


Extremely Efficient Constructions of Hash Functions, with Applications to Hardness Magnification and PRFs

Lijie Chen   

CSAIL, MIT, Cambridge, MA, USA

Jiatu Li   

IIS, Tsinghua University, Beijing, China

Tianqi Yang   

IIS, Tsinghua University, Beijing, China

Abstract

In a recent work, Fan, Li, and Yang (STOC 2022) constructed a family of almost-universal hash functions such that each function in the family is computable by $(2n + o(n))$ -gate circuits of fan-in 2 over the B_2 basis. Applying this family, they established the existence of pseudorandom functions computable by circuits of the same complexity, under the standard assumption that OWFs exist. However, a major disadvantage of the hash family construction by Fan, Li, and Yang (STOC 2022) is that it requires a seed length of $\text{poly}(n)$, which limits its potential applications.

We address this issue by giving an improved construction of almost-universal hash functions with seed length $\text{polylog}(n)$, such that each function in the family is computable with POLYLOGTIME -uniform $(2n + o(n))$ -gate circuits. Our new construction has the following applications in both complexity theory and cryptography.

- **(Hardness magnification).** Let $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ be any function such that $\alpha(n) \leq \log n / \log \log n$. We show that if there is an $n^{\alpha(n)}$ -sparse NP language that does not have probabilistic circuits of $2n + O(n / \log \log n)$ gates, then we have (1) $\text{NTIME}[2^n] \not\subseteq \text{SIZE}[2^{n^{1/5}}]$ and (2) $\text{NP} \not\subseteq \text{SIZE}[n^k]$ for every constant k . Complementing this magnification phenomenon, we present an $O(n)$ -sparse language in P which requires probabilistic circuits of size at least $2n - 2$. This is the first result in hardness magnification showing that even a *sub-linear additive* improvement on known circuit size lower bounds would imply $\text{NEXP} \not\subseteq \text{P}_{/\text{poly}}$. Following Chen, Jin, and Williams (STOC 2020), we also establish a sharp threshold for **explicit obstructions**: we give an explicit obstruction against $(2n - 2)$ -size circuits, and prove that a sub-linear additive improvement on the circuit size would imply (1) $\text{DTIME}[2^n] \not\subseteq \text{SIZE}[2^{n^{1/5}}]$ and (2) $\text{P} \not\subseteq \text{SIZE}[n^k]$ for every constant k .
- **(Extremely efficient construction of pseudorandom functions).** Assuming that one of integer factoring, decisional Diffie-Hellman, or ring learning-with-errors is sub-exponentially hard, we show the existence of pseudorandom functions computable by POLYLOGTIME -uniform $\text{AC}^0[2]$ circuits with $2n + o(n)$ wires, with key length $\text{polylog}(n)$. We also show that PRFs computable by POLYLOGTIME -uniform B_2 circuits of $2n + o(n)$ gates follows from the existence of sub-exponentially secure one-way functions.

2012 ACM Subject Classification Theory of computation \rightarrow Circuit complexity; Theory of computation \rightarrow Pseudorandomness and derandomization

Keywords and phrases Almost universal hash functions, hardness magnification, pseudorandom functions

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.23

Funding Lijie Chen is supported by NSF CCF-2127597 and an IBM Fellowship.

Acknowledgements We are grateful for Ryan Williams for insightful discussions during this project and many helpful comments on a draft of this paper. We would also like to thank Ce Jin for discussions during the early stage of this research project and anonymous reviewers for their comments.



© Lijie Chen, Jiatu Li, and Tianqi Yang;
licensed under Creative Commons License CC-BY 4.0
37th Computational Complexity Conference (CCC 2022).

Editor: Shachar Lovett; Article No. 23; pp. 23:1–23:37

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1 **Introduction**

Background and motivation. *Universal hash families*¹, introduced by Carter and Wegman [6], are among the most useful and well-studied objects in computer science, with applications to both real-world systems and the theory of computation (e.g., algorithm design, complexity theory, and cryptography). Its wide applications motivate the problem of constructing universal hash functions with the smallest computational overhead. After decades of research, optimal constructions (up to a constant factor) have been obtained in both the RAM model [36] and the Boolean circuit model [23].

In particular, the investigation of hash functions with small circuit complexity was motivated by the program of constructing low-complexity cryptographic primitives. Ishai, Kushilevitz, Ostrovsky, and Sahai [23] showed that universal hash functions can be constructed with linear-size circuits, which, combined with Levin’s trick of domain extension (see, e.g., [5]), led to linear-size constructions of pseudorandom functions (PRFs) and many other cryptographic primitives, under the standard assumption that one-way functions (OWFs) exist.

In a recent work, Fan, Li, and Yang [13] proved that *almost* universal hash functions² are sufficient for constructing extremely efficient PRFs. They then constructed almost universal hash functions with $2n + o(n)$ circuit complexity in both general and $\text{CC}^0[2]$ circuits³, and proved the optimality of the constant factor 2 (i.e., there is no $2n - O(1)$ -size almost-universal hash construction). As a consequence of their almost-universal hash construction, they presented $2n + o(n)$ -size constructions of PRFs assuming OWFs exist. However, a *major disadvantage* of the hash constructions in [13] is that they require a seed of length $O(n \log n)$ to sample a circuit from the hash family, which greatly limits their applications.

Overview of our results. The main technical contribution of this paper is to improve the hash constructions in [13] for both general and $\text{CC}^0[2]$ circuits by reducing the seed length from $O(n \log n)$ to $\text{polylog}(n)$. Moreover, we observe that the improvement of the seed length makes the hash family explicit in a strong sense: the local topology of the circuit computing the hash function can be obtained with a uniform algorithm (e.g., given a gate index, compute its gate type and which gates’ outputs are fed into this gate) in poly-logarithmic time given a seed of poly-logarithmic length. We call such hash family **POLYLOGTIME-uniform**; see Theorem 1.1 for the details.

Despite being weaker than universal hash functions, our new randomness-efficient low-complexity hash constructions allow us to obtain two important consequences in complexity theory and cryptography.

- **(Hardness magnification)** Following the kernelization method developed by Chen, Jin, and Williams [9, 10], we present extremely sharp bootstrapping results for hardness magnification and explicit obstruction. In particular, we show that a $2.01n$ lower bound

¹ Recall that a universal hash family has the property that for a function drawn from the family, the hash values of any distinct pair of inputs collide with probability 2^{-m} , where m is the bit length of the hash value.

² In an almost universal hash family, the hash collision probability is a negligible function (e.g., $1/2^{\log^2 n}$) instead of $1/2^m$ for m -bit hash values. It is weaker than a universal hash family, but is still useful in many applications.

³ For general circuits we mean B_2 circuits, in which each gate is of fan-in 2 and can compute an arbitrary Boolean function in $B_2 \triangleq \mathbb{F}_2 \times \mathbb{F}_2 \rightarrow \mathbb{F}_2$. $\text{CC}^0[2]$ is a sub-class of $\text{AC}^0[2]$ representing the constant-depth circuits consisting of only unbounded fan-in XOR gates. The complexity of $\text{CC}^0[2]$ circuits are measured in the number of wires instead of gates.

for any sparse NP language against probabilistic B_2 circuits would imply a major breakthrough: NP does not have n^k -size circuits for every fixed constant k . We also obtain stronger consequences with the same lower bound for MCSP using the explicitness of our hash family.⁴

- **(Extremely efficient PRFs)** Following [23, 13], our new hash constructions imply extremely efficient PRFs as well. Under the sub-exponential decisional Diffie-Hellman assumption (see, e.g., [5]), we can construct a PRF computable by $\text{AC}^0[2]$ circuits of wire complexity $2n + o(n)$ with $\text{polylog}(n)$ seed length (instead of $\text{poly}(n)$ in [13]).⁵ Furthermore, the $\text{AC}^0[2]$ -computable PRF is POLYLOGTIME -uniform, which implies a parallel algorithm to print the $\text{AC}^0[2]$ circuit. Similar results also hold for general circuits assuming sub-exponentially secure one-way function exists.

Both of the above consequences crucially rely on our improvement upon [13] on seed length and explicitness; we believe that our new hash construction will find further applications in other areas of computer science.

1.1 Randomness-efficient and strongly-explicit almost universal hash functions

Before stating our main theorem, we first define the notion of an almost universal hash family and its properties. A family of hash functions is defined as $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$, where each \mathcal{H}_n is a distribution of functions from $\{0, 1\}^n$ to $\{0, 1\}^m$ for output length $m = m(n)$. For a function $\varepsilon: \mathbb{N} \rightarrow \mathbb{R}$, we say \mathcal{H} is ε -almost universal if for all sufficiently large $n \in \mathbb{N}$, and for every two distinct inputs $x, y \in \{0, 1\}^n$, it holds that

$$\Pr_{h \leftarrow \mathcal{H}_n} [h(x) = h(y)] \leq \varepsilon(n).$$

We say \mathcal{H} is *linear* if every function in the family is linear over the field \mathbb{F}_2 .

We are now ready to present our construction of almost universal hash functions. The theorem is formally proved as Theorem 3.9.

► **Theorem 1.1** (Unconditional construction of extremely efficient almost universal hash). *Let $\ell \triangleq \log^2 n / \log \log n$. There is a family of linear $\Theta(\ell)$ -output $\exp(-\Omega(\ell))$ -almost universal hash functions $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$ satisfying the following properties.*

(Low complexity) *Each function $h \in \text{supp}(\mathcal{H}_n)$ is computable by a $\text{CC}^0[2]$ circuit with at most $2n + o(n)$ wires.*

(Randomness efficient) *\mathcal{H}_n is samplable with seed length $r = O(\ell \log^2 n)$ in polynomial time. More formally, there exists a polynomial-time algorithm \mathcal{G} that takes 1^n and a seed v of length r and outputs (the description of) a function $h_v \in \text{supp}(\mathcal{H}_n)$, such that \mathcal{H}_n and $\mathcal{G}(1^n, \mathcal{U}_r)$ are identical distributions (if we identify a function with its description), where \mathcal{U}_r is the uniform distribution over $\{0, 1\}^r$. The function h_v is said to be corresponding to the seed v .*

⁴ The Minimum Circuit Size Problem with size parameter $s(n)$ ($\text{MCSP}[s(n)]$) is defined as follows: given a string x of length $N = 2^n$, determine whether there exists an n -input $s(n)$ size circuit with x as its truth table.

⁵ The existence of PRFs computable by $\text{AC}^0[2]$ circuits might sound surprising, given that there exists a natural property against sub-exponential-size $\text{AC}^0[2]$ and natural properties can be used to break PRFs [38]. However, a closer look at the arguments in [38] shows that the natural property against $\text{AC}^0[2]$ only gives a large quasi-polynomial-time (i.e., $2^{\log^k n}$ for some large constant k) adversary breaking PRFs computable by $\text{AC}^0[2]$. This does not contradict our PRF construction in $\text{AC}^0[2]$, as our construction is only secure against $\exp(\log^2 n / \log \log n)$ -time adversaries; see Theorem 1.8 for details.

(POLYLOGTIME-uniform) *There exists a polynomial-time algorithm \mathcal{A} which takes a tuple (n, v, i, j) as the input⁶, and outputs the source of the j^{th} in-wire of the i^{th} gate in the $\text{CC}^0[2]$ -circuit (with $2n + o(n)$ wires) computing the function $h_v \in \text{supp}(\mathcal{H}_n)$ corresponding to the seed v .*

(Strongly explicit) *There exists a polynomial-time algorithm $\mathcal{B}(n, v, i, j)$ satisfying the following: Let v be a seed, $h_v \in \text{supp}(\mathcal{H}_n)$ be the \mathbb{F}_2 -linear function corresponding to v , and \mathbf{M} be the $m \times n$ \mathbb{F}_2 -matrix such that $h_v(x)_i = \sum_{j=1}^n \mathbf{M}_{i,j} x_j$ for any $i \in [m]$. It holds that $\mathcal{B}(n, v, i, j)$ outputs $\mathbf{M}_{i,j}$.*

The POLYLOGTIME-uniformity and strongly explicitness characterize the explicitness of our hash construction in two different senses. The former one captures the explicitness of the $2n + o(n)$ -size circuit computing the hash function, while the latter one focuses on the linear transformation corresponding to the hash function.

Here we briefly discuss why the notion of POLYLOGTIME-uniformity is important for our (and other potential) applications. POLYLOGTIME-uniformity is the most natural definition of parallel uniformity. It says that our hash function is parallel-efficient not only in the evaluation phase (since it is in sparse $\text{CC}^0[2]$) but also in the pre-processing phase, i.e., to construct the circuit according to the seed. With a POLYLOGTIME-uniform hash family, we are able to construct POLYLOGTIME-uniform low-complexity PRFs under standard assumptions, which is beneficial to the overall efficiency of other cryptographic systems involving PRFs (see, e.g., [5]). The “strongly explicitness” of our construction will be useful in proving hardness magnification theorems for MCSP, see Section 4.2.

Table 1 summarizes the differences between our new construction and known low-complexity constructions in [23, 9, 13].

■ **Table 1** Comparison of known constructions of almost universal hash functions. Note that the collision probability of these hash function are $\exp(-\Omega(m))$ for output length m . The construction in [23] is in addition a pairwise-independent hash function.

	Output size m	Seed Length	Circuit Class	Uniformity/Explicitness
Folklore ⁷	$m \leq n$	$\log n + O(m)$	linear-size NC^1	P-uniform ⁸
[23]	$m \leq n$	$O(n)$	linear-size NC^1	P-uniform
[13]	$m = O\left(\frac{\log^2 n}{\log \log n}\right)$	$O(n \log n)$	$2n + o(n)$ size $\text{CC}^0[2]$	P-uniform
Ours	$m = O\left(\frac{\log^2 n}{\log \log n}\right)$	$\text{polylog}(n)$	$2n + o(n)$ size $\text{CC}^0[2]$	POLYLOGTIME-uniform and strongly explicit

1.2 Implications on sharp bootstrapping results

Prior works in Hardness Magnification. Proving strong circuit lower bounds against explicit functions is one of the most significant challenges in complexity theory. However, despite decades of efforts, it remains unknown whether NP has linear-size circuits. The

⁶ The input lengths to \mathcal{A} and the algorithm \mathcal{B} in the next bullet are both $O(\log n) + r = \text{polylog}(n)$, so their running times are actually polylogarithmic in n .

⁷ This is done by sampling over the output bits of Spielman’s ECC [40] with a random sampling based on an expander walk (see Lemma 3.2 of [9]).

⁸ A hash construction is P-uniform if there is a polynomial time algorithm that prints the circuit computing a hash function given its seed v .

strongest explicit circuit lower bounds against general fan-in 2 circuits (known as B_2 circuits) is $3.1n - o(n)$ [28], following [12, 14]. A $5n - o(n)$ lower bound is known against U_2 circuits [26, 24]⁹.

A recently discovered phenomena in computational complexity called *hardness magnification* (see, e.g., [35, 34, 30, 9, 8]) provides new insights in bridging the gap between what we can prove and what we want to prove regarding circuit lower bounds. It says that a relatively weak circuit lower bound (e.g., $n \cdot \text{polylog}(n)$ size against B_2 circuits) for some special problems would imply major breakthroughs like $\text{NP} \not\subseteq \text{P}_{/\text{poly}}$.

Most interestingly, the required lower bounds for hardness magnification are only slightly stronger than provable lower bounds. For instance, Chen, Jin, and Williams [10] showed that an $n^{2+\varepsilon}$ lower bound against probabilistic De Morgan formulas for MCSP would imply $\text{NP} \not\subseteq \text{SIZE}[n^k]$ for all k , while an $n^{2-o(1)}$ lower bound for the same problem can be proved using a variant of the random restriction method in [18, 41, 22, 21, 34]. For more related works on hardness magnification, see [8] for a comprehensive summary.

Still, there are asymptotically significant gaps between what are provable and what suffice to bootstrap in all known results. For example, [30] says that proving a slightly super-linear circuit lower bound for a sparse version of MCSP would already imply $\text{NP} \not\subseteq \text{P}_{/\text{poly}}$. However, the best unconditional lower bound for any language in NP is only $3.1n - o(n)$ [28], which is still far from the superlinear threshold for bootstrapping. This is formally discussed in [17], which shows that the current technique for proving unconditional circuit lower bounds is not capable of breaking the linear barrier. Hence, even proving a super-linear lower bound for MCSP seems out of reach.

Overview of results in this section. By slightly generalizing the computation model to *probabilistic circuits*, we observe that even a *sub-linear* improvement over the known lower bounds would be enough to imply breakthroughs in complexity theory. For example, we show that improving a known $2n - O(1)$ probabilistic circuit lower bound for particular sparse languages to $2n + O(n/\log \log n)$ would imply $\text{NP} \not\subseteq \text{SIZE}[n^k]$ for every constant k . Regarding circuit lower bounds for deterministic time classes, we also obtain a similar threshold phenomenon by studying *explicit obstructions*; see Section 1.2.2 for details.

1.2.1 Sharp magnification threshold for probabilistic circuits

Notation. Formally, a *probabilistic circuit deciding a language L* is a distribution \mathcal{D} over circuits that outputs the wrong answer with probability at most $1/\text{poly}(n)$ on any input x , i.e., for all $x \in \{0, 1\}^n$ it holds that $\Pr_{D \leftarrow \mathcal{D}} [D(x) \neq L(x)] < 1/\text{poly}(n)$.¹⁰ We say a language L is $s(n)$ -sparse if for all sufficiently large n , $L \cap \{0, 1\}^n$ has size at most $s(n)$.

We first present the most standard form of our result, which gives fixed polynomial lower bounds for NP. This theorem is a special case of our general theorem of hardness magnification, which is stated and proved as Theorem 4.1.

⁹ A U_2 circuit consists of fan-in 2 gates computing all binary functions except for XOR and its complement.

¹⁰ Note that in contrast to robust classes like BPP, our result is actually sensitive to the error probability in the definition of probabilistic circuits, since the complexity overhead of error reduction is costly in the linear-size setting. For simplicity, we stick to the definition with error $1/\text{poly}(n)$.

► **Theorem 1.2** (Hardness magnification, high-end). *Let $\alpha(n) \triangleq \log n / \log \log n$. If there exists an $n^{\alpha(n)}$ -sparse language in NP that does not have probabilistic circuits¹¹ of size $2n + O(n/\log \log n)$, then we have (1) $\text{NTIME}[2^n] \not\subseteq \text{SIZE}[2^{n^{1/5}}]$ and (2) $\text{NP} \not\subseteq \text{SIZE}[n^k]$ for every constant k .¹²*

Let us compare this theorem with Theorem 1.2 of [9]. They proved that $\text{NP} \not\subseteq \text{SIZE}[n^k]$ follows from the non-existence of $O(n)$ -time randomized algorithms with n^ϵ bits of advice and $O(\log n)$ bits of randomness for $2^{n^{o(1)}}$ -sparse NP languages. In our theorem, we trade the sparsity for the exact constants in the running time of the randomized algorithm, showing that even a $2n + o(n)$ probabilistic circuit lower bound is enough.

Theorem 1.2 allows us to bootstrap a non-trivial lower bound for sparse languages in NP to strong sub-exponential lower bounds. If we only want a super-polynomial circuit lower bound for NEXP, then the assumption can be considerably weakened. The following theorem is another extreme case of Theorem 4.1.

► **Theorem 1.3** (Hardness magnification, low-end). *Let $\alpha(n) \triangleq \log n / \log \log n$. If there is an $n^{\alpha(n)}$ -sparse language in $\text{NTIME}[2^{n^{o(1)}}]$ that is not computable by probabilistic circuits of size $2n + O(n/\log \log n)$, then $\text{NEXP} \not\subseteq \text{P}_{/\text{poly}}$.*

We show that a nearly-matching lower bound is provable, from the proof of the $2n - O(1)$ lower bound for pseudorandom functions in [13]. This complements the bootstrapping above, and shows that there is only a gap of an *additive sub-linear term* towards breakthrough circuit lower bounds. The following theorem is formally proved as Corollary 4.11.

► **Theorem 1.4.** *There is an explicit $O(n)$ -sparse language L computable in P, such that every probabilistic circuit deciding L must have size at least $2n - 2$.*

Comparison with known lower bounds. We remark that a better-than- $2n$ circuit lower bound against probabilistic B_2 circuits can indeed be proved for certain non-sparse languages in P from known average-case lower bounds. This is because given a probabilistic circuit with small error probability, we can guarantee the existence of a particular circuit that approximates the language well by an averaging argument. In particular, the following lower bound of Chen and Kabanets [11] implies that no probabilistic circuits of size $2.49n$ can decide an explicit language in P.

► **Theorem 1.5** (Theorem 4.8, [11]). *There is a language L in P such that for any B_2 circuit C of size $2.49n$,*

$$\Pr_{x \leftarrow \{0,1\}^n} [C(x) = L(x)] \leq \frac{1}{2} + \frac{1}{2^{\Omega(n)}}.$$

Note that every sparse language must be *easy* on average as a trivial circuit outputting 0 can approximate it well. As a result, it is impossible to obtain a breakthrough directly from our hardness magnification result and an average-case lower bound. Nevertheless, linear-size

¹¹ Although it is not explicitly stated for simplicity, we mention that the lower bound required to obtain a breakthrough in Theorem 1.2, Theorem 1.3, and Theorem 1.6 can be weakened to the one-sided error case. That is, we only need to show that for any sparse language $L \in \text{NP}$, no probabilistic circuit of small size could output 1 with certainty for any $x \in L$ and output 0 with high probability otherwise.

¹² We remark that we can indeed obtain a conclusion that is stronger than both (1) and (2), albeit it is more technical: for all $c > 0$ there exists some $c' > c$ such that $\text{NTIME}[2^{c'n^{1/5}}] \not\subseteq \text{SIZE}[2^{cn^{1/5}}]$.

lower bounds against general circuits of size greater than $2n$ is a widely-studied problem for decades, so this magnification phenomenon may give us more insights on the setting of small linear size circuits. Following [8], our results achieve a “hardness magnification frontier”.

Better magnification results for meta-complexity problems. We also show that weak circuit lower bounds for specific meta-complexity problems such as MCSP imply consequences stronger than that of Theorem 1.2. Formally, we define $\text{MCSP}[s(n)]$ as the language taking a truth table of length $N = 2^n$ as input, and decides whether it admits a circuit of size at most $s(n)$. The following theorem is formally proved as Theorem 4.3.

► **Theorem 1.6.** *Let $N = 2^n$ be the truth table length of n -input Boolean functions and $n \leq s(n) \leq n^2/\log n$ be any size function. If $\text{MCSP}[s(n)]$ does not have probabilistic circuits of size $2N + O(N/\log \log N)$, then there exists some $c > 0$ such that $\oplus\text{P} \not\subseteq \text{SIZE}[2^{N^c}]$.*

Unfortunately, our lower bound technique in proving Theorem 1.4 cannot be used to derive a lower bound for MCSP. We leave proving an unconditional $2n - o(n)$ circuit size lower bound for MCSP, even against deterministic circuits, as an interesting open problem.

1.2.2 Sharp magnification thresholds for explicit obstruction

A drawback of Theorem 1.2 and Theorem 1.3 is that the conclusion only gives circuit lower bounds for nondeterministic time classes such as $\text{NTIME}[2^n]$. By studying a stronger notion called explicit obstruction, we are able to obtain a bootstrapping theorem with tight threshold, which gives circuit lower bounds for deterministic time classes such as $\text{TIME}[2^n]$.

Explicit obstruction. An *explicit obstruction of size $S(n)$ against \mathcal{C}* is an algorithm \mathcal{A} running in $\text{poly}(n, S(n))$ time, such that on input 1^n , \mathcal{A} outputs a set $E_n = \{(x_i, y_i)\}$ of size $S(n)$ such that $x_i \in \{0, 1\}^n$ and $y_i \in \{0, 1\}$ for every i , and $x_i \neq x_j$ for every $i \neq j$. The set has the property that, for all circuit $C \in \mathcal{C}$, there exists some $i \in [S(n)]$ such that $C(x_i) \neq y_i$. This can be viewed as an explicit proof of the hardness of C computing any n -bit function f that is consistent with E_n , since one can always efficiently find a counter-example from the explicit obstruction. Indeed, giving an explicit obstruction of polynomial size would directly imply $\text{P} \not\subseteq \mathcal{C}$. This concept is first suggested by Mulmuley [32] in the context of geometric complexity theory, where he argued that explicit obstructions might be essential in proving arithmetic circuit lower bounds. For more discussions on explicit obstruction, see [10].

Our results. Our theorem below shows that in the case of general Boolean circuits, even presenting such an obstruction for very small linear-size circuits would imply breakthrough circuit lower bounds. It is proved formally as Corollary 4.9 and Theorem 4.5.

- **Theorem 1.7.** *Let $\alpha(n) = \log n / \log \log n$. The following holds.*
- *There is an explicit obstruction of $\text{poly}(n)$ -size against $(2n - 2)$ -size B_2 circuits.*
 - *If for some $\beta(n) \geq \omega(n/\log \log n)$, there is an explicit obstruction of size $n^{\alpha(n)}$ against $2n + \beta(n)$ -size B_2 circuits, then we have (1) $\text{DTIME}[2^n] \not\subseteq \text{SIZE}[2^{n^{1/5}}]$ and (2) $\text{P} \not\subseteq \text{SIZE}[n^k]$ for every k .¹³*

¹³Similar to Theorem 1.2, we can indeed obtain a stronger conclusion that for all $c > 0$ there exists a $c' > c$ such that $\text{DTIME}[2^{c'n^{1/5}}] \not\subseteq \text{SIZE}[2^{cn^{1/5}}]$.

1.3 Strongly uniform pseudorandom functions

Utilizing the POLYLOGTIME-uniformity and short seed length of our hash constructions, we can apply them to the framework in [23, 13] to obtain extremely uniform low-complexity pseudorandom functions (PRFs) with short key length. Recall that a PRF is a family of distributions over functions that are indistinguishable from uniformly random functions by efficient adversaries. We show, from standard cryptography assumptions, that each member of a PRF can be constructed by circuits of size $2n + o(n)$; moreover, the construction itself can be POLYLOGTIME-uniform as well.

Formally, an m -output t -secure pseudorandom function is a family $\mathcal{F} = \{\mathcal{F}_n\}_{n \geq 1}$ of distributions \mathcal{F}_n on n -input m -output Boolean functions that fools every probabilistic $t(n)$ -time adversary \mathcal{A} , i.e.,

$$\left| \Pr_{f \leftarrow \mathcal{F}_n} [\mathcal{A}^f(1^n) \text{ accepts}] - \Pr_{g: \{0,1\}^n \rightarrow \{0,1\}^m} [\mathcal{A}^g(1^n) \text{ accepts}] \right| < \frac{1}{t}.$$

We say that \mathcal{C} has *key length* $s(n)$ if \mathcal{F}_n is samplable (by a $\text{poly}(n)$ -time algorithm) with $s(n)$ random bits. Similar to the hash functions, we say that \mathcal{F} is computable by POLYLOGTIME-uniform \mathcal{C} -circuits, if $s(n) \leq \text{polylog}(n)$ and there exists an algorithm $\mathcal{A}(n, v, i, j)$ running in $\text{polylog}(n)$ time (hence polynomial in its input length), that outputs the type of the i^{th} gate and the j^{th} descendant of the i^{th} gate in the \mathcal{C} -circuit computing the function $f_v \in \text{supp}(\mathcal{F}_n)$ keyed by v .

Since our hash functions can be implemented in $\text{CC}^0[2]$, we can obtain uniform PRFs in different circuit models based on different assumptions.

► **Theorem 1.8** (Uniform low-complexity PRFs). *There is a $t = t(n) = \exp\left(\Omega\left(\frac{\log^2 n}{\log \log n}\right)\right)$ and the following candidates of t -secure PRFs. Let ε be an arbitrarily small constant.*

B_2 circuits. *Assuming OWFs against $\exp(n^\varepsilon)$ -time adversaries exist, there exists a family of t -secure PRFs with key length $\text{polylog}(n)$ computable by POLYLOGTIME-uniform B_2 circuits of size $2n + o(n)$ and depth $\text{polylog}(n)$ simultaneously.*

NC^1 circuits. *Assuming $\exp(n^\varepsilon)$ -secure PRFs in NC^1 exist, there exists a family of t -secure PRFs with key length $\text{polylog}(n)$ computable by POLYLOGTIME-uniform B_2 circuits of size $2n + o(n)$ and depth $\log n + O(\log \log n)$ simultaneously.*

$\text{AC}^0[2]$ circuits. *Assuming $\exp(n^\varepsilon)$ -secure PRFs in NC^1 exist, there exists a family of t -secure PRFs with key length $\text{polylog}(n)$ computable by POLYLOGTIME-uniform $\text{AC}^0[2]$ circuits of wire complexity $2n + o(n)$.*

Note that the assumption for our NC^1 and $\text{AC}^0[2]$ constructions can be derived from standard assumptions such as sub-exponentially hard decisional Diffie-Hellman [33] or Ring Learning-with-Error [4], as well as other constructions like [31, 2]. We also mention that quasi-polynomial security is known to be optimal for $\text{AC}^0[2]$ PRFs by [38] (see, e.g., [5]).

Our PRF constructions are the first to achieve extremely low-complexity (efficient evaluation), short key length, and POLYLOGTIME-uniformity (parallel pre-processing) simultaneously from standard assumptions. Prior to our result, only PRF candidates with $O(n \log n)$ key length computable by P-uniform circuits of the same sizes were known [13] in these three circuit classes, improved on the linear-size NC^1 PRFs by Ishai, Kushilevitz, Ostrovsky, and Sahai [23] and the $\text{AC}^0[2]$ PRF due to Viola [43] with $n \cdot \text{polylog}(n)$ wire complexity and key length. Our improvement could also improve the efficiency of other cryptographic primitives involving PRFs such as message authentication code (MAC) and CCA-secure encryption (see, e.g., [23, 5]). See Section 1.3 [13] for more discussions on low-complexity PRFs.

► **Remark 1.9.** As pointed out by an anonymous reviewer, assuming a PRF against $\exp(n^\epsilon)$ -time adversaries (as we did in Theorem 1.8), we can reduce the key length to $\text{polylog}(n)$ generically by directly applying such a PRF to the keys (here we are actually using the PRF as a PRG of sub-exponential stretch). Particularly, let $\mathcal{F} = \{\mathcal{F}_n\}_{n \geq 1}$ be the low-complexity PRF in [13], $m = m(n) \leq \text{poly}(n)$ be the key length of \mathcal{F}_n , and $\mathcal{G} = \{\mathcal{G}_n\}_{n \geq 1}$ be a PRF against $\exp(n^\epsilon)$ -time adversaries. Let $f_{n,k}$ be the function in \mathcal{F}_n with key k , and $\text{tt}_m(f)$ be the first m bits in the truth table of the function f . We can define $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$ as

$$\mathcal{H}_n = \left\{ f_{n, \text{tt}_m(g)} \mid g \in \mathcal{G}_{\lceil \log^{2/\epsilon} m \rceil} \right\}.$$

Then \mathcal{H} is a PRF with the same circuit complexity as \mathcal{F} , while the key length is only $\text{poly}(\lceil \log^{2/\epsilon} m \rceil) = \text{polylog}(n)$. However, it is not clear how to directly reduce the uniformity to POLYLOGTIME-uniform in this way.

1.4 Intuition

We now briefly discuss the ideas behind our new results. Since our hash construction is based on the previous construction by Fan, Li, and Yang [13], we will first recall their original construction and then discuss how to reduce the key length and make the construction POLYLOGTIME-uniform and strongly explicit. After that, we will explain how to derive sharp bootstrapping results and construct POLYLOGTIME-uniform PRFs from the new hash construction.

1.4.1 Construction of randomness efficient almost-universal hash functions

The $2n + o(n)$ almost universal hash in [13]. Our starting point is the low-complexity hash function \mathcal{H} in [13] from high-girth graphs¹⁴. The constructed \mathcal{H} is linear over \mathbb{F}_2 . Hence, in order to make it almost universal, we only need to guarantee that for every non-zero input $x \in \{0, 1\}^n \setminus 0^n$, $\mathcal{H}(x) \neq 0$ with high probability.

Their hash function \mathcal{H} is the concatenation of two “hash functions” $\mathcal{H}^{\text{light}}$ and $\mathcal{H}^{\text{heavy}}$. The former one ensures that $\mathcal{H}^{\text{light}}(x) \neq 0$ with high probability for any non-zero x with small Hamming weights ($0 < |x| \leq n/2$), and the latter one ensures that $\mathcal{H}^{\text{heavy}}(x) \neq 0$ for any x with large Hamming weights ($|x| > n/2$). The construction of $\mathcal{H}^{\text{heavy}}$ is quite simple: for any non-zero input with Hamming weight larger than $n/2$, a random sampling of $n/\log n$ positions over all input bits includes a 1 with high probability; the hash $\mathcal{H}^{\text{light}}$ that deals with non-zero inputs with Hamming weights smaller than $n/2$ is in fact a combinatorial primitive called *1-detector*: a distribution \mathcal{L} over n -bit functions such that for any non-zero x with Hamming weight at most r (which is called the *range* of the 1-detector), $\mathcal{L}(x) \neq 0$ with high probability.

Fan, Li, and Yang [13] presented a construction of low-complexity 1-detectors using high-girth graphs. Consider an undirected graph with m vertices and n edges, the *girth* of G is the length of the shortest cycle in it. Let G be a graph with $n = \Theta(m \log m)$ and girth $g = \Omega\left(\frac{\log n}{\log \log n}\right)$ (see, e.g., [27, 7]), then the required 1-detector (with range $r = n/2$) is a depth-1 $\text{CC}^0[2]$ circuit whose topology is the vertex-edge incident graph of G , with input bits being randomly permuted. More formally, the edges of G are randomly permuted and

¹⁴The *girth* of an undirected graph is the length of the shortest cycles in it. A high-girth graph usually means a graph $G = (V, E)$ with girth $\Omega(\log_k n)$, where $k = 2|E|/|V|$ is the average degree of vertices.

23:10 Extremely Efficient Constructions of Hash Functions with Applications

assigned to the n input bits of the circuit, and each of the m nodes in G is assigned to an output gate computing the XOR of the input bits corresponding to the adjacent edges of the node.

The crucial observation leading to the analysis of their construction is that every input x with an all-zero output would imply a subset S of edges in G of size $|x|$ such that every vertex is adjacent to even number of edges in S , which further leads to a cycle in G of size at most $|x|$ that does not likely to exist in a high-girth graph.

Now we present the analysis more formally. We call a subset S of edges *good* if at least one of the vertices is adjacent to an odd number of edges in S . It is easy to see that an input x with non-zero output corresponds to a good subset of size $|x|$. Therefore, the 1-detector above has range r if and only if for all $0 < \ell \leq r$, a randomly chosen subset of size ℓ is good with high probability. Consider the cases for $\ell < g$ and $g \leq \ell \leq n/2$ separately.

1. Note that the graph has girth at least $g = \Omega\left(\frac{\log n}{\log \log n}\right)$. Every subset S of size $\ell < g$ is good, since the induced subgraph of a bad subset S would contain a cycle of size at most $|S|$.
2. Otherwise let $\ell \in [g, n/2]$. We claim that if we have chosen all but the first $\lceil g/3 \rceil$ edges, there will be *at most one* bad subset containing the $\ell - \lceil g/3 \rceil$ chosen edges: if both S_1 and S_2 are two distinct bad subsets containing them, their symmetric difference $S_1 \oplus S_2$ would also be a bad subset of size at most $2\lceil g/3 \rceil < g$, which is impossible as discussed above. This means that if a subset of edges of size ℓ is randomly chosen, it will not hit a bad subset with high probability.

Derandomizing the construction. The original hash in [13] requires $O(n \log n)$ bits of seed in $\mathcal{H}^{\text{light}}$ to permute the input bits and another $O(n)$ bits of seed in $\mathcal{H}^{\text{heavy}}$ to randomly sample $n/\log n$ input bits. To reduce the overall seed length to $\text{polylog}(n)$, we need to derandomize both of these two parts.

To derandomize $\mathcal{H}^{\text{light}}$ (i.e., the 1-detector \mathcal{L} based on high-girth graphs), we need to look into the correctness proof of it. Let x be an input with Hamming weight smaller than the range r of the 1-detector. It can be verified that the analysis of the 1-detector is correct as long as the 1-indices of x are randomly permuted (i.e., the 1-indices are randomly assigned to distinct edges in G). So if we restrict ourselves to the cases $r = \log^3 n$ (instead of $n/2$ in [13]), then a $\log^3 n$ -wise (almost) independent permutation¹⁵ would already be sufficient. In particular, we use the k -wise ε -dependent permutation by Kaplan, Naor, and Reingold [25] with seed length $O(k \log n + \log(1/\varepsilon)) = O(\log^4 n)$, where $k = \log^3 n$ and $\varepsilon = \exp(-\log^2 n)$ in our setting.

Apart from the 1-detector, we also need to derandomize the random sampling part in $\mathcal{H}^{\text{heavy}}$. We need to carefully choose the parameters so that we can handle all x with Hamming weights greater than $\log^3 n$ (since our 1-detector can only handle those below this threshold). In order to achieve negligible collision probability, we can sample $n/\log n$ number of bits. This is done with a folklore sampling trick via k -wise independent hash functions, which requires $O(\log^4 n)$ bits of randomness using [1].

Note that the collision probability of our derandomized hash function is $\exp(-\Omega(\frac{\log^2 n}{\log \log n}))$, which is the same as the original construction in [13] up to the constant hidden in $\Omega(\cdot)$.

¹⁵ A k -wise almost independent permutation is a distribution \mathcal{F} of permutations over $[n]$ such that for all $0 \leq i_0 < i_1 < \dots < i_{k-1} < n$, the distribution $(\mathcal{F}(i_0), \mathcal{F}(i_1), \dots, \mathcal{F}(i_{k-1}))$ is statistically close to $(\mathcal{R}(i_0), \mathcal{R}(i_1), \dots, \mathcal{R}(i_{k-1}))$, where \mathcal{R} is a truly random permutation.

Reducing the output length. The hash function above has $\Theta(n/\log n)$ output bits, whereas an ordinary universal hash with the same collision probability has output length only $\text{polylog}(n)$. Fortunately, the output length can be reduced to $\text{polylog}(n)$ with little overhead in seed length and circuit complexity. Since the composition of two almost universal hash functions is still almost universal, we compose the hash above with itself to reduce the output length to $\Theta(n/\log^2 n)$. We can then compose the resulting hash function with an almost universal hash with output length $\text{polylog}(n)$ and $(\text{CC}^0[2])$ circuit complexity $o(n \log^2 n)$ in [9] based on ε -biased sets [1] and expander walks, the overall circuit complexity can still be bounded by $2n + o(n)$.

In fact, there is a trade-off between circuit depth and seed length in our hash construction by setting up the parameters more carefully. The approach described above can be computed by depth-3 $\text{CC}^0[2]$ circuits and requires a seed length $O(\log^4 n)$. If we allow the depth to be 300, the seed length can be reduced to $\log^{3.01} n$.

Making the construction explicit and uniform. There are several main components in our hash construction that are not obviously strongly explicit and POLYLOGTIME -uniform. The first one is the high-girth graph construction of 1-detectors in $\mathcal{H}^{\text{light}}$. We observe that the local topology around an edge or vertex in the high-girth graphs constructed by [27] can be obtained within poly-logarithmic time; see Appendix A for a formal argument. The k -wise almost-independent random permutation used in our 1-detectors for $\mathcal{H}^{\text{light}}$ is also strongly explicit according to [25]. Therefore $\mathcal{H}^{\text{light}}$ is strongly explicit. It is also easy to verify that $\mathcal{H}^{\text{heavy}}$ is strongly explicit given the k -wise ε -dependent distribution in [1]. Finally, the hash function in [9] for shrinkage reduction is strongly explicit, since both of its two components (i.e., ε -biased sets and expander graphs) are strongly explicit (see [25, 3]).

1.4.2 Applications to hardness magnification and construction of PRFs

Sharp hardness magnification. We first show how to apply our new hash construction to obtain an extremely sharp hardness magnification result for all sparse NP languages (i.e., Theorem 1.2 and Theorem 1.3), following [9, 10]. Recall that [9, Theorem 1.2] proved that if there is a $2^{n^{o(1)}}$ -sparse NP language L that cannot be computed by cn -size probabilistic circuits for some big constant $c \gg 1$, then $\text{NP} \not\subseteq \text{SIZE}[n^k]$ for all constant k .

[9, Theorem 1.2] is proved by taking the contrapositive: assuming that $\text{NP} \subseteq \text{SIZE}[n^k]$ for some k , we first sample h from a proper hash function family to *kernelize* any sparse language L in NP (i.e., a randomized reduction from the sparse language L to another (non-sparse) NP problem L' with much smaller input size $m \ll n^{1/k}$), then use the $\text{SIZE}[n^k]$ circuit D for L' (which has size $m^k \ll n$) to solve L . Now, composing the hash function h together with D gives a linear-size probabilistic circuit computing L . This proves the contrapositive of our desired theorem. Due to technical reasons, in the proof above one also has to combine the hash function h with an error-correcting code. While there is an efficient construction of linear-size error-correcting codes [40], it has size cn for some constant $c \gg 2$. Hence, the size of encoding circuits for error-correcting codes become the bottleneck which prevents further improvement.

We are able to avoid using error-correcting codes by utilizing properties of the new almost-universal hash construction (the construction of [9] works for all k -perfect hash). In more details, let $H(v, x) : \{0, 1\}^{O(\ell \log^2 n)} \times \{0, 1\}^n \rightarrow \{0, 1\}^{m(n)}$ be the hash function given by Theorem 1.1 with $\ell = \log^2 n / \log \log n$. For any sparse language L , we define an intermediate problem as follows: $L' = \{(v, h) \mid \exists x \in \{0, 1\}^n \cap L \text{ s.t. } H(v, x) = h\}$. By the definition, for any no-instance $x \in \{0, 1\}^n \setminus L$, the probability that there exists

23:12 Extremely Efficient Constructions of Hash Functions with Applications

an yes-instance $x' \in L \cap \{0, 1\}^n$ having the same hash value with x can be bounded by $\exp(-\Omega(\ell)) \cdot |L \cap \{0, 1\}^n| = n^{-\omega(1)}$. Hence by checking whether $(v, H(v, x)) \in L'$ for a random seed v , we can determine with high probability whether $x \in L$. This gives us a simple probabilistic circuit to decide L that only needs one evaluation of the hash function H (which is of circuit complexity $2n + o(n)$) and one oracle query to L' . Given the assumption that $\text{NP} \subseteq \text{SIZE}[n^k]$, we can show that L' can be decided by circuits of size $o(n)$, hence the overall circuit complexity would be $2n + o(n)$.

Uniform pseudorandom functions. We now briefly describe how our hash function can be used to obtain POLYLOGTIME-uniform pseudorandom functions with $\text{polylog}(n)$ key length. Following the framework established in [23, 13], we use Levin's trick, which says that the composition of a PRF and an almost universal hash function is also pseudorandom. More formally, let $\mathcal{F} = \{\mathcal{F}_n\}_{n \geq 1}$ be a family of PRF secure against any $\exp(n^\varepsilon)$ -time adversary, and $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$ be an almost universal hash function with output length $\log^c n$, then the composition $\mathcal{F} \circ \mathcal{H}$ is also a PRF against any polynomial-time adversary as long as $c > \varepsilon^{-1}$. Let $c = 2\varepsilon^{-1}$ and \mathcal{C} be any circuit class. By assuming a PRF (secure against any $\exp(n^\varepsilon)$ -time adversary) with polynomial key length computable by P-uniform \mathcal{C} -circuits of size $o(\exp(n^\varepsilon))$, we can then obtain a PRF (secure against any polynomial-time adversary) computable by POLYLOGTIME-uniform $2n + o(n)$ size \mathcal{C} -circuits with $\text{polylog}(n)$ key length, using our POLYLOGTIME-uniform efficient hash with $\text{polylog}(n)$ seed length. Note that for $\mathcal{C} \in \{B_2, \text{NC}^1, \text{AC}^0[2]\}$, we can implement such PRF candidates by plugging in standard constructions based on the existence of OWFs against any sub-exponential adversary for B_2 circuits, and decisional Diffie-Hellman [33] or Ring Learning-with-Errors [4] for NC^1 and $\text{AC}^0[2]$ circuits.

2 Preliminaries

Notation. We define $[n] \triangleq \{0, 1, \dots, n-1\}$, $B_{n,m}$ as the set of all functions from \mathbb{F}_2^n to \mathbb{F}_2^m and $B_n \triangleq B_{n,1}$. The Hamming weight of $x \in \mathbb{F}_2^n$, denoted by $|x|$, is defined as the number of 1's in x ; the Hamming distance $\Delta(x, y)$ of x and y from \mathbb{F}_2^n is defined as the Hamming weight of the bitwise XOR of them. The concatenation of two strings x and y is denoted by $x||y$, and the i^{th} bit of x (0-indexed) is denoted by x_i . Graphs are undirected by default. We assume that all functions used as parameters of our constructions are poly-logarithmic time constructible, i.e., for any function $\ell(n)$ that is used as a parameter in our results, there is a polynomial-time algorithm \mathcal{A} such that $\mathcal{A}(n)$ outputs the binary representation of $\ell(n)$.

We use $x \leftarrow \mathcal{D}$ to denote a random element x sampled from a distribution \mathcal{D} , and $\mathcal{D}(x_0) \triangleq \Pr_{x \leftarrow \mathcal{D}}[x = x_0]$. Natural numbers are represented in binary when being fed into Turing machines or circuits; and 1^n represents the unary representation of n . We assume basic familiarity with cryptographic primitives like one-way functions and pseudorandom functions, and typical complexity classes like P, NP, and $\oplus\text{P}$ (see, e.g., [3]).

2.1 Probability Theory

Let $\mathcal{U}(S)$ be the uniform distribution supported on a set S , and $\mathcal{U}_\ell \triangleq \mathcal{U}(\{0, 1\}^\ell)$. A family $\mathcal{D} = \{\mathcal{D}_n\}_{n \geq 1}$ of distributions is said to be *samplable with seed length $\ell(n)$* (or *$\ell(n)$ -samplable*) if there is a polynomial-time algorithm \mathcal{A} such that $\mathcal{A}(1^n, \mathcal{U}_{\ell(n)})$ samples \mathcal{D}_n . For every $s \in \{0, 1\}^\ell$, we say $\mathcal{A}(1^n, s)$ is the element corresponding to the seed s .

The *statistical distance* between two distributions \mathcal{D}_1 and \mathcal{D}_2 over a set S , denoted by $\text{SD}(\mathcal{D}_1, \mathcal{D}_2)$, is defined as

$$\text{SD}(\mathcal{D}_1, \mathcal{D}_2) \triangleq \frac{1}{2} \sum_{u \in S} |\mathcal{D}_1(u) - \mathcal{D}_2(u)|.$$

A distribution \mathcal{D}_2 is said to be δ -close to \mathcal{D}_1 if their statistical distance is at most δ .

Statistical distance characterizes the intractability of distinguishing two distributions by any test (even computationally unbounded): if \mathcal{D}_1 and \mathcal{D}_2 have statistical distance at most δ , then for any stochastic process T , $|\Pr_{u \leftarrow \mathcal{D}_1, T}[T(u) = 1] - \Pr_{u \leftarrow \mathcal{D}_2, T}[T(u) = 1]| \leq \delta$.

2.2 Circuit Classes

In this paper we will work with various circuit classes. In general, a circuit is an acyclic graph where each of its nodes can be an input variable, a constant $c \in \{0, 1\}$, or a gate. One or more nodes are marked as output nodes (together with an index i denoting the corresponding output bit). The *depth* of a circuit is the number of edges on the longest path from any input variable to an output node. An n -input m -output circuit computes a function in $B_{n,m}$.

B_2 circuits. A B_2 circuit (or general circuit) contains fan-in-2 gates that can compute any binary function $f \in B_2$. The *size* of a B_2 circuit refers to the number of gates involved.

NC^1 circuits. An NC^1 circuit is a B_2 circuit with $O(\log n)$ depth. Note that a single-output NC^1 circuit of depth d can be converted into a formula of size $O(2^d)$.

$\text{CC}^0[2]$ circuits. A $\text{CC}^0[2]$ circuit is a constant-depth circuit with only XOR gates of unbounded fan-in. It is easy to see that $\text{CC}^0[2]$ circuits can only compute linear functions over \mathbb{F}_2 . The complexity of a $\text{CC}^0[2]$ circuit is usually measured by the number of wires.

$\text{AC}^0[m]$ circuits. An $\text{AC}^0[m]$ circuit is a constant-depth circuit with fan-in-1 NOT gates and unbounded fan-in gates over $\{\text{AND}, \text{OR}, \text{MOD}_m\}$, where $\text{MOD}_m(x_1, \dots, x_k) = 1$ if and only if $x_1 + x_2 + \dots + x_k \equiv 0 \pmod{m}$. Similar to $\text{CC}^0[2]$ circuits, the complexity of an $\text{AC}^0[m]$ circuit is measured by the number of wires.

A *probabilistic B_2 circuit* (or simply a probabilistic circuit) with input size n and output size m is a distribution \mathcal{C}_n over n -input m -output B_2 circuits. The circuit complexity of a probabilistic circuit is defined as the maximum complexity of functions in its support. A family $\mathcal{C} = \{\mathcal{C}_n\}_{n \geq 1}$ of n -input 1-output probabilistic circuit is said to *decide a language L with error probability $\varepsilon = \varepsilon(n)$* if for sufficiently large n and all $x \in \mathbb{F}_2^n$, $\Pr_{C \leftarrow \mathcal{C}_n}[C(x) \neq L(x)] \leq \varepsilon$.

2.3 Hash and 1-detector

► **Definition 2.1** (Almost universal hash function). *Let $m = m(n)$ and $\varepsilon = \varepsilon(n)$ be two parameters. An m -output ε -almost universal hash function is a family of distributions $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$, where \mathcal{H}_n is supported on $B_{n,m}$, such that $\Pr_{h \leftarrow \mathcal{H}_n}[h(x) = h(y)] \leq \varepsilon(n)$ for all $x \neq y$ and sufficiently large n . The parameter ε is called its collision probability. It is linear if every function in the support of \mathcal{H}_n is linear. It is said to be $s(n)$ -samplable if the family \mathcal{H} of distributions is samplable with seed length $s(n)$.*

We will make heavy use of the notion of 1-detectors in [13], as parts of our almost universal hash construction.

► **Definition 2.2** (1-detector). *An m -output (randomized) 1-detector with range r and error ε is a family of distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n \geq 1}$, where \mathcal{D}_n is supported on linear functions in $B_{n,m}$, such that for sufficiently large n , $\Pr_{L \leftarrow \mathcal{D}_n}[L(x) \neq 0] \leq \varepsilon(n)$ for all $x \in \mathbb{F}_2^n$ satisfying $0 < |x| \leq r$. It is said to be $s(n)$ -samplable if the family \mathcal{D} of distributions is samplable with seed length $s(n)$.*

We emphasize that the definition requires the functions in the family of distributions to be linear. This ensures that, for any randomized 1-detector \mathcal{D}_n with range r and error ε , and any $x_1 \neq x_2$ with Hamming distance at most r , it holds that $\Pr_{L \leftarrow \mathcal{D}_n} [L(x_1) \neq L(x_2)] \leq \varepsilon(n)$.

The *shrinkage* of a hash function (or 1-detector) is defined as the input length n divided by the output length m (e.g., poly-logarithmic shrinkage means $m = n/\text{polylog}(n)$).

2.4 ε -biased set and k -wise independence

To derandomize the low-complexity hash function in [13], we need several standard derandomization tools including strongly explicit ε -biased sets, k -wise independent distributions, and k -wise independent permutations.

► **Definition 2.3** (ε -biased set). *For any $n \geq 1$ and $\varepsilon \in (0, 1/2)$, a set $S \subseteq \mathbb{F}_2^n$ is said to be ε -biased if for all non-zero $v \in \mathbb{F}_2^n$, $\Pr_{w \leftarrow S}[(w, v) = 0] \in [1/2 - \varepsilon, 1/2 + \varepsilon]$.*

► **Theorem 2.4** (Alon, Goldreich, Håstad, and Peralta [1]). *For any constant $\varepsilon \in (0, 1/2)$, there is a family $\{S_n \subseteq \mathbb{F}_2^n\}_{n \geq 1}$ of ε -biased sets such that $|S_n| = \tilde{O}(n^2)$. Moreover, there is an algorithm $\mathcal{A}(n, i, j)$ running in time $\text{poly}(\log n)$ that computes the j^{th} bit of the i^{th} vector in S_n .*

► **Definition 2.5.** *A k -wise ε -dependent distribution with length n and alphabet size p is a distribution \mathcal{D} over $[p]^n$ such that for all $0 \leq i_1 < i_2 < \dots < i_k < n$, the distribution over $[p]^k$ obtained by restricting \mathcal{D} to the $i_1^{\text{th}}, i_2^{\text{th}}, \dots, i_k^{\text{th}}$ coordinates is ε -close to the uniform distribution over $[p]^k$. It is said to be a k -wise independent distribution if $\varepsilon = 0$.*

► **Theorem 2.6** (Alon, Goldreich, Håstad, and Peralta [1]). *Let $\varepsilon = \varepsilon(n) > 0$ be a parameter. There is an algorithm \mathcal{A} such that $\mathcal{A}(1^n, k, \mathcal{U}_r)$ runs in time $\text{poly}(n, \log(1/\varepsilon))$ and samples a k -wise ε -dependent distribution with length n and alphabet size 2 for every $1 \leq k \leq n$, where $r = O(k + \log \log n + \log(1/\varepsilon))$. Moreover, there is an algorithm \mathcal{B} such that $\mathcal{B}(n, k, v, i)$ runs in $\text{poly}(\log n, k, \log(1/\varepsilon))$ time and computes the i^{th} coordinate of the string corresponding to the seed v .*

► **Definition 2.7.** *Let $n \geq 1$ be the number of elements, and P_n be the set of all permutations over $[n]$. A k -wise ε -dependent permutation is a distribution \mathcal{D} over P_n such that for all $0 \leq i_1 < i_2 < \dots < i_k < n$, the distribution of $(f(i_1), f(i_2), \dots, f(i_k))$ with $f \leftarrow \mathcal{D}$ is ε -close to the uniform distribution over $\{(j_1, j_2, \dots, j_k) \mid j_1, \dots, j_k \in [n] \text{ are pairwise distinct}\}$.*

► **Theorem 2.8** (Kaplan, Naor, and Reingold [25], Theorem 5.9). *Let n be a power of 2, $2 \leq k \leq n$, and $\varepsilon > 2^{-n}$. There is an n -element k -wise ε -dependent permutation that is samplable with seed length $O(k \log n + \log(1/\varepsilon))$. Moreover, there is an algorithm $\mathcal{A}(n, v, i)$ that runs in $\text{poly}(\log n, k, \log(1/\varepsilon))$ time and outputs $\rho_v(i)$ for the permutation ρ_v corresponding to the seed v .*

2.5 Expander Graphs

We will need strongly explicit expander graphs in our proofs. We first recall the definition of expander graphs.

► **Definition 2.9.** *An n -vertex d -regular graph G is called an (n, d, λ) expander graph (or (n, d, λ) -graph) if $\lambda_2(G) \leq \lambda$, where $\lambda_2(G)$ denotes the second largest eigenvalue of normalized adjacency matrix (i.e., adjacency matrix divided by d) of G . A family of graphs $\{G_n\}_{n \geq 1}$ is an expander graph family if there exists constants d and $\lambda < 1$ such that for all sufficiently large n , G_n is an (n, d, λ) -graph.*

We will make use of the following construction of strongly explicit expander graphs.

► **Theorem 2.10** (Strongly explicit expander; see [3, Theorem 21.19]). *There exists an expander graph family $\{G_n\}_{n \geq 1}$ and an algorithm $\mathcal{A}(n, v, i)$ that runs in $\text{polylog}(n)$ time (i.e., polynomial in input length) and outputs the i^{th} neighbor of the node v in G_n , where $v \in [n]$ and $i \in [d]$.*

Performing random walk on expanders is a standard derandomization technique. Consider the task to find a good element from n elements in which there is a constant fraction of them being good. A trivial approach is to sample ℓ independently random elements, which has $\exp(-\Omega(\ell))$ error probability but requires $\ell \log n$ random bits. By applying the following lemma, we can reduce the randomness complexity to $O(\log n + \ell)$ while keeping the error probability to be exponentially small.

► **Lemma 2.11** (Expander walk; see [3, Theorem 21.12]). *Let G be an (n, d, λ) -graph and S be a subset of vertices of size at most βn for some $\beta \in (0, 1)$. Let X_1, X_2, \dots, X_k be random variables denoting a random walk in G (where X_1 is uniformly chosen), then*

$$\Pr[\forall 1 \leq i \leq k, X_i \in S] \leq \left((1 - \lambda)\sqrt{\beta} + \lambda \right)^{k-1}.$$

2.6 Graph with large girth

The *girth* of an undirected graph is the length of the minimum cycle in it. We need the following construction of strongly explicit graphs with large girth.

► **Theorem 2.12** (Adapted from [27]; see Appendix A). *Let $r = r(n) = n^{o(1)}$ be a parameter. For every sufficiently large n , there exists an $m = \Theta(\frac{n}{r})$ and a regular graph $G_{m,n}$ with m vertices, n edges, and girth $\Omega(\frac{\log n}{\log r})$. Moreover, there exists a $\text{polylog}(n)$ -time algorithm $\mathcal{A}(n, i)$ for $i \in [n]$ that outputs the indices of the two endpoints of the i^{th} edge in $G_{m,n}$, and a $\text{polylog}(n)$ -time algorithm $\mathcal{B}(n, i, j)$ for $i \in [m]$ that outputs the j^{th} edge attaching to the i^{th} vertex.*

3 Randomness-efficient low-complexity hash functions

In this section, we present constructions of randomness-efficient low-complexity hash functions with various parameters and properties.

In Section 3.1, we show almost universal hash functions can be constructed from 1-detectors (Lemma 3.2). In Section 3.2 we give a derandomized version of construction of 1-detectors from [13] based on high-girth graphs (Lemma 3.3), and use that to construct a low-complexity hash function with poly-logarithmic seed-length and $n/\text{polylog}(n)$ output length (Theorem 3.4).

In Section 3.3, we show how to reduce the output length from $n/\text{polylog}(n)$ to $\text{polylog}(n)$, by composing almost universal hash families. Finally, in Section 3.4 and 3.5, we establish the uniformity and explicitness of our constructions, which are essential for our applications to hardness magnification and PRF constructions.

Notation. In this section, for a construction \mathcal{H} with parameters a, b, c , we will use $\mathcal{H}_{\hat{a}, \hat{b}, \hat{c}}$ to denote the specific construction \mathcal{H} with the parameters specified to \hat{a}, \hat{b} and \hat{c} . When the parameters are obvious from the context, we often omit the subscripts and simply write it as \mathcal{H} .

3.1 General construction from 1-detectors

We first give a general construction of almost universal hash functions from 1-detectors, using the following sampling procedure.

► **Lemma 3.1.** *For all integers n, b and r such that $b \leq n$ and $\max\{10n/b, 10 \log \log n\} \leq r \leq n$, there is a distribution $\mathcal{D}_{n,b,r}^{\text{samp}}$ supported on $[n]^b$ samplable by $O(r \log(n/b))$ bits, such that the following conditions hold:*

1. (**Hitting Condition**). *For all $x \in \mathbb{F}_2^n$ with $|x| \geq r$, it holds that*

$$\Pr_{(w_0, \dots, w_{b-1}) \leftarrow \mathcal{D}_{n,b,r}^{\text{samp}}} \left[\bigwedge_{j \in [b]} [x_{w_j} = 0] \right] \leq 2 \exp\left(-\frac{br}{2n}\right). \quad (1)$$

2. (**Ordering**) *For every $(w_0, \dots, w_{b-1}) \in \text{supp}(\mathcal{D}_{n,b,r}^{\text{samp}})$, it holds that $w_0 < w_1 < \dots < w_{b-1}$.*
3. (**Explicitness**) *There are algorithms $\mathcal{A}_{n,b,r}^{\text{samp}}(v, i)$ and $\mathcal{B}_{n,b,r}^{\text{samp}}(v, j)$ running in $\text{poly}(\log n, r)$ time such that*

$$\mathcal{A}_{n,b,r}^{\text{samp}}(v, i) \text{ outputs } \begin{cases} j & \text{if } w_j = i \text{ for some } j \in [b], \\ \perp & \text{otherwise;} \end{cases}$$

$$\mathcal{B}_{n,b,r}^{\text{samp}}(v, j) \text{ outputs } w_j.$$

where $(w_0, \dots, w_{b-1}) \in \text{supp}(\mathcal{D}_{n,b,r}^{\text{samp}})$ is the vector corresponding to the seed v .

Proof. We firstly describe a sampling procedure that satisfies Equation (1) but has a long seed length and then reduce the seed length to $O(r \log(n/b))$ bits using the explicit k -wise ε -dependent distribution from Theorem 2.6. The sampling procedure is simple: we first partition $[n]$ into b consecutive groups g_0, g_1, \dots, g_{b-1} of size either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$; we then uniformly choose an $i_j \in g_j$ for each $j \in [b]$; finally we output the tuple $(i_0, i_1, \dots, i_{b-1})$.

For any $x \in \mathbb{F}_2^n$ with Hamming weight at least r , for some $\ell \leq r$, there are ℓ groups such that there are at least r 1-indices (i.e., the corresponding bits of x are 1) in these groups. Assume that there are a_j 1-indices in the j^{th} among these ℓ groups. The probability that none of the 1-indices is sampled is at most

$$\prod_{j=1}^{\ell} \left(1 - \frac{a_j b}{2n}\right) \leq \exp\left(\sum_{j=1}^{\ell} \log\left(1 - \frac{a_j b}{2n}\right)\right) \leq \exp\left(-\sum_{j=1}^{\ell} \frac{a_j b}{2n}\right) \leq \exp\left(-\frac{br}{2n}\right). \quad (2)$$

A trivial implementation of the sampling procedure above needs seed length $O(b \log(n/b))$: for each group, we sample a random binary string of length $t \triangleq \lceil \log(n/b) \rceil$ indicating the index we want to choose. Therefore, we would need to sample bt bits in total. The observation here is that the argument in (2) only involves $\ell \leq r$ groups. Hence, the analysis still works if we sample those required bt bits from an rt -wise $\exp(-br/(2n))$ -dependent distribution with alphabet 2. Note that this incurs an additional error of $\exp(-br/(2n))$.

By Theorem 2.6, the distribution is samplable with seed length

$$O(rt + \log \log(bt) + br/(2n)) = O(r \log(n/b)),$$

and the algorithms $\mathcal{A}_{n,b,r}^{\text{samp}}$ and $\mathcal{B}_{n,b,r}^{\text{samp}}$ can be constructed straightforwardly from the algorithm \mathcal{B} in Theorem 2.6.¹⁶ ◀

¹⁶To compute $\mathcal{A}_{n,b,r}^{\text{samp}}(v, i)$ for some i belonging to the group g_a , we only need to check if i is the selected element of the group g_a , and then output a if the answer is yes, and output \perp otherwise.

Now we are ready to give the construction of our hash function from 1-detectors. Recall that their definitions are given in Section 2.3.

► **Construction 1** (Hash function $\mathcal{H}^{\mathcal{L}}$ from 1-detectors \mathcal{L}). *Let n be the input length and $b(n) = o(n)$ be a parameter. Given a randomized 1-detector $\mathcal{L} = \{\mathcal{L}_n\}_{n \geq 1}$ with range \hat{r} , we construct a hash function $\mathcal{H}_b^{\mathcal{L}} = \{\mathcal{H}_n\}_{n \geq 1}$ as follows.*

- *Let $\mathcal{D}^{\text{samp}}$ be the distribution in Lemma 3.1 with parameters n, b and $r = \hat{r}$. For each L in the support of \mathcal{L} and each $D = (j_0, \dots, j_{b-1})$ in the support of $\mathcal{D}^{\text{samp}}$, we define a function $h_{L,D}$ as*

$$h_{L,D}(x) \triangleq L(x) \|x_{j_0}\|x_{j_1}\| \dots \|x_{j_{b-1}}.$$

- *\mathcal{H}_n is then defined to be the distribution generated as follows: sample $L \leftarrow \mathcal{L}_n$ and $D \leftarrow \mathcal{D}^{\text{samp}}$, and then output $h_{L,D}$.*

We show that Construction 1 indeed gives almost universal hash functions.

► **Lemma 3.2** ($\mathcal{H}^{\mathcal{L}}$ is a linear almost universal hash). *Let $\mathcal{L} = \{\mathcal{L}_n\}_{n \geq 1}$ be an \hat{m} -output \hat{s} -samplable randomized 1-detector with error $\hat{\varepsilon}$ and range \hat{r} , such that $\hat{m} = o(n)$ and $\hat{r} = \omega(\log n)$. Let $b = b(n) = o(n)$ be a parameter and $r = r(n) = \hat{r}(n)$. The followings hold for $\mathcal{H}_b^{\mathcal{L}}$:*

1. $\mathcal{H}_b^{\mathcal{L}}$ has output length $m = \hat{m} + b$.
2. $\mathcal{H}_b^{\mathcal{L}}$ is $(\hat{s} + \hat{r} \log(n/b))$ -samplable.
3. $\mathcal{H}_b^{\mathcal{L}}$ is a linear ε -almost universal hash function, where $\varepsilon \triangleq \hat{\varepsilon} + \exp(-\Omega(br/n))$.

Proof. The first two items follow directly from the definition of $\mathcal{H}^{\mathcal{L}}$ from Construction 1. So we will only establish the last item. We need to show that for any two inputs $x_1 \neq x_2$ from $\{0, 1\}^n$, their hash values collide with probability at most ε .

Consider the following two cases depending on whether the Hamming distance between x_1 and x_2 is small or large:

- If $0 < \Delta(x_1, x_2) \leq r$, then by the definition of randomized 1-detector (Definition 2.2), $L(x_1)$ equals to $L(x_2)$ with probability at most $\hat{\varepsilon}$ for $L \leftarrow \mathcal{L}_n$ (note that L is a linear function over \mathbb{F}_2). Since the hash values contain the output of the 1-detector as the first \hat{m} bits, the hash values collide with probability at most $\hat{\varepsilon}$.
- If $\Delta(x_1, x_2) > r$, then $y \triangleq x_1 \oplus x_2$ has Hamming weight at least r . By Lemma 3.1, a random $D \leftarrow \mathcal{D}_{n,b,r}^{\text{samp}}$ fails to contain all 1-indices of y with probability at most $\exp(-\Omega(br/n))$, which means that the last b bits of the hash values of x_1 and x_2 collide with probability at most $\exp(-\Omega(br/n))$.

It follows immediately that the collision probability is at most $\varepsilon = \hat{\varepsilon} + \exp(-\Omega(br/n))$. ◀

3.2 Randomness-efficient low-complexity 1-detectors

Now we will present the construction of randomness-efficient low-complexity 1-detectors. Applying Lemma 3.2, this construction yields a hash function with inverse-super-polynomial collision probability, short seed length, and moderately large shrinkage.

Intuition. Our construction is essentially a derandomized version of the randomized 1-detector based on high-girth graphs in [13]. The randomized 1-detector supports on depth-1 $\text{CC}^0[2]$ circuits with $o(n)$ gates and wire-complexity $2n$. The topology of any depth-1 $\text{CC}^0[2]$ circuit can be considered as the edge-vertex incident graph of an undirected graph $G_{m,n} = (V, E)$, that is, we identify the m gates with the vertices, and the n variables with

23:18 Extremely Efficient Constructions of Hash Functions with Applications

the edges in the graph. One can check that if $G_{m,n}$ has girth g , the corresponding circuit would directly make a (deterministic) m -output 1-detector with range $g - 1$: assume that it is not the case, the bad input x with Hamming weight $w < g$ specifies a subset of edges $T \subseteq E$ which forms a Eulerian cycle of size $w < g$. By a similar argument, [13] boosts the range to $n/2$ by randomly permuting the input bits.

The construction of [13] does not suffice for our application because we need $\Theta(n \log n)$ bits to sample a random permutation. Fortunately, it can be derandomized. By looking into its correctness proof, we can see that if we replace the random permutation by an r -wise almost independent permutation (see Theorem 2.8), it can still achieve 1-detection for range r and roughly quasi-polynomial error probability.

Now we specify the construction of our 1-detector \mathcal{L}^{hg} (hg stands for high-girth graphs).

► **Construction 2** (Construction of the 1-detector \mathcal{L}^{hg} from high-girth graphs). *Let $k = k(n) = n^{o(1)}$ be the shrinkage parameter, $\log n \leq r(n) \leq n/2$ be the range. Let $\gamma = \gamma(n) \triangleq \log^2 n / \log k$. $\mathcal{L}_{k,r}^{\text{hg}} = \{\mathcal{L}_n\}_{n \geq 1}$ is constructed as follows.*

- *Let n' be the smallest power of two that is larger than n . Let $G_{m,n'}$ be the regular graph with $m = \Theta(n'/k)$ vertices, n' edges, and girth $g = \Theta(\log n / \log k)$ from Theorem 2.12. Let $\mathcal{D} = \{\mathcal{D}_{n'}\}$ be the explicit family of r -wise $2^{-\gamma}$ -dependent permutation in Theorem 2.8.*
- *Let $\Gamma(i)$ be the set of indices of edges incident to the i^{th} vertex in $G_{m,n'}$. For each permutation $\sigma \in \text{supp}(\mathcal{D}_{n'})$, we define a function $L_\sigma : \{0, 1\}^n \rightarrow \{0, 1\}^m$ as*

$$L_\sigma(x)_i \triangleq \bigoplus_{j \in [n] \text{ s.t. } \sigma(j) \in \Gamma(i)} x_j \quad \forall i \in [m].$$

- *\mathcal{L}_n is then defined to be the distribution generated as follows: sample $\sigma \leftarrow \mathcal{D}_{n'}$ and then output L_σ .*

► **Lemma 3.3** (\mathcal{L}^{hg} is a 1-detector). *Let $k = k(n) = n^{o(1)}$ be the shrinkage parameter, $\log n \leq r(n) \leq n/2$ be the range. The followings hold for $\mathcal{L}_{k,r}^{\text{hg}} = \{\mathcal{L}_n\}_{n \geq 1}$:*

1. $\mathcal{L}_{k,r}^{\text{hg}}$ has output size $\Theta(n/k)$ and seed length $O(r \log n)$.
2. Every $L \in \mathcal{L}_n$ can be computed by depth-1 $\text{CC}^0[2]$ circuits of wire complexity at most $2n$, or B_2 circuits of size $2n$ and depth $\log k + O(1)$.
3. $\mathcal{L}_{k,r}^{\text{hg}}$ is a randomized 1-detector with range r and error $\exp(-\Omega(\log^2 n / \log k))$.

Proof. Recall that $\gamma = \log^2 n / \log k$ from Construction 2. For the first item, the output size follows directly from the definition of $\mathcal{L}_{k,r}^{\text{hg}}$. The seed length of $\mathcal{L}_{k,r}^{\text{hg}}$ equals the seed length of the required r -wise $2^{-\gamma}$ -dependent permutation, which is $O(r \log n + \gamma) = O(r \log n)$ from Theorem 2.8 (note that $r \geq \log n$). The second item follows directly from the definition of L_σ in Construction 2 together with the fact that $G_{m,n'}$ has maximum degree $O(k)$ (since it is regular).

In the rest of the proof we establish the third item. We need to show that for any non-zero $x \in \mathbb{F}_2^n$, the probability that $L(x) = 0$ with $L \leftarrow \mathcal{L}_n$ is at most $\exp(-\Omega(\gamma))$. For any input x with Hamming weight $0 < |x| < g$ (where g is the girth of $G_{m,n}$), we must have $L_\sigma(x) \neq 0$ for all $\sigma \in \mathcal{D}_{n'}$, since otherwise we can extract from the edges $\{\sigma(i) \mid x_i = 1 \wedge i \in [n]\}$ a cycle of size length less than g . Now we consider the case when $g \leq |x| \leq r$. Let $i_1, i_2, \dots, i_{|x|}$ be the indices corresponding to the 1's in x (i.e., $x_{i_j} = 1$ for all j). Since our construction is linear, only these bits can influence the output. Also, by the r -wise $2^{-\gamma}$ -dependence of $\mathcal{D}_{n'}$, we have

$$\begin{aligned}
\Pr_{\sigma \leftarrow D_n} [L_\sigma(x) = 0] &= \Pr_{\sigma \leftarrow D_{n'}} \left[\bigwedge_{i' \in [m]} [|\{i_j \mid \sigma(i_j) \in \Gamma(i')\}| \text{ is even}] \right] \\
&\leq \Pr_{S \subseteq [n'], |S|=|x|} \left[\bigwedge_{i' \in [m]} [|S \cap \Gamma(i')| \text{ is even}] \right] + 2^{-\gamma} \\
&= \mathbb{E}_{\substack{S \subseteq [n'] \\ |S|=|x|-\alpha}} \left[\Pr_{\substack{S' \subseteq [n'] \setminus S \\ |S'|=\alpha}} \left[\bigwedge_{i' \in [m]} [(S \cup S') \cap \Gamma(i') \text{ is even}] \right] \right] + 2^{-\gamma}. \\
&\hspace{20em} (\alpha \triangleq \lceil \frac{g}{3} \rceil)
\end{aligned}$$

The final observation is that for any fixed S , there should be at most one S' satisfying the condition in the summation. Indeed, if two sets S'_1 and S'_2 satisfy the condition at the same time, then the symmetric difference of them contains a cycle in $G_{m,n'}$ of length $2\alpha < g$ for sufficiently large n , contradicting the fact that $G_{m,n'}$ has girth g . In particular, it means for every fixed S , we have

$$\Pr_{\substack{S' \subseteq [n'] \setminus S \\ |S'|=\alpha}} \left[\bigwedge_{i' \in [m]} [(S \cup S') \cap \Gamma(i') \text{ is even}] \right] \leq \frac{1}{\binom{n' - |x| + \alpha}{\alpha}}.$$

Since $\alpha = \lceil g/3 \rceil = \Theta(\log n / \log k)$, putting everything together, for sufficiently large n we have

$$\Pr_{\sigma \leftarrow D_n} [L_\sigma(x) = 0] \leq \frac{1}{\binom{n' - |x| + \alpha}{\alpha}} + 2^{-\gamma} = \exp(-\Omega(\gamma)). \quad \blacktriangleleft$$

Plugging \mathcal{L}^{hg} into Construction 1 with appropriately chosen parameters, we immediately obtain a linear hash \mathcal{H}^{md} with poly-logarithmic seed length and shrinkage as follows. (Here md stands for moderate, meaning that the hash has moderate (poly-logarithmic) shrinkage.)

► **Construction 3** (Construction of hash \mathcal{H}^{md} with poly-logarithmic shrinkage). *Let $\beta \in (0, 2]$ be a constant and $\omega(\log n) \leq \delta(n) \leq O(\frac{\log^2 n}{\log \log n})$ be the error parameter. $\mathcal{H}_{\beta, \delta}^{\text{md}} = \{\mathcal{H}_n\}_{n \geq 1}$ is constructed as follows.*

- Setting $k = \log^\beta n$ and $r = \delta \log^\beta n$, $\mathcal{L}^{\text{hg}} = \mathcal{L}_{k,r}^{\text{hg}}$ (from Construction 2) is an $O(\delta \log^{\beta+1} n)$ -samplable $\Theta(n / \log^\beta n)$ -output randomized 1-detector with range r and error $\exp(-\Omega(\delta))$.
- Setting $b = n / \log^\beta n$, \mathcal{H}^{md} is now defined to be $\mathcal{H}_b^{\mathcal{L}^{\text{hg}}}$ (from Construction 1).

The following theorem follows directly from the Construction 3, Lemma 3.2, and Lemma 3.3.

► **Theorem 3.4** (Properties of the intermediate hash construction \mathcal{H}^{md}). *Let $\beta \in (0, 2]$ be the shrinkage parameter and $\omega(\log n) \leq \delta(n) \leq O(\frac{\log^2 n}{\log \log n})$ be the error parameter. The followings hold for $\mathcal{H}_{\beta, \delta}^{\text{md}} = \{\mathcal{H}_n\}_{n \geq 1}$:*

1. $\mathcal{H}_{\beta, \delta}^{\text{md}}$ is a linear $O(\delta \log^{\beta+1} n)$ -samplable $\Theta(n / \log^\beta n)$ -output $\exp(-\Omega(\delta))$ -almost universal hash function.
2. Every $H \in \mathcal{H}_n$ can be computed either by a depth-1 $\text{CC}^0[2]$ circuit of wire complexity $2n$, or by a B_2 circuit of size $2n$ and depth $\beta \log \log n + O(1)$.

3.3 Shrinkage reduction of hash function

Now we reduce the output length of the hash function \mathcal{H}^{md} in Construction 3 from $n/\text{polylog}(n)$ to $\text{polylog}(n)$ with little overhead in its circuit complexity and seed length, which is crucial for our applications. The idea is simple: we composite it with a hash with large shrinkage, short seed length, and relatively larger circuit complexity. In particular, we can use the following hash construction $\mathcal{H}^{\text{expw}}$ of Chen, Jin, and Williams [9]. (Here expw stands for expander walk.)

► **Construction 4** (Hash function $\mathcal{H}^{\text{expw}}$ from expander walk [9]). *Let $\ell = \ell(n)$ be the output length. $\mathcal{H}_\ell^{\text{expw}} = \{\mathcal{H}_n\}_{n \geq 1}$ is constructed as follows.*

- *Let $\{S_n\}_{n \geq 1}$ be a family of 0.1-biased set from Theorem 2.4 and $\{G_n\}_{n \geq 1}$ be the family of strongly explicit expanders from Theorem 2.10. Assume that $S_n = \{w_0, w_1, \dots, w_{t-1}\}$ for $t = \tilde{O}(n^2)$.*
- *For each walk $v = (v_0, v_1, \dots, v_{\ell-1}) \in [t]^\ell$ of length ℓ on G_t , we define a hash function $h_v: \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ as*

$$h_v(x) \triangleq \langle w_{v_0}, x \rangle \| \langle w_{v_1}, x \rangle \| \dots \| \langle w_{v_{\ell-1}}, x \rangle,$$

where $\langle a, b \rangle$ denotes the inner product over \mathbb{F}_2 .

- *\mathcal{H}_n is then defined to be the distribution generated as follows: samples a random walk v of length ℓ on G_t uniformly at random, then outputs h_v .*

► **Lemma 3.5** (Properties of the hash construction $\mathcal{H}^{\text{expw}}$ from [9]). *Let $\ell = \ell(n)$ be the output length. The followings hold for $\mathcal{H}_\ell^{\text{expw}} = \{\mathcal{H}_n\}_{n \geq 1}$:*

1. *Every $H \in \text{supp}(\mathcal{H}_n)$ can implemented by a depth-1 $\text{CC}^0[2]$ circuit of wire complexity $n\ell$ or a B_2 circuit of size $n\ell$ and depth $\log n + O(1)$.*
2. *$\mathcal{H}_\ell^{\text{expw}}$ is a linear $O(\log n + \ell)$ -samplable ℓ -output $\exp(-\Omega(\ell))$ -almost universal hash function.*

Proof. The first item follows directly from the definition of h_v from Construction 4. In the following we establish the second item.

The linearity, seed length, and output length are straightforward to verify, thus we only analyze the collision probability. Since the hash function is linear, we only need to show that for any non-zero $x \in \mathbb{F}_2^n$, $\Pr_{h \leftarrow \mathcal{H}_n}[h(x) = 0] \leq \exp(-\Omega(\ell))$. By Theorem 2.4, we know that for any non-zero x , at least 0.4 fraction of vectors $w \in S_n$ satisfies $\langle w, x \rangle = 1$. Then by Theorem 2.11, a random walk of length ℓ will find one of such w with probability $1 - \exp(-\Omega(\ell))$. This immediately implies that $h(x) \neq 0$ with probability at most $\exp(-\Omega(\ell))$ for $h \leftarrow \mathcal{H}_n$. ◀

Next we formally define the composition of two hash function families.

► **Definition 3.6** (Composition of hash families). *Let $\mathcal{F} = \{\mathcal{F}_n\}_{n \geq 1}$ and $\mathcal{G} = \{\mathcal{G}_n\}_{n \geq 1}$ be two families of hash functions. The composition of \mathcal{F} and \mathcal{G} is defined as $\mathcal{F} \circ \mathcal{G} = \{(\mathcal{F} \circ \mathcal{G})_n\}_{n \geq 1}$, where $(\mathcal{F} \circ \mathcal{G})_n$ is the following distribution: let $m = m(n)$ be the output length of \mathcal{G} , we sample $f \leftarrow \mathcal{F}_m$ and $g \leftarrow \mathcal{G}_n$, and then output $f \circ g$.*

The following proposition is crucial for the analysis of our final hash construction.

► **Proposition 3.7.** *The composition $\mathcal{H}' \circ \mathcal{H}$ of an ε_1 -almost universal hash function \mathcal{H}' and an ε_2 -almost universal hash function \mathcal{H} is an $(\varepsilon_2(n) + \varepsilon_1(m(n)))$ -almost universal hash function, where $m(n)$ is the output length of \mathcal{H} . Moreover, $\mathcal{H}' \circ \mathcal{H}$ is linear if both of \mathcal{H}' and \mathcal{H} are linear.*

The following is a simple corollary of Proposition 3.7.

► **Corollary 3.8.** *For any $t \geq 2$, non-increasing $\varepsilon = \varepsilon(n)$, and output length parameter $\ell(n) \leq n$, the t^{th} order composition of an ℓ -output ε -almost universal hash \mathcal{H} with itself, denoted by \mathcal{H}^{ot} , is also a hash function with collision probability $t \cdot \varepsilon(\hat{\ell})$, where $\hat{\ell} = \ell \circ \ell \circ \dots \circ \ell(n)$ ($t - 1$ times) is the input of the outer-most hash. Moreover, \mathcal{H}^{ot} is linear if \mathcal{H} is linear.*

We are now ready to specify our final hash construction $\mathcal{H}^{\text{final}}$ with $\text{polylog}(n)$ output length.

► **Construction 5** (Construction of $\mathcal{H}^{\text{final}}$ with $\text{polylog}(n)$ output length). *Let $\beta \in (0, 2]$ be a parameter and $\omega(\log n) \leq \ell(n) \leq O(\frac{\log^2 n}{\log \log n})$ be a non-decreasing function. We define*

$$\mathcal{H}_{\beta, \ell}^{\text{final}} \triangleq \mathcal{H}_{\ell}^{\text{expw}} \circ (\mathcal{H}_{\beta, \ell}^{\text{md}})^{\circ \lceil \frac{2}{\beta} \rceil}.$$

We analyze the properties of $\mathcal{H}^{\text{final}}$ constructed above by combining the composition proposition (Proposition 3.7 and Corollary 3.8) together with properties of $\mathcal{H}^{\text{expw}}$ (Lemma 3.5) and \mathcal{H}^{md} (Theorem 3.4).

► **Theorem 3.9** (Properties of the final hash construction $\mathcal{H}^{\text{final}}$). *Let $\beta \in (0, 2]$ be a constant and $\omega(\log n) \leq \ell(n) \leq O(\frac{\log^2 n}{\log \log n})$ be a non-decreasing function. The followings hold for $\mathcal{H}_{\beta, \ell}^{\text{final}} = \{\mathcal{H}_n\}_{n \geq 1}$:*

1. $\mathcal{H}_{\beta, \ell}^{\text{final}}$ is a linear $O(\ell \log^{\beta+1} n)$ -samplable $\Theta(\hat{\ell})$ -output $\exp(-\Omega(\hat{\ell}))$ -almost universal hash function, where $\hat{\ell} = \ell(\Theta(n/\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n))$.
2. Every $H \in \mathcal{H}_n$ can be computed by depth- $(1 + \lceil \frac{2}{\beta} \rceil)$ $\text{CC}^0[2]$ circuits of wire complexity $2n + O\left(\frac{n\hat{\ell}}{\log^2 n}\right)$, or by B_2 circuits of size $2n + O\left(\frac{n\hat{\ell}}{\log^2 n}\right)$ and depth $\log n + O(1)$.

Proof. Let $\mathcal{H}^{\text{md}} = \mathcal{H}_{\beta, \ell}^{\text{md}}$ and $\mathcal{H}^{\text{expw}} = \mathcal{H}_{\ell}^{\text{expw}}$ be the hash functions described in Construction 5. We also use $\mathcal{H}_n^{\text{md}}$ and $\mathcal{H}_n^{\text{expw}}$ to denote the distributions over n -input hash functions in \mathcal{H}^{md} and $\mathcal{H}^{\text{expw}}$, respectively.

$\mathcal{H}^{\text{final}}$ is almost universal with $\text{polylog}(n)$ output length. Let \hat{m} be the output length of $(\mathcal{H}^{\text{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$, and $\hat{\ell} \triangleq \ell(\hat{m}) = \ell(\Theta(n/\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n))$. By a simple induction and Theorem 3.4 we know that for any $1 \leq d \leq \lceil \frac{2}{\beta} \rceil$, $(\mathcal{H}^{\text{md}})^{\circ d}$ is a linear $O(\ell \log^{\beta+1} n)$ -samplable $\Theta(n/\log^{\beta d} n)$ -output $\exp(-\Omega(\hat{\ell}))$ -almost universal hash function. In particular, when $d = \lceil \frac{2}{\beta} \rceil$, we know that $(\mathcal{H}^{\text{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$ is a linear $O(\ell \log^{\beta+1} n)$ -samplable $\Theta(n/\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n)$ -output $\exp(-\Omega(\hat{\ell}))$ -almost universal hash function.

Now apply Proposition 3.7 and Lemma 3.5, we know that $\mathcal{H}^{\text{final}} = \mathcal{H}^{\text{expw}} \circ (\mathcal{H}^{\text{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$ is a linear $O(\ell \log^{\beta+1} n)$ -samplable $\Theta(\hat{\ell})$ -output $\exp(-\Omega(\hat{\ell}))$ -almost universal hash function.

Complexity with $\text{CC}^0[2]$ circuits. By Theorem 3.4, every $H \in \text{supp}(\mathcal{H}_n^{\text{md}})$ can be computed by a depth-1 $\text{CC}^0[2]$ circuit with wire complexity $2n$. Also, by Lemma 3.5, every $H \in \text{supp}(\mathcal{H}_n^{\text{expw}})$ can be computed by a depth-1 $\text{CC}^0[2]$ circuit with wire complexity $n\ell$. Since the output length of $(\mathcal{H}^{\text{md}})^{\circ d}$ is $\Theta(n/\log^{\beta d} n)$, the total wire complexity of a hash function from the support of $\mathcal{H}^{\text{expw}} \circ (\mathcal{H}^{\text{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$ can be bounded by

$$2n + O\left(\sum_{d=1}^{\lceil \frac{2}{\beta} \rceil - 1} \frac{n}{\log^{\beta d} n}\right) + \frac{n}{\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n} \cdot \hat{\ell} = 2n + O\left(\frac{n\hat{\ell}}{\log^2 n}\right).$$

The depth complexity can be verified straightforwardly.

Complexity with B_2 circuits. We only analyze the depth since the analysis of the size complexity is similar to the analysis above. By Theorem 3.4 and Lemma 3.5, every $H \in \text{supp}(\mathcal{H}_n^{\text{md}})$ can be computed by a B_2 circuit with depth $\beta \log \log n + O(1)$, and every $H \in \text{supp}(\mathcal{H}_n^{\text{expw}})$ can be computed by a B_2 circuit with depth $\log n + O(1)$. Therefore the total depth of a hash function from the support of $\mathcal{H}^{\text{expw}} \circ (\mathcal{H}^{\text{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$ is at most

$$\sum_{d=0}^{\lceil \frac{2}{\beta} \rceil - 1} \left[\beta \log \log \left(\Theta \left(\frac{n}{\log^{\beta d} n} \right) \right) + O(1) \right] + \log \left(\Theta \left(\frac{n}{\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n} \right) \right) = \log n + O(1). \quad \blacktriangleleft$$

3.4 Explicitness of our construction

Apart from the seed length, shrinkage and collision probability, the *explicitness* of the a hash function is also critical for many applications. That is, given a seed v (which is usually much shorter than the input, say $|v| = \text{polylog}(n)$), whether we can obtain information about the hash function corresponding to the seed v efficiently. In this section we show that our hash constructions are indeed explicit in a strong sense, which is crucial for our application to hardness magnification.

► **Definition 3.10** (Locally explicit hash function). *A family of $\text{polylog}(n)$ -samplable distributions over m -output linear functions (e.g., linear hash functions or 1-detectors) $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$ is said to be locally explicit if each input bit only influences $\text{polylog}(n)$ output bits, and there exists an algorithm $\mathcal{A}(n, v, i)$ running in $\text{polylog}(n)$ time that returns the list of output bits influenced by the i^{th} input bit in the hash function corresponding to the seed v .*

Note that any linear function can be realized as $x \mapsto \mathbf{M}x$ for some transformation matrix \mathbf{M} over \mathbb{F}_2 . Clearly, a locally explicit linear hash has a sparse transformation matrix, and we can efficiently list all 1-entries in any column of it. By performing a sparse matrix multiplication, we can immediately obtain the following proposition.

► **Proposition 3.11.** *The composition of two locally explicit hash families is locally explicit.*

Now we verify that our constructions of the randomness-efficient low-complexity hash functions are indeed locally explicit.

► **Theorem 3.12.** *Let $k = k(n) = n^{o(1)}$ be the shrinkage parameter and $\log n \leq r(n) \leq n/2$ be the range. The 1-detector $\mathcal{L}_{k,r}^{\text{hg}}$ in Construction 2 is locally explicit.*

Proof. Recall that the 1-detector in Construction 2 consists of two components: a depth-1 $\text{CC}^0[2]$ circuit whose topology is determined by the high-girth graph from Theorem 2.12, and a random permutation of the input bits according to the k -wise almost independent permutation from Theorem 2.8. Given the seed v of the 1-detector (i.e., the seed for the permutation) and an index i . According to Theorem 2.8, we can obtain $\sigma_v(i)$ in $\text{poly}(\log n, |v|)$ time, where σ_v is the permutation corresponding to the seed v . To find all the output bits influenced by the i^{th} input bit, we only need to find the two endpoints of the $\sigma_v(i)$ -th edge in the high-girth graph, which can be done in $\text{polylog}(n)$ time by Theorem 2.12. \blacktriangleleft

► **Theorem 3.13.** *Let $\beta > 0$ be the shrinkage parameter and $\delta = \delta(n) \leq O(\frac{\log^2 n}{\log \log n})$ be the error parameter. The hash family $\mathcal{H}_{\beta,\delta}^{\text{md}}$ in Construction 3 is locally explicit.*

Proof. Recall that the hash function \mathcal{H} in Construction 3 is obtained by the 1-detector \mathcal{L}^{hg} in Construction 2 and the reduction in Construction 1. The output of the hash $\mathcal{H}_{\beta,\delta}^{\text{md}}$ consists of two parts: the output of the 1-detector \mathcal{L}^{hg} , and several randomly sampled bits from the

input. Since \mathcal{L}^{hg} is locally explicit by Theorem 3.12, it is sufficient to show that the sampling procedure is locally explicit. More precisely, given a seed v of the sampling procedure and an index i , we need to compute efficiently all the indices j such that $w_j = i$ for the vector (w_0, \dots, w_{b-1}) corresponding to the seed v . This can be done using the algorithm $\mathcal{A}^{\text{samp}}$ from Lemma 3.1. \blacktriangleleft

► **Theorem 3.14.** *Let $\ell = \ell(n) = \text{polylog}(n)$. The hash family $\mathcal{H}_\ell^{\text{expw}}$ in Construction 4 is locally explicit.*

Proof. Let $S_n = \{w_0, \dots, w_{t-1}\}$ with $t = \tilde{O}(n^2)$ is a 0.1-biased set from Theorem 2.4 and G_t is the expander in Theorem 2.10. The hash function in Construction 4 is defined as

$$h_v(x) \triangleq \langle w_{v_0}, x \rangle \parallel \langle w_{v_1}, x \rangle \parallel \dots \parallel \langle w_{v_{\ell-1}}, x \rangle,$$

where $v = (v_0, \dots, v_{\ell-1}) \in [t]^\ell$ is a random walk on G_t . Given the seed v (i.e., the seed of a random walk), we can produce $v_0, v_1, \dots, v_{\ell-1}$ in $\text{polylog}(n, \ell) = \text{polylog}(n)$ time by Theorem 2.10. Then using the algorithm \mathcal{A} in Theorem 2.4 we can check for each j whether the i^{th} bit of w_{v_j} is 1, which indicates whether the j^{th} output bit is influenced by the i^{th} input bit. Enumerating all $j \in [\ell]$ gives a required $\text{polylog}(n)$ time algorithm listing all the outputs influenced by a particular input bit. \blacktriangleleft

Using Theorem 3.13, Theorem 3.14, and Proposition 3.11, we immediately obtain the explicitness of Construction 5.

► **Corollary 3.15.** *Let $\beta \in (0, 2]$ be a parameter and $\ell = \ell(n) \leq O(\frac{\log^2 n}{\log \log n})$ be a non-decreasing function. The hash family $\mathcal{H}_{\beta, \ell}^{\text{final}}$ in Construction 5 is locally explicit.*

3.5 Uniformity of our construction

We have shown in Theorem 3.9 that our hash function can be computed by extremely sparse $\text{CC}^0[2]$ circuit or B_2 circuits with small size and depth simultaneously. It remains to clarify the *uniformity* of our hash construction, i.e., the complexity we need to construct the circuit given the input length and the seed.

The definitions of uniformity vary with respect to the complexity measures, which can be chosen according to the applications. Typical choices include space complexity (e.g., LOGSPACE-uniform, see Section 6.2.1 of [3]), time complexity (e.g., polynomial-time uniform), parallel time (see, e.g., Section 3 of [29]), and descriptive complexity (see, e.g., [20]). If we choose the time complexity, for instance, we can define P-uniformity as the existence of the polynomial-time algorithm that prints the circuit given 1^n and the seed v . Indeed, it is easy to check that the $2n + o(n)$ size B_2 and $\text{CC}^0[2]$ circuits for Construction 5 in Theorem 3.9 is P-uniform.

Since our hash function can be evaluated in low-depth circuit models such as NC^1 and $\text{CC}^0[2]$, the parallel time to generate the circuit also seems to be crucial for its further applications. Therefore in this section we will discuss the POLYLOGTIME-uniformity of our hash functions, defined as follows.

► **Definition 3.16.** *Let \mathcal{C} be a circuit class. A family of $\text{polylog}(n)$ -samplable distributions over m -output functions (e.g., hash functions or 1-detectors) $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$ is said to be computable with POLYLOGTIME-uniform $S(n)$ -size (measured in size, depth, wire complexity, etc) \mathcal{C} -circuits if each of the functions $h_v \in \text{supp}(\mathcal{H}_n)$ corresponding to the seed v is computable by an $S(n)$ -size \mathcal{C} -circuit, supplemented with the following $\text{polylog}(S(n))$ -time algorithms:*

SIZE(n, v) returns the size of the \mathcal{C} -circuit C_v computing h_v ;

TYPE(n, v, i) returns the type of the i^{th} node in the circuit C_v , which indicates (1) whether it is a gate, an input variable or a constant; (2) whether it is an output node; and (3) the type (if it is a gate or a constant) or index (if it is an input variable and/or output node) of it;

EDGE(n, v, i, j) returns the j^{th} input of the i^{th} node if the i^{th} node is a gate in C_v .

OUT(n, v, i) returns the number of the i^{th} output node.

It is guaranteed that the nodes are numbered in topological orders, and in particular, the input variables are numbered from 1 to n .

We claim that all our results of circuit complexity for the constructions in the previous parts of this section are in fact POLYLOGTIME-uniform. It is straightforward to check the claim, so we only sketch the proof and left the details to the readers.

► **Theorem 3.17** (Uniformity of Theorem 3.9). *Let $\beta \in (0, 2]$ be a parameter and $\ell = \ell(n) \leq O(\frac{\log^2 n}{\log \log n})$ be a non-decreasing function. The hash function $\mathcal{H}_{\beta, \ell}^{\text{final}}$ in Construction 5 can be computed either by POLYLOGTIME-uniform depth- $(1 + \lceil \frac{2}{\beta} \rceil)$ $\text{CC}^0[2]$ circuits of wire complexity $2n + O(n/\log \log n)$, or by POLYLOGTIME-uniform B_2 circuits of size $2n + O(n/\log \log n)$ and depth $\log n + O(\beta^{-1})$ simultaneously.*

Sketch. Since the POLYLOGTIME-uniformity is close under composition, we only need to verify that Construction 3 and Construction 4 are POLYLOGTIME-uniform. The uniformity of Construction 3 directly follows from the algorithm \mathcal{B} in Theorem 2.12, the algorithm \mathcal{A} in Theorem 2.8, and the algorithm \mathcal{A} in Lemma 3.1. The uniformity of Construction 4 follows from the algorithm \mathcal{A} in Theorem 2.4 and the algorithm \mathcal{A} in Theorem 2.10. Note that the circuit for Construction 4 should be slightly modified to make $\text{EDGE}(n, \cdot, \cdot)$ polylog(n)-time computable. Take the $\text{CC}^0[2]$ case for example: each of the output gates has fan-in *exactly* n (instead of $|w_{v_i}|$ for the i^{th} output bit, see Construction 4 for the notation); the j^{th} input wire of the i^{th} output gate is connected to the j^{th} input or a constant 0 according to the j^{th} bit of w_{v_i} . ◀

4 Sharp bootstrapping results from hash functions

We now show the extremely sharp bootstrapping results for small linear size circuits based on the almost universal hash function constructed above, using a refined *kernelization method* of Chen, Jin, and Williams [9, 10].

We will first prove the general hardness magnification theorem for all sparse NP languages in Section 4.1. In Section 4.2, we will present stronger hardness magnification results for MCSP, which will utilize the explicitness of our hash construction. Then we will show in Section 4.3 that similar techniques can be applied to obtain a bootstrapping result for *explicit obstructions* (see, e.g., [10]), which formalizes the *explicit* proofs of circuit lower bounds. In Section 4.4, we construct explicit obstructions and prove circuit lower bounds that tightly match these bootstrapping results.

4.1 Hardness magnification for all sparse NP languages

We first prove the most general version of the hardness magnification result. More discussions are presented after the proof.

► **Theorem 4.1.** *Let $s = s(n)$ and $T = T(n)$ be two functions such that*

- $\omega(\log n) \leq s(n) \leq O(\log^2 n / \log \log n)$, $s(\Theta(n / \log^2 n)) = \Theta(s(n))$, s is non-decreasing;
- $n^\gamma \leq T(n) \leq 2^{O(n)}$, where $\gamma > 1$ is an absolute large constant.

Then, if there is a $2^{s(n)}$ -sparse language L in $\text{NTIME}[T(n)]$, such that L cannot be computed by probabilistic circuits of size $2n + O(ns / \log^2 n)$ within error $\exp(-\Omega(s))$, it then follows that $\text{NTIME}\left[T\left(2^{O(n^{1/5})}\right)\right] \not\subseteq \text{SIZE}\left[2^{cn^{1/5}}\right]$ for all $c > 1$.

Proof. Towards a contradiction we assume that $\text{NTIME}\left[T\left(2^{O(n^{1/5})}\right)\right] \subseteq \text{SIZE}\left[2^{cn^{1/5}}\right]$ for some $c > 1$. Let L be a $2^{s(n)}$ -sparse language in $\text{NTIME}[T(n)]$. We now show that L can be computed by probabilistic circuits of size $2n + O(ns / \log^2 n)$ within error $\exp(-\Omega(s))$.

Let $\mathcal{H} = \mathcal{H}_{\beta, \ell}^{\text{final}} = \{\mathcal{H}_n\}_{n \geq 1}$ be the linear $O(\ell \log^{1+\beta} n)$ -samplable $\Theta(\ell)$ -output $\exp(-\Omega(\ell))$ -almost universal hash function in Theorem 3.9 with $\ell(n) = O(s)$ and $\beta = 2$, such that the collision probability is at most 2^{-2s} ¹⁷. The circuit complexity of each $h \in \text{supp}(\mathcal{H}_n)$ is bounded by $2n + O(ns / \log^2 n)$. Since \mathcal{H} is efficiently samplable, there is a polynomial time algorithm $M(1^n, v, \cdot)$ computing the hash function corresponding to the seed v , where $|v| = O(\ell \log^3 n) = O(s \log^3 n)$.¹⁸

We now define an intermediate language $L' = \{(n, v, h) \mid \exists x \in \{0, 1\}^n \cap L, M(1^n, v, x) = h\}$. For any sufficiently large n , we pad the language to make the corresponding length of the tuple in L' has length exactly $m = m(n) = \lfloor (\log(n) / (2c))^5 \rfloor$ to obtain L'' . That is, for sufficiently large n and v, h of appropriate length, we define

$$z_{n,v,h} \triangleq (n, v, h) \parallel 1 \parallel 0^{m(n) - |(n,v,h)| - 1}$$

and we have

$$z_{n,v,h} \in L'' \Leftrightarrow (n, v, h) \in L'.$$

Using the straightforward non-deterministic algorithm for L' (guessing $x \in \{0, 1\}^n$ and determine whether $M(1^n, v, x) = h$), we can show that $L'' \in \text{NTIME}\left[T\left(2^{O(m^{1/5})}\right)\right] \subseteq \text{SIZE}\left[2^{cm^{1/5}}\right]$, the second containment follows from our assumption.

The key in our argument is an algorithm for the sparse language L with an oracle access to this intermediate problem L'' , and then replace the oracle with the small size circuit by assumption. This is formalized in the following lemma.

► **Lemma 4.2.** *There is a uniform family $\mathcal{D} = \{\mathcal{D}_n\}_{n \geq 1}$ of probabilistic oracle circuits of size $2n + O(ns / \log^2 n)$, such that for every input length n , the followings hold:*

1. *Every $D \in \text{supp}(\mathcal{D}_n)$ contains at most one L'' oracle gate of fan-in $\lfloor \log^5(n) / (32c^5) \rfloor$.*
2. *\mathcal{D} decides the language L with error at most 2^{-s} .*

Proof. Consider the following probabilistic circuit family $\mathcal{D} = \{\mathcal{D}_n\}_{n \geq 1}$. Given input x of length n , \mathcal{D}_n is constructed as follows: we randomly choose a seed v to sample a hash function h_v from \mathcal{H}_n ; and then query whether $z_{n,v,h_v(x)}$ is in L'' via the oracle gate, where h_v is the hash function corresponding to the seed v . The circuit complexity of \mathcal{D}_n equals the circuit complexity of the hash function h_v , which is at most $2n + O(ns / \log^2 n)$ by

¹⁷In particular, assume that α is a constant such that \mathcal{H} is $\exp(-\alpha\ell)$ -almost universal hash function, then we take $\ell(n) = \lceil 2s/\alpha \rceil$.

¹⁸That is, for every $x \in \{0, 1\}^n$ and every seed v , we have $M(1^n, v, x) = h_v(x)$, where h_v is the hash function corresponding to the seed v .

Theorem 3.9, and every $D \in \text{supp}(\mathcal{D}_n)$ only needs to call the L'' oracle once with input length $m = \lceil \log^5(n)/(32c^5) \rceil$. Clearly, by the definition of L'' , for any $x \in L$, \mathcal{D}_n accepts x with probability 1. For any $x \notin L$, we have

$$\begin{aligned} & \Pr_v[\mathcal{D}_n \text{ accepts } x] \\ &= \Pr_{h \leftarrow \mathcal{H}_n}[\exists x' \in L \cap \{0,1\}^n \text{ s.t. } h(x) = h(x')] \\ &\leq \sum_{x' \in L \cap \{0,1\}^n} \Pr_{h \leftarrow \mathcal{H}_n}[h(x) = h(x')] \\ &= 2^s \cdot 2^{-2s} \\ &\leq 2^{-s}. \end{aligned} \quad \blacktriangleleft$$

Now applying Lemma 4.2 and replacing the oracle query by circuits using the fact that $L'' \in \text{SIZE}[2^{cn^{1/5}}]$, we finish the proof of Theorem 4.1. \blacktriangleleft

We now discuss some typical choices of parameters in the above theorem. Let the sparsity parameter $s(n) = \frac{\log^2 n}{\log \log n}$.¹⁹

- A standard form of hardness magnification result similar to the one in [9] but quantitatively stronger can be obtained by choosing $T(n) = \text{poly}(n)$. Then the theorem says that, if there exists an $n^{\log n / \log \log n}$ -sparse language in NP that does not have probabilistic circuits of size $2n + O(n/\log \log n)$, then $\text{NTIME}[2^{O(n^{1/5})}] \not\subseteq \text{SIZE}[2^{cn^{1/5}}]$ for all $c > 0$. By a padding argument, this implies that $\text{NP} \not\subseteq \text{SIZE}[n^c]$ for all $c > 0$. This establishes Theorem 1.2.
- Being less ambitious, we can let $T(n) = 2^{n^{o(1)}}$ and show that, even the existence of an $n^{\log n / \log \log n}$ -sparse language in $\text{NTIME}[2^{n^{o(1)}}]$ that does not have probabilistic circuits of size $2n + O(n/\log \log n)$, would be enough to imply that $\text{NTIME}[2^{2^{o(n^{1/5})}}] \not\subseteq \text{SIZE}[2^{cn^{1/5}}]$, and hence $\text{NTIME}[2^{n^{o(1)}}] \not\subseteq \text{SIZE}[n^c]$ for all $c > 0$ and $\text{NEXP} \not\subseteq \text{P}_{/\text{poly}}$, which would already be a major breakthrough in circuit complexity. This establishes Theorem 1.3.

4.2 Hardness magnification for MCSP

We now utilized the strongly explicit hash function constructed above to obtain a stronger hardness magnification theorem for MCSP. Intuitively, this is possible since the yes-instances of MCSP can be efficiently encoded in a much shorter string. Indeed, for $\text{MCSP}[s(n)]$, one only need $O(s(n) \log s(n))$ bits to encode a small circuit, instead of 2^n bits for a whole truth table.

We assume a paddable encoding of circuits, i.e., any string x and $x||0$ encode the same circuit. This can be done with only a constant overhead. For a circuit C , we use the notation $\langle C \rangle$ to denote its encoding.

► **Theorem 4.3.** *Let $n \leq s(n) \leq O(n^2/\log^2 n)$ be a non-decreasing size parameter that satisfies $s(\log(\Theta(n/\log^2 n))) = \Theta(s(\log n))$. Let $N = 2^n$ be the truth table length. Let $g = g(n) = s(n) \log s(n)$. If $\text{MCSP}[s(n)]$ cannot be computed by probabilistic circuits of size $2N + O(Ng/\log^2 N)$ within error $\exp(-\Omega(g))$, then there is some $c \in (0, 1)$ such that $\oplus\text{P} \not\subseteq \text{SIZE}[2^{N^c}]$.*

¹⁹ For a typical $2^{s(n)}$ -sparse language in NP, consider $\text{MCSP}[n^{1.9}]$ on input length $N = 2^n$.

Intuition. The proof is similar to the one for Theorem 4.1, but we need to make the oracle from Lemma 3.1 computable in $\oplus\text{P}$ instead of in super-polynomial non-deterministic time. Concretely, we need to slightly modify the intermediate problem so that we can make use of the strongly explicitness of our hash construction $\mathcal{H}^{\text{final}}$.

Proof. Towards a contradiction, suppose that $\oplus\text{P} \subseteq \text{SIZE}[2^{N^c}]$ for all $c \in (0, 1)$, we now prove that $\text{MCSP}[s(n)]$ can be computed by probabilistic circuits of size $2N + O(Ng/\log^2 N)$ within error $2^{-\Omega(g(n))}$.

Let $N = 2^n$ be the truth table length of n -input functions. Take $\mathcal{H} = \mathcal{H}_{\beta, \ell}^{\text{final}} = \{\mathcal{H}_N\}_{N \geq 1}$ to be the linear $O(\ell \log^{1+\beta} N)$ -samplable $\Theta(\ell)$ -output $\exp(-\Omega(\ell))$ -almost universal hash function in Theorem 3.9 with $\ell(N) = O(g(n)) = O(g(\log N))$ and $\beta = 2$, such that the collision probability is at most 2^{-2g} .²⁰

We denote the hash function of input length N corresponding to the seed v as $H_{N,v}(\cdot)$. By the strongly explicitness (Corollary 3.15), there exists a algorithm $M(N, v, i, j)$ running in $\text{polylog}(N) = \text{poly}(n)$ time that decides whether the i^{th} output bit depends on the j^{th} input bit in $H_{N,v}$. Note that for a circuit C of input length n ,²¹

$$H_{N,v}(\text{tt}(C))_i = \sum_{j=0}^{N-1} M(N, v, i, j) \cdot C(j) \pmod{2},$$

and $M(N, v, i, j) \cdot C(j)$ can be computed in $\text{poly}(n, |C|)$ time, so the language $L^h = \{(N, i, v, \langle C \rangle) \mid H_{N,v}(\text{tt}(C))_i = 1\}$ is decidable in $\oplus\text{P}$. Hence if we define an intermediate language $L' = \{(N, v, h) \mid \exists C \in \text{SIZE}[s(n)], H_{N,v}(\text{tt}(C)) = h\}$ similar to the proof of Theorem 4.1, it is now decidable in NP given oracle access to $\oplus\text{P}$.

Let $m = O(\ell \log^3 n)$ be the input length of L' . Note that $L' \in \text{NP}^{\oplus\text{P}} \subseteq \text{BPP}^{\oplus\text{P}} \subseteq \text{P}_{/\text{poly}}^{\oplus\text{P}} \subseteq \oplus\text{P}_{/\text{poly}}$, where $\text{NP}^{\oplus\text{P}} \subseteq \text{BPP}^{\oplus\text{P}}$ follows from [42] (see also [15]) and $\text{P}_{/\text{poly}}^{\oplus\text{P}} \subseteq \oplus\text{P}_{/\text{poly}}$ follows from $\oplus\text{P}^{\oplus\text{P}} \subseteq \oplus\text{P}$ [37]²² Since $\oplus\text{P} \subseteq [2^{N^c}]$ for all $c \in (0, 1)$ by the assumption, we know that the evaluation of $\oplus\text{P}_{/\text{poly}}$ circuits can be computable in $\text{SIZE}[2^{N^c}]$ for all $c \in (0, 1)$, which implies that $L' \in \text{SIZE}[\sqrt{N}]$. The rest of the proof follows similar to Theorem 4.1 and the fact that the $m = O(\ell \log^3 N) = o(\log^5 N)$. \blacktriangleleft

4.3 Explicit obstruction

We now formally state and prove our results regarding explicit obstructions. We begin by formally defining the notion of *explicit obstructions*.

► **Definition 4.4** (Explicit obstruction). *An explicit obstruction of size $S(n)$ computable in \mathcal{C} against \mathcal{D} is a family of lists of input-output pairs $\mathcal{O} = \{\mathcal{O}_n\}_{n \geq 1}$ satisfying the following.*

- $|\mathcal{O}_n| \leq S(n)$ for all sufficiently large n .
- There is a machine in \mathcal{C} that prints the set \mathcal{O}_n given input 1^n .
- For every n , $\mathcal{O}_n = \{(x_i, y_i)\}$ satisfies that $x_i \neq x_j$ for all $i \neq j$.
- For all sufficiently large n , and for every n -input \mathcal{D} circuit f , there is a pair $(x_i, y_i) \in \mathcal{O}_n$ such that $f(x_i) \neq y_i$.

²⁰ Note that since $s(n) \leq O(n^2/\log^2 n)$, we have $\ell(N) = O(\log^2 N/\log \log N)$ and ℓ is non-decreasing. Hence, ℓ satisfies the requirements in Theorem 3.9.

²¹ Here we use the notation $\text{tt}(C)$ to represent the truth table of C . Particularly, if C is a single-output circuit of n inputs, then $\text{tt}(C)$ is a string of length 2^n such that $\text{tt}(C)_i = C(i)$.

²² Since $\oplus\text{P}^{\oplus\text{P}} \subseteq \oplus\text{P}$ implies that $\text{P}_{/\text{poly}}^{\oplus\text{P}} \subseteq \oplus\text{P}$, it follows that the evaluation of polynomial-size circuits with $\oplus\text{P}$ oracles can be computable in $\oplus\text{P}$, which further implies that $\text{P}_{/\text{poly}}^{\oplus\text{P}} \subseteq \oplus\text{P}_{/\text{poly}}$.

► **Theorem 4.5.** *There is an absolute constant $\gamma > 0$ such that the following holds. Let $n^\gamma \leq T(n) \leq 2^n$ and $\log n \leq s(n) \leq \min\{O(\log^2 n / \log \log n), \log T(n)\}$ being non-decreasing and satisfying $s(\Theta(n/\log^2 n)) = \Theta(s(n))$. If there is an explicit obstruction of size $2^{s(n)}$ computable in $\text{DTIME}[T(n)]$ against $2n + O(n/\log \log n)$ -size circuits, then $\text{DTIME}\left[T\left(2^{O(n^{1/5})}\right)\right] \not\subseteq \text{SIZE}\left[2^{cn^{1/5}}\right]$ for all $c > 1$.*

Proof. Let $t(n) = 2^{s(n)}$. Towards a contradiction we assume that there is some constant $c > 1$, such that $\text{DTIME}\left[T\left(2^{O(n^{1/5})}\right)\right] \subseteq \text{SIZE}\left[2^{cn^{1/5}}\right]$. Suppose that $\mathcal{O} = \{\mathcal{O}_n\}_{n \geq 1}$ is an explicit obstruction against $2n + O(n/\log \log n)$ size circuits. For any sufficiently large input length n , we suppose that $\mathcal{O}_n = \{(x_{n,1}, y_{n,1}), (x_{n,2}, y_{n,2}), \dots, (x_{n,t(n)}, y_{n,t(n)})\}$. Our goal is to design a circuit with extremely small size, but agree with \mathcal{O}_n on all its input-output pairs. Following the similar proof outline as for Theorem 4.1, we begin by using an almost universal hash function to kernalize the inputs from \mathcal{O}_n .

Let $\mathcal{H} = \mathcal{H}_{\ell, \beta}^{\text{final}} = \{\mathcal{H}_n\}_{n \geq 1}$ be the linear $O(\ell \log^{1+\beta} n)$ -samplable $\Theta(\ell)$ -output $\exp(-\Omega(\ell))$ -almost universal hash function in Theorem 3.9 with $\ell(n) = O(s)$ and $\beta = 2$ such that the collision probability is at most 2^{-3s} . Let n be a sufficiently large input length. We call a hash function h *good* if it is perfect on the inputs in \mathcal{O}_n , i.e., any two distinct inputs in \mathcal{O}_n have different hash values assigned by h . For a randomly chosen function $h \leftarrow \mathcal{H}_n$, the probability of it not being good is bounded by

$$\begin{aligned} & \Pr_{h \leftarrow \mathcal{H}_n} [\exists 1 \leq i < j \leq t(n) \text{ s.t. } h(x_{n,i}) = h(x_{n,j})] \\ & \leq \sum_{1 \leq i < j \leq t(n)} \Pr_{h \leftarrow \mathcal{H}_n} [h(x_{n,i}) = h(x_{n,j})] \\ & \leq \binom{2^s}{2} 2^{-3s} \leq 2^{-s}. \end{aligned}$$

So a good hash function always exists in the support of \mathcal{H}_n for all sufficiently large n . For any large input length n , we arbitrarily fix such a good hash and denote its seed by v_n^{good} .

Let h_v be the hash function corresponding to the seed v . We define an intermediate language $L' = \{(n, v, h) \mid \exists 1 \leq i \leq t(n), y_{n,i} = 1 \wedge h_v(x_{n,i}) = h\}$. We again pad its input to have length exactly $m = \lfloor (\log(n)/(2c))^5 \rfloor$, and form a padded language L'' . Then $L'' \in \text{DTIME}\left[T\left(2^{O(m^{1/5})}\right)\right] \subseteq \text{SIZE}\left[2^{cm^{1/5}}\right]$. We also note that $2^{cm^{1/5}} \leq \sqrt{n}$.

Then we consider the function $f_n(x) \triangleq L'(n, v_n^{\text{good}}, H_n(v_n^{\text{good}}, x))$, where $H_n(v, \cdot)$ is the hash function in \mathcal{H}_n corresponding to the seed v . By an argument similar to that of Theorem 4.1, we can show that f_n can be decided by a circuit of size $2n + O(n/\log \log n) + \sqrt{n} = 2n + O(n/\log \log n)$, but totally agrees with \mathcal{O}_n , contradicting to the assumption that \mathcal{O} is an explicit obstruction against circuits of such size. ◀

4.4 Unconditional lower bounds for sparse languages

Now we complement the results mentioned in previous subsections by constructing an explicit obstruction against B_2 circuits of size $2n - O(1)$ and a corresponding sparse language in \mathbb{P} with a $2n - O(1)$ probabilistic circuit lower bound. They form sharp bootstrapping thresholds together with Theorem 4.1 and Theorem 4.5.

The main idea behind our explicit obstruction is the investigation of a combinatorial structure in Boolean circuits called *critical path* introduced in [13], which was used to prove $2n - O(1)$ circuit lower bounds for PRFs and hash functions.

For the simplicity of presentation, we assume without loss of generality that our circuits are *normalized*, in the sense that there is no non-output gates with out-degree 0. This is without loss of generality since redundant gates with out-degree 0 can be removed.

► **Definition 4.6** (Critical path). *Let C be a circuit, and u be an input variable of it. The critical path of u in C is a sequence of vertices v_0, v_1, \dots, v_k satisfying the following conditions:*

1. $v_0 = u$, and v_i is a descendent of v_{i-1} for all $i \geq 1$, and
2. $\text{out-degree}(v_i) = 1$ for all $0 \leq i < k$, and $\text{out-degree}(v_k) \neq 1$.

Fix a circuit with n input bits, we can obtain a total of n critical paths. The crucial observation in [13] is that, if all critical paths do not intersect with one another, then the circuit must contain at least $2n - O(1)$ gates. This is formalized below.

► **Lemma 4.7** ([13], Lemma 6.4). *For any normalized n -input single-output circuit C with no intersecting critical paths and no input variables with out-degree 0, the number of gates in the circuit is at least $2n - 2$.*

Let $\mathcal{O} = \{\mathcal{O}_n \subseteq \{0, 1\}^n \times \{0, 1\}\}_{n \geq 1}$, where

$$\mathcal{O}_n = \{(x, 0) \mid |x| \in \{0, 2, n-2, n-1\}\} \cup \{(x, 1) \mid |x| \in \{1, n\}\}.$$

In the remaining part of the section, we will first prove that \mathcal{O} is an explicit obstruction against $2n - 2$ size circuits, and then present a general connection between explicit obstruction and probabilistic circuit lower bounds.

Explicit obstruction. According to Lemma 4.7, we only need to prove that any circuit with intersecting critical paths or input variables with out-degree 0 does not fully agree with \mathcal{O} . For circuits with input variables of out-degree 0, this can be verified straightforwardly. Now we prove the case for circuits with intersecting critical paths.

► **Lemma 4.8.** *For any circuit $C_n : \{0, 1\}^n \rightarrow \{0, 1\}$ with intersecting critical paths, there must exist a pair $(x, b) \in \mathcal{O}_n$ such that $C_n(x) \neq b$.*

Proof. Suppose that the critical paths of u and v in C_n intersect. Let G be the first gate on the intersection. Let f_G be the function computed by G . Assume that we take a restriction ρ to all variables except u and v , and consider the restricted function $C_n \upharpoonright_\rho$. The key observation is that, for any ρ , there are functions $\phi_\rho, \psi_\rho, \chi_\rho : \{0, 1\} \rightarrow \{0, 1\}$ such that

$$C_n \upharpoonright_\rho(u, v) = \chi_\rho(f_G(\phi_\rho(u), \psi_\rho(v))).$$

Based on the characterization of different functions in B_2 , we call a function f *quadratic* if it has the form $f(u, v) = ((u \oplus c_1) \wedge (v \oplus c_2)) \oplus c_3$. We call a function *linear* if it has the form $f(u, v) = u \oplus v \oplus c$. The pivotal point in [13] is that for a fixed pair of u and v , $C_n \upharpoonright_\rho$ cannot be quadratic under some restriction ρ_1 , then become linear under another restriction ρ_2 . However, by checking the truth table it can be verified that our construction of \mathcal{O}_n forces $C_n \upharpoonright_\rho$ to be the linear function $(u, v) \mapsto u \oplus v$ for all-zero restriction, and the quadratic function $(u, v) \mapsto u \wedge v$ for all-one restriction, which leads to contradiction. ◀

► **Corollary 4.9.** *\mathcal{O} is an explicit obstruction against $2n - 2$ size B_2 circuits.*

Probabilistic circuit lower bound. We now present a general reduction from explicit obstruction to probabilistic circuit lower bounds for sparse languages in P.

► **Lemma 4.10.** *For any circuit class \mathcal{C} , if there exists an explicit obstruction \mathcal{O} of size $S(n)$ against \mathcal{C} , then every language agreeing with \mathcal{O} cannot be computed by probabilistic \mathcal{C} -circuits with error probability less than $\frac{1}{S(n)}$, even infinitely often.*

Proof. Towards contradiction, let L be a language agreeing with \mathcal{O} , and assume that there exists a probabilistic \mathcal{C} -circuit C computing L with error probability smaller than $\frac{1}{S(n)}$, for infinitely many n . Then for those n , we have

$$\mathbb{E}_{\mathcal{C}}[|\{(x, b) \in \mathcal{O}_n \mid C(x) \neq b\}|] < 1.$$

By the averaging argument, there exists a deterministic circuit that agrees with all pairs in \mathcal{O}_n . This guarantees the existence of a family of circuit in \mathcal{C} that agrees with \mathcal{O} on those n , which leads to a contradiction. ◀

In particular, let $\mathcal{L} = \{\mathcal{L}_n\}_{n \geq 1}$ be a language such that $\mathcal{L}_n \triangleq \{x \in \{0, 1\}^n \mid (x, 1) \in \mathcal{O}_n\}$, then it is a sparse language in P that agrees with \mathcal{O} . Together with Corollary 4.9, we obtain the following lower bound.

► **Corollary 4.11.** *There exists a $O(n)$ -sparse language in P that cannot be computed (even infinitely often) by probabilistic B_2 circuits of size $2n - 2$ with error probability smaller than $\frac{1}{2n^2}$.*

► **Remark 4.12.** We note that the error probability in the lower bound cannot be trivially boosted to $1/3$, since the complexity overhead of error reduction is not affordable when we are dealing with small linear-size circuits. Therefore it might be significantly more difficult to prove a similar circuit lower bound with constant error probability.

5 Low-complexity PRFs from hash functions

In this section, we will present the consequences of our hash constructions for low-complexity constructions of pseudorandom functions. We will discuss the Levin’s trick for PRF construction in Section 5.1, and then present our constructions of PRFs for B_2 and low-depth circuits in Section 5.2 and 5.3, respectively.

5.1 PRF and Levin’s trick

► **Definition 5.1** (Pseudorandom functions). *Let $s = s(n)$, $m = m(n)$, and $\varepsilon = \varepsilon(n)$. An (s, ε) -secure m -output pseudorandom function (PRF) is a family $\mathcal{F} = \{\mathcal{F}_n\}_{n \geq 1}$ of distributions \mathcal{F}_n over $B_{n,m}$ such that no probabilistic $s(n)$ -time adversary could distinguish $f \leftarrow \mathcal{F}_n$ from a truly random function $g \leftarrow \mathcal{U}(B_{n,m})$ with advantage $\varepsilon(n)$ given oracle access to the functions, i.e.,*

$$\left| \Pr_{f \leftarrow \mathcal{F}_n} [\mathcal{A}^f(1^n) \text{ accepts}] - \Pr_{g \leftarrow \mathcal{U}(B_{n,m})} [\mathcal{A}^g(1^n) \text{ accepts}] \right| < \varepsilon.$$

for all probabilistic $s(n)$ -time algorithm \mathcal{A} and sufficiently large n .

A PRF is said to be s -secure if it is $(s, 1/s)$ -secure. In particular, a PRF is said to be polynomially secure if it is n^k -secure for all constants $k \geq 1$, and it is said to be sub-exponentially secure if it is $\exp(n^\varepsilon)$ -secure for some constant $\varepsilon \in (0, 1)$. Similar to the hash functions, we define the key (seed) length, circuit complexity, composition, P-uniformity, and POLYLOGTIME-uniformity of PRFs.

Following [23, 13], the key to construct low-complexity PRFs is Levin's trick: the composition of a PRF and an almost universal hash function is still a PRF.

► **Lemma 5.2** (Levin's trick, see, e.g., [5, 13]). *Let $s = s(n)$, $m = m(n)$, $\varepsilon = \varepsilon(n)$, and $\delta = \delta(n)$. The composition $\mathcal{F} \circ \mathcal{H}$ of an (s, ε) -secure PRF \mathcal{F} and a polynomial-time computable m -output δ -almost universal hash \mathcal{H} is an $(\hat{s}, \hat{\varepsilon})$ -secure PRF if $\hat{s}(n) \leq s(m(n)) - \text{poly}(n)$ and $\hat{\varepsilon}(n) \geq \varepsilon(m(n)) + \hat{s}(n)^2 \cdot \delta(n)$ for sufficiently large n .*

Note that the circuit complexity of the resulting PRF mostly depends on the complexity of the hash function, since we can reduce the complexity of the "pseudorandom kernel" by reducing the length of the hash value. Indeed, we need to balance the complexity and security by tuning the output length of the hash function: it should be sufficiently small to reduce the complexity of the original PRF, while it has to be moderately large to guarantee the security of the constructed PRF.

5.2 Low-complexity PRFs in B_2 circuit

Now we discuss the low-complexity PRF construction in B_2 circuits. We first introduce the assumption for the security of our PRF construction.

► **Assumption 5.3.** *There exists a sub-exponentially secure polynomial-time computable pseudorandom function with polynomial key length.*

Note that by the celebrated works of Goldreich, Goldwasser, and Micali [16], the existence of sub-exponentially secure PRF is equivalent to the existence of sub-exponentially secure pseudorandom generator, which is further equivalent to the existence of sub-exponentially secure one-way function [19].

► **Theorem 5.4.** *Assuming Assumption 5.3, there exists an $\exp\left(\Omega\left(\frac{\log^2 n}{\log \log n}\right)\right)$ -secure PRF with key length $\text{polylog}(n)$ computable by POLYLOGTIME-uniform B_2 circuits of size $2n + O(n/\log \log n)$ and depth $\text{polylog}(n)$ simultaneously*

Proof. Since any polynomial-time computable function can be computed by a P-uniform polynomial-size B_2 circuit, the existence of polynomial-time PRF implies the existence of PRF computable by P-uniform polynomial-size B_2 circuits. Under Assumption 5.3 we can obtain an $(\exp(n^\varepsilon), \exp(n^\varepsilon))$ -secure PRF \mathcal{F} computable by P-uniform polynomial-size B_2 circuits for an $\varepsilon \in (0, 1)$. From Construction 5, Theorem 3.9, and Theorem 3.17 with $\ell = \frac{\log^2 n}{\log \log n}$ and $\beta = 1$, we can construct a $\text{polylog}(n)$ -samplable $\Theta(\ell)$ -output $\exp(-\Omega(\ell))$ -almost universal hash function \mathcal{H} computable by POLYLOGTIME-uniform $2n + O(n/\log \log n)$ size B_2 circuits. We increase the output length of \mathcal{H} to $m = \lceil \log^{2/\varepsilon}(n) \rceil$ to obtain \mathcal{H}' by padding constant outputs. We now prove that $\mathcal{F} \circ \mathcal{H}'$ is the desired $\exp(-\Omega(\ell))$ -secure PRF.

Recall that \mathcal{H}' is an m -output $\exp(-c\ell)$ -almost universal hash function for some constant $c \in (0, 1)$. According to Lemma 5.2, we know that for $\hat{s}(n) \triangleq \exp(c\ell/4)$, $\mathcal{F} \circ \mathcal{H}'$ is an \hat{s} -secure PRF, since for sufficiently large n ,

$$\begin{aligned} \exp(m^\varepsilon) - \text{poly}(n) &\geq \exp(\log^2 n) - \text{poly}(n) \geq \hat{s}, \\ \exp(-m^\varepsilon) + \hat{s}^2 \cdot \exp(-c\ell) &\leq \exp(-\log^2 n) + \exp(-c\ell/2) \leq \frac{1}{\hat{s}}. \end{aligned}$$

Because the key of $\mathcal{F} \circ \mathcal{H}'$ consists of the seed of \mathcal{H}' (with input length n) and the key of \mathcal{F} (with input length m), the key length would be $\text{polylog}(n) + \text{poly}(m) = \text{polylog}(n)$.

It is straightforward to verify that $\mathcal{F} \circ \mathcal{H}'$ can be computed by B_2 circuits of size $2n + O(n/\log \log n)$ and depth $\text{polylog}(n) + \text{poly}(m) = \text{polylog}(n)$, so it remains to show that the circuit computing it is POLYLOGTIME-uniform. We only demonstrate the evaluation of function $\text{EDGE}(n, v, i, j)$ (i.e., the j^{th} input node of the i^{th} node in the circuit computing function keyed by v) and remark that the evaluations of other functions (SIZE, TYPE, and OUT) can be done in a similar way.

Given a key $v = v_1 \| v_2$ where v_1 is the seed for \mathcal{H}' and v_2 is the key for \mathcal{F} , we firstly decide whether the i^{th} node is inside the hash \mathcal{H}' or the PRF \mathcal{F} (which can be done since we can compute the number of nodes in \mathcal{H} in $\text{polylog}(n)$ time given v_1). In the former case, we can use the $\text{EDGE}(n, v_1, \cdot, \cdot)$ function for the hash function \mathcal{H}' , since it is POLYLOGTIME-uniform. In the latter case, we can draw the circuit computing \mathcal{F} given key v_2 in $\text{polylog}(n)$ time (since \mathcal{F} is P-uniform, and the input length is $m = \text{polylog}(n)$), and then find out the j^{th} input node of the i^{th} node. ◀

► **Remark 5.5.** It can be easily verified that if we do not require the PRF to be computable within $\text{polylog}(n)$ depth, we can in fact rely on the following (possibly) weaker assumption: the existence of $\exp(n^{\varepsilon/4})$ -time computable $\exp(n^\varepsilon)$ -secure PRFs with polynomial key length for an $\varepsilon > 0^{23}$. By [16, 19] (implicitly), this assumption follows from the existence of $\exp(n^{\varepsilon/8})$ -time computable OWFs against any $\exp(n^\varepsilon)$ -time adversary for an $\varepsilon > 0$. We stick to Assumption 5.3 since it has been a quite standard assumption.

5.3 Low-complexity PRFs in low-depth circuits

To construct efficient PRFs in low-depth circuit classes such as NC^1 and $\text{AC}^0[2]$ using our low-complexity hash functions, we need to rely on the existence of low-depth PRFs. In particular, we will need the existence of (sub-exponentially secure) NC^1 PRFs to construct efficient NC^1 and $\text{AC}^0[2]$ PRFs.

► **Assumption 5.6** (NC^1 PRF). *There exists a sub-exponentially secure PRF with polynomial key length computable by P-uniform polynomial-size NC^1 circuits.*

Note that it is unknown whether such assumption can be reduced to more elementary ones such as the existence of certain kinds of one-way functions. Nevertheless, it follows from standard cryptographic assumptions such as sub-exponential decisional Diffie-Hellman [33] or sub-exponential Ring Learning-with-Error [4].

► **Theorem 5.7.** *There exists a $\exp\left(\Omega\left(\frac{\log^2 n}{\log \log n}\right)\right)$ -secure PRF with $\text{polylog}(n)$ key length computable by POLYLOGTIME-uniform B_2 circuits of size $2n + O(n/\log \log n)$ and depth $\log n + O(\log \log n)$ simultaneously under Assumption 5.6.*

Proof. We will use the construction in the proof of Theorem 5.4 by replacing the PRF \mathcal{F} with a PRF computable by P-uniform NC^1 circuits. We only check the circuit depth of the construction since other properties can be established similarly as in the proof of Theorem 5.4. By Theorem 3.17, the B_2 circuit computing the hash function \mathcal{H} (and \mathcal{H}') in

²³The POLYLOGTIME-uniformity follows from the folklore simulation of P algorithms by POLYLOGTIME-uniform circuits (see, e.g., [29]).

the proof of Theorem 5.4 has depth $\log n + O(1)$. Since the output length of \mathcal{H}' (i.e., the input length of the original PRF) is $\text{polylog}(n)$, the depth of the original PRF would be $\log(\text{polylog}(n)) = O(\log \log n)$. Therefore the total depth would be $\log n + O(\log \log n)$. ◀

To construct $\text{AC}^0[2]$ PRFs from Assumption 5.6, we need a folklore reduction from logarithmic depth circuits to AC^0 circuits. In particular, we show how to transform a polynomial-size NC^1 circuit to an AC^0 circuit of size $2^{O(n^\varepsilon)}$ and depth $O(1/\varepsilon)$.

► **Lemma 5.8.** *Let f be a function computable by P-uniform NC^1 circuits. For any constant $\varepsilon > 0$, f is computable by POLYLOGTIME-uniform AC^0 circuits of $2^{O(n^\varepsilon)}$ size and $O(1/\varepsilon)$ depth.*

Proof. Suppose that f is computable by an NC^1 circuit C of depth $d \log n$. Let $k = d/\varepsilon$. We partition C into k layers of chunks, each of depth $\varepsilon \log n$. In such case, each chunk only depends on $O(2^{\varepsilon \log n}) = O(n^\varepsilon)$ number of gates, so we can expand it into a CNF of size $2^{O(n^\varepsilon)}$. After expanding all the chunks, we get an AC^0 circuit C' of size $2^{O(n^\varepsilon)}$ and depth $2k = O(1/\varepsilon)$.

Let $N = 2^{O(n^\varepsilon)}$ be the size of C' . To show the uniformity, we observe that to compute the local structure of a gate, we only need to evaluate one of the chunks on a given input, which can be done in $\text{poly}(n) = \text{polylog}(N)$ time. ◀

► **Theorem 5.9.** *There exists a $\exp\left(\Omega\left(\frac{\log^2 n}{\log \log n}\right)\right)$ -secure PRF with $\text{polylog}(n)$ key length computable by POLYLOGTIME-uniform $\text{AC}^0[2]$ circuits with $2n + O(n/\log \log n)$ wires under Assumption 5.6.*

Proof. We will use the construction in the proof of Theorem 5.4 by replacing the PRF \mathcal{F} with a PRF computable by P-uniform NC^1 circuits. Suppose that the parameter m in the proof of Theorem 5.4 is at most $\log^c n$. Initiating Lemma 5.8 with $\varepsilon = 1/(2c)$, we can compute the PRF \mathcal{F} with an POLYLOGTIME-uniform AC^0 circuit with $2^{O(\sqrt{\log n})} = n^{o(1)}$ wires. It immediately follows that our PRF construction is computable by POLYLOGTIME-uniform $\text{AC}^0[2]$ circuits with $2n + O(n/\log \log n)$ wires. The other parts are the same as the proof of Theorem 5.4. ◀

References

- 1 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k -wise independent random variables. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 544–553. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89575.
- 2 Benny Applebaum and Pavel Raykov. Fast pseudorandom functions based on expander graphs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 27–56, 2016. doi:10.1007/978-3-662-53641-4_2.
- 3 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
- 4 Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012. doi:10.1007/978-3-642-29011-4_42.

- 5 Andrej Bogdanov and Alon Rosen. Pseudorandom functions: Three decades later. In *Tutorials on the Foundations of Cryptography*, pages 79–158. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_3.
- 6 Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979. doi:10.1016/0022-0000(79)90044-8.
- 7 L. Sunil Chandran. A high girth graph construction. *SIAM J. Discret. Math.*, 16(3):366–370, 2003. doi:10.1137/S0895480101387893.
- 8 Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPICs*, pages 70:1–70:48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ITCS.2020.70.
- 9 Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness magnification for all sparse NP languages. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1240–1255. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00077.
- 10 Lijie Chen, Ce Jin, and R. Ryan Williams. Sharp threshold results for computational complexity. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1335–1348. ACM, 2020. doi:10.1145/3357713.3384283.
- 11 Ruiwen Chen and Valentine Kabanets. Correlation bounds and #SAT algorithms for small linear-size circuits. *Theor. Comput. Sci.*, 654:2–10, 2016. doi:10.1016/j.tcs.2016.05.005.
- 12 Evgeny Demenkov and Alexander S. Kulikov. An elementary proof of a $3n - o(n)$ lower bound on the circuit complexity of affine dispersers. In *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 256–265. Springer, 2011. doi:10.1007/978-3-642-22993-0_25.
- 13 Zhiyuan Fan, Jiayu Li, and Tianqi Yang. The exact complexity of pseudorandom functions and the black-box natural proof barrier for bootstrapping results in computational complexity. *Electron. Colloquium Comput. Complex.*, page 125, 2021. To appear in STOC 2022. URL: <https://eccc.weizmann.ac.il/report/2021/125>.
- 14 Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than- $3n$ lower bound for the circuit complexity of an explicit function. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 89–98. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.19.
- 15 Lance Fortnow. A simple proof of Toda’s theorem. *Theory Comput.*, 5(1):135–140, 2009. doi:10.4086/toc.2009.v005a007.
- 16 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.
- 17 Alexander Golovnev, Edward A. Hirsch, Alexander Knop, and Alexander S. Kulikov. On the limits of gate elimination. *J. Comput. Syst. Sci.*, 96:107–119, 2018. doi:10.1016/j.jcss.2018.04.005.
- 18 Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.
- 19 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.
- 20 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002. doi:10.1016/S0022-0000(02)00025-9.

- 21 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPICs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.CCC.2017.7.
- 22 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. *J. ACM*, 66(2):11:1–11:16, 2019. doi:10.1145/3230630.
- 23 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 433–442. ACM, 2008. doi:10.1145/1374376.1374438.
- 24 Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of $5n - o(n)$ for boolean circuits. In *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, volume 2420 of *Lecture Notes in Computer Science*, pages 353–364. Springer, 2002. doi:10.1007/3-540-45687-2_29.
- 25 Eyal Kaplan, Moni Naor, and Omer Reingold. Derandomized constructions of k -wise (almost) independent permutations. *Algorithmica*, 55(1):113–133, 2009. doi:10.1007/s00453-008-9267-y.
- 26 Oded Lachish and Ran Raz. Explicit lower bound of $4.5n - o(n)$ for boolean circuits. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 399–408. ACM, 2001. doi:10.1145/380752.380832.
- 27 Felix Lazebnik and Vasilij A. Ustimenko. Explicit construction of graphs with an arbitrary large girth and of large size. *Discret. Appl. Math.*, 60(1-3):275–284, 1995. doi:10.1016/0166-218X(94)00058-L.
- 28 Jiayu Li and Tianqi Yang. $3.1n - o(n)$ circuit lower bounds for explicit functions. *Electron. Colloquium Comput. Complex.*, page 23, 2021. To appear in STOC 2022. URL: <https://ecc.weizmann.ac.il/report/2021/023>.
- 29 Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time, with applications. *Comput. Complex.*, 22(2):311–343, 2013. doi:10.1007/s00037-013-0069-5.
- 30 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1215–1225. ACM, 2019. doi:10.1145/3313276.3316396.
- 31 Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. *J. ACM*, 62(6):46:1–46:29, 2015. doi:10.1145/2792978.
- 32 Ketan Mulmuley. On P vs. NP and geometric complexity theory: Dedicated to sri ramakrishna. *J. ACM*, 58(2):5:1–5:26, 2011. doi:10.1145/1944345.1944346.
- 33 Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. doi:10.1145/972639.972643.
- 34 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Theory Comput.*, 17:1–38, 2021.
- 35 Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 65–76, 2018. doi:10.1109/FOCS.2018.00016.
- 36 Anna Pagh and Rasmus Pagh. Uniform hashing in constant time and optimal space. *SIAM J. Comput.*, 38(1):85–96, 2008. doi:10.1137/060658400.
- 37 Christos H. Papadimitriou and Stathis Zachos. Two remarks on the power of counting. In *Theoretical Computer Science, 6th GI-Conference, Dortmund, Germany, January 5-7, 1983, Proceedings*, volume 145 of *Lecture Notes in Computer Science*, pages 269–276. Springer, 1983. doi:10.1007/BFb0009651.
- 38 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

- 39 Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 283–290. IEEE Computer Society, 1988. doi:10.1109/SFCS.1988.21944.
- 40 Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inf. Theory*, 42(6):1723–1731, 1996. doi:10.1109/18.556668.
- 41 Avishay Tal. Shrinkage of de Morgan formulae by spectral techniques. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 551–560. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.65.
- 42 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.
- 43 Emanuele Viola. The communication complexity of addition. *Comb.*, 35(6):703–747, 2015. doi:10.1007/s00493-014-3078-3.

A Strongly explicit high-girth graphs

In this section, we will examine the construction of bipartite high-girth graphs of Lazebnik and Ustimenko [27], and verify that it is indeed strongly explicit, thereby proving Theorem 2.12.

Let q be an odd prime power. Let P and L be two infinite sequences of elements from \mathbb{F}_q indexed as follows:

$$\begin{aligned} P &= \langle p_1, p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, p'_{2,2}, p_{3,2}, \dots, p_{i,i+1}, p_{i+1,i}, p_{i+1,i+1}, p'_{i+1,i+1}, p_{i+1,i+2}, \dots \rangle \\ L &= \langle l_1, l_{1,1}, l_{1,2}, l_{2,1}, l_{2,2}, l'_{2,2}, l_{3,2}, \dots, l_{i,i+1}, l_{i+1,i}, l_{i+1,i+1}, l'_{i+1,i+1}, l_{i+1,i+2}, \dots \rangle. \end{aligned}$$

The names P and L mean points and lines, respectively, due to certain geometric intuition of the construction. We say P is incident with L if they satisfy the following set of equations

$$E_1 = \begin{cases} l_{1,1} - p_{1,1} = l_1 p_1 \\ l_{1,2} - p_{1,2} = l_{1,1} p_1 \\ l_{2,1} - p_{2,1} = l_1 p_{1,1} \end{cases}, \quad E_i = \begin{cases} l_{i,i} - p_{i,i} = l_1 p_{i-1,i} \\ l'_{i,i} - p'_{i,i} = l_{i,i-1} p_1 \\ l_{i,i+1} - p_{i,i+1} = l_{i,i} p_1 \\ l_{i+1,i} - p_{i+1,i} = l_1 p'_{i,i} \end{cases} \quad (\forall i \geq 2) \quad (3)$$

One can verify that for every point P , a line L incident with it is uniquely determined by l_1 , since all other coordinates of L can be computed from the equations above iteratively. Similarly for every line L , a point P incident with it is uniquely determined by p_1 .

Let $P_k \in \mathbb{F}_q^k$ be the length- k prefix of P , and $L_k \in \mathbb{F}_q^k$ be the length- k prefix of L . We define $D(k, q) = (V_1, V_2, E \subseteq V_1 \times V_2)$ to be the following bipartite graph:

$$\begin{aligned} V_1 &= V_2 = \mathbb{F}_q^k; \\ (u, v) \in E &\iff \exists (P, L), P \text{ is incident with } L, u = P_k, \text{ and } v = L_k. \end{aligned}$$

Equivalently, u and v are connected if and only if they satisfy the first k equations of $\cup_i E_i$ in (3). Note that $|V_1| = |V_2| = q^k$, $|E| = q^{k+1}$, and the graph is q -regular.

► **Theorem A.1** (Lazebnik and Ustimenko [27]). *Let $k \geq 3$ be an odd integer and q be an odd prime power. Then the girth of $D(k, q)$ is at least $k + 5$.*

For our purpose, it is sufficient to consider $q = p^r$ for prime $p = O(1)$. To construct the graph explicitly, we need to evaluation field operations over \mathbb{F}_q for a prime power q in $\text{polylog}(q)$ time, for which we need a representation of \mathbb{F}_q . Recall that \mathbb{F}_q is isomorphic to $\mathbb{F}_p[x]/(Q(x))$ for any irreducible degree- r polynomial $Q \in \mathbb{F}_p[x]$. So the explicit representation of \mathbb{F}_q follows from the construction of irreducible polynomials by Shoup [39].

► **Theorem A.2** (Shoup [39]). *There is a deterministic algorithm that constructs a degree- n irreducible polynomial over \mathbb{F}_p given an integer n and a prime power p in $\text{poly}(n, p)$ time.*

Now we check that $D(k, q)$ is strongly explicit, in the sense that given the index i of an edge, we can obtain the indices j_1 and j_2 in $\text{poly}(k, \log q)$ time such that the i^{th} edge connects the j_1^{th} vertex in V_1 and the j_2^{th} vertex in V_2 . Let $q = p^r$ for a prime $p = O(1)$. We number the vertices and edges as follows.

1. We identify the elements in the finite field \mathbb{F}_q as length- r vectors from $[p]^r$, and number all the elements in the lexicographic order.
2. We identify the vertices in both V_1 and V_2 (that are length k sequences of elements in \mathbb{F}_q) as length- kr vectors from $[p]^{kr}$, and number them in the lexicographic order.
3. For any $i \in [q^k]$ and $j \in [q]$, the $(iq + j)^{\text{th}}$ edge connects the i^{th} vertex P in V_1 and the unique vertex $L = (l_1, \dots)$ in V_2 connected to P such that l_1 is the j^{th} element in \mathbb{F}_q .

Under such a numbering scheme, given the index $iq + j$ of an edge with $i \in [q^k]$ and $j \in [q]$, we can easily determine its two endpoints in $\text{poly}(k, \log q)$ time.

Now we are ready to prove Theorem 2.12.

► **Remainder of Theorem 2.12.** Let $r = r(n) = n^{o(1)}$ be a parameter. For every sufficiently large n , there exists an $m = \Theta(\frac{n}{r})$ and a regular graph $G_{m,n}$ with m vertices, n edges, and girth $\Omega(\frac{\log n}{\log r})$. Moreover, there exists a $\text{polylog}(n)$ -time algorithm $\mathcal{A}(n, i)$ for $i \in [n]$ that outputs the indices of the two endpoints of the i^{th} edge in $G_{m,n}$, and a $\text{polylog}(n)$ -time algorithm $\mathcal{B}(n, i, j)$ for $i \in [m]$ that outputs the j^{th} edge attaching to the i^{th} vertex.

Proof. Let q be the power of 3 in the interval $[r, 3r)$. Applying Theorem A.2, we construct a fixed representation of \mathbb{F}_q in $\text{polylog}(q)$ time so that the field operations over \mathbb{F}_q can be evaluated in $\text{polylog}(q)$ time. Let $k \triangleq \lfloor \log_q n \rfloor - 1$ (i.e., k is the largest integer such that $q^{k+1} \leq n$), $\ell \triangleq \lceil n/q^{k+1} \rceil$, and $m \triangleq 2q^k \ell$. Note that

$$m \leq 2q^k \left(\frac{n}{q^{k+1}} + 1 \right) = \frac{2n}{q} + \frac{2q^{k+1}}{q} \leq O\left(\frac{n}{r}\right),$$

$$m \geq 2q^k \cdot \frac{n}{q^{k+1}} \geq \Omega\left(\frac{n}{r}\right),$$

and therefore $m = \Theta(n/r)$. Now we construct a graph $G_{m,n}$ with ℓ connected components as follows. Each of the first $\ell - 1$ connected components is a copy of $D(q, k)$ with $2q^k$ vertices and q^{k+1} edges. The last connected component is a subgraph of $D(q, k)$ with $2q^k$ vertices but only the first $n - q^{k+1}(\ell - 1)$ edges.

By Theorem A.1, it is easy to see that the girth of $G_{m,n}$ is at least $k = \Omega(\log n / \log q) = \Omega(\frac{\log n}{\log r})$. According to our numbering scheme and related discussions, it is also easy to verify that given an edge index i , we can compute the two endpoints of the i^{th} edge in $G_{m,n}$ in $\text{polylog}(n)$ time. Furthermore, the j^{th} edge attaching to the i^{th} vertex can be easily computed since each of the q edges attaching to the i^{th} vertex is uniquely determined by $l_1 \in \mathbb{F}_q$ or $p_1 \in \mathbb{F}_q$ (see Equation 3). ◀