Leakage Resilience, Targeted Pseudorandom Generators, and Mild Derandomization of Arthur-Merlin Protocols

Dieter van Melkebeek
University of Wisconsin-Madison, WI, USA

Nicollas Mocelin Sdroievski
University of Wisconsin-Madison, WI, USA

Abstract

Many derandomization results for probabilistic decision processes have been ported to the setting of Arthur-Merlin protocols. Whereas the ultimate goal in the first setting consists of efficient simulations on deterministic machines (BPP vs. P problem), in the second setting it is efficient simulations on nondeterministic machines (AM vs. NP problem). Two notable exceptions that have not yet been ported from the first to the second setting are the equivalence between whitebox derandomization and leakage resilience (Liu and Pass, 2023), and the equivalence between whitebox derandomization and targeted pseudorandom generators (Goldreich, 2011). We develop both equivalences for mild derandomizations of Arthur-Merlin protocols, i.e., simulations on Σ_2-machines. Our techniques also apply to natural simulation models that are intermediate between nondeterministic machines and Σ_2-machines.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Hardness versus randomness tradeoff, leakage resilience, Arthur-Merlin protocol, targeted hitting set generator

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.29

Funding Partial support for this research was provided by the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation, and by the National Science Foundation under Grant No. 2312540.

1 Introduction

Understanding whether bounded-error probabilistic algorithms can be simulated deterministically with a polynomial overhead in time (the BPP versus P problem) is a central problem in the theory of computing. An analogous question in relation to interactive protocols is whether constant-round interactive protocols (commonly referred to as Arthur-Merlin protocols), can be simulated *nondeterministically* with a polynomial overhead in time (the AM versus NP problem). An influential line of research on hardness vs. randomness tradeoffs concludes derandomization from lower bounds, and in some cases equivalences are known. As the lower bounds are plausible, it is conjectured that BPP = P and AM = NP. Still, even *mild* derandomizations that require more nondeterminism than what is conjectured (such as BPP $\subseteq$ NP and AM $\subseteq \Sigma_2$P $=$ NP$^{\mathsf{NP}}$) remain open. We first describe the situation for the BPP setting and then move on to the AM setting, which is the focus of this work.

BPP setting. The first hardness vs. randomness tradeoffs for BPP were in the *blackbox* setting, where a pseudorandom generator (PRG) obliviously constructs a small set of strings that "fools" every efficient probabilistic algorithm in the sense that the algorithm behaves approximately the same under the uniform distribution and the generator's distribution. To construct pseudorandom generators, early works relied on non-uniform hardness assumptions such as linear-exponential circuit lower-bounds for $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ (which suffices to conclude $\mathsf{BPP} = \mathsf{P}$) [12]. The preceding parameter setting is known as the high end of the derandomization spectrum. At the low end of the spectrum, subexponential-time blackbox simulations for BPP follow from $\mathsf{E} \not\subseteq \mathsf{P/poly}$, i.e., polynomial-size circuit lower bounds for linear-exponential time [3]. Pseudorandom generators are in fact *equivalent* to non-uniform lower bounds, and the equivalence holds across the entire derandomization spectrum [22].

The discussion presents an equivalence between *blackbox* derandomization and non-uniform lower bounds. It leaves the question of whether general (*whitebox*) derandomization implies non-uniform lower bounds. Given the equivalence between such lower bounds and PRGs, another way to phrase the question is whether PRGs are "complete" for derandomization: If derandomization is possible, then is it possible through PRGs?

Goldreich [8, 9] showed that whitebox derandomization of prBPP (the promise-problem version of BPP) is equivalent to the existence of *targeted* pseudorandom generators, which are pseudorandom generators that receive as input a target composed of an algorithm A and an input x, and are only required to "fool" A on input x. The equivalence establishes a "normal form" for derandomizing prBPP. In principle, a deterministic simulation A_{det} of a probabilistic algorithm A can access A in other ways than by running A on the given input x and some random bit strings ρ. Goldreich's result, however, establishes that if derandomization is possible, i.e., $\mathsf{prBPP} \subseteq \mathsf{P}$, then we can always take A_{det} to be the algorithm that, on input x, computes a targeted pseudorandom set $S_{A,x}$ and outputs the majority answer obtained by simulating A on input x while replacing its random inputs by each element in $S_{A,x}$.

Recently, Chen and Tell [5] proposed a candidate hardness assumption that is equivalent to whitebox derandomization of prBPP: *uniform* lower bounds for *multi-bit* functions f (rather than usual decision problems) against probabilistic algorithms that hold on *almost-all inputs*, i.e., any probabilistic algorithm limited to a certain running time can only compute f correctly on a finite number of inputs. Chen and Tell show that if there exists a length-preserving function f computable by logspace-uniform circuits of polynomial size and depth n^b for some constant b that is hard on almost-all inputs against $\mathsf{prBPTIME}[n^{b+O(1)}]$, then $\mathsf{prBPP} \subseteq \mathsf{P}$. They also observe that $\mathsf{prBPP} \subseteq \mathsf{P}$ implies that for every constant c, there is a length-preserving function $f \in \mathsf{P}$ that is hard on almost-all inputs against $\mathsf{prBPTIME}[n^c]$. Modulo the uniform depth requirement, their result establishes a near-equivalence between hardness on almost-all inputs and derandomization. The approach scales fairly well to the rest of the spectrum, but not perfectly. In particular, the low end remains open.

Under a different hardness assumption, Liu and Pass managed to obtain *full* equivalences for derandomization of prBPP and hardness on almost-all inputs. The hardness assumption that they employ is the existence of computational problems (relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$) that are hard even in the presence of efficiently-computable leakage [15]. We define a version of the notion in the setting of hardness on almost-all inputs.

▶ **Definition 1** (Leakage-resilient hardness). *A relation R is (T, ℓ)-leakage-resilient hard on almost-all inputs against a class of algorithms if for all pairs (Leak, A) of algorithms in the class that run in time T and such that $|\mathrm{Leak}(x,y)| \leq \ell(|x|)$, the following holds for almost-all inputs x and every $y \in R(x) = \{y \mid (x,y) \in R\}$: The probability that $A(x, \mathrm{Leak}(x,y)) = y$ is at most $1/3$.*

In cryptographic terms, the algorithm A can be viewed as an attacker that attempts to compute $y \in R(x)$ by receiving a few bits of information about y from the leakage-providing algorithm Leak. In some cases, such as when the relation R is computable in P, one may take R to be a function.

At the high end of the derandomization spectrum, the main result of Liu and Pass [15] establishes an equivalence between the derandomization $\mathsf{prBPP} \subseteq \mathsf{P}$ and the existence of a length-preserving function $f \in \mathsf{P}$ that is leakage-resilient hard on almost all inputs against probabilistic algorithms running in fixed-polynomial time and with sublinear leakage. The equivalence works across the entire derandomization spectrum. For completeness we mention that, in another work, Liu and Pass establish an equivalence between derandomization and hardness related to approximating conditional Levin-Kolmogorov complexity (Kt) [14]. The latter equivalence does not scale well toward the low end, though.

One can also ask about efficient simulations of prBPP on *nondeterministic* machines. Such simulations are referred to as *mild* derandomizations. We remark that establishing mild derandomization results for prBPP is a required step if we hope to show stronger deterministic simulations for the class. At the low-end of the derandomization spectrum, an equivalence is known between whitebox and blackbox mild derandomizations for prBPP that work for infinitely-many input lengths [11]. The technique, however, does not scale and is only known to work in the infinitely-often setting. The Liu-Pass result on the connection between leakage-resilient hardness and derandomization extends to the mild setting. In contrast to the low-end equivalence of [11], Theorem 2 extends to the entire derandomization spectrum. We state the high-end version.

▶ **Theorem 2** (Follows from [15, Theorem 1.6]). *There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

- $\mathsf{prBPP} \subseteq \mathsf{NP}$.
- *There exists a total length-preserving relation $R \in \mathsf{NP}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against* prBPP.

AM setting. In the *blackbox* setting, an equivalence between derandomization and non-uniform lower bounds is known throughout the entire spectrum [13, 16, 18]. The class E is replaced by NE∩coNE, and the circuits are nondeterministic (or single-valued nondeterministic, or deterministic with oracle access to an NP-complete problem like SAT). These assumptions yield hitting-set generators (HSGs) for AM, the one-sided error counterparts of pseudorandom generators. These objects suffice for derandomizing AM as every Arthur-Merlin protocol can be efficiently transformed into an equivalent one with perfect completeness [7].

Until recently, not much was known in the *whitebox* setting for AM. Van Melkebeek and Sdroievski have established a near-equivalence between (plain) almost-all inputs hardness and derandomization [23]. In one direction, they show that the existence of a length-preserving function computable in nondeterministic time n^b that is hard on almost-all inputs against prAM protocols that run in time $n^{O((\log b)^2)}$ implies that $\mathsf{prAM} \subseteq \mathsf{NP}$. They also establish a converse, though the hard function they obtain requires a small amount of advice and the hardness is only guaranteed against AM protocols instead of prAM protocols. The hardness-to-derandomization direction of their result scales very well toward the low-end.

It remains open whether whitebox derandomization of prAM can always be obtained through targeted hitting-set generators if it can be obtained at all, a question raised by Goldreich [8]. As a byproduct of their main result, the authors of [23] obtain a first step toward a positive resolution: They show that any low-end derandomization of prAM implies the existence of a targeted hitting-set generator that achieves a slightly weaker derandomization.

Similar to the BPP setting, one may ask for derandomizations of AM that require more resources than what is conjectured to suffice. While it is believed that $\mathsf{prAM} \subseteq \mathsf{NP}$ and we know that $\mathsf{prAM} \subseteq \Pi_2\mathsf{P} = \mathsf{coNP}^{\mathsf{NP}}$, it remains open whether $\mathsf{prAM} \subseteq \mathsf{P}^{\mathsf{NP}}$ or even whether $\mathsf{prAM} \subseteq \Sigma_2\mathsf{P} = \mathsf{NP}^{\mathsf{NP}}$, both of which are required steps toward showing that $\mathsf{prAM} \subseteq \mathsf{NP}$.

We note that even a subexponential-time Σ_2-simulation with subpolynomial advice for prAM that works for infinitely many input lengths remains open.[1] In this setting, an equivalence between whitebox and blackbox derandomization is known: The simulation is equivalent to polynomial size *nondeterministic* circuit lower bounds for the class $\Sigma_2\mathsf{E}$. In symbols, $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all ϵ if and only if $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$ [2]. The equivalence does not scale well, and it is unknown whether it holds for other parameter settings such as the high end.

Our results. Our main contribution is establishing *full* equivalences for *mild* derandomization of prAM. Specifically, we obtain an equivalence between mild derandomization of prAM and leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, the class of polynomial-time probabilistic algorithms with non-adaptive oracle access to SAT. The equivalence scales smoothly across the entire derandomization spectrum, and applies to other classes between NP and $\Sigma_2\mathsf{P}$ in addition to $\Sigma_2\mathsf{P}$. We state the high-end version.

▶ **Theorem 3.** *Let $\mathcal{C} \in \{\mathsf{P}^{\mathsf{NP}}, \mathsf{ZPP}^{\mathsf{NP}}, \Sigma_2\mathsf{P}\}$. There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

- $\mathsf{prAM} \subseteq \mathcal{C}$.
- *There exists a total length-preserving relation $R \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against* $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.

Theorem 3 can be viewed as a counterpart to Theorem 2 in the mild setting for prAM. Indeed, when compared to the traditional setting, both require an extra level of nondeterminism in the hardness assumption as well as in the simulation.

The class $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ was used in the initial derandomization results for Arthur-Merlin protocols [13] and is seemingly more powerful than prAM. However, derandomization results for prAM typically translate into corresponding derandomization results for $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. In particular, the derandomization $\mathsf{prAM} \subseteq \Sigma_2\mathsf{P}$ of Theorem 3 implies that $\mathsf{prBPP}_{||}^{\mathsf{SAT}} \subseteq \Sigma_2\mathsf{P}$. Also, lower bounds for linear-exponential time classes such as E and NE against nondeterministic circuits (which suffice for derandomizing prAM) imply non-uniform lower bounds for the same class against deterministic circuits with non-adaptive oracle access to SAT (which suffice for derandomizing $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$) [19].

We show that a connection between $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ and prAM holds in the leakage-resilient hardness setting as well. Specifically, we show that the derandomization conclusion of Theorem 3 holds under a weaker hardness assumption on *learn-and-evaluate* Arthur-Merlin protocols. Learn-and-evaluate protocols are a two-phase type of Arthur-Merlin protocol that arises naturally in hardness vs. randomness tradeoffs and fits nicely into the leakage-resilient hardness paradigm. In the first phase, a probabilistic algorithm with oracle access to a function f produces a short sketch π that consists of evaluations of f. In our case, f is the function that maps an index i to the i-th bit of a string y such that $(x, y) \in R$, and the first phase can be viewed as a small amount of leakage on y. In the second phase, an Arthur-Merlin protocol computes f given π, which naturally translates into a protocol that attempts to recover y from a "small" amount of leakage. The result, taken together with the derandomization-to-hardness direction of Theorem 3, implies an equivalence between leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ and learn-and-evaluate protocols.

[1] The best known unconditional results are worst-case simulations of AM in $\text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^c$ for all ϵ and some fixed c [24] and average-case simulations of prAM in $\text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all ϵ [23].

▶ **Theorem 4.** *Let $\mathcal{C} \in \{\mathsf{P}^{\mathsf{NP}}, \mathsf{ZPP}^{\mathsf{NP}}, \Sigma_2\mathsf{P}\}$. There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

1. *There exists a total length-preserving relation $R \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*
2. *There exists a total length-preserving relation $R \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against learn-and-evaluate protocols.*

Theorem 4 partially addresses an open problem posed by Shaltiel and Umans of obtaining uniform equivalences between hardness against Arthur-Merlin protocols and algorithms with non-adaptive oracle access to SAT [19]. They ask whether an implication such as $\mathsf{E} \not\subseteq \mathsf{AM} \implies \mathsf{E} \not\subseteq \mathsf{P}_{||}^{\mathsf{SAT}}$ holds. Theorem 4 establishes an analogous result in the mild leakage-resilient hardness setting.

We also address, in the mild setting, the open problem posed by Goldreich about prAM [8]. We show that mild derandomization of prAM is equivalent to the existence of targeted hitting-set generators achieving the same derandomization. We state such a result at the high end of the derandomization spectrum, but it extends to the entire range.

▶ **Theorem 5.** *Let $\mathcal{C} \in \{\mathsf{P}^{\mathsf{NP}}, \mathsf{ZPP}^{\mathsf{NP}}, \Sigma_2\mathsf{P}\}$. If $\mathsf{prAM} \subseteq \mathcal{C}$, then there exist targeted hitting-set generators that achieve this mild derandomization result.*

Finally, we connect leakage-resilient hardness to the known equivalence between whitebox and blackbox mild derandomization of prAM at the low end. As mentioned before, the simulation $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$ is equivalent to the non-uniform lower bound $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$ [2]. We show that those conditions are also equivalent to leakage-resilient hardness where the algorithm A needs to produce each individual bit of a solution $y \in R(x)$ very efficiently. The equivalence of [2] holds in the infinitely-often setting. Correspondingly, the equivalent leakage-resilient hardness condition holds for all inputs of infinitely-many input lengths.

▶ **Definition 6** (Local Leakage-resilient hardness). *A relation R is t-local (T, ℓ)-leakage-resilient hard on all inputs of infinitely-many input lengths against a class of algorithms if for all pairs (Leak, A) of algorithms in the class such that Leak runs in time T, $|\mathrm{Leak}(x, f(x))| \leq \ell(|x|)$ and A runs in time t, the following holds for infinitely many $n \in \mathbb{N}$ and every $x \in \{0,1\}^n$: $R(x)$ is non-empty and for every such $y \in R(x)$ there exists $i \in [|y|]$ such that the probability that $A(x, \mathrm{Leak}(x, y), i) = y_i$ is at most $1/3$.*

This definition allows for separate running times for the leakage-providing algorithm Leak and A, and in particular allows A to run in time that is sublinear in the length of a solution $y \in R(x)$.

▶ **Theorem 7.** *The following are equivalent:*

1. $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ *for all $\epsilon > 0$.*
2. $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$.
3. *For all $\epsilon > 0$ there exists a relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is $\mathsf{poly}(n)$-local $(\infty, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*
4. *For all $\epsilon > 0$ there exists a relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is $\mathsf{poly}(n)$-local $(2^{n^\epsilon}, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*
5. *For all $\epsilon > 0$ and $c \geq 1$ there exists a length-preserving relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is n^c-local $(n^c, \ell(n))$-leakage resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, where $n^{\Omega(1)} \leq \ell(n) \leq n - \omega(1)$ is polynomial-time computable.*

We only know Theorem 7 for the low-end of the derandomization spectrum due to the use of [2].

Techniques. We combine the approach of Liu and Pass in the leakage-resilient setting for BPP (see Section 2 for an overview) with constructions that are suitable for the AM setting.

In the hardness-to-derandomization direction, given a hard relation R, we use the value of $y \in R(x)$ as a basis for instantiating the nondeterministic version of the Shaltiel-Umans (SU) generator for prAM [18]. As the SU generator is originally meant for the the non-uniform setting, the original reconstruction argument only guarantees the existence of a small nondeterministic circuit that computes y in case the generator fails. Thus, in order to use the construction we need to "uniformize" the reconstruction procedure and show that such a circuit can be constructed by an efficient algorithm with non-adaptive oracle access to SAT. To obtain the results under hardness against learn-and-evaluate protocols, we employ the same approach but instead use a recursive version of the Miltersen-Vinodchandran generator [16], which we refer to as RMV and is due to Shaltiel and Umans [20]. RMV does not scale as well as SU but still suffices for obtaining Theorem 4. To employ the construction, we need to cast the RMV reconstruction process as a learn-and-evaluate protocol.

For the derandomization-to-hardness direction, similar to [15], we frame the problem of computing a leakage-resilient hard function as a $\mathsf{BPP}_{||}^{\mathsf{SAT}}$ search problem, and then make use of a search-to-decision reduction à la [8] together with the derandomization assumption. We first show that for a fixed x, a random choice r of $y \in R(x)$ suffices: With high probability, the first few algorithms Leak and A according to some canonical enumeration fail to compute r in the sense that $A(x, \mathrm{Leak}(x, r)) \neq r$. Then we show that checking whether a candidate r for $R(x)$ is hard (again w.r.t. the first few algorithms) can be done by a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ algorithm. To conclude the argument for $R \in \Sigma_2\mathsf{P}$, we guess-and-verify a "good" value of $y \in R(x)$ by exploiting the connection between $\mathsf{BPP}_{||}^{\mathsf{SAT}}$ and prAM [4] together with the derandomization assumption on prAM.

We remark that both directions of the Liu-Pass result for prBPP relativize, which immediately implies an equivalence between the derandomization $\mathsf{prBPP}^{\mathsf{SAT}} \subseteq \mathsf{P}^{\mathsf{SAT}}$ and the existence of leakage-resilient hard functions computable in $\mathsf{P}^{\mathsf{SAT}}$ that are hard on almost-all inputs against probabilistic algorithms with SAT oracle. Since our objective is to obtain equivalences with respect to derandomizing prAM, and it is unknown whether (mild) derandomization of prAM implies derandomization of $\mathsf{prBPP}^{\mathsf{SAT}}$, the relativization approach is insufficient for our purposes.

Organization. In Section 2, we develop the ideas behind our results and relate them to existing techniques. We start the formal treatment in Section 3 with definitions, notation, and other preliminaries. In Section 4, we develop the equivalence between leakage-resilient hardness on almost-all inputs and mild derandomization of prAM (Theorem 3), and the equivalence between mild derandomization of prAM and the existence of targeted hitting-set generators (Theorem 5). The development of the equivalence between leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ and against learn-and-evaluate protocols (Theorem 4) and the discussion of local leakage resilience (Theorem 7) are relegated to the appendix. The details of the uniformization of SU and proofs of secondary results are deferred to the full version.

2 Technical overview

In this section, we present an overview of the techniques underlying our results. In order to describe the Liu-Pass approach and ours, we start with an overview of the learning reconstructive paradigm used in prior hardness vs. randomness tradeoffs for BPP and AM.

We then provide an overview of the hardness-to-derandomization direction of the Liu-Pass result in a way that facilitates the transition to our results on the mild derandomization setting for AM.

Learning reconstructive paradigm. Pseudorandom generator constructions G typically have a hard function h as a basis (where how "hard" h is depends on the context) and produce a pseudorandom distribution G^h that depends on h. The proof of correctness for such generators is *reconstructive*: It presents an algorithm (the reconstructor) that, given access to any process D that distinguishes the distribution G^h from a truly random distribution, as well as to some additional information a, computes the hard function h. It is common for the additional information a to be composed of evaluations of h at a small number of points, in which case we also say that the reconstruction is *learning*. Thus, unless G^h "fools" an efficient randomized process P on input x, the function h can be reconstructed efficiently from the distinguisher $D(r) \doteq P(x, r)$ and the evaluations.

Targeted setting. One way to obtain a targeted PRG is to use a hardness-based (oblivious) pseudorandom generator construction and instantiate it with a function h_x that depends on the input x (which we view as a circuit capturing the behavior of the algorithm A on the actual input as a function of the random bit string) [5, 15, 23]. In the leakage-resilient setting and given a hard function f, we take h_x to be the function whose truth-table is $f(x)$. The leakage-providing algorithm Leak uses access to $f(x)$ to provide the answers to the learning queries, which are then fed into the reconstructor to compute h_x and thus $f(x)$.

Liu and Pass [15] employ the Nisan-Wigderson (NW) pseudo-random generator construction [17] combined with the locally-list-decodable encoding of [21], and instantiate the PRG with the function $h_x : \{0,1\}^{\log n} \to \{0,1\}$ that computes the mapping $i \mapsto f(x)_i$. In the reconstruction, the leakage-providing algorithm Leak uses access to h_x to answer the learning queries for the NW reconstruction algorithm as well to identify a "good" random string to be used in the list-decoding step and the position of h_x in the resulting list. This allows A to compute the value of $f(x)$ by running the NW reconstruction followed by the list-decoding process with the "good" random string, outputting the element in the correct position of the resulting list.

In porting the Liu-Pass approach to the mild derandomization setting for prAM, we follow a similar idea but replace the NW construction by other hardness-based pseudorandom generator (PRG) or hitting-set generator (HSG) constructions that exhibit the learning property: In the case of failure of the PRG/HSG on an input x, the underlying hard function h can be reconstructed efficiently from the values of h at a small number of points. In particular, we make use of the following HSGs that have the learning property and are tailored for Arthur-Merlin protocols:

- The construction by Shaltiel and Umans [20], which we dub SU and use to derive our main results under leakage-resilient hardness against $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$. This construction scales throughout the entire derandomization spectrum and, in particular, allows for obtaining tighter connections with blackbox derandomization via our local-leakage-resilient hardness results. To employ this generator, we present a uniform version for its reconstruction.
- A recursive version of the construction by Miltersen and Vinodchandran [16] due to Shaltiel and Umans [20], which we dub RMV and use to derive our results under hardness against a specific type of Arthur-Merlin protocol that we call a *learn-and-evaluate* protocol. The original construction, denoted MV, is geared toward the high end of the derandomization spectrum, whereas the recursive variant works in a broader range.

We remark that the SU construction scales better than RMV toward the low end, and thus we choose to employ SU for our main results. The disadvantage is that SU requires hardness against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ instead of against Arthur-Merlin protocols. However, as evidenced in Theorem 3, such a hardness assumption is also required in the mild setting.

We now describe the two constructions in more detail and how we employ them to obtain our results.

Shaltiel-Umans construction. The SU generator takes any low-degree polynomial $\hat{g} : \mathbb{F}^r \to \mathbb{F}$, which is usually obtained via a low-degree extension/Reed-Müller encoding (in our case, of $y \in R(x)$ of length n), and produces a set of strings of length m, where m is a parameter. The reconstruction for the generator shows the existence of a small (of size $\mathsf{poly}(m, \log n)$) nondeterministic circuit computing $\hat{g}$ in case the construction fails as a hitting-set generator.

The original reconstruction for the generator requires access to a few pieces of information in relation to the field $\mathbb{F}^r$ and the polynomial $\hat{g}$: A generating matrix M for the set $\mathbb{F}^r \setminus \{\vec{0}\}$, two "good" low-degree curves C_1 and C_2 over $\mathbb{F}^r$ and evaluations of $\hat{g}$ over points that depend on M, C_1 and C_2. For our purposes, the leakage-providing algorithm can compute the matrix M efficiently enough via a brute-force search. As for the curves C_1 and C_2, Shaltiel and Umans show that a random choice has the required properties with high probability, which allows Leak to sample those at random. Finally, the evaluations of $\hat{g}$ can be computed efficiently given access to the value of $y \in R(x)$ by having Leak compute the low-degree extension $\hat{g}$ of y.

In the nondeterministic setting, there is an extra complication: The reconstruction also requires oracle access to a list of r predictors $P_1, \ldots, P_r$, all of which can be obtained from a distinguisher D for the generator. As the predictors themselves are nondeterministic, it is unclear how, for example, negative predictions can be computed nondeterministically. Shaltiel and Umans use a strategy (originating in [6]) to approximately and nondeterministically evaluate the predictors when given as non-uniform additional input approximations up to $O(\log n)$ bits of precision for the probability that each predictor outputs 1 over a random choice of inputs for the generator. The original reconstruction argument can handle a small amount of error in these probabilities due to the use of error correction. This is where the SAT oracle comes in: We can efficiently obtain the approximations required to run the reconstruction in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.

In the end, we obtain a probabilistic version of the reconstruction with non-adaptive oracle access to SAT that, given oracle access to $\hat{g}$ and to a distinguisher D, outputs with high probability a small nondeterministic circuit C that computes $\hat{g}$. We then take Leak to be the algorithm that computes C using the low-degree extension of $y \in R(x)$ to answer the learning queries. Recall that C has size $\mathsf{poly}(m, \log n)$ and y has length n. By picking a small value of m, the amount of leakage is indeed small. We take A to be a circuit-evaluation algorithm that evaluates C to recover the value of y. Note that since C computes $\hat{g}$, in particular it computes the mapping $i \mapsto y_i$ very efficiently, and thus the reconstruction can also be cast into the local leakage-resilience framework.

Recursive Miltersen-Vinodchandran construction. The RMV generator also takes any low-degree polynomial $\hat{g} : \mathbb{F}^r \to \mathbb{F}$ and a parameter m and outputs a set of strings of length m.

The RMV reconstructor for $\hat{g}$ forms a *commit-and-evaluate protocol*, a notion introduced in [20] for this reason. It is an Arthur-Merlin protocol that consists of two phases. The first phase is the commitment phase, where Arthur and Merlin interact to produce a commitment π. The commitment is then given as input to the evaluation phase, in which Arthur and Merlin compute evaluations of a function g_π that is determined by π and supposedly equals $\hat{g}$. As Merlin could cheat in the commitment phase, a key property of the protocol is its *resilience*: Given a commitment π, the evaluation protocol can only produce evaluations of a fixed function g_π, except with low probability. As g_π might differ from $\hat{g}$, one usually combines the construction with a checker (as in [10]) or a PCP system (as in [23]) to verify that $g_\pi = \hat{g}$.

In the commitment phase, Arthur samples a small (of size $\mathsf{poly}(m, \log n)$) set of points in $\mathbb{F}^r$ and Merlin provides evaluations of $\hat{g}$ at those points as the commitment π. With high probability over the set chosen by Arthur, given the correct commitment π as additional input and access to a distinguisher D, the evaluation protocol computes $\hat{g}$ with high confidence, no matter what strategy Merlin uses.

By replacing the interaction in the commitment phase by a probabilistic algorithm with oracle access to $\hat{g}$, the RMV reconstruction can be cast as a *learn-and-evaluate protocol*. A learn-and-evaluate protocol is a reconstructor for a function h that consists of two phases, where only the second phase requires access to the distinguisher D. The two phases are:

1. A learning phase, which is a randomized algorithm that can make queries to h and outputs a string, which we refer to as a sketch.
2. An evaluation phase, which is a promise Arthur-Merlin protocol with access to D that takes a sketch and an evaluation point z as input, and is supposed to output $h(z)$.

With high probability, given the correct answers to the queries, the learning phase should output a sketch such that the evaluator with access to a distinguisher D is complete and sound.

The learn-and-evaluate version of the reconstructor naturally fits the leakage-resilient hardness-to-derandomization framework, where the leakage-providing algorithm Leak is a probabilistic algorithm with access to $y \in R(x)$ that samples the set of points and produces evaluations of the low-degree extension $\hat{g}$ of y on those points and A is an Arthur-Merlin protocol that takes those evaluations as additional input to compute y. The amount of leakage is upper bounded by $\mathsf{poly}(m, \log n)$, which can be made small in relation to $|y| = n$ by picking a sufficiently small m.

3 Preliminaries

In this section, we present preliminary definitions and results that are necessary for developing our contributions. We start by defining targeted hitting-set generators, then move on to low-degree extensions and finally present some results on the class $\mathsf{BPP}_{||}^{\mathsf{SAT}}$. We assume familiarity with standard notions such as circuits and complexity classes such as NP, AM, and prAM. We measure circuit size by the number of gates. Most of our results extend to "robust" time bounds: We define a set of time bounds $\mathcal{T}$ as robust if for all polynomials p, q it holds that if $T \in \mathcal{T}$, then $p(n) \cdot T(q(n)) \in \mathcal{T}$.

Targeted hitting-set generators. Without loss of generality, Arthur-Merlin protocols have perfect completeness [7]. In this case, targeted hitting-set generators, the one-sided error variant of targeted PRGs, suffice for derandomization. We may naturally view an Arthur-Merlin protocol as a nondeterministic computation that receives an input x and a random sequence ρ, guesses a response from Merlin and accepts if and only if the final deterministic verification of Arthur accepts. Since such a computation can only make a mistake on a negative instance x, we wish for a generator that hits rejection, i.e., it produces a set S such that there is $\rho \in S$ that is rejected by the nondeterministic computation. As hitting-set generators are usually considered for the accepting side of computations, we instead consider generators for co-nondeterministic computations.

A co-nondeterministic computation on the random bits ρ for a given input x can be captured by a co-nondeterministic circuit. For this reason, in the context of derandomizing Arthur-Merlin protocols, we let targeted hitting-set generators take as input a co-nondeterministic circuit D that accepts at least a fraction $1/2$ of its inputs, and output a set of strings that "hits" D. If the co-nondeterministic computation runs in time $t(n)$ for inputs x of length n, then the co-nondeterministic circuit has size at most $t(n)^2$ for all but finitely many n. To accommodate different computational models for *generating* the hitting sets (including nondeterministic ones), we define targeted generators based on a relation between a circuit D and a hitting-set S. In the following definition, we let $\mathcal{C}$ denote a machine model and $\mathcal{C}\mathsf{TIME}[T]$ denote computational problems computable by machines in $\mathcal{C}$ that run in time T.

▶ **Definition 8.** *Let H be an algorithm that computes a relation $R \in \mathcal{C}\mathsf{TIME}[T(m)]$ between co-nondeterministic circuits of size m and sets of strings of length m. We say that H is a targeted hitting-set generator for co-nondeterministic circuits computable in $\mathcal{C}\mathsf{TIME}[T(m)]$ if the following two conditions hold for all sufficiently large $m \in \mathbb{N}$:*

- *For all co-nondeterministic circuits D of size m, there exists a non-empty S such that $(D, S) \in R$.*
- *For all co-nondeterministic circuits D of size m that accept at least a $1/2$ fraction of their inputs, it holds that for every $S \in R(D)$ there exists $\rho \in S$ such that $D(\rho)$ accepts.*

For some classes of algorithms, such as nondeterministic algorithms, the notion of computing a relation is intuitive: Upon receiving an input, the algorithm eventually halts and produces an output in the relation on every accepting computation path. For other classes of algorithms, such as $\Sigma_2\mathsf{P}$, the notion of computing a relation is perhaps less intuitive. We say that a Σ_2-algorithm N computes a relation R if for all $x \in \{0,1\}^*$, any accepting configuration of $N(x)$ outputs a value y such that $(x, y) \in R$.

We remark that the existence of a targeted hitting-set generator for co-nondeterministic circuits suffices for derandomizing all of prAM. For future reference, we include a statement and proof in the Σ_2 setting.

▶ **Proposition 9.** *Assume that there exists a targeted hitting-set generator H for co-nondeterministic circuits computable in $\Sigma_2\mathsf{TIME}[T(m)]$. Then $\mathsf{prAM} \subseteq \bigcup_{k\in\mathbb{N}} \Sigma_2\mathsf{TIME}[T(n^k)]$.*

Proof. Let $\Pi \in \mathsf{prAM}$, and let P be an Arthur-Merlin protocol (with perfect completeness) that runs in time n^k for some constant k and decides Π. It is standard that we can obtain from P and a sufficiently large input $x \in \{0,1\}^*$, in time $O(|x|^{2k})$, a co-nondeterministic circuit $D_{P,x}$ of size at most $|x|^{2k}$ such that:

$$x \in \Pi_Y \implies \Pr_r[D_{P,x}(r) = 1] = 0.$$
$$x \in \Pi_N \implies \Pr_r[D_{P,x}(r) = 1] \geq 1/2.$$

The Σ_2-simulation for Π works as follows on input x: First, it computes $D_{P,x}$, and guesses a set S output by $H(D_{P,x})$. Using another nondeterministic guess, the simulation verifies that $D_{P,x}$ rejects every $\rho \in S$, rejecting otherwise. In parallel, the simulation verifies, using an existential and a universal guess, that the guessed set S is an output of H for $D_{P,x}$, rejecting otherwise.

If $x \in \Pi_Y$, then $D_{P,x}$ rejects every $\rho \in S$ for any S output by $H(D_{P,x})$, in which case the overall simulation accepts. If $x \in \Pi_N$, then correctness of H implies that $D_{P,x}$ accepts some $\rho \in S$ for every S output by $H(D_{P,x})$, in which case the overall simulation rejects. The running time for the simulation is $T(n^{2k})$, which completes the proof. ◀

Low-degree extension. We also need standard low-degree extensions. Let $g : \{0,1\}^\ell \to \{0,1\}$ be a function, $\mathbb{F} = \mathbb{F}_p$ be the field with p elements (for prime p), h and r integers such that $h^r \geq 2^\ell$, and $[h]^r$ the r-tuples of elements in $\{1,\ldots,h\}$. The low-degree extension of g with respect to p, h, r is the unique r-variate polynomial $\hat{g} : \mathbb{F}^r \to \mathbb{F}$ with degree $h-1$ in each variable, for which $\hat{g}(\vec{v}) = g(y)$ for all $\vec{v} \in [h]^r$ representing a $y \in \{0,1\}^\ell$, and $\hat{g}(\vec{v}) = 0$ for $\vec{v} \in [h]^r$ that do not represent a string y. The total degree of $\hat{g}$ is $\Delta = hr$ and $\hat{g}$ is computable in time $\mathsf{poly}(h^r, \log p, r)$ given oracle access to g.

Non-adaptive oracle access to SAT. We first define the class $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$: We say that a promise problem $\Pi = (\Pi_Y, \Pi_N)$ is in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ if there exists a probabilistic algorithm M with non-adaptive oracle access to SAT such that for all $x \in \{0,1\}^*$:

$$x \in \Pi_Y \implies \Pr[M(x) = 1] \geq 2/3.$$
$$x \in \Pi_N \implies \Pr[M(x) = 1] \leq 1/3.$$

By non-adaptive oracle access, we mean that the queries made by M cannot depend on the answers to previous queries, i.e., the queries must all be made in parallel.

Arthur-Merlin protocols can be simulated by probabilistic algorithms with oracle access to SAT: To simulate the interaction, we sample a random sequence for Arthur and query the SAT oracle on whether there exists a response of Merlin that would lead to acceptance. In a similar fashion, $\mathsf{P}_{||}^{\mathsf{prAM}}$ can be simulated in $\mathsf{BPP}_{||}^{\mathsf{SAT}}$, the class of problems decidable by bounded-error probabilistic polynomial-time algorithms with non-adaptive oracle access to SAT. In fact, a converse also holds and is crucial in allowing our results to work under the assumption that prAM can be mildly derandomized.

▶ **Lemma 10** ([4]). $\mathsf{prBPP}_{||}^{\mathsf{SAT}} \subseteq \mathsf{P}_{||}^{\mathsf{prAM}}$.

In Lemma 10, the deterministic machines with oracle access to prAM on the right-hand side are guaranteed to work correctly irrespective of how the queries outside of the promise are answered, even if those queries are answered inconsistently, i.e., different answers may be given when the same query is made multiple times.

4 Equivalences for mild derandomization of prAM

In this section, we develop our main results, equivalences between mild derandomization of prAM and the existence of leakage-resilient hard functions/relations. We first state a more general version of Theorem 3 for classes $\mathcal{C}$ such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. For such classes, we obtain an equivalence with respect to a hard function.

▶ **Theorem 11.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

- $\mathsf{prAM} \subseteq \mathcal{C}$.
- *There exists a length-preserving function $f \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

Theorem 11 follows from Theorem 14 in Section 4.1 and Theorem 19 in Section 4.2. Examples of classes for which Theorem 11 applies are P^{NP} and $\mathsf{ZPP}^{\mathsf{NP}}$ and their time-$2^{\mathsf{polylog}(n)}$ and time-$2^{n^{o(1)}}$ variants.

In the case of $\Sigma_2\mathsf{P}$, we state a more general equivalence with respect to a hard relation.

▶ **Theorem 12.** *Let $\mathcal{T}$ be a robust set of time bounds. There exists a constant c such that for all $0 < \epsilon < 1$, the following are equivalent.*

- $\mathsf{prAM} \subseteq \bigcup_{T \in \mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$.
- *There exists $T' \in \mathcal{T}$ and a total length-preserving relation $R \in \Sigma_2\mathsf{TIME}[T'(n)]$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

Theorem 12 follows from Theorem 15 in Section 4.1 and Theorem 21 in Section 4.2. Similar to Theorem 11, Theorem 12 applies to the time-$\mathsf{poly}(n)$, time-$2^{\mathsf{polylog}(n)}$ and time-$2^{n^{o(1)}}$ variants of $\Sigma_2\mathsf{P}$. Theorem 3 follows by instantiating Theorems 11 and 12 with polynomial time bounds.

4.1 From leakage-resilient hardness to derandomization

Assuming the existence of a leakage-resilient hard function/relation against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, we show that we can obtain mild derandomization of prAM. Our approach is to instantiate the Shaltiel-Umans generator with the value of $f(x)$ (or some $y \in R(x)$ in case of a hard relation R). In case the generator fails, there exists a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ algorithm (the reconstructor) that given $f(x)$ as input outputs a small nondeterministic circuit computing the mapping $i \mapsto f(x)_i$. We take the reconstructor as the leakage-providing algorithm Leak and define A as a circuit-evaluation algorithm that evaluates the circuit output by Leak.

The next lemma presents a uniform version of the SU reconstruction that is particularly suited to our objectives. See the full version for a proof.

▶ **Lemma 13.** *There exists a deterministic algorithm H_{det} and a probabilistic algorithm* Learn *with non-adaptive oracle access to* SAT *such that at least one of the following holds for every $y \in \{0,1\}^n$, $m \in \mathbb{N}$ and co-nondeterministic circuit D of size m that accepts at least half of its inputs:*

1. *$H_{det}(1^m, y)$ outputs a set that hits D.*
2. *With probability at least $2/3$,* Learn$(1^m, y, D)$ *outputs a nondeterministic circuit C that computes the mapping $i \mapsto y_i$.*

H_{det} and Learn *run in time $\mathsf{poly}(m, n)$ and C has size $\mathsf{poly}(m, \log n)$. Moreover,* Learn *and C only need blackbox access to the deterministic predicate that underlies D.*

We are now ready to prove the hardness-to-derandomization direction of Theorem 11.

▶ **Theorem 14.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$ and $\mathsf{NP} \subseteq \mathcal{C}$. There exists a constant $c \geq 1$ such that the following holds. Assume that there exist a constant $0 < \epsilon < 1$ and a length-preserving function $f \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. Then there exists a targeted hitting-set generator H for co-nondeterministic circuits computable in $\mathcal{C}$, implying that $\mathsf{prAM} \subseteq \mathcal{C}$.*

Proof. Fix a binary string representation for co-nondeterministic circuits that describes a circuit of size m by a string of length $m' = \Theta(m \log m)$. The generator H, on input a string x of length m' describing a co-nondeterministic circuit D_x of size m, sets $n = m^a$ for a sufficiently large constant a to be defined later and sets $x' = x0^{n-m'}$. It then computes $f(x')$ and instantiates the generator H_{det} of Lemma 13, outputting the set $S_x = H_{\text{det}}(1^m, f(x'))$. Computing $f(x')$ can be done in $\mathcal{C}$ by the assumption that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$ and since $|x'| = \mathsf{poly}(m)$, and computing S_x from $f(x')$ takes deterministic time $\mathsf{poly}(m, n) = \mathsf{poly}(m)$, and therefore the generator H is computable in $\mathcal{C}$.

Assume, with the intent of deriving a contradiction, that H fails as a targeted HSG for co-nondeterministic circuits. This means that there exists an infinite set $\mathcal{D}$ of co-nondeterministic circuits that accept at least a $1/2$ fraction of their inputs such that $H(D)$ fails to hit every $D \in \mathcal{D}$. We show that there exist probabilistic algorithms Leak and A with non-adaptive oracle access to SAT running in time n^c for a constant c to be defined later such that for infinitely many x', $\mathrm{Leak}(x', f(x'))$ produces $|x'|^\epsilon$ bits of leakage and $A(x', \mathrm{Leak}(x', f(x')))$ computes $f(x')$ with high probability.

The algorithm Leak parses the input $x' \in \{0,1\}^n$ as $x' = x0^{n-m'}$ for $m' = \Theta(m \log m)$ and $m = n^{1/a}$. If x' is not of the expected type, it outputs 0^n. Then, it outputs $\mathrm{Learn}(1^m, f(x'), D_x)$, where D_x is the circuit described by x. The algorithm A is a circuit evaluation algorithm that, on input x' of length n and a nondeterministic circuit C, makes parallel queries to its SAT oracle about the output of C for each $i \in [n]$, outputting the concatenation of those (to hopefully obtain $f(x')$).

Notice that we measure the running times of Leak and A in terms of $n = m^a$. By Lemma 13, Leak runs in time $\mathsf{poly}(m, n) = \mathsf{poly}(n) \leq n^c$ for a sufficiently large constant c. By the same lemma, n^c serves as an upper bound for the running time of A, which is $n \cdot \mathsf{poly}(m, \log n)$. Moreover, the amount of leakage is $\mathsf{poly}(m, \log n) \leq (m \log n)^k$ for a sufficiently large constant k. By taking $a = 2k/\epsilon$, this is at most n^ϵ.

As for correctness, Lemma 13 guarantees that for all x such that $H(D_x)$ fails to hit D_x, $\mathrm{Learn}(1^m, x', f(x'))$ for $x' = x0^{n-m'}$ outputs with probability at least $2/3$ a circuit C that computes the mapping $i \mapsto f(x')_i$. In that case, A with inputs x' and C outputs $f(x')$. The conclusion $\mathsf{prAM} \subseteq \mathcal{C}$ follows as in Proposition 9: Given an Arthur-Merlin protocol P witnessing $\Pi \in \mathsf{prAM}$ and an input x, we construct $D_{P,x}$ and compute a hitting set $S = H(D_{P,x})$. Finally, we use the fact that $\mathsf{NP} \subseteq \mathcal{C}$ to verify in $\mathcal{C}$ that $D_{P,x}$ rejects all $\rho \in S$, rejecting otherwise. ◀

Essentially the same argument establishes the result for a leakage-resilient hard relation $R \in \Sigma_2\mathsf{TIME}[T(n)]$.

▶ **Theorem 15.** *There exists a constant $c \geq 1$ such that the following holds. Assume that there exist a constant $0 < \epsilon < 1$ and a length-preserving relation $R \in \Sigma_2\mathsf{TIME}[T(n)]$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. Then there exists targeted hitting-set generator H for co-nondeterministic circuits computable in $\Sigma_2\mathsf{TIME}[T(\mathsf{poly}(m)) + \mathsf{poly}(m)]$, implying that $\mathsf{prAM} \subseteq \bigcup_{k \in \mathbb{N}} \Sigma_2\mathsf{TIME}[T(n^k)]$.*

Proof (sketch). The proof follows the argument of Theorem 14 closely. Instead of computing $H_{\det}(1^m, f(x'))$, the generator H guesses and verifies $y \in R(x')$ and then outputs $H_{\det}(1^m, y)$, which leads to a generator computable in time $O(T(m^a) + \mathsf{poly}(m))$. The reconstruction is identical, and allows for computing $y \in R(x')$ given a small amount of leakage on y. ◀

We remark that the arguments for Theorems 14 and 15 show, in particular, that it is possible to compute $f(x')$ (or $y \in R(x')$) *locally* in time $\mathsf{poly}(m, \log n)$ by letting A be a regular circuit evaluation algorithm (instead of one that outputs the concatenation of evaluating the input circuit on every i).

4.2 From derandomization to leakage-resilient hardness

We first establish the derandomization-to-hardness direction of Theorem 11. As mentioned in Section 1, we frame the problem of computing a leakage-resilient hard function as a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem and then make use of a search-to-decision reduction as in [8].

We start by defining a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem.

▶ **Definition 16.** *Let R_Y and R_N be two disjoint binary relations. We say that (R_Y, R_N) is a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem if the following two conditions hold.*

1. *The decisional promise problem represented by (R_Y, R_N) is in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$; that is, there exists a probabilistic polynomial-time algorithm V with non-adaptive oracle access to SAT such that for every $(x, y) \in R_Y$ it holds that $\Pr[V(x, y) = 1] \geq 2/3$ and for every $(x, y) \in R_N$ it holds that $\Pr[V(x, y) = 1] \leq 1/3$.*
2. *There exists a probabilistic polynomial-time algorithm A with non-adaptive oracle access to SAT such that, for every x for which $R_Y(x) \neq \emptyset$, it holds that $\Pr[A(x) \in R_Y(x)] \geq 2/3$, where $R_Y(x) = \{y \mid (x, y) \in R_Y\}$.*

We observe that the search-to-decision strategy developed in [8] ports over to $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ problems.

▶ **Proposition 17.** *For every $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem (R_Y, R_N), there exists a binary relation R such that $R_Y \subseteq R \subseteq (\{0,1\}^* \times \{0,1\}^*) \setminus R_N$ and solving the search problem of R is (adaptively) deterministically polynomial-time reducible to some decisional problem in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

The following standard result will be useful to compose computations that have non-adaptive access to a SAT oracle.

▶ **Proposition 18** (See e.g., [19, Lemma 7.2])**.** *There exists a non-adaptive SAT-oracle algorithm M with the following behavior. On input $x \in \{0,1\}^n$ and the description of two non-adaptive SAT-oracle algorithms M_1 and M_2 such that M_1 runs in time time $t_1(n)$ and produces outputs of length $t_1(n)$ and M_2 takes inputs of length $t_1(n)$ and runs time $t_2(n)$, M runs in time $\mathsf{poly}(n, t_1(n), t_2(n))$ and outputs $M_2(M_1(x))$.*

Now, we are ready to prove Theorem 19.

▶ **Theorem 19.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. If $\mathsf{prAM} \subseteq \mathcal{C}$, then for all constants c and $0 < \epsilon < 1$ there exists a length-preserving function $f \in \mathcal{C}$ that is $(n^c, n - \omega(1))$-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, where $\omega(1)$ is polynomial-time computable.*

Proof. Our approach is to cast computing a leakage-resilient hard function f as a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem, which allows us to instantiate Proposition 17 and show that there is a $\mathsf{P}^{\mathsf{prBPP}_{||}^{\mathsf{SAT}}} \subseteq \mathsf{P}^{\mathsf{P}_{||}^{\mathsf{prAM}}} \subseteq \mathsf{P}^{\mathsf{prAM}}$ algorithm that solves it. Finally, we use the derandomization assumption together with the assumption on $\mathcal{C}$ to conclude that f is computable in $\mathcal{C}$.

First, we argue that for any x, a random choice of $f(x)$ is hard w.r.t. a fixed pair (Leak, A) of probabilistic algorithms with non-adaptive oracle access to SAT. Fix a constant c, an input $x \in \{0,1\}^n$ and a polynomial-time computable function $\nu(n) = \omega(1)$. Let ρ_{Leak} and ρ_A denote the random bits input to Leak and A, respectively. Let

$$P(r, \rho_{\mathrm{Leak}}, \rho_A) \equiv |\mathrm{Leak}(x, r; \rho_{\mathrm{Leak}})| \leq \ell \wedge A(x, \mathrm{Leak}(x, r; \rho_{\mathrm{Leak}}); \rho_A) = r,$$

for $\ell = n - \nu(n)$. We say that r fails if $\Pr_{\rho_{\mathrm{Leak}}, \rho_A}[P(r, \rho_{\mathrm{Leak}}, \rho_A)] \geq 1/6$.

We have that

$$\begin{aligned}
\Pr_r[r \text{ fails}] &= \Pr_r[\Pr_{\rho_{\text{Leak}},\rho_A}[P(r,\rho_{\text{Leak}},\rho_A)] \geq 1/6] && \text{[definition of failing } r\text{]} \\
&\leq 6 \cdot \mathbb{E}_r[\Pr_{\rho_{\text{Leak}},\rho_A}[P(r,\rho_{\text{Leak}},\rho_A)] && \text{[Markov's inequality]} \\
&= 6 \cdot \mathbb{E}_{r,\rho_{\text{Leak}},\rho_A}[P(r,\rho_{\text{Leak}},\rho_A)] && \text{[expectation of indicator variable]} \\
&= 6 \cdot \mathbb{E}_{\rho_{\text{Leak}},\rho_A}[\mathbb{E}_r[P(r,\rho_{\text{Leak}},\rho_A)]] && \text{[reordering random bits]} \\
&< 6\frac{2^{\ell+1}}{2^n} && \text{[pigeonhole argument]}
\end{aligned}$$

The pigeonhole argument is that, after fixing the random bits ρ_{Leak} and ρ_A, for each of the at most $2^{\ell+1}$ strings y of length at most ℓ, among all the strings r that Leak maps to y, there is at most one that A maps back to r. Setting $\ell = n - \nu(n)$ implies that the probability that a string r is "bad" is at most $12/2^{\nu(n)}$.

We then define the search problem (R_Y, R_N) such that $(x,r) \in R_Y$ if $|x| = |r| = n$ and for the first $\nu(n)$ pairs of probabilistic machines with non-adaptive oracle access to SAT (Leak, A) clocked to run in time n^c, it holds that

$$\Pr_{\rho_{\text{Leak}},\rho_A}[|\text{Leak}(x,r;\rho_{\text{Leak}})| \leq \ell \wedge A(x,\text{Leak}(x,r);\rho_A) = r] < \frac{1}{6}. \tag{1}$$

As for "no" instances, $(x,r) \in R_N$ if for at least one pair out of the first $\nu(n)$ (Leak, A), equation (1) with 1/6 replaced by 1/3 does not hold.

Now, we show that this problem is a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ search problem. On input $x \in \{0,1\}^n$, the algorithm for finding a solution samples a random $r \in \{0,1\}^n$. By a union bound over the $\nu(n)$ many algorithms Leak and A and the fact that r fails for a particular pair with probability at most $12/2^{\nu(n)}$, it holds that the solution-finding algorithm succeeds with probability at least 2/3 for sufficiently large n. On input (x,r), the verification algorithm enumerates the first $\nu(n)$ probabilistic machines with non-adaptive oracle access to SAT, Leak and A, all clocked to run in time n^c. It then estimates the value

$$p_{\text{Leak},A} = \Pr_{\rho_{\text{Leak}},\rho_A}[|\text{Leak}(x,r;\rho_{\text{Leak}})| \leq \ell \wedge A(x,\text{Leak}(x,r);\rho_A) = r]$$

up to error 1/12 and with failure probability at most $1/n$ for each pair (Leak, A). By a standard Chernoff bound, it suffices to perform the following steps a polynomial (in n) number of times per pair: Let Leak$'$ be the algorithm that, on input x and a random sequence ρ_{Leak} for Leak, computes an output $y = \text{Leak}(x,r;\rho_{\text{Leak}})$ and outputs (x,y). Let A' be an algorithm that, on input x, y and random sequence ρ_A, rejects if $|y| > \ell$ and outputs $A(x,y;\rho_A)$ otherwise, and let M be the algorithm of Proposition 18. Sample random strings ρ_{Leak} and ρ_A for Leak and A, respectively, compute $A'(x,\text{Leak}'(x,r;\rho_{\text{Leak}});\rho_A)$ by feeding M inputs $x, r, \rho_{\text{Leak}}, \rho_A$ and the codes of Leak$'$ and A' and compare the output with r. Each such execution requires time $\mathsf{poly}(n)$ for a total running time of $\mathsf{poly}(n)$. Moreover, all oracle queries made by M can be made in parallel.

After estimating the average acceptance probability for each pair (Leak, A), the verification algorithm outputs 1 if the estimated values are less than 1/4 (the midpoint between 1/6 and 1/3) for all pairs of algorithms. By a union bound over the $\nu(n)$ pairs of algorithms, the algorithm accepts $(x,r) \in R_Y$ and rejects $(x,r) \in R_N$ with probability at least 2/3.

Finally, we use Proposition 17 together with the derandomization assumption and the assumption on $\mathcal{C}$ to conclude that there is a function $f \in \mathcal{C}$ that solves (R_Y, R_N), i.e., f is a leakage-resilient hard function. ◀

The proof of Theorem 19 shows that, given x, verifying whether a candidate r for $R(x)$ is hard w.r.t. the first $\nu(n)$ algorithms Leak, A can be done in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. The proof also shows that a "good" r always exists for sufficiently large n. To extend Theorem 19 to obtain a hard relation $R \in \Sigma_2$, we first show that a Σ_2-derandomization assumption on prAM implies the same derandomization for $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. See the full version for a proof.

▶ **Proposition 20.** *Let $\mathcal{T}$ be a robust set of time bounds. If $\mathsf{prAM} \subseteq \bigcup_{T\in\mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$, then $\mathsf{prBPP}_{||}^{\mathsf{SAT}} \subseteq \bigcup_{T\in\mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$.*

Under a Σ_2-derandomization assumption for prAM, Proposition 20 allows a Σ_2 algorithm to compute a leakage-resilient hard relation R by guessing a candidate y and verifying in Σ_2 that it is a "good" solution. We therefore establish Theorem 21.

▶ **Theorem 21.** *Let $\mathcal{T}$ be a robust set of time bounds. If $\mathsf{prAM} \subseteq \bigcup_{T\in\mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$, then for every constant c there exists a total length-preserving relation $R \in \Sigma_2\mathsf{TIME}[T']$ for some $T' \in \mathcal{T}$ that is $(n^c, n-\omega(1))$-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, where $\omega(1)$ is polynomial-time computable.*

4.3 Targeted hitting-set generators from derandomization

In this section, we show that mild derandomization of prAM implies the existence of targeted hitting-set generators that suffice to obtain the original derandomization result. The results follow as consequences of the equivalence between leakage-resilient hardness and derandomization since we show the hardness-to-derandomization direction by constructing such targeted generators.

▶ **Corollary 22.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. If $\mathsf{prAM} \subseteq \mathcal{C}$, then there exists a targeted hitting-set generator for co-nondeterministic circuits computable in $\mathcal{C}$.*

Proof. By Theorem 14, there is a constant c such that if there is a length-preserving function $f \in \mathcal{C}$ that is $(n^c, n^{1/2})$-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$, then there is a targeted HSG as in the conclusion. By Theorem 19, the assumption $\mathsf{prAM} \subseteq \mathcal{C}$ implies the existence of f as required. ◀

Corollary 23 is established in the exact same way.

▶ **Corollary 23.** *Let $\mathcal{T}$ be a robust set of time bounds. If $\mathsf{prAM} \subseteq \bigcup_{T\in\mathcal{T}} \Sigma_2\mathsf{TIME}[T(n)]$, then there exists a targeted hitting-set generator for co-nondeterministic circuits computable in $\Sigma_2\mathsf{TIME}[T']$ for $T' \in \mathcal{T}$.*

Theorem 5 follows by combining Corollaries 22 and 23 with polynomial time bounds.

References

1 Leonard Adleman. Two theorems on random polynomial time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978. doi:10.1109/SFCS.1978.37.

2 Barış Aydınlıoğlu and Dieter van Melkebeek. Nondeterministic circuit lower bounds from mildly derandomizing Arthur-Merlin games. *Computational Complexity*, 26(1):79–118, 2017. doi:10.1007/s00037-014-0095-y.

3 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993. doi:10.1007/BF01275486.

4 Venkatesan T. Chakaravarthy and Sambuddha Roy. Arthur and Merlin as oracles. *Computational Complexity*, 20(3):505–558, 2011. doi:10.1007/s00037-011-0015-3.

5 Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *Symposium on Foundations of Computer Science (FOCS)*, pages 125–136, 2022. doi:10.1109/FOCS52979.2021.00021.

6 Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993. doi:10.1137/0222061.

7 Martin Furer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stahis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989.

8 Oded Goldreich. In a world of P=BPP. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 191–232. Springer, 2011. Part of the Lecture Notes in Computer Science book series (LNCS, volume 6650). doi:10.1007/978-3-642-22670-0_20.

9 Oded Goldreich. Two comments on targeted canonical derandomizers. In *Computational Complexity and Property Testing: On the Interplay Between Randomness and Computation*, pages 24–35. Springer, 2020. doi:10.1007/978-3-030-43662-9_4.

10 Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12(3):85–130, 2003. doi:10.1007/s00037-003-0178-7.

11 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002. doi:10.1016/S0022-0000(02)00024-7.

12 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing (STOC)*, page 220–229, 1997. doi:10.1145/258533.258590.

13 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

14 Yanyi Liu and Rafael Pass. Characterizing derandomization through hardness of Levin-Kolmogorov complexity. In *Computational Complexity Conference (CCC)*, volume 234, pages 35:1–35:17, 2022. doi:10.4230/LIPIcs.CCC.2022.35.

15 Yanyi Liu and Rafael Pass. Leakage-Resilient Hardness vs Randomness. In *Computational Complexity Conference (CCC)*, volume 264, pages 32:1–32:20, 2023. doi:10.4230/LIPIcs.CCC.2023.32.

16 Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur–Merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005. doi:10.1007/s00037-005-0197-7.

17 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

18 Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005. doi:10.1145/1059513.1059516.

19 Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006. doi:10.1007/s00037-007-0218-9.

20 Ronen Shaltiel and Christopher Umans. Low-end uniform hardness versus randomness tradeoffs for AM. *SIAM Journal on Computing*, 39(3):1006–1037, 2009. doi:10.1137/070698348.

21 Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom Generators without the XOR Lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.

22 Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003. doi:10.1016/S0022-0000(03)00046-1.

23 Dieter van Melkebeek and Nicollas Mocelin Sdroievski. Instance-wise hardness versus randomness tradeoffs for Arthur-Merlin protocols. In *Computational Complexity Conference (CCC)*, volume 264, pages 17:1–17:36, 2023. doi:10.4230/LIPIcs.CCC.2023.17.

24 R. Ryan Williams. Natural proofs versus derandomization. *SIAM Journal on Computing*, 45(2):497–529, 2016. doi:10.1137/130938219.

Appendices

In Appendix A, we develop the equivalence between leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$ and against learn-and-evaluate protocols (Theorem 4). In Appendix B, we present our results relating local leakage resilience to non-uniform lower bounds (Theorem 7).

A Leakage-resilient hardness against learn-and-evaluate protocols

In this appendix, we prove that the hardness-to-derandomization direction of Theorems 11 and 12 holds under hardness against learn-and-evaluate protocols, a specific type of Arthur-Merlin protocol. By combining such a result with the derandomization-to-hardness direction of Theorems 11 and 12, we obtain an equivalence in the leakage-resilient hardness setting between hardness against learn-and-evaluate protocols and hardness against $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$.

We start with some definitions. A *learn-and-evaluate protocol* is composed of two phases: A *learning* phase followed by an *evaluation* phase. In the learning phase, a probabilistic algorithm makes queries to a function f and produces an output (which we call a sketch). The evaluation phase then consists of an Arthur-Merlin protocol that computes $f(x)$ correctly on every input x when given the sketch as additional input. To define this notion, we first define what it means for an Arthur-Merlin protocol to produce an output.

▶ **Definition 24** (Arthur-Merlin protocol with output). *Let P be an Arthur-Merlin protocol. We say that on a given input $x \in \{0,1\}^*$:*

- *P outputs v with completeness c if there exists a Merlin strategy such that the probability that P succeeds and outputs v is at least c. In symbols: $(\exists M)\Pr[P(x,M)=v] \geq c$.*
- *P outputs v with soundness s if, no matter what strategy Merlin uses, the probability that P succeeds and outputs a value other than v is at most s. In symbols: $(\forall M)\Pr[P(x,M) \notin \{v,\perp\}] \leq s$.*

For a given function $f : X \to \{0,1\}^*$ where $X \subseteq \{0,1\}^*$, we say that P computes f with completeness $c(n)$ and soundness $s(n)$ if on every input $x \in X$, P outputs $f(x)$ with completeness $c(|x|)$ and soundness $s(|x|)$.

Now, we define learn-and-evaluate protocols.

▶ **Definition 25** (Learn-and-evaluate protocol). *A learn-and-evaluate protocol P consists of a probabilistic algorithm A_{learn} and an Arthur-Merlin protocol P_{eval}. Let $g : X \to \{0,1\}^*$ where $X \subseteq \{0,1\}^*$. We say that P computes g with error $e(n)$ for completeness $c(n)$ and soundness $s(n)$ if on every input $x \in X$ of length n the following hold: The probability over the randomness of A_{learn} that P_{eval} with input x and additional input $\pi = A^f_{learn}(1^n)$ outputs $f(x)$ with completeness $c(n)$ and soundness $s(n)$ is at least $1 - e(n)$.*

We define (T, ℓ)-leakage-resilient hardness on almost-all inputs against learn-and-evaluate protocols in a completely analogous way to hardness against $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$ (Definition 1), where a probabilistic algorithm $\text{Learn}(x, y)$ (which we take to be A^y_{learn}) takes on the role of Leak, running in time T and producing, on input x of length n, an output of length at most $\ell(n)$, and P_{eval} takes on the role of algorithm A, also running in time T.

We make use of the following lemma, which hinges on the RMV generator. See the full version for a proof.

▶ **Lemma 26.** *There exists a deterministic algorithm H_{det}, a probabilistic algorithm* Learn *and an Arthur-Merlin protocol P_{eval} such that at least one of the following holds for every $y \in \{0,1\}^n$, $m \in \mathbb{N}$ and co-nondeterministic circuit D of size m that accepts at least half of its inputs:*

1. *$H_{det}(1^m, y)$ outputs a set that hits D.*
2. *With probability at least $2/3$,* Learn$(1^m, y)$ *outputs a sketch π such that $P_{eval}(\pi, D, \cdot)$ computes the mapping $i \mapsto y_i$ with completeness 1 and soundness $1/n^2$.*

H_{det} and A_{learn} run in time $\mathsf{poly}(n, m)$, and A_{learn} produces a sketch π of length $\mathsf{poly}(m, \log n)$. P_{eval} runs in time $(m \cdot \log n)^{O(\log^2 r)}$ for $r = O(\log n / \log m)$. Moreover, P_{eval} only needs blackbox access to the deterministic predicate that underlies D.

We remark that the RMV reconstruction does not scale as well as the SU reconstruction, where the analogue of P_{eval} runs in time $\mathsf{poly}(m, \log n)$.

We now state the hardness-to-derandomization result that we obtain from Lemma 26.

▶ **Theorem 27.** *Theorems 14 and 15 continue to hold when $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ is replaced by learn-and-evaluate protocols.*

Proof. The proof follows closely that of Theorems 14 and 15 but uses the generator H_{det} of Lemma 26 instead of Lemma 13. For the reconstructor's running time, since we pick $n = \mathsf{poly}(m)$, we can upper bound the running time of Leak (which equals the algorithm Learn of Lemma 26) and the leakage-receiving protocol (which invokes P_{eval} n times) by n^c for some constant c. The bound on the leakage is calculated in the exact same way. ◀

As a consequence, we obtain equivalence between hardness on almost-all inputs against polynomial-time learn-and-evaluate protocols and leakage-resilient hardness on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. For simplicity, we state the result for classes $\mathcal{C}$ such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$, but it holds for Σ_2 as well.

▶ **Corollary 28.** *Let $\mathcal{C}$ be a complexity class such that $\mathsf{P}^{\mathcal{C}} \subseteq \mathcal{C}$. There exists a constant c such that the following are equivalent for all $0 < \epsilon < 1$:*

1. *There exists a length-preserving function $f \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against learn-and-evaluate protocols.*
2. *There exists a length-preserving function $f \in \mathcal{C}$ that is (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

Proof. We start with the $1 \implies 2$ implication. If 1 holds, then by Theorem 27 it follows that $\mathsf{prAM} \subseteq \mathcal{C}$. This in turn implies 2 by Theorem 11. As for the other direction, if a function f is not (n^c, n^ϵ)-leakage-resilient hard on almost-all inputs against learn-and-evaluate protocols, then it is also not (n^c, n^ϵ)-leakage-resilient hard against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ since a $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$ algorithm can use the SAT oracle to determine the output of the evaluation protocol. ◀

B Connection to non-uniform lower bounds

In this appendix, we prove Theorem 7, which establishes further equivalences between whitebox and blackbox mild derandomization of prAM. In particular, we show that uniform leakage-resilient hardness assumptions imply the $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$ separation.

We start by presenting the equivalence between lower bounds for $\Sigma_2\mathsf{E}$ against nondeterministic circuits and against deterministic circuits with non-adaptive oracle access to SAT. Since we only need this equivalence in the case of polynomial-size circuit lower bounds, we state it in this setting.

▶ **Lemma 29** (Instantiation of [19, Theorem 3.2]). *$\Sigma_2\mathsf{E} \subseteq \mathsf{NP/poly}$ if and only if $\Sigma_2\mathsf{E} \subseteq \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$.*

Proof. The direction $\Sigma_2\mathsf{E} \subseteq \mathsf{NP/poly} \implies \Sigma_2\mathsf{E} \subseteq \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ is trivial, so we focus on the converse implication. Theorem 3.2 in [19] shows that if a low-degree extension $\hat{g}$ with specific parameters of a function g has non-adaptive SAT-oracle circuits of size s, then g itself has nondeterministic circuits of size $\mathsf{poly}(s)$. Assume that $\Sigma_2\mathsf{E} \subset \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$. By a standard argument, $\Sigma_2\mathsf{E}/n \subseteq \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ and there exists a constant c such that every language in $\Sigma_2\mathsf{E}/n$ has non-adaptive SAT-oracle circuits of size n^c.

Let $L \in \Sigma_2\mathsf{E}$. By having as advice n bits describing the number of strings of length n in L, a Σ_2-machine can compute the characteristic function $g_L : \{0,1\}^n \to \{0,1\}$ of L by guessing which strings of length n are in L and verifying each one in $\Sigma_2\mathsf{E}$, outputting 1 if and only if the input string is in the list of guessed-and-verified strings. We then set parameters exactly as in [19], that is, for a parameter $r' = 2(n + \log(32n^{5c}))$, we set $h = (4r')^2(9n^c)^4$, $r = n/\log h + 3$ and p to the smallest prime greater than or equal to $9hdr'$ to obtain the low-degree extension $\hat{g}$ of g. With these parameters, it follows that the function $\hat{g}_{\text{bool}}$ that maps the binary representation of a $\vec{y} \in \mathbb{F}_p^r$ and an index $i \in [\log p]$ to the i-th bit of $\hat{g}(\vec{y})$ is in $\Sigma_2\mathsf{E}/n$. Together with the assumption on $\Sigma_2\mathsf{E}$, we have that $\hat{g}$ has non-adaptive SAT-oracle circuits of size $s(n) = O(n^{c+1})$, and thus Theorem 3.2 in [19] guarantees that g (and thus L) has nondeterministic circuits of size $\mathsf{poly}(s(n)) = \mathsf{poly}(n)$. ◀

Now, we provide some intuition for the proof of Theorem 7. The first step is to understand how lower bounds such as $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$ imply leakage-resilient hardness. First, by Lemma 29, the assumption implies also that $\Sigma_2\mathsf{E} \not\subseteq \mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$. Define a function f that maps every input x of length ℓ to the truth-table of a hard language $L \in \Sigma_2\mathsf{E}$. With ℓ bits of advice (indicating how many strings of length ℓ are in L), this function can be computed in Σ_2-time $2^{O(\ell)}$. Assume, with the intent of deriving a contradiction, that f is not locally leakage-resilient hard on almost-all inputs against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. For each input x where hardness fails, there exists a small leakage string a such that $A(x,a)$ locally computes f. That is, $L \in \mathsf{BPP}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ and thus also in $\mathsf{P}_{||}^{\mathsf{SAT}}/\mathsf{poly}$ by Adleman's argument [1]. As for the other direction, by using the techniques developed in Section 4, we show that leakage-resilient hardness is sufficient for obtaining the mild derandomization of item 1, and thus equivalent to non-uniform lower bounds in the low end by [2].

We now detail how we prove Theorem 7. It is already known that items 1 and 2 are equivalent by the main result of [2]. We show that $2 \implies 3$ in Lemma 30, that $4 \implies 1$ in Lemma 31 and that $1 \iff 5$ in Lemma 33. The $3 \implies 4$ implication is trivial.

We start by obtaining leakage-resilient hardness from the assumption $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$.

▶ **Lemma 30.** *Assume $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP/poly}$. Then for all $\epsilon > 0$ there exists a relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is $\mathsf{poly}(n)$-local $(\infty, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$.*

Proof. Let $L \in \Sigma_2\mathsf{E}$ and fix $\epsilon > 0$. We construct a function f from L such that, if f is not hard as in the theorem statement, then $L \in \mathsf{NP/poly}$. The existence of a hard relation then follows. Define f as the function that maps any input $x \in \{0,1\}^n$ to the truth-table of L at input length $n^{\epsilon/2}$. By having as advice the number N of strings of length $n^{\epsilon/2}$ in L, it is possible to compute this function in Σ_2-time $2^{O(n^{\epsilon/2})} \leq 2^{n^\epsilon}$ for sufficiently large n by guessing which N of the $2^{n^{\epsilon/2}}$ strings of length $n^{\epsilon/2}$ are in L and verifying those using the linear-exponential time Σ_2-algorithm for L. Note that since $f(x)$ is constant for all inputs of a given length, the same advice string applies to all such inputs.

Now, assume that f is not $\mathsf{poly}(n)$-local $(\infty, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$. This means there exist a constant c, a (potentially uncomputable) function Leak and a probabilistic algorithm A with non-adaptive oracle access to SAT such that for almost-all input lengths n there exists $x \in \{0,1\}^n$ such that $|\text{Leak}(x, f(x))| \leq n^c$ and $A(x, \text{Leak}(x, f(x)))$ computes $f(x)$ locally in time n^c. By providing x and a "good" leakage string a as advice, algorithm $A(x, a)$ computes f locally in time n^c, and thus computes L at input length $n^{\epsilon/2}$ in time $\mathsf{poly}(n)$. This implies that $L \in \mathsf{BPP}^{\mathsf{SAT}}_{||}/\mathsf{poly}$ and thus $L \in \mathsf{P}^{\mathsf{SAT}}_{||}/\mathsf{poly}$ [1]. By Lemma 29, $L \in \mathsf{NP}/\mathsf{poly}$. ◀

Now, we show that a weaker hardness assumption, where the leakage-providing function is computable in subexponential time, suffices to derandomize prAM.

▶ **Lemma 31.** *If for all $\epsilon > 0$ there exists a relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is $\mathsf{poly}(n)$-local $(2^{n^\epsilon}, \mathsf{poly}(n))$-leakage-resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$, then $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$.*

Proof. The proof is almost identical to that of Theorem 15. Fix some $\epsilon > 0$, the idea to construct a targeted hitting-set generator is to, on input $x \in \{0,1\}^{m'}$ representing a circuit D_x of size m for $m' = \Theta(m \log m)$, guess-and-verify a value $y \in R(x)$ and instantiate the generator H_{det} of Lemma 13 with y and m. The process takes time $2^{O(m^\epsilon)}$ and requires $O(m^\epsilon)$ bits of advice (for computing $y \in R(x)$). As with Theorem 15, there exist probabilistic algorithms Leak and A with non-adaptive oracle access to SAT such that for any $x \in \{0,1\}^n$ representing a circuit D_x not hit by the generator there exists $y \in R(x)$ for which the following holds. On input (x, y), Leak runs in time $2^{O(n^\epsilon)}$ and produces $\mathsf{poly}(n)$ bits of leakage that, when given as input to A, allow it to locally compute y in time $\mathsf{poly}(n, \log(2^{n^\epsilon})) = \mathsf{poly}(n)$.

As the hardness assumption holds for every input of infinitely-many input lengths (and R is total on the same input lengths), the targeted generator also works for infinitely-many circuit sizes m, and thus the derandomization in the conclusion of the theorem follows along the lines of Proposition 9 together with padding. ◀

Finally, we show that the weak derandomization assumption of item 1 in Theorem 7 is equivalent to local leakage-resilient hardness against almost-maximal leakage. Before doing so, we present a variant of Proposition 20 for derandomizations with advice that only work for infinitely-many input lengths.

▶ **Proposition 32.** *Assume that $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$. Then $\mathsf{prBPP}^{\mathsf{SAT}}_{||} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$.*

Proof (sketch). In [23, Lemma 38], it is shown that the hardness assumption $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP}/\mathsf{poly}$ implies that $\mathsf{prBPP}^{\mathsf{SAT}}_{||} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ for all $\epsilon > 0$ by using traditional hardness vs. randomness tradeoffs in the blackbox setting [22]. As the premise $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ implies that $\Sigma_2\mathsf{E} \not\subseteq \mathsf{NP}/\mathsf{poly}$, we are done. ◀

▶ **Lemma 33.** *The following are equivalent:*

1. $\mathsf{prAM} \subseteq \text{io-}\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ *for all $\epsilon > 0$.*
5. *For all $\epsilon > 0$ and $c \geq 1$ there exists a length-preserving relation $R \in \Sigma_2\mathsf{TIME}[2^{n^\epsilon}]/n^\epsilon$ that is n^c-local $(n^c, \ell(n))$-leakage resilient hard on all inputs of infinitely-many input lengths against $\mathsf{prBPP}^{\mathsf{SAT}}_{||}$, where $n^{\Omega(1)} \leq \ell(n) \leq \omega(1)$ is polynomial-time computable.*

Proof. The proof follows closely the arguments of Theorems 21 (in the derandomization-to-hardness direction) and 15 (in the hardness-to-derandomization direction).

In the derandomization-to-hardness direction, recall that the proof of Theorem 21 shows that for every constant c, checking whether a value y is $(n^c, n - \omega(1))$ leakage-resilient hard for a string x (w.r.t. the first few algorithms) can be done in $\mathsf{prBPP}_{||}^{\mathsf{SAT}}$. Such a value y must also be n^{c-2}-local leakage-resilient hard. Otherwise, we could amplify the algorithm A that locally computes y so that it outputs every bit of y correctly with high probability and obtain an algorithm that computes y in its entirety in time n^c. Fix some $\epsilon > 0$. By the derandomization assumption and Proposition 32, the check can be replaced by a $\Sigma_2\mathsf{TIME}[2^{n^\epsilon}]$ algorithm with n^ϵ bits of advice that is guaranteed to work on infinitely-many input lengths. By guessing a value of y and running the Σ_2 verification, we obtain a relation R that is defined and hard for all inputs of infinitely-many input lengths.

The hardness-to-derandomization direction follows Theorem 15 even more closely. Theorem 15 establishes the hardness-to-targeted-derandomization connection on an instance-wise basis, i.e., the targeted generator hits every co-nondeterministic circuit D_x described by a padded string x' for which the hardness assumption holds. Since hardness holds for all inputs of infinitely-many input lengths, we obtain a targeted generator that works for infinitely many circuit sizes, which as in Lemma 31 suffices to obtain the derandomization result. ◀

We remark that due to the gap between the length of a solution $y \in R(x)$ and the leakage-resilient hardness in items 3 and 4 together with the sub-optimal scaling of the RMV reconstructor (Lemma 26) when compared to the SU reconstructor (Lemma 13), this result does not extend to hardness against learn-and-evaluate protocols.