate on this. Let $U \subseteq \mathbb{F}_2^n$ be a subspace. Imagine that at every point $u \in U$ we place a source of light that emits radiation to its surrounding, with intensity that decays with distance. Then, every point $x \in \mathbb{F}_2^n$ senses the sum of radiations coming to it from all points in U. From this perspective, finding a point that is far from U boils down to locating a point that senses a small amount of radiation, that is, a dark point. The formal definition of this energy function is as follows.

Definition 7.7. *For a parameter $\rho \in (0, 1)$ and a subspace $U \subseteq \mathbb{F}_2^n$, define the function* $\mathsf{energy}_{U,\rho} : \mathbb{F}_2^n \to \mathbb{R}$ *as follows*

$$\mathsf{energy}_{U,\rho}(x) = \frac{1}{\mathsf{W}_\rho(U)} \cdot \sum_{u \in U} \rho^{|u+x|}\,.$$

[4]Here the word field takes its meaning from physics and has nothing to do with algebraic fields.

When it is not needed to specify one or more of the parameters ρ, U, we omit them. We note that $\mathsf{energy}_U(x) \in (0,1]$, and that $\mathsf{energy}_U(x) = 1$ if and only if $x \in U$. (The lower bound is obvious, whereas the upper bound and the characterization of equality follows from Equation (7.3) below.) Thus, not surprisingly, a maximum amount of radiation is sensed on the subspace U itself. Moreover, for a uniformly sampled $x \in \mathbb{F}_2^n$, $\mathsf{energy}_U(x)$ is exponential in $\Omega(k-n)$, where $k = \mathsf{dim}(U)$. That is, a typical point in $\mathbb{F}_2^n$ senses a small amount of radiation, and so most of $\mathbb{F}_2^n$ is dark.

For the proof of Theorem 7.2, we will need the following theorem due to MacWilliams (see, e.g., [MS77]), that relates the weight enumerator of a subspace with that of its dual. We state the theorem for the binary field only.

Theorem 7.5 (MacWilliams's Theorem)**.** *Let $U \subseteq \mathbb{F}_2^n$ be a subspace of dimension k. For every $0 < \rho < 1$ it holds that*

$$\mathsf{W}_\rho(U^\perp) = \frac{(1+\rho)^n}{2^k} \cdot \mathsf{W}_{\frac{1-\rho}{1+\rho}}(U).$$

We are now ready to prove Theorem 7.2.

Proof of Theorem 7.2. Let $1_U : \mathbb{F}_2^n \to \{0,1\}$ be the characteristic function for U. That is, $1_U(x) = 1$ if and only if $x \in U$. Then,

$$\begin{aligned}
T_\rho(1_U)(x) &= \sum_{y \in \mathbb{F}_2^n} \left(\frac{1-\rho}{2}\right)^{|y|} \cdot \left(\frac{1+\rho}{2}\right)^{n-|y|} \cdot 1_U(x+y) \\
&= \left(\frac{1+\rho}{2}\right)^n \cdot \sum_{y \in \mathbb{F}_2^n} \left(\frac{1-\rho}{1+\rho}\right)^{|y|} \cdot 1_U(x+y) \\
&= \left(\frac{1+\rho}{2}\right)^n \cdot \sum_{u \in U} \left(\frac{1-\rho}{1+\rho}\right)^{|u+x|} \\
&= \left(\frac{1+\rho}{2}\right)^n \cdot \mathsf{W}_{\frac{1-\rho}{1+\rho}}(U) \cdot \mathsf{energy}_{U,\frac{1-\rho}{1+\rho}}(x). \qquad (7.1)
\end{aligned}$$

On the other hand, it is easy to see that

$$\widehat{1_U}(\alpha) = \begin{cases} 2^{k-n}, & \alpha \in U^\perp; \\ 0, & \text{otherwise.} \end{cases}$$

Hence, by Fact 7.6

$$\begin{aligned} T_\rho(1_U)(x) &= \sum_{\alpha\in\mathbb{F}_2^n} \widehat{T_\rho(1_U)}(\alpha)\cdot(-1)^{<\alpha,x>} \\ &= \sum_{\alpha\in\mathbb{F}_2^n} \widehat{1_U}(\alpha)\cdot\rho^{|\alpha|}\cdot(-1)^{<\alpha,x>} \\ &= 2^{k-n}\cdot\sum_{\alpha\in U^\perp}\rho^{|\alpha|}\cdot(-1)^{<\alpha,x>} \\ &= 2^{k-n}\cdot \mathsf{W}_\rho(U^\perp)\cdot p_{U^\perp,\rho}(x) \\ &= \left(\frac{1+\rho}{2}\right)^n\cdot \mathsf{W}_{\frac{1-\rho}{1+\rho}}(U)\cdot p_{U^\perp,\rho}(x), \end{aligned} \tag{7.2}$$

where the last equality follows by Theorem 7.5. By Equations (7.1),(7.2) we have that

$$\mathsf{energy}_{U,\frac{1-\rho}{1+\rho}}(x) = p_{U^\perp,\rho}(x).\ ^{5} \tag{7.3}$$

Assume now that $\mathsf{dist}_{\mathsf{H}}(x,U) = d$. Then there exists $w\in U$ such that $|x+w| = d$. Therefore,

$$\begin{aligned} \mathsf{W}_{\frac{1-\rho}{1+\rho}}(U)\cdot\mathsf{energy}_{U,\frac{1-\rho}{1+\rho}}(x) &= \sum_{u\in U}\left(\frac{1-\rho}{1+\rho}\right)^{|u+x|} \\ &\overset{(1)}{=} \sum_{u\in U}\left(\frac{1-\rho}{1+\rho}\right)^{|u+x+w|} \\ &\overset{(2)}{\geq} \sum_{u\in U}\left(\frac{1-\rho}{1+\rho}\right)^{|u|+|x+w|} \\ &= \left(\frac{1-\rho}{1+\rho}\right)^d\cdot\sum_{u\in U}\left(\frac{1-\rho}{1+\rho}\right)^{|u|} \\ &= \left(\frac{1-\rho}{1+\rho}\right)^d\cdot\mathsf{W}_{\frac{1-\rho}{1+\rho}}(U). \end{aligned}$$

Equality (1) uses the fact that U is a subspace, and in particular, the fact that for every $w\in U$, the function $f(u) = u+w$ is a bijection from U to U. Inequality (2) holds by the triangle inequality, and by the fact that $(1-\rho)/(1+\rho) < 1$. Thus, by Equation (7.3),

$$p_{U^\perp,\rho}(x) \geq \left(\frac{1-\rho}{1+\rho}\right)^d,$$

which concludes the proof of the theorem. □

[5] By this equality, it follows that U-polynomials are positive.

One may ask whether there is a quantitative loss in the reduction from the problem of constructing rigid sets to the problem of constructing hitting sets for U-polynomials. Similarly, is there a quantitative loss in the reduction from the problem of constructing strong rigid sets to the problem of constructing pseudorandom generators for U-polynomials ? We answer this question negatively in Corollary 7.7.2. To this end, we need the following claim about the weight enumerator.

Claim 7.7.1. *For any $\rho \in (0,1)$ and for any subspace $U \subseteq \mathbb{F}_2^n$ of dimension $n/2$, it holds that*

$$\mathsf{W}_\rho(U) \geq \left(\frac{1+\rho}{\sqrt{2}}\right)^n.$$

Proof. Consider a coset $x+U$ of the subspace U. By Equation (7.3),

$$\frac{1}{\mathsf{W}_\rho(U)} \cdot \sum_{u \in U} \rho^{|u+x|} = \mathsf{energy}_{U,\rho}(x) = p_{U^\perp, \frac{1-\rho}{1+\rho}}(x).$$

The right hand side is clearly bounded above by 1, and so $\sum_{u\in U}\rho^{|u+x|} \leq \sum_{u\in U}\rho^{|u|}$ for all x. However, the summation of $\sum_{u\in U}\rho^{|u+x|}$ over all $2^{n/2}$ cosets is exactly $\sum_{w\in\mathbb{F}_2^n}\rho^{|w|} = (1+\rho)^n$, which completes the proof. □

Claim 7.7.2. *Let $\rho \in (\sqrt{2}-1, 1)$ be a constant parameter. Then, with high probability, a random set $S \subset \mathbb{F}_2^n$ of size $O(n)$ has the following property: for every $p_U \in \mathcal{P}_{n/2}$, for at least half of the elements $s \in S$ it holds that $p_U(s) \leq 2^{-\Omega(n)}$.*

Proof. Let $p_U \in \mathcal{P}_{n/2}$. Then

$$\begin{aligned}
\mu &\triangleq \mathbb{E}_{x\sim\mathbb{F}_2^n}\left[p_U(x)\right] \\
&= \mathbb{E}_{x\sim\mathbb{F}_2^n}\left[\frac{1}{\mathsf{W}_\rho(U)} \cdot \sum_{u\in U}\rho^{|u|}(-1)^{<u,x>}\right] \\
&= \frac{1}{\mathsf{W}_\rho(U)} \cdot \sum_{u\in U}\rho^{|u|} \cdot \mathbb{E}_{x\sim\mathbb{F}_2^n}[(-1)^{<u,x>}] \\
&= \frac{1}{\mathsf{W}_\rho(U)},
\end{aligned}$$

where the last equality holds as all summands are zero but for $u=0$, which contributes 1 to the sum. By Corollary 7.7.1,

$$\mu = \frac{1}{\mathsf{W}_\rho(U)} \leq \left(\frac{\sqrt{2}}{1+\rho}\right)^n.$$

For any $\rho > \sqrt{2}-1$ the base of the exponent in the above equation is smaller than 1, and so, for any such ρ, there exists a constant $\alpha = \alpha(\rho) > 0$ such that $\mu < 2^{-\alpha n}$. Thus, by Markov's inequality,

$$\Pr_{x\sim\mathbb{F}_2^n}\left[p_U(x) > 2^{-\alpha n/2}\right] \leq 2^{-\alpha n/2}.$$

Let m be an integer to be determined later.

$$\Pr_{x_1,\ldots,x_m\sim\mathbb{F}_2^n}\left[\exists S\subseteq[m],|S|=\frac{m}{2}\ \text{ s.t. }\forall i\in S\ \ p_U(x_i)>2^{-\alpha n/2}\right] \tag{7.4}$$
$$\leq\binom{m}{m/2}\cdot\left(2^{-\alpha n/2}\right)^{m/2}.$$

The number of subspaces of dimension $n/2$ in $\mathbb{F}_2^n$ is bounded by $\binom{2^n}{n/2}$ [6], and so by the union bound, the probability that there exists U of dimension $n/2$ for which the event in Equation (7.4) holds is bounded above by

$$\binom{2^n}{n/2}\cdot\binom{m}{m/2}\cdot\left(2^{-\alpha n/2}\right)^{m/2}<2^{n^2/2}\cdot 2^m\cdot 2^{-\alpha nm/4}.$$

For $m=(7/\alpha)\cdot n$ the right hand side in the above expression is bounded by 2^{-n^2}, for large enough n. This concludes the proof of the claim. □

7.4 Small-Bias Sets as Rigid Sets

Small-bias sets introduced by [NN93], are pseudorandom objects that have found numerous applications in theoretical computer science (see, e.g., Chapter 3). For ease of readability we recall here the formal definition (see also Section 2).

Definition 7.8 (Small-bias sets, [NN93])**.** *Let $S\subseteq\mathbb{F}_2^n$. We say that S is an* ε-biased set *if for every $0\neq\alpha\in\mathbb{F}_2^n$ it holds that*

$$\left|\mathbb{E}_{s\sim S}\left[(-1)^{<\alpha,s>}\right]\right|\leq\varepsilon.$$

A minor technicality when working with small-bias sets is repetition of elements in the set. To avoid ambiguity, when working with small-bias sets we do not ignore repetitions of elements, that is, we consider small-bias sets as multi-sets. Put differently, we think of small-bias sets as sample spaces, where an element is sampled with probability that is proportional to the element's multiplicity in the set.

A simple probabilistic argument shows that there exist ε-biased sets in $\mathbb{F}_2^n$ with size $O(n/\varepsilon^2)$. Several explicit constructions of small-bias sets were introduced by [AGHP92, ABN+92, NN93, BT09]. Unfortunately, none of the explicit constructions achieves the size obtained by the probabilistic argument.

In this section we give three proofs for the fact that any small-bias set is a rigid set, with related parameters, as formalized in Theorem 7.3. As mentioned, a proof for Theorem 7.3 is implicit in [AS10]. In that paper, the authors consider the remote point problem, which is defined as follows. Given a basis for a subspace $U\subseteq\mathbb{F}_2^n$, find in time $\mathrm{poly}(n)$ a point $r\in\mathbb{F}_2^n$ such that $\mathsf{dist}_{\mathsf{H}}(r,U)$ is large. Informally speaking, the remote

[6] In fact, a tighter bound of roughly $2^{n^2/4}$ can be easily proven.

point problem can be seen as a relaxation of the problem of constructing rigid sets. Indeed, in the remote point problem, the subspace is given as an input to the algorithm (in a succinct representation), whereas rigid sets contain a point that is far from *any* subspace with small enough dimension. The authors showed that any small-bias set with appropriate parameters contain a point that is far from the given subspace U. Since the small-bias set used in their proof depends only on the dimension of U, their proof yields Theorem 7.3, and with the same parameters.

Using, for example, the construction of [ABN+92] for small-bias sets, Theorem 7.3 yields an $(n, n/2, d)$-strong rigid set with size $n \cdot \mathsf{exp}(d)$. This matches the construction of [APY09]. Applying the reduction from Section 7.6, we get an explicit construction of a strong (n, k, d)-rigid set with size $n \cdot \mathsf{exp}(d \cdot k/n)$.

7.4.1 Proof of Theorem 7.3 based on U-polynomials

Proof of Theorem 7.3. Let $S \subseteq \mathbb{F}_2^n$ be an ε-biased set, and U a subspace of dimension $n/2$. Then,

$$\begin{aligned}\mathbb{E}_{x\sim S}[p_U(x)] &= \frac{1}{\mathsf{W}_\rho(U)} \cdot \mathbb{E}_{x\sim S}\left[\sum_{u\in U} \rho^{|u|} \cdot (-1)^{<u,x>}\right] \\ &= \frac{1}{\mathsf{W}_\rho(U)} \cdot \sum_{u\in U} \rho^{|u|} \cdot \mathbb{E}_{x\sim S}\left[(-1)^{<u,x>}\right].\end{aligned}$$

Any summand except for $u = 0$ is bounded in absolute value by ε. Thus,

$$\mathbb{E}_{x\sim S}[p_U(x)] < \varepsilon + \frac{1}{\mathsf{W}_\rho(U)}.$$

Assume for now that we will pick $\varepsilon > 1/\mathsf{W}_\rho(U)$, and so we can further simplify to get $\mathbb{E}_{x\sim S}[p_U(x)] < 2\varepsilon$. Since $\log(1/x)$ is a convex function, we get, by Jensen's inequality that

$$\mathbb{E}_{x\sim S}\left[\log\left(\frac{1}{p_U(x)}\right)\right] \geq \log\left(\frac{1}{\mathbb{E}_{x\sim S}[p_U(x)]}\right) \geq \log\left(\frac{1}{2\varepsilon}\right).$$

Since we are working with subspaces of dimension $n/2$, the above equation also holds for the dual of every subspace of dimension $n/2$. Thus, by Theorem 7.2, for every subspace $U \subset \mathbb{F}_2^n$ with dimension $n/2$, it holds that

$$\mathbb{E}_{x\sim S}\left[\mathsf{dist}_{\mathsf{H}}(x, U)\right] = \Omega\left(\log\frac{1}{\varepsilon}\right).$$

Recall that in our case $m = O(n/\varepsilon^3)$, and so setting $m = n \cdot 2^{\Theta(d)}$ would give that S is an $(n, n/2, d)$-strong rigid set with size m.

We now return to the assumption we made, namely, that $\varepsilon > 1/\mathsf{W}_\rho(U)$. Eventually we chose $\varepsilon = \mathsf{exp}(-d)$, and so to justify the assumption, it is enough to show that $\mathsf{W}_\rho(U) > \mathsf{exp}(d)$. By Corollary 7.7.1 we have that $\mathsf{W}_\rho(U) \geq ((1+\rho)/\sqrt{2})^n$. For $\rho > \sqrt{2} - 1$, the base of the exponent is larger than 1. For any such ρ, there exists a constant $c = c(\rho) > 0$ such that our assumption is met as long as $d \leq c \cdot n$. □

7.4.2 The bias-reduction proof

Our second proof relies on the Parity Lemma (c.f., for example, [NN93]).

Lemma 7.9 (The Parity Lemma)**.** *Let $S \subseteq \{0,1\}^n$ be an ε-biased set. Let $T \subseteq [n]$ be a non-empty set of size k. Denote by S_T the projection of S on the index set T. Then,*

$$\mathsf{SD}(S_T, \mathrm{U}_k) \leq \varepsilon \cdot 2^{k/2}.$$

Roughly speaking, Lemma 7.9 states that the projection of a small-bias set on a small number of coordinates is close, in statistical distance, to the uniform distribution. Since a random vector is, with high probability, far from any given subspace with small dimension, one would hope that a typical vector in a small-bias set would also be far from any given subspace. This idea fails because although the bound on the statistical distance guaranteed by the Parity Lemma depends linearly on the bias of the small-bias set, it depends exponentially on n, the length of the vectors.

A natural suggestion for circumventing this problem is to partition the set of indices $[n]$ to blocks and apply the argument above to each block separately. This way, the statistical distance guaranteed by the Parity Lemma will be exponential in the block length, which can be controlled, as opposed to being exponential in n. However, this suggestion fails as well since one must take the block length large enough so that the projection of the subspace on a block would still have small dimension with respect to the block length. Indeed, otherwise a random vector would not necessarily be far from the projection.

As mentioned, the statistical distance guaranteed by the Parity Lemma depends linearly on the bias of the small-bias set and exponentially on n. The natural idea above tried to obtain a better guarantee on the statistical distance by decreasing the exponential part as it naturally seems to cause the problem. However, this idea failed. The idea behind the "bias-reduction proof" as its name suggests, is to reduce the bias enough so as to cancel the exponential loss incurred by the Parity Lemma. The way we reduce the bias is by applying the above argument not to the original small-bias set S, but rather to the set $S + \cdots + S$, where the number of summands depends on the distance, d, that we want to achieve. The bias of this sum decreases exponentially with the number of summands (see Claim 7.9.1 below). This cancels out the exponential loss we incur by the Parity Lemma, as desired. This shows that $S + \cdots + S$ is a strong rigid set with very good parameters. We then show that this implies that S itself must also be a strong rigid set, albeit with weaker parameters. We now make this formal. We need the following claim.

Claim 7.9.1. *Let S be an ε-biased set. Then, for every integer $c \geq 1$, $c \cdot S$ is an ε^c-biased set.*

Proof. For any $0 \neq \alpha \in \mathbb{F}_2^n$

$$\begin{aligned}
\left|\mathbb{E}_{x\sim c\cdot S}\left[(-1)^{<\alpha,x>}\right]\right| &= \left|\mathbb{E}_{s_1,\ldots,s_c\sim S}\left[(-1)^{<\alpha,s_1+\cdots+s_c>}\right]\right| \\
&= \left|\mathbb{E}_{s_1,\ldots,s_c\sim S}\left[\prod_{i=1}^{c}(-1)^{<\alpha,s_i>}\right]\right| \\
&= \prod_{i=1}^{c}\left|\mathbb{E}_{s_i\sim S}\left[(-1)^{<\alpha,s_i>}\right]\right| \leq \varepsilon^c.
\end{aligned}$$

□

We are now ready to give the bias-reduction proof for Theorem 7.3.

Proof of Theorem 7.3. Let S be a $2^{-c'd}$-biased set for a constant $c' > 0$ to be determined later on. Let $S' = (n/20d) \cdot S$. By Claim 7.9.1, S' is a $2^{-c'n/20}$-biased set. Let $U \subset \mathbb{F}_2^n$ be a subspace of dimension $n/2$. By standard counting arguments one can show that

$$\Pr_{x\sim\mathbb{F}_2^n}\left[\mathsf{dist}_{\mathsf{H}}(x,U) > \frac{n}{10}\right] > 0.6.$$

By the Parity Lemma (Lemma 7.9), we have that

$$\mathsf{SD}\left(S',\mathbb{F}_2^n\right) \leq 2^{-c'n/20+n/2} < 0.1,$$

where the last inequality holds for a sufficiently large constant c'. We choose c' accordingly. Thus,

$$\Pr_{x\sim S'}\left[\mathsf{dist}_{\mathsf{H}}(x,U) > \frac{n}{10}\right] > 0.5.$$

In particular, the latter implies that

$$\mathbb{E}_{x\sim S'}[\mathsf{dist}_{\mathsf{H}}(x,U)] > \frac{n}{20}.$$

Recall that $S' = (n/20d) \cdot S$, and so the above equation can be written as

$$\mathbb{E}_{s_1,\ldots,s_{n/20d}\sim S}\left[\mathsf{dist}_{\mathsf{H}}\left(\sum_{i=1}^{n/20d} s_i, U\right)\right] > \frac{n}{20}. \tag{7.5}$$

At this point we note that for every $s_1,\ldots,s_{n/20d} \in S$

$$\sum_{i=1}^{n/20d}\mathsf{dist}_{\mathsf{H}}(s_i,U) \geq \mathsf{dist}_{\mathsf{H}}\left(\sum_{i=1}^{n/20d} s_i, U\right).$$

Indeed, for $i \in [n/20d]$, let $u_i \in U$ be such that $\mathsf{dist}_{\mathsf{H}}(s_i,U) = |s_i + u_i|$. Then,

$$\sum_{i=1}^{n/20d}\mathsf{dist}_{\mathsf{H}}(s_i,U) = \sum_{i=1}^{n/20d}|s_i+u_i| \geq \left|\sum_{i=1}^{n/20d} s_i + \sum_{i=1}^{n/d} u_i\right| \geq \mathsf{dist}_{\mathsf{H}}\left(\sum_{i=1}^{n/20d} s_i, U\right),$$

where the last inequality follows since U is closed under addition. Plugging this into Equation (7.5) and using linearity of expectation, we get that $\mathbb{E}_{s\sim S}[\mathsf{dist}_{\mathsf{H}}(s,U)] > d$, which concludes the proof. □

7.4.3 The covering proof

We now give a third proof for Theorem 7.3. We need some preliminary definitions and results regarding expander graphs. For more information regarding expander graphs we refer the reader to the survey by [HLW06].

Let $G = (V, E)$ be an undirected D-regular graph on N vertices. Let A_G be the normalized adjacency matrix of G. That is, for $u, v \in V$, $(A_G)_{uv}$ equals the number of edges connecting the vertices u, v, divided by D. It is well-known that the eigenvalues of A_G are all real numbers, and that the maximum eigenvalue is 1. The graph G is called (N, D, λ)-expander if the second largest eigenvalue in absolute value is at most λ.

For a subset $S \subset V$, let $e(S)$ be the number of edges in the induced subgraph of G on S. The quantity $e(S)$ measures the density of this induced subgraph. In [AC88], the following lemma was proved. Roughly speaking, the lemma states that induced subgraphs of expanders have approximately the "right" density.

Theorem 7.6 ([AC88], Lemma 2.3)**.** *Let $G = (V, E)$ be an (N, D, λ)-expander. Then, for any set $S \subseteq V$ with size $|S| = \alpha N$*

$$\left| e(S) - \frac{1}{2} D\alpha^2 N \right| \leq \frac{1}{2} \lambda D \alpha (1 - \alpha) N.$$

We also need the following theorem proved in [AR94].

Theorem 7.7 ([AR94])**.** *Let $S \subseteq \mathbb{F}_2^n$ be an ε-biased set. Define the graph $G_S = (V, E)$ as follows. $V = \mathbb{F}_2^n$, and an edge connects a pair of vertices u, v if and only if $u + v \in S$. Then, G_S is a $(2^n, |S|, \varepsilon)$-expander.*

With the two theorems above we are ready to prove the following lemma. A similar lemma was proved by [PR04] and by [AS10]. Here we give a somewhat simpler proof.

Lemma 7.10. *Let $S \subseteq \mathbb{F}_2^n$ be an ε-biased set. Then, for any subspace $U \subseteq \mathbb{F}_2^n$,*

$$\left| \frac{|S \cap U|}{|S|} - \frac{|U|}{2^n} \right| \leq \varepsilon.$$

Remark. Recall that a set S is ε-biased if for every nonzero $x \in \{0,1\}^n$ it holds that $|\Pr_{s \sim S}[\langle x, s \rangle = 0] - \Pr_{s \sim S}[\langle x, s \rangle = 1]| \leq \varepsilon$. Equivalently, S is ε-biased if for every subspace U with co-dimension 1 it holds that

$$\left| \frac{|S \cap U|}{|S|} - \frac{|U|}{2^n} \right| \leq \frac{\varepsilon}{2}.$$

Lemma 7.10 shows that if S is ε-biased, the above equation holds (up to a factor of 2) for all subspaces, regardless of their dimension.

Proof. Define the graph $G_S = (V, E)$ as in Theorem 7.7. That is $V = \mathbb{F}_2^n$, and an edge connects a pair of vertices u, v if and only if $u + v \in S$. By Theorem 7.7, G_S is a $(2^n, |S|, \varepsilon)$-expander. Let $U \subset \mathbb{F}_2^n = V$ be a subspace of dimension k. For $u \in U$, the degree of u in the induced subgraph of G_S on U is

$$|\{s \in S : u + s \in U\}| = |\{s \in S : s \in U\}| = |S \cap U|.$$

Thus,

$$e(S) = \frac{1}{2} \cdot |U| \cdot |S \cap U|.$$

By Theorem 7.6,

$$\left| |U| \cdot |S \cap U| - |S| \cdot \left(\frac{|U|}{2^n}\right)^2 \cdot 2^n \right| \leq \varepsilon \cdot |S| \cdot |U|,$$

or equivalently,

$$\left| \frac{|S \cap U|}{|S|} - \frac{|U|}{2^n} \right| \leq \varepsilon,$$

which concludes the proof of the lemma. □

Proof of Theorem 7.3. Let $U \subset \mathbb{F}_2^n$ be a subspace of dimension $n/2$. We now describe the covering of the neighborhood of U, proposed in [APY09]. Partition the n unit vectors of $\mathbb{F}_2^n$ into $8d$ sets $B_1, \ldots, B_{8d}$ of size $n/8d$ each. For every set $I \subseteq [8d]$ with size $|I| = 2d$, define

$$U_I = \mathsf{Span}\left(U \cup \bigcup_{i \in I} B_i\right).$$

We note that $\mathsf{dim}(U_I) \leq 3n/4$ for every I, as we add to U, which has dimension $n/2$, $(n/8d) \cdot 2d$ unit vectors, thus increasing U's dimension by at most $n/4$. Moreover, it is easy to see that every vector x satisfying $\mathsf{dist}_{\mathsf{H}}(x, U) \leq 2d$ is contained in U_I for some I. Let S be an ε-biased set. By Lemma 7.10, for every I as above,

$$|S \cap U_I| \leq |S| \cdot \left(2^{-n/4} + \varepsilon\right).$$

There are $\binom{8d}{2d} < 120^d$ such sets I, and as mentioned, they cover the $2d$-neighborhood of U. Therefore, S intersects the $2d$-neighborhood of U in at most $120^d \cdot |S| \cdot \left(2^{-n/4} + \varepsilon\right)$ vectors. As we assume $d \leq c \cdot n$, for small enough constant c, setting $\varepsilon = 120^{-d}/4$ implies that at most half of the vectors in S are contained in the $2d$-neighborhood of U. Thus, $\mathbb{E}_{s \sim S}\left[\mathsf{dist}_{\mathsf{H}}(s, U)\right] \geq d$. □

7.5 Rigid Sets from Unbalanced Expanders

We start this section by recalling some preliminary definitions and results regarding unbalanced expanders (see also Section 2). Let $G = (L, R, E)$ be a bipartite graph with $|L| = m$, $|R| = n$, and left-degree d. For a set $S \subseteq L$ define

$$\Gamma(S) = \{r \in R : \exists s \in S \text{ such that } sr \in E\},$$

and

$$\Gamma_1(S) = \{r \in R : \exists! s \in S \text{ such that } sr \in E\}.$$

G is called $(k_{max}, 1-\varepsilon)$-*bipartite-expander* if for every $S \subseteq L$ with size at most $k_{\max}$, it holds that $|\Gamma(S)| \geq (1-\varepsilon)d|S|$. G is called $(k_{max}, 1-\varepsilon)$-*unique neighbor expander* if for every $S \subseteq L$ with size at most $k_{\max}$ it holds that $|\Gamma_1(S)| \geq (1-\varepsilon)d|S|$. The following simple well-known fact relates the two definitions.

Fact 7.11. *Every $(k_{max}, 1-\varepsilon)$-bipartite expander is a $(k_{max}, 1-2\varepsilon)$-unique neighbor expander.*

Proof. Consider a nonempty set of left-vertices $S \subseteq L$ of size at most $k_{\max}$. The number of outgoing edges from S is $d \cdot |S|$. Hence,

$$d \cdot |S| \geq 1 \cdot |\Gamma_1(S)| + 2 \cdot (|\Gamma(S)| - |\Gamma_1(S)|).$$

The proof then follows since $|\Gamma(S)| \geq (1-\varepsilon)d|S|$. □

We will be interested in the case where $m = \omega(n)$. Such bipartite expanders are called *unbalanced expanders*. The following fact shows that given any plausible $n, d, k_{\max} \in \mathbb{N}$ and $\varepsilon \in (0,1)$, there exist highly unbalanced expanders, that is, $(k_{\max}, 1-\varepsilon)$-bipartite expanders with large m.

Fact 7.12. *Let $n, d \in \mathbb{N}$, and let $\frac{1}{d} < \varepsilon < 1$. For any $k_{max} \leq e^{-2/\varepsilon} \cdot \frac{n}{d}$ there exists a $(k_{max}, 1-\varepsilon)$-bipartite expander with $m = \Omega(k_{max} \cdot e^d)$.*

In particular, for constant ε, Fact 7.12 implies that for large enough n, d, there exist $(k_{\max}, 1-\varepsilon)$-bipartite expanders with $k_{\max} = \Omega(\frac{n}{d})$, and $m = \Omega(\frac{n}{d} \cdot e^d)$.

Proof. Let $G = (L, R, E)$ be a bipartite graph with $|L| = m, |R| = n$, where every vertex in L is connected to d vertices in R, sampled uniformly and independently at random. We show that, with positive probability, G is a $(k_{\max}, 1-\varepsilon)$-bipartite expander. This will conclude the existential claim.

Fix $1 \leq k \leq k_{\max}$ and a subset $S \subset L$ of size k. By union bound, the probability that $|\Gamma(S)| < (1-\varepsilon)dk$ is bounded above by

$$\binom{n}{(1-\varepsilon)dk} \cdot \left(\frac{(1-\varepsilon)dk}{n}\right)^{dk}.$$

Thus, the probability that $|\Gamma(S)| < (1-\varepsilon)dk$ for *some* subset $S \subset L$ of size k is bounded above by

$$\binom{m}{k} \cdot \binom{n}{(1-\varepsilon)dk} \cdot \left(\frac{(1-\varepsilon)dk}{n}\right)^{dk}. \tag{7.6}$$

By demanding that for every $1 \leq k \leq k_{\max}$, Equation (7.6) is bounded above by 4^{-k}, we obtain a bound of $\sum_{k=1}^{k_{\max}} 4^{-k} < \frac{1}{3}$ on the probability that G is not a $(k_{\max}, 1-\varepsilon)$-bipartite expander. Since for every $a, b \in \mathbb{N}$ it holds that $\binom{a}{b} \leq \left(\frac{ea}{b}\right)^b$, it is enough to require that

$$m \leq \frac{k}{4e} \cdot e^{-d} \cdot \left(\frac{n}{dk}\right)^{\varepsilon d}$$

for all $1 \leq k \leq k_{\max}$. Since $\varepsilon > 1/d$, the right hand side in the equation above decreases as k increases, and thus it is enough to require

$$m \leq \frac{k_{\max}}{4e} \cdot e^{-d} \cdot \left(\frac{n}{dk_{\max}}\right)^{\varepsilon d}.$$

The proof then follows by the assumption $k_{\max} < e^{-2/\varepsilon} \cdot \frac{n}{d}$. □

As mentioned, the state of the art explicit construction for unbalanced expanders is due to [GUV09], given by the following theorem.

Theorem 7.8 ([GUV09])**.** *Let $\alpha > 0$ and $\varepsilon < 1$ be any constants. Let n, d be large enough integers. For any $k_{max} \leq \left(\frac{n}{d^2}\right)^{1/(1+\alpha)}$ there exists an explicit construction of a $(k_{max}, 1-\varepsilon)$-bipartite expander with $m = \exp\left(\frac{d^{\alpha/(1+\alpha)}}{\log k_{max}}\right)$.*

Unfortunately, the explicit construction in Section 7.8 falls short from achieving the parameters of the probabilistic construction presented in Fact 7.12.

We are now ready to prove Theorem 7.4.

Proof of Theorem 7.4. By Fact 7.11, we have that G is a $(k_{\max}, 1/3)$-unique neighbor expander. Let $U \subseteq \mathbb{F}_2^n$ be a subspace of dimension k. Assume for contradiction that for every $c \in C$ there exists $u_c \in U$ such that $|c + u_c| \leq d$. In case there is more than one element in U that is of distance at most d from c, we choose one such element arbitrarily. Define $U' = \{u_c : c \in C\}$.

Claim 7.12.1. $|U'| = |C| = m$

Proof. Let c, c' be two distinct elements in C. To prove the claim it is enough to show that $u_c \neq u_{c'}$. Assume for contradiction that $u_c = u_{c'}$. Then, by the triangle inequality,

$$|c + c'| \leq |c + u_c| + |c' + u_{c'}| + |u_c + u_{c'}| \leq 2d. \tag{7.7}$$

On the other hand, G is a $(k_{\max}, 1/3)$-unique neighbor expander. Hence,

$$|c + c'| \geq \frac{1}{3} \cdot 4d \cdot 2 > 2d,$$

contradicting Claim 7.7. □

Define

$$U'' = \left\{ \sum_{i=1}^{t} u_i \;\middle|\; t \in [k_{\max}/2] \text{ and } u_1, \ldots, u_t \in U' \right\}.$$

Claim 7.12.2.

$$|U''| = \sum_{i=0}^{k_{max}/2} \binom{m}{i}.$$

Before proving Claim 7.12.2 we note that it completes the proof of Theorem 7.4. Indeed, on one hand $U'' \subseteq U$, and so $|U''| \leq |U|$. On the other hand, by Claim 7.12.2 and by the assumption of Theorem 7.4, $|U''| > |U|$.

Proof of Claim 7.12.2. We first note that it is enough to prove that for every $\emptyset \neq S \subseteq U''$ with size at most $k_{\max}$, it holds that

$$\sum_{u \in S} u \neq 0. \tag{7.8}$$

Indeed, assume that there exist two distinct subsets $R, T \subseteq U''$ such that $R = \{u_1, \ldots, u_r\}$, $T = \{v_1, \ldots, v_t\}$, and $r, t \leq k_{\max}/2$. If

$$\sum_{i=1}^{r} u_i = \sum_{j=1}^{t} v_j,$$

then the symmetric difference of R, T is a non-empty set of size at most $k_{\max}$ such that the sum of its elements is 0, contradicting Equation (7.8). As in Claim 7.12.1, assume by contradiction that there exists a set S as above for which Equation (7.8) does not hold. Then, by the triangle inequality,

$$\left|\sum_{u \in S} c_u\right| \leq \sum_{u \in S} |u + c_u| + \left|\sum_{u \in S} u\right| \leq d \cdot |S|. \tag{7.9}$$

On the other hand, since G is $(k_{\max}, 1/3)$ unique-neighbor expander,

$$\left|\sum_{u \in S} c_u\right| \geq \frac{1}{3} \cdot 4d \cdot |S| > d \cdot |S|,$$

contradicting Equation (7.9). □

This completes the proof of Theorem 7.4. □

How small are the rigid sets obtained by Theorem 7.4? By Fact 7.12, there exists a $(k_{\max}, 2/3)$-unbalanced expander with $k_{\max} = \Omega(n/d)$, left-degree $4d$ and $m = \Theta(k_{\max} \cdot e^{4d})$. By Theorem 7.4, such bipartite expander induces an (n, k, d)-rigid set with size m, given that $\binom{m}{k_{\max}/2} > 2^k$. For this inequality to hold, it is enough to have $\left(\frac{m}{k_{\max}/2}\right)^{k_{\max}/2} = \Theta(e^{2d \cdot k_{\max}}) > 2^k$. As $k_{\max} = \Omega(n/d)$, the latter holds for any $k < c \cdot n$, for some absolute constant c. That is, one gets an $(n, \Omega(n), d)$-rigid set with size $n \cdot \mathsf{exp}(d)$. For general k, one can obtain an (n, k, d)-rigid set with size $n \cdot \mathsf{exp}(d \cdot k/n)$ by considering only $k_{\max} \cdot \mathsf{exp}(d \cdot k/n)$ of the left vertices of the expander.

This construction however is not explicit. Plugging the unbalanced expanders of [GUV09] (see Section 7.8) only gives (n, k, d)-rigid sets with size $m = n \cdot \mathsf{exp}\left(k \cdot (d^2/n)^{1/(1+\alpha)}\right)$, for any constant $\alpha > 0$. By considering $\alpha \to 0$, one can see the quadratic lose in the distance parameter, d.

7.6 From General Dimension k to Dimension $n/2$

In this section we discuss the problem of constructing (n, k, d)-rigid sets for an arbitrary k. A natural approach would be to reduce this problem to the problem of constructing $(n, n/2, d')$-rigid sets. However, it is not clear whether or not there exists such a reduction. More formally, it is not clear how can one use a $\text{poly}(n)$-time algorithm that is given n, d as inputs and computes an $(n, n/2, d)$-rigid set in $\mathbb{F}_2^n$ to devise a $\text{poly}(n)$-time algorithm that given n, k, d as inputs, where $k < n/2$, computes an (n, k, d)-rigid set with small size. However, it turns out that for *strong* rigid sets such a reduction exists. This is the statement of the following lemma.

Lemma 7.13. *Assume that there exists an algorithm $\mathcal{A}$ that given inputs n, d, runs in $\text{poly}(n)$-time and computes a strong $(n, n/2, d)$-rigid set with size $m = m(n, d)$. Then, there exists an algorithm $\mathcal{A}'$ that given n, k, d as inputs, such that $k \leq n/2$, runs in $\text{poly}(n)$-time and computes a strong (n, k, d)-rigid set with size $m(2k, d \cdot 2k/n)$.*

Proof. The algorithm $\mathcal{A}'$ works as follows. $\mathcal{A}'$ makes a call to $\mathcal{A}$ on input $2k, d \cdot 2k/n$ to compute a strong $(2k, k, d \cdot 2k/n)$-rigid set S. The output of $\mathcal{A}'$ is the set

$$S' = \left\{ \underbrace{s \circ s \cdots \circ s}_{n/2k \text{ copies}} : s \in S \right\},$$

where $\circ$ denotes string concatenation. Note that $|S'| = |S| = m(2k, d{\cdot}2k/n)$ as stated. We now show that S' is a strong (n, k, d)-rigid set. Let $U \subseteq \mathbb{F}_2^n$ be a subspace of dimension k. Partition the set of indices $[n]$ into $n/2k$ consecutive blocks of size $2k$ each. For $i \in [n/2k]$ denote by $U|_i$ the projection of U on the i^{th} block. Note that for every $i \in [n/2k]$, $U|_i \subseteq \mathbb{F}_2^{2k}$ is a subspace of dimension at most k. For $s \in S$ let $u_s \in U$ be a closest vector in U to $s \circ \cdots \circ s$, namely,

$$\mathsf{dist}_{\mathsf{H}}(s \circ \cdots \circ s, U) = |s \circ \cdots \circ s + u_s|.$$

For $i \in [n/2k]$, let $u_s|_i$ be the projection of u_s to the i^{th} block. Then,

$$\mathsf{dist}_{\mathsf{H}}(s \circ \cdots \circ s, U) = \sum_{i=1}^{n/2k} \left| u_s|_i + s \right| \geq \sum_{i=1}^{n/2k} \mathsf{dist}_{\mathsf{H}}(s, U|_i).$$

Thus, by linearity of expectation

$$\begin{aligned}
\mathbb{E}_{s' \sim S'} \left[\mathsf{dist}_{\mathsf{H}}(s', U) \right] &= \mathbb{E}_{s \sim S} \left[\mathsf{dist}_{\mathsf{H}}(s \circ \cdots \circ s, U) \right] \\
&\geq \mathbb{E}_{s \sim S} \left[\sum_{i=1}^{n/2k} \mathsf{dist}_{\mathsf{H}}(s, U|_i) \right] \\
&= \sum_{i=1}^{n/2k} \mathbb{E}_{s \sim S} \left[\mathsf{dist}_{\mathsf{H}}(s, U|_i) \right] \\
&\geq \frac{n}{2k} \cdot \frac{2kd}{n} = d.
\end{aligned}$$

□

Theorem 7.3 together with Lemma 7.13 yield the following corollary.

Corollary 7.9. *Let n, k, d be such that $k \leq n/2$ and $d \leq c \cdot n$ for some suitable constant $0 < c < 1$. Then there exists an explicit construction of an (n, k, d)-strong rigid set with size $n \cdot \exp(d \cdot k/n)$.*

In fact, one can generalize each of the proofs we gave for Theorem 7.3 to show that an $\exp(-d \cdot k/n)$-biased set is an (n, k, d)-strong rigid set. Nevertheless, the reduction in Lemma 7.13 might be of use in the construction of (n, k, d)-strong rigid sets from arbitrary $(n, n/2, d)$-rigid sets.

Chapter 8

Gradual small-bias sample spaces

8.1 Introduction

An *ε-biased sample space* S over $\{0,1\}^n$ is a sample space with the following property: for every $\emptyset \neq T \subseteq [n]$, the random variable $s_T \triangleq \oplus_{i \in T} s_i$, where s is sampled from S, has bias at most ε (see also Section 2.4). In other words, a sample space is ε-biased if it ε-fools every nontrivial linear test. When it is not desired or not important to specify ε, one usually refers to such a sample space as a *small-bias sample space*.

The notion of a small-bias sample space was introduced in the seminal paper of Naor and Naor [NN93] and has become a fundamental notion in theoretical computer science, with a variety of applications.

Several explicit constructions of small-bias sample spaces that attempt to minimize the sample space size in terms of n and ε are known [AGHP92, ABN+92, NN93, BT09]. These constructions give incomparable sizes. Unfortunately, all known constructions fall short from achieving sample spaces of size $O(n/\varepsilon^2)$, which are guaranteed to exist by a simple probabilistic argument. Another research direction, which this work falls into, studies variations and generalizations of small-bias sample spaces [AIK+90, RSW93, EGL+92, AM95, MST06, Shp06].

A relaxation of the notion of a small-bias sample space requires only that *small* linear tests will be fooled. Formally, a *(k,ε)-biased sample space* is a sample space S over $\{0,1\}^n$ such that for every $\emptyset \neq T \subseteq [n]$ of size at most k, the random variable s_T has bias at most ε, where again s is sampled from S. The advantage of this relaxed notion is that fooling only small tests, rather than every nontrivial test, can be achieved by much smaller sample spaces. The original motivation for studying (k,ε)-biased sample spaces was to obtain almost k-wise independent random variables. However, (k,ε)-biased sample spaces had proved to be useful in their own right, and found several applications, especially for constructing randomness extractors [SZ94, Raz05, GRS06, CRS12] (see Chapter 3).

Naor and Naor [NN93] explicitly constructed (k,ε)-biased sample spaces with seed that is exponentially smaller in terms of n than what is possible for ε-biased sample spaces. They showed that $O(\log k + \log\log n + \log \varepsilon^{-1})$ random bits are sufficient in order to fool tests of size k, while it is known that a seed of length $\Omega(\log n + \log \varepsilon^{-1})$ is necessary

in order to fool every nontrivial linear test (see, e.g, [AGHP92, Alo09]).

Gradual small-bias sample spaces Consider two pairs (k_1, ε_1) and (k_2, ε_2) such that

$$s = \log k_1 + \log \varepsilon_1^{-1} = \log k_2 + \log \varepsilon_2^{-1}.$$

Potentially, one could hope that a seed of length $O(s + \log \log n)$ would be sufficient to ε_1-fool tests of size k_1 and *simultaneously* to ε_2-fool tests of size k_2. In other words, we are considering a (k, ε)-biased sample space that has the following property: for tests of size $t < k$, the "spare" $\log k - \log t$ bits of the seed are utilized to reduce the bias. In this paper we initiate the study of such sample spaces, which have a better bound on the bias for smaller tests.

Definition 8.1. *A sample space S over $\{0,1\}^n$ is called* gradual (k, ε)-biased *if for every $\emptyset \neq T \subseteq [n]$ of size at most k,*

$$\left| \mathbb{E}_{s \sim S} \left[(-1)^{\sum_{i \in T} s_i} \right] \right| \leq \varepsilon \cdot \frac{|T|}{k}.$$

A few words about the definition are in order. First, note that when T is of size exactly k, the bound on the bias is ε, i.e., a gradual (k, ε)-biased sample space is, in particular, (k, ε)-biased. On the other hand, a $(k, \varepsilon/k)$-biased sample space is a gradual (k, ε)-biased sample space.

One may consider a more general definition, which allows an arbitrary decaying function $\phi : \mathbb{N} \to [0, 1]$ as the bound on the bias (say, $\phi(|T|) = \varepsilon \cdot (|T|/k)^d$ for some parameter d). We choose this function to be $\varepsilon \cdot |T|/k$ in the definition and discuss a more general definition in Section 8.4.

8.1.1 Main Result

Theorem 8.1. *For any integers n and $k \leq n$, for any $\varepsilon > 0$, and for any constant $\delta > 0$* [1] *there exists an explicit construction of a gradual (k, ε) sample space of size*

$$m = O_\delta \left(\left(\frac{k}{\varepsilon} \right)^{2+\delta} + \left(\frac{\log n}{\log k} \right)^{2+4/\delta} k^{1+\delta} \right),$$

where the O_δ hides a multiplicative constant that depends only on δ.

Obviously, one can find a value for δ that minimizes m as a function of n, k and ε. However, when no assumptions are made on the relations between n, k and ε, the expression one would get is cumbersome and non-informative. Moreover, when conducting such minimization one can no longer ignore the multiplicative dependency in δ that is hidden under the big O_δ notation. We therefore choose to specify our bound in the more readable way presented above.

[1] In fact, the construction works without assuming δ is constant, and this assumption appears only to simplify the presentation of the theorem. See Theorem 8.4 for a more general statement.

8.2 Previous Results Used By the Construction

8.2.1 Quadratic Characters

We denote by χ_q the quadratic character over $\mathbb{F}_q$. Namely, $\chi_q(0) = 0$ and for all $0 \neq x \in \mathbb{F}_q$, $\chi_q(x) = 1$ if $\exists y \in \mathbb{F}_q \setminus \{0\}$ such that $x = y^2$, and $\chi_q(x) = -1$ otherwise. When the field is understood from the context, we omit the subscript and simply denote this character by χ. We use a special case of Weil's Theorem regarding character sums (see e.g., [Sch76]).

Theorem 8.2 (Weil's Theorem)**.** *Let q be an odd prime power. Let $f \in \mathbb{F}_q[x]$ be a degree d polynomial. Assume that $f(x) \neq c \cdot g(x)^2$ for any $c \in \mathbb{F}_q, g \in \mathbb{F}_q[x]$. Then, $\left|\sum_{x \in \mathbb{F}_q} \chi(f(x))\right| \leq (d-1)\sqrt{q}$.*

8.2.2 Expanders and Codes

We associate a bipartite graph $G = (L, R, E)$ with $|L|$ left-vertices, $|R|$ right-vertices and left-degree d with the adjacency function $G : L \times [d] \to R$, where $G(x, i) = y$ if and only if y is the ith neighbor of x. For a set of left-vertices $A \subseteq L$ we denote by $G(A)$ the set of neighbors of A.

Definition 8.2. *A bipartite graph $G : L \times [d] \to R$ is a k-unique-neighbor expander if for any nonempty subset $A \subseteq L$ of size at most k, there exists some $y \in R$ that is adjacent to exactly one vertex in A.*

Definition 8.3. *A bipartite graph $G : L \times [d] \to R$ is a $(\leq k, \alpha)$ expander if for any subset $A \subseteq L$ of size at most k, $|G(A)| \geq \alpha \cdot |A|$.*

We will need the well-known fact that a graph whose expansion is greater than half of the degree is also a unique-neighbor expander.

Fact 8.4. *If $G : L \times [d] \to R$ is a $(\leq k, \alpha)$ expander for $\alpha > d/2$ then G is a k-unique-neighbor expander.*

We will make use of the following expanders, constructed by [GUV09]. We remark that these expanders are related to the extractors mentioned in Section 2.

Theorem 8.3 ([GUV09, Theorem 3.2])**.** *Let q be a prime power.*[2] *For every integers $\ell, r, h \geq 1$ there exists an explicit construction of a graph $G : [q^\ell] \times [q] \to [q^{r+1}]$ which is an $(\leq h^r, q - (\ell-1)(h-1)r)$ expander. In particular, G is an h^r-unique-neighbor expander when $q > (\ell-1)(h-1)r/2$.*

[2]For this construction to be explicit, the characteristic of $\mathbb{F}_q$ should be small. In our construction we take it to be 3.

8.3 The Construction

In this section we describe our construction of a gradual (k, ε)-biased sample space, and prove Theorem 8.1. Let $r \geq 2$ be an integer. Let q be an odd prime power to be determined later. Set $\ell = \lceil \frac{\log n}{\log q} \rceil$. For the construction, we assume that we have a bipartite graph $G = (L, R, E)$ which is a k-unique-neighbor expander with $|L| = q^\ell$, $|R| = q^{r+1}$, and left-degree q. By our choice of ℓ we have $|L| \geq n$. Fix an arbitrary subset L' of L such that $|L'| = n$. Set $m = q^{r+1}$ and identify R with the finite field $\mathbb{F}_m$. For every vertex $v \in L'$ define the polynomial $p_v(x) \in \mathbb{F}_m[x]$ by $p_v(x) = \prod_{w \,:\, (v,w)\in E} (x - w)$.

We now describe the sample space S over $\{0,1\}^n$. [3] Each element in S corresponds to a field element in $\mathbb{F}_m$, that is, $S = \{s_x : x \in \mathbb{F}_m\}$. The string s_x is indexed by elements from the set L'. In particular, for every $x \in \mathbb{F}_m$ and $v \in L'$, we define

$$(s_x)_v = \begin{cases} \dfrac{1 - \chi_m\left(p_v(x)\right)}{2}, & p_v(x) \neq 0; \\ 0, & \text{otherwise.} \end{cases} \tag{8.1}$$

The following theorem readily implies Theorem 8.1 by setting $\delta = 4/(r-1)$.

Theorem 8.4. *For every integers n, k, r such that $n \geq k$ and $r \geq 2$, and for any $\varepsilon > 0$, one can (efficiently) find a value for q such that the construction defined above is an explicit gradual (k, ε) sample space over $\{0,1\}^n$ with size*

$$m \leq \max\left\{ (10r^2)^{r+1} \left(\frac{\log n}{\log k}\right)^{r+1} k^{1+1/r},\ 3^{r+1} \left(\frac{2k}{\varepsilon}\right)^{2+4/(r-1)} \right\}.$$

To prove Theorem 8.4 we prove the following two claims.

Claim 8.4.1. *If $q \geq (2k/\varepsilon)^{2/(r-1)}$ then the sample space defined above is gradual (k, ε)-biased.*

Claim 8.4.2. *If $q \geq 3.3 \cdot \frac{\log n}{\log k} \cdot k^{1/r} r^2$, then we have an explicit construction of the k-unique-neighbor expander graph $G = (L, R, E)$ required by the above construction.*

Before proving the two claims we derive Theorem 8.4 from them. By choosing

$$q \geq \max\left\{ 3.3 \cdot \frac{\log n}{\log k} \cdot k^{1/r} r^2, \left(\frac{2k}{\varepsilon}\right)^{2/(r-1)} \right\}, \tag{8.2}$$

Claim 8.4.2 assures us that we can obtain the graph G that we need in the construction. Having this graph, Claim 8.4.1 guarantees that the above sample space is gradual (k, ε)-biased. Certainly one can efficiently find a choice for $q = 3^z$ which is at most three times

[3] In fact, we define S as a multi-set. The sample space is induced in the natural way, namely, to sample from the sample space, one sample an element $s \in S$ with probability proportional to the multiplicity of s in S.

the right hand side of Equation (8.2). [4] As $m = q^{r+1}$ we get the following upper bound on m, the sample space size

$$m \leq \max\left\{(10r^2)^{r+1}\left(\frac{\log n}{\log k}\right)^{r+1} k^{1+1/r}, 3^{r+1}\left(\frac{2k}{\varepsilon}\right)^{2+4/(r-1)}\right\},$$

hence Theorem 8.4 follows.

Proof of Claim 8.4.1. Let $\emptyset \neq T \subseteq L'$, $|T| \leq k$. Define $p_T(x) = \prod_{v\in T} p_v(x)$. Since $p_T(x)$ is defined as a product of $|T|$ polynomials, each of degree at most q, we have that $\deg(p_T(x)) \leq q \cdot |T|$. Moreover, we claim that $p_T(x)$ has a simple root. Indeed, T is a nonempty set of size at most k of $L' \subseteq L$. By our assumption, G is a k-unique-neighbor expander, and so there exists a vertex $w \in R$ with exactly one neighbor, v, in T. This implies that w is a simple root of $p_v(x)$, while for every $u \in T\backslash\{v\}$, $p_u(w) \neq 0$. Hence, by the definition of $p_T(x)$ we have that w is a simple root of $p_T(x)$. Now, up to normalization, the bias of the linear test defined by T is

$$\sum_{x\in\mathbb{F}_m} (-1)^{\sum_{v\in T}(s_x)_v} = \sum_{x\in\mathbb{F}_m}\prod_{v\in T}(-1)^{(s_x)_v}. \tag{8.3}$$

Suppose x is not a root of $p_T(x)$. Then, the value that such an x contributes to the sum in Equation (8.3) is

$$\prod_{v\in T}(-1)^{(s_x)_v} = \prod_{v\in T}\chi_m(p_v(x)) = \chi_m\left(\prod_{v\in T} p_v(x)\right) = \chi_m(p_T(x)),$$

where the middle equality follows from the fact that χ is a multiplicative homomorphism. As $p_T(x)$ has at most $\deg(p_T) \leq q \cdot |T|$ roots, we have that $\left|\sum_{x\in\mathbb{F}_m}(-1)^{\sum_{v\in T}(s_x)_v}\right| \leq \left|\sum_{x\in\mathbb{F}_m}\chi_m(p_T(x))\right| + q\cdot|T|$. Since $p_T(x)$ has a simple root, $p_T(x)$ is not of the form $c\cdot g(x)^2$ for any $c \in \mathbb{F}_m$ and $g \in \mathbb{F}_m[x]$. Therefore, we can apply Weil's Theorem (Theorem 8.2) to get $\left|\sum_{x\in\mathbb{F}_m}\chi_m(p_T(x))\right| < q\cdot|T|\cdot\sqrt{m}$. Hence,

$$\frac{1}{m}\left|\sum_{x\in\mathbb{F}_m}(-1)^{\sum_{v\in T}(s_x)_v}\right| \leq \frac{2q\cdot|T|}{\sqrt{m}} = 2|T|\cdot q^{(1-r)/2}.$$

To get a bound of at most ε on the bias for tests of size exactly k, we require that $2k\cdot q^{(1-r)/2} \leq \varepsilon$, or $q \geq \left(\frac{2k}{\varepsilon}\right)^{2/(r-1)}$. □

Proof of Claim 8.4.2. We use the expanders from Theorem 8.3 with $h = \lceil k^{1/r}\rceil$. If $q - (\ell-1)(h-1)r \geq 0.51q$ then G is a k-unique-neighbor expander. By the definition of ℓ, for the above equation to hold, it is enough to require $q\log q \geq 2.05\cdot\log n\cdot k^{1/r}r$, which holds for any $q \geq 3.28\cdot\frac{\log n}{\log k}\cdot k^{1/r}r^2$. □

[4] Observe also that this solves the minor issue regarding the need for small characteristic for the explicitness requirements of Theorem 8.3.

8.4 Non-Linear Bias Decay

The definition of a gradual (k, ε)-biased sample space that appears in the introduction requires a bound of the form $\varepsilon \cdot |T|/k$ on the bias for any nonempty set T of size at most k. The construction we suggest in this paper indeed has such linear decay. However, one may consider a more general definition where the decay exponent is a non-negative real parameter d.

Definition 8.5. *A sample space S over $\{0,1\}^n$ is called* gradual (k, d, ε)-biased *if for every $\emptyset \neq T \subseteq [n]$ of size at most k,*

$$\left|\mathbb{E}_{s\sim S}\left[(-1)^{\sum_{i\in T} s_i}\right]\right| \leq \varepsilon \cdot \left(\frac{|T|}{k}\right)^d.$$

We call d the decay exponent.

A straightforward probabilistic argument shows that a random sample space S over $\{0,1\}^n$ of size $m = O(k^{2d} \cdot \varepsilon^{-2} \cdot \log n)$ is, with high probability, a (k, d, ε)-biased sample space. We start this section by proving an almost matching lower bound on the size of (k, d, ε)-biased sample spaces (Theorem 8.6 below). We then turn to present two simple methods that transform a gradual (k, d, ε)-biased sample space to a gradual (k, d', ε)-biased sample space for $d' > d$. These methods, together with the construction for the case $d = 1$ (Theorem 8.1) yields constructions with larger decay exponents (Corollary 8.7).

8.4.1 A Lower Bound

To prove a lower bound on the size of a gradual small-bias sample space, we use the following known lower bound on the size of (non-gradual) (k, ε)-biased sample spaces.

Theorem 8.5 ([AAK+07, Alo09])**.** *Let S be a (k, ε)-biased sample space over $\{0,1\}^n$ of size m. If $\varepsilon \geq \binom{n}{k/2}^{-1/2}$ then $m \geq \Omega\left(\frac{k \log(n/k)}{\varepsilon^2 \cdot \log 1/\varepsilon}\right)$.*

We now state and prove a lower bound for the size of gradual (k, d, ε) sample spaces.

Theorem 8.6. *Let S be a gradual (k, d, ε)-biased sample space over $\{0,1\}^n$ of size m. If $d \leq k/\log k$ and $\varepsilon > (d \log k/n)^{O(d \log k)}$ then*

$$m \geq \Omega\left(\frac{\log n}{\varepsilon^2 \cdot \log 1/\varepsilon} \cdot \left(\frac{k}{d \cdot \log k}\right)^{2d}\right).$$

Proof. Let S be a gradual (k, d, ε)-biased sample space over $\{0,1\}^n$ of size m. Then, in particular, S is a (k', ε')-biased sample space with $k' = d \log k$ and $\varepsilon' = \varepsilon \cdot \left(\frac{d \log k}{k}\right)^d$. As $\varepsilon' \geq \binom{n}{k'/2}^{-1/2}$ we can use Theorem 8.5 to deduce that

$$m \geq \Omega\left(\frac{k' \log(n/k')}{(\varepsilon')^2 \cdot \log 1/\varepsilon'}\right) \geq \Omega\left(\left(\frac{k}{d \cdot \log k}\right)^{2d} \cdot \frac{1}{\varepsilon^2 \cdot \log(1/\varepsilon)} \cdot \frac{\log k \cdot \log\left(\frac{n}{d \log k}\right)}{\log\left(\frac{k}{d \log k}\right)}\right).$$

Since we assume that $d \leq k/\log k$, and since $k \leq n$, we have that

$$\frac{\log k \cdot \log\left(\frac{n}{d\log k}\right)}{\log\left(\frac{k}{d\log k}\right)} \geq \log n,$$

thus we have the desired lower bound on m. □

8.4.2 Amplifying the Decay Exponent

To amplify the decay exponent, we note that every gradual $(k, d, \varepsilon/k)$-biased sample space S on n variables is a gradual $(k, d+1, \varepsilon)$-biased sample space on n variables. Indeed, for every nonempty $T \subseteq [n]$ of size at most k,

$$\left|\mathbb{E}_{s\sim S}\left[(-1)^{\sum_{i\in T} s_i}\right]\right| \leq \frac{\varepsilon}{k} \cdot \left(\frac{|T|}{k}\right)^d \leq \varepsilon \cdot \left(\frac{|T|}{k}\right)^{d+1}.$$

That is, choosing a smaller error to begin with, will result in a larger decay exponent. This observation, together with Theorem 8.1 immediately implies the following corollary.

Corollary 8.7. *For any integers n, k, d, such that $k \leq n$, for any $\varepsilon > 0$, and for any constant $\delta > 0$, there exists an explicit construction of a gradual (k, d, ε) sample space of size*

$$m = O_\delta\left(\left(\frac{k^d}{\varepsilon}\right)^{2+\delta} + \left(\frac{\log n}{\log k}\right)^{2+4/\delta} k^{1+\delta}\right).$$

Another method for amplifying the decay exponent, which is more suitable than the above in cases where the original gradual sample space has bad dependency in ε, is based on the following observation. Let S be a gradual $(k, d, \sqrt{\varepsilon})$-biased sample space. Then $S + S$ [5] is a gradual $(k, 2d, \varepsilon)$-biased sample space. This follows because

$$\left|\mathbb{E}_{s\sim S+S}\left[(-1)^{\sum_{i\in T} s_i}\right]\right| = \left(\mathbb{E}_{s\sim S}\left[(-1)^{\sum_{i\in T} s_i}\right]\right)^2 \leq \left(\sqrt{\varepsilon} \cdot \left(\frac{|T|}{k}\right)^d\right)^2 = \varepsilon \cdot \left(\frac{|T|}{k}\right)^{2d}.$$

[5] The sample space $S+S$ is defined by sampling s_1 and s_2, independently, from S and then outputting $s_1 + s_2$, where the addition is a bitwise addition over $\mathbb{F}_2$.

Bibliography

[AAK+07] N. Alon, A. Andoni, T. Kaufman, K. Matulef, R. Rubinfeld, and N. Xie. Testing k-wise and almost k-wise independence. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 496–505. ACM, 2007.

[Aar10] S. Aaronson. BQP and the Polynomial Hierarchy. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 141–150. ACM, 2010.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ABN+92] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38:509–516, 1992.

[AC88] N. Alon and F.R.K. Chung. Explicit construction of linear sized tolerant networks. *Discrete Mathematics*, 72(1):15–19, 1988.

[AC13] N. Alon and G. Cohen. On rigid matrices and U-polynomials. In *Conference on Computational Complexity (CCC), 2013 IEEE*, pages 197–206. IEEE, 2013.

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[AIK+90] M. Ajtai, H. Iwaniec, J. Komlos, J. Pintz, and E. Szemeredi. Construction of a thin set with small fourier coefficients. *Bulletin of the London Mathematical Society*, 22:583–590, 1990.

[AKS83] Miklós Ajtai, János Komlós, and Endre Szemerédi. An o(n log n) sorting network. In *STOC*, pages 1–9, 1983.

[Alo09] N. Alon. Perturbed identity matrices have high rank: proof and applications. *Combinatorics, Probability and Computing*, 18(1-2):3–15, 2009.

[AM95] N. Alon and Y. Mansour. epsilon-discrepancy sets and their application for interpolation of sparse polynomials. *Information Processing Letters*, 54(6):337–342, 1995.

[APY09] N. Alon, R. Panigrahy, and S. Yekhanin. Deterministic approximation algorithms for the nearest codeword problem. In *APPROX-RANDOM*, pages 339–351, 2009.

[AR63] S. Akers and T. Robbins. Logical design with three-input majority gates. *Computer Design*, 45(3):12–27, 1963.

[AR94] N. Alon and Y. Roichman. Random cayley graphs and expanders. *Random Structures and Algorithms*, 5(2):271–285, 1994.

[AS10] V. Arvind and S. Srinivasan. The remote point problem, small bias spaces, and expanding generator sets. In *27th STACS*, pages 59–70, 2010.

[BAC12] Avraham Ben-Aroya and Gil Cohen. Gradual small-bias sample spaces. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 19, page 50, 2012.

[BBCM95] C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[BBR85] C. H. Bennett, G. Brassard, and J. M. Robert. How to reduce your enemys information. In *Advances in Cryptology (CRYPTO)*, volume 218, pages 468–476. Springer, 1985.

[BBR88] C. H. Bennett, G. Brassard, and J. M. Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, 1988.

[BCS14] I. Benjamini, G. Cohen, and I. Shinkar. Bi-lipschitz bijection between the Boolean cube and the Hamming ball. In *IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 81–89. IEEE, 2014.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BIW10] O. Barkol, Y. Ishai, and E. Weinreb. On locally decodable codes, self-correctable codes, and t-private PIR. *Algorithmica*, 58(4):831–859, 2010.

[BKS+05] B. Barak, G. Kindler, R. Shaltiel, B. Sudakov, and A. Wigderson. Simulating independence: New constructions of condensers, Ramsey graphs, dispersers, and extractors. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 1–10. ACM, 2005.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[Bop97] R. Boppana. The average sensitivity of bounded-depth circuits. *Information Processing Letters*, 63(5):257–261, 1997.

[Bou05] J. Bourgain. More on the sum-product phenomenon in prime fields and its applications. *International Journal of Number Theory*, 1(01):1–32, 2005.

[Bra87] Gabriel Bracha. An O(log n) expected rounds randomized byzantine generals protocol. *J. ACM*, 34(4):910–920, 1987.

[BRSW12] 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. *Annals of Mathematics*, 176(3):1483–1544, 2012.

[BSK12] E. Ben-Sasson and S. Kopparty. Affine dispersers from subspace polynomials. *SIAM Journal on Computing*, 41(4):880–914, 2012.

[BT09] A. Ben-Aroya and A. Ta-Shma. Constructing small-bias sets from algebraic-geometric codes. In *Proceedings of the 50th annual IEEE symposium on foundations of computer science (FOCS)*, 2009.

[BV10] J. Brody and E. Verbin. The coin problem and pseudorandomness for branching programs. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 30–39. IEEE, 2010.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

[CDI+13] G. Cohen, I. B. Damgård, Y. Ishai, J. Kölker, P. B. Miltersen, R. Raz, and R. D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae. In *Advances in Cryptology – CRYPTO 2013*, pages 185–202. Springer, 2013.

[CDN12] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing - An Information Theoretic Appoach*. 2012. Book draft, available at `http://www.daimi.au.dk/~ivan/MPCbook.pdf`.

[CFF+05] Jeffrey Considine, Matthias Fitzi, Matthew K. Franklin, Leonid A. Levin, Ueli M. Maurer, and David Metcalf. Byzantine agreement given partial broadcast. *J. Cryptology*, 18(3):191–217, 2005.

[CFIK03] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT*, pages 596–613, 2003.

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

[CGH+85] B. Chor, O. Goldreich, J. Håstad, J. Freidmann, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 396–407. IEEE, 1985.

[CGR14] G. Cohen, A. Ganor, and R. Raz. Two sides of the coin problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 618–629, 2014.

[Cha89] David Chaum. The spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. In *CRYPTO*, pages 591–602, 1989.

[CKK+13] R. Chen, V. Kabanets, A. Kolokolova, R. Shaltiel, and D. Zuckerman. Mining circuit lower bound proofs for meta-algorithms. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 20, page 57, 2013.

[CKOR10] N. Chandran, B. Kanukurthi, R. Ostrovsky, and L. Reyzin. Privacy amplification with asymptotically optimal entropy loss. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pages 785–794, 2010.

[Coh15] G. Cohen. Local correlation breakers and applications to three-source extractors and mergers. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 38, 2015.

[CRS12] G. Cohen, R. Raz, and G. Segev. Non-malleable extractors with short seeds and applications to privacy amplification. In *Computational Complexity (CCC), 2012 IEEE 27th Annual Conference on*, pages 298–308. IEEE, 2012.

[CRS14] G. Cohen, R. Raz, and G. Segev. Nonmalleable extractors with short seeds and applications to privacy amplification. *SIAM Journal on Computing*, 43(2):450–476, 2014.

[CS14] G. Cohen and I. Shinkar. The complexity of DNF of parities. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 21, page 99, 2014.

[CS15] G. Cohen and I. Shinkar. Zero-fixing extractors for sub-logarithmic entropy. In *The 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, 2015.

[CT14] G. Cohen and A. Tal. Two structural results for low degree polynomials and applications. *arXiv preprint arXiv:1404.0654*, 2014.

[DIK+08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In *CRYPTO*, pages 241–261, 2008.

[DKRS06] Y. Dodis, J. Katz, L. Reyzin, and A. Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In *Advance in Cryptology – CRYPTO '06*, pages 232–250, 2006.

[DKSS09] Z. Dvir, S. Kopparty, S. Saraf, and M. Sudan. Extensions to the method of multiplicities, with applications to Kakeya sets and mergers. In *50th Annual IEEE Symposium on Foundations of Computer Science*, pages 181–190. IEEE, 2009.

[DLWZ11a] Y. Dodis, X. Li, T. D. Wooley, and D. Zuckerman. Non-malleable extractors via character sums. In *Proceedings of the 52th Annual Symposium on Foundations of Computer Science (FOCS)*, 2011.

[DLWZ11b] Y. Dodis, X. Li, T. D. Wooley, and D. Zuckerman. Privacy amplification and non-malleable extractors via character sums, 2011. http://arxiv.org/abs/1102.5415.

[DLWZ11c] Y. Dodis, X. Li, T. D. Wooley, and D. Zuckerman. Privacy amplification and non-malleable extractors via character sums. In *Proceedings of the 52th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 668–677. IEEE, 2011.

[Dol82] Danny Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.

[DORS08] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[DP07] S. Dziembowski and K. Pietrzak. Intrusion-resilient secret sharing. In *48th Annual IEEE Symposium on Foundations of Computer Science*, pages 227–237, 2007.

[DPS12a] Yvo Desmedt, Josef Pieprzyk, and Ron Steinfeld. Active security in multiparty computation over black-box groups. In *SCN*, pages 503–521, 2012.

[DPS+12b] Yvo Desmedt, Josef Pieprzyk, Ron Steinfeld, Xiaoming Sun, Christophe Tartary, Huaxiong Wang, and Andrew Chi-Chih Yao. Graph coloring applied to secure computation in non-abelian groups. *J. Cryptology*, 25(4):557–600, 2012.

[DPSW07] Yvo Desmedt, Josef Pieprzyk, Ron Steinfeld, and Huaxiong Wang. On secure multi-party computation in black-box groups. In *CRYPTO*, pages 591–612, 2007.

[DR08] Z. Dvir and R. Raz. Analyzing linear mergers. *Random Structures & Algorithms*, 32(3):334–345, 2008.

[DS07] Z. Dvir and A. Shpilka. An improved analysis of linear mergers. *computational complexity*, 16(1):34–59, 2007.

[Dvi10] Z. Dvir. On matrix rigidity and locally self-correctable codes. In *Proceedings of the 25th Annual CCC*, pages 291–298, 2010.

[Dvi12] Z. Dvir. Extractors for varieties. *computational complexity*, 21(4):515–572, 2012.

[DW09] 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*, pages 601–610, 2009.

[DW11] Z. Dvir and A. Wigderson. Kakeya sets, new mergers, and old extractors. *SIAM Journal on Computing*, 40(3):778–792, 2011.

[DY13] Y. Dodis and Y. Yu. Overcoming weak expectations. In *Theory of Cryptography*, pages 1–22. Springer, 2013.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *CACM: Communications of the ACM*, 28, 1985.

[EGL+92] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Velickovic. Approximations of general independent distributions. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 10–16, 1992.

[EL75] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10:609–627, 1975.

[ER52] P. Erdős and R. Rado. Combinatorial theorems on classifications of subsets of a given set. *Proceedings of the London Mathematical Society*, 3(2):417–439, 1952.

[Erd47] P. Erdös. Some remarks on the theory of graphs. *Bulletin of the American Mathematical Society*, 53(4):292–294, 1947.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *STOC*, pages 554–563, 1994.

[FM98] Matthias Fitzi and Ueli M. Maurer. Efficient byzantine agreement secure against general adversaries. In *DISC*, pages 134–148, 1998.

[FM00] Matthias Fitzi and Ueli M. Maurer. From partial consistency to global broadcast. In *STOC*, pages 494–503, 2000.

[Fri92] J. Friedman. On the bit extraction problem. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 314–319. IEEE, 1992.

[Fri93] J. Friedman. A note on matrix rigidity. *Combinatorica*, 13(2):235–239, 1993.

[GM96] A. Gupta and S. Mahajan. Using amplification to compute majority with small majority gates. *Computational Complexity*, 6(1):46–63, 1996.

[GM98] J. A. Garay and Y. Moses. Fully polynomial byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. Comput.*, 27(1):247–290, 1998.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[Gol08] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[Gol11a] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In Oded Goldreich, editor, *Studies in Complexity and Cryptography*, volume 6650 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2011.

[Gol11b] Oded Goldreich. Valiant's polynomial-size monotone formula for majority. http://www.wisdom.weizmann.ac.il/~oded/PDF/mono-maj.pdf, 2011.

[GRS06] A. Gabizon, R. Raz, and R. Shaltiel. Deterministic extractors for bit-fixing sources by obtaining an independent seed. *SIAM Journal on Computing*, 36(4):1072–1094, 2006.

[GUV09] V. Guruswami, C. Umans, and S. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *Journal of the ACM (JACM)*, 56(4):1–34, 2009.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. OT-combiners via secure computation. In *TCC*, pages 393–411, 2008.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulleting of the American Mathematical Society*, 43:439–561, 2006.

[HM00] M. Hirt and U. M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptology*, 13(1):31–60, 2000.

[HS10] E. Haramaty and A. Shpilka. On the structure of cubic and quartic polynomials. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 331–340. ACM, 2010.

[IKOS09] Yuval Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[IN96] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 356–364. ACM, 1994.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009.

[Juk06] S. Jukna. On graph complexity. *Combinatorics, Probability and Computing*, 15(06):855–876, 2006.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[KL08] T. Kaufman and S. Lovett. Worst case to average case reductions for polynomials. In *Foundations of Computer Science (FOCS), 2008 49th Annual IEEE Symposium on*, pages 166–175. IEEE, 2008.

[KLR10] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. *SIAM J. Comput.*, 39(5):2090–2112, 2010.

[KPS85] Richard M. Karp, N. Pippinger, and Nicholas Sipser. A time-randomness tradeoff. In *AMS Conference on Probabilistic Computational Complexity*, 1985.

[KR98] B.S. Kashin and A.A. Razborov. Improved lower bounds on the rigidity of Hadamard matrices. *Mathematical Notes*, 63(4):471–475, 1998.

[KR09] B. Kanukurthi and L. Reyzin. Key agreement from close secrets over unsecured channels. In *Advance in Cryptology – EUROCRYPT '09*, pages 206–223, 2009.

[KRT13] I. Komargodski, R. Raz, and A. Tal. Improved average-case lower bounds for DeMorgan formula size. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 588–597. IEEE, 2013.

[KZ06] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM Journal on Computing*, 36(5):1231–1247, 2006.

[Li11] X. Li. Improved constructions of three source extractors. In *IEEE 26th Annual Conference on Computational Complexity*, pages 126–136, 2011.

[Li12a] X. Li. Design extractors, non-malleable condensers and privacy amplification. In *Proceedings of the 44th symposium on Theory of Computing*, pages 837–854. ACM, 2012.

[Li12b] X. Li. Non-malleable condensers for arbitrary min-entropy, and almost optimal protocols for privacy amplification. *arXiv preprint arXiv:1211.0651*, 2012.

[Li12c] X. Li. Non-malleable extractors, two-source extractors and privacy amplification. In *Proceedings of the 53rd IEEE Symposium on Foundation of Computer Science*, 2012.

[Li13] X. Li. Extractors for a constant number of independent sources with polylogarithmic min-entropy. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 100–109. IEEE, 2013.

[Li15] X. Li. Three-source extractors for polylogarithmic min-entropy. *Electronic Colloquium on Computational Complexity (ECCC)*, 2015.

[Lok95] S. V. Lokam. Spectral methods for matrix rigidity with applications to size-depth tradeoffs and communication complexity. In *36th Annual FOCS*, pages 6–15, 1995.

[Lok09] S. V. Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1-2):1–155, 2009.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In *CRYPTO*, pages 259–276, 2011.

[LRM10] Christoph Lucas, Dominik Raub, and Ueli M. Maurer. Hybrid-secure mpc: trading information-theoretic robustness for computational privacy. In *PODC*, pages 219–228, 2010.

[LRVW03] 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*, pages 602–611. ACM, 2003.

[LV12] S. Lovett and E. Viola. Bounded-depth circuits cannot sample good codes. *Computational Complexity*, 21(2):245–266, 2012.

[Mau92] U. M. Maurer. Protocols for secret key agreement by public discussion based on common information. In *Advance in Cryptology – CRYPTO '92*, pages 461–470, 1992.

[Mau97] U. M. Maurer. Information-theoretically secure secret-key agreement by NOT authenticated public discussion. In *Advance in Cryptology – EUROCRYPT '97*, pages 209–225, 1997.

[Mau06] Ueli M. Maurer. Secure multi-party computation made simple. *Discrete Applied Mathematics*, 154(2):370–381, 2006.

[Mil92] P. B. Miltersen. Lecutre notes. Available from author, 1992.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes, Part II.* North-Holland, 1977.

[MST06] E. Mossel, A. Shpilka, and L. Trevisan. On epsilon-biased generators in NC^0. *Random Structures and Algorithms*, 29(1):56–81, 2006.

[MV13] Eric Miles and Emanuele Viola. Shielding circuits with groups. *IACR Cryptology ePrint Archive*, 2013:1, 2013. To appear in STOC 2013.

[MW97] U. M. Maurer and S. Wolf. Privacy amplification secure against active adversaries. In *Advance in Cryptology – CRYPTO '97*, pages 307–321, 1997.

[MW03] U. M. Maurer and S. Wolf. Secret-key agreement over unauthenticated public channels III: Privacy amplification. *IEEE Transactions on Information Theory*, 49(4):839–851, 2003.

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

[NN93] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. on Computing*, 22(4):838–856, 1993.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[O'D] R. O'Donnell. Analyis of boolean functions. http://analysisofbooleanfunctions.org/.

[PR04] P. Pudlák and V. Rödl. Pseudorandom sets and explicit constructions of Ramsey graphs. *Quad. Mat*, 13:327–346, 2004.

[PSL80] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[Rab81] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[Rao07] A. Rao. An exposition of Bourgains 2-source extractor. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 14, page 034, 2007.

[Rao09a] A. Rao. Extractors for a constant number of polynomially small min-entropy independent sources. *SIAM Journal on Computing*, 39(1):168–194, 2009.

[Rao09b] A. Rao. Extractors for low-weight affine sources. In *Proceedings of 24th Annual IEEE Conference on Computational Complexity, (CCC '09)*, pages 95–101. IEEE, 2009.

[Raz05] R. Raz. Extractors with weak random seeds. In *Proceedings of the 37th Annual STOC*, pages 11–20, 2005.

[RBO89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In David S. Johnson, editor, *STOC*, pages 73–85. ACM, 1989.

[RSW93] A. A. Razborov, E. Szemerédi, and A. Wigderson. Constructing small sets that are uniform in arithmetic progressions. *Combinatorics, Probability & Computing*, 2:513–518, 1993.

[RV13] Y. Reshef and S. Vadhan. On extractors and exposure-resilient functions for sublogarithmic entropy. *Random Structures & Algorithms*, 42(3):386–401, 2013.

[RW03] R. Renner and S. Wolf. Unconditional authenticity and privacy from an arbitrarily weak secret. In *Advance in Cryptology – CRYPTO '03*, pages 78–95, 2003.

[Sch76] W. M. Schmidt. *Equations over Finite Fields: An elementary approach.* Springer-Verlag, 1976.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha11] R. Shaltiel. Dispersers for affine sources with sub-polynomial entropy. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 247–256. IEEE, 2011.

[Shp06] A. Shpilka. Constructions of low-degree and error-correcting in-biased generators. In *21st Annual IEEE Conference on Computational Complexity*, pages 33–45, 2006.

[Spe77] J. Spencer. Intersection theorems for systems of sets. *Canadian Math Bulletin*, 20(2):249–254, 1977.

[SSS97] M.A. Shokrollahi, D. Spielman, and V. Stemann. A remark on matrix rigidity. *Information Processing Letters*, 64(6):283–285, 1997.

[Ste13] J. Steinberger. The distinguishability of product distributions by read-once branching programs. In *Computational Complexity (CCC), 2013 IEEE Conference on*, pages 248–254. IEEE, 2013.

[SV10] R. Shaltiel and E. Viola. Hardness amplification proofs require majority. *SIAM Journal on Computing*, 39(7):3122–3154, 2010.

[SYT08] Xiaoming Sun, Andrew Chi-Chih Yao, and Christophe Tartary. Graph design for secure multiparty computation over non-abelian groups. In *ASIACRYPT*, pages 37–53, 2008.

[SZ94] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. In *35th Annual Symposium on Foundations of Computer Science*, pages 264–275. IEEE, 1994.

[TS96] A. Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the twenty-eighth annual ACM Symposium on Theory of Computing*, pages 276–285, 1996.

[Uma03] C. Umans. Pseudo-random generators for all hardnesses. *J. of Computer and System Sciences*, 67(2):419–440, 2003.

[Vad11] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 2011.

[Val77] L. G. Valiant. Graph-theoretic arguments in low-level complexity. In *Lecture notes in Computer Science*, volume 53, pages 162–176. Springer, 1977.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Vaz85] V. U. Vazirani. Towards a strong communication complexity theory or generating quasi-random sequences from two communicating slightly-random sources. In *Proceedings of the seventeenth annual ACM symposium on Theory of Computing*, pages 366–378. ACM, 1985.

[Wol98] S. Wolf. Strong security against active attacks in information-theoretic secret-key agreement. In *Advance in Cryptology – ASIACRYPT '98*, pages 405–419, 1998.

[Yao82a] A. C. Yao. Protocols for secure computations. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[Yao82b] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

[Yao82c] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91, 1982.

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

[Zuc07] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.

[Zwi96] Uri Zwick. Lecture notes. http://www.cs.tau.ac.il/~zwick/circ-comp-new/six.ps, 1996.