Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 193 (2023)

The hardness of range avoidance for randomized algorithms implies Minicrypt[*]

Eldon Chung[†] Alexander Golovnev[‡] Zeyong Li[§] Maciej Obremski[¶]
Sidhant Saraogi[‖] Noah Stephens-Davidowitz[**]

Abstract

We study the Range Avoidance Problem (Avoid), in which the input is an expanding circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$, and the goal is to find a $y \in \{0,1\}^{n+1}$ that is *not* in the image of C. We are interested in the *randomized* complexity of this problem, i.e., in the question of whether there exist efficient randomized algorithms that output a valid solution to Avoid with probability significantly greater than $1/2$. (Notice that achieving probability $1/2$ is trivial by random guessing.)

Our main result shows that cryptographic one-way functions exist unless Avoid can be solved efficiently with probability $1 - 1/n^c$ (on efficiently sampleable input distributions). In other words, even a relatively weak notion of hardness of Avoid (for randomized algorithms) already implies the existence of all cryptographic primitives in Minicrypt.

In fact, we show something stronger than this. In particular, we introduce two new natural problems, which we call CollisionAvoid and AffineAvoid. Like Avoid, these are total search problems in the polynomial hierarchy. They are provably at least as hard as Avoid, and seem to be notably harder. We show that auxiliary-input one-way functions exist if either of these problems is weakly hard in the worst case, and standard one-way functions exist if either of these problems is weakly hard on average.

We also show that (1) Avoid can be solved by an efficient zero-error randomized algorithm given access to a suitable oracle that approximates the Kolmogorov-Levin complexity of a bit string; and that (2) simple reductions from hard problems in FNP to Avoid are unlikely. These latter results can be viewed as variants of known results from Ren, Santhanam, and Wang (FOCS 2022), and Ilango, Li, and Williams (STOC 2023).

[*]An earlier version of this work used the title "On the randomized complexity of range avoidance, with applications to cryptography and metacomplexity."

[†]Centre for Quantum Technologies, National University of Singapore. Email: `echung.math@gmail.com`.

[‡]Georgetown University. Email: `alexgolovnev@gmail.com`.

[§]Centre for Quantum Technologies, National University of Singapore. Email: `li.zeyong@u.nus.edu`.

[¶]Centre for Quantum Technologies, National University of Singapore. Email: `obremski.math@gmail.com`.

[‖]Georgetown University. Email: `ss4456@georgetown.edu`.

[**]Cornell University. Email: `noahsd@gmail.com`.

ISSN 1433-8092

Contents

1 Introduction

We study the Range Avoidance Problem (Avoid). This is the computational search problem in which the input is a circuit $C : \{0,1\}^n \to \{0,1\}^{n+s}$ with some *stretch* $s > 0$, and the goal is to find $y \in \{0,1\}^{n+s}$ that is *not* in the image of C (i.e., for all $x \in \{0,1\}^n$, $C(x) \neq y$). Avoid can be thought of as the computational problem corresponding to the "dual pigeonhole principle," which states that an expanding function cannot be surjective. In this introduction, we will primarily be interested in the case when the stretch is taken to be $s = 1$.

Avoid was recently introduced by Kleinberg, Korten, Mitropolsky, and Papadimitriou [KKMP21] as an example of a total search problem that is unlikely to be in FNP but *does* lie in the polynomial hierarchy, i.e. Avoid $\in \mathsf{TF}\Sigma_2$. There has since been much follow-up work on the complexity of Avoid because of surprising connections to derandomization [Kor22, GLW22, GGNS23], circuit complexity [RSW22, CHLR23, CHR24, Li24, KP24, LORS24], and other areas of complexity theory [ILW23, CL24].

In this work, we study the *randomized* complexity of Avoid, i.e., the hardness (or easiness?) of solving Avoid using randomized algorithms. Here, one must be rather precise about what one means. In particular, notice that the very simple (and efficient!) algorithm that simply outputs a random string in $\{0,1\}^{n+s}$ already solves Avoid with probability at least $1 - 2^{-s} \geq 1/2$. To see this, simply notice that the set of all images of C has size at most 2^n, while the range $\{0,1\}^{n+s}$ has size 2^{n+s}, so that at most a 2^{-s} fraction of the bitstrings of length $n+s$ are in the image of C. This property makes Avoid stand out quite a bit among search problems that are thought to be "hard" (e.g., hard for deterministic algorithms).[1] This peculiarity was already observed in [KKMP21], and it is the starting point of our work.

Specifically, we are interested in the question of how high the success probability of an efficient randomized algorithm can be. As far as the authors know, it is open to find even a $2^{(1-\varepsilon)n}$-time algorithm that solves Avoid with probability even, say, $1/2 + 1/n$ with stretch $s = 1$.[2]

1.1 Our results

1.1.1 Hardness of Avoid implies cryptography

Our main contribution is a proof that if Avoid is "even slightly hard" for randomized algorithms in the worst case, then auxillary-input one-way functions exist.

Theorem 1.1. *If there is a constant $c > 0$ such that no (randomized) polynomial-time algorithm solves* Avoid *with probability larger than $1 - 1/n^c$, then auxiliary-input one-way functions exist.*

Similarly, if Avoid is "even slightly hard" for randomized algorithms for some efficiently sampleable family of distributions, then cryptographic one-way functions exist.

[1] Much of the prior work on Avoid was interested in the complexity of Avoid *relative to an* NP oracle. However, the same idea shows that there is a trivial $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm for Avoid (just sample a random string y and use the NP oracle to check the coNP statement of whether y is outside the image of the circuit). In fact, subject to some caveats, there is even a pseudodeterministic algorithm for Avoid relative to an NP oracle [CHR24, Li24]. So, in our context of the randomized complexity of Avoid, we do not allow ourselves an NP oracle.

[2] Of course, it is trivial to solve Avoid in roughly 2^n time by simply enumerating the entire image of the input circuit C. And, in less than 2^n time, one can do very slightly better than $1-2^{-s}$. In particular, for any $T \leq 2^n$, one can solve Avoid with probability at least $1-2^{-s}+T/2^{n+s}$ in time roughly T simply by returning a random element $y \in \{0,1\}^{n+s} \setminus S$ where $S := \{C(x_i)\}_{i=1}^{T}$ is the set of images of any choice of distinct inputs $x_1, \ldots, x_T \in \{0,1\}^n$.

Theorem 1.2. *If there is some efficiently sampleable family of distributions of expanding circuits $\mathcal{D}_n$ and a constant $c > 0$ such that no (randomized) polynomial-time algorithm solves* Avoid *with probability larger than $1 - 1/n^c$ on input distribution $\mathcal{D}_n$, then one-way functions exist.*

Recall that, as far as we know, no efficient algorithm solves Avoid with probability even slightly larger than $1/2$. So, these assumptions seem relatively weak. Of course, if one-way functions exist, then it is well known that many other cryptographic primitives of interest exist as well, such as pseudorandom generators, pseudorandom functions, secret-key encryption, digital signatures, etc. I.e., if Avoid is suitably hard, then all cryptographic primitives in MiniCrypt exist [Imp95].

At a high level, our proof of Theorem 1.1 is quite intuitive. Indeed, recall that there is a simple algorithm that solves Avoid (with $s = 1$) with probability $1/2$ by simply outputting a uniformly random string $y \sim \{0,1\}^{n+1}$. To boost the success probability of this algorithm, we can try to determine whether y is in fact in the image of the input circuit C. If we guess that y is in the image, then we can resample and try again. The basic idea behind our proof is simply to treat C as a candidate one-way function and to observe that an oracle that inverts C with non-negligible probability on random inputs can be used to guess whether y is in the image of C with non-negligible advantage, i.e., with probability $1/2 + 1/\operatorname{poly}(n)$. So, if no such guesser exists, then C itself must be a one-way function. (Of course, here we are hiding important technical issues that arise in the proof. E.g., we must use tricks from cryptography to boost the success probability from $1/2 + 1/\operatorname{poly}(n)$ to $1 - 1/\operatorname{poly}(n)$.)

1.1.2 Two harder variants of Avoid (whose hardness also implies cryptography)

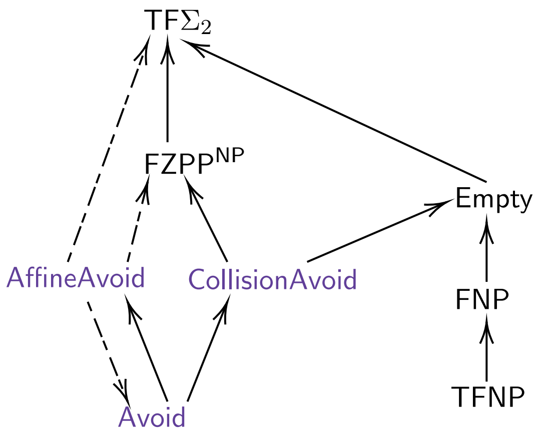

Figure 1: Relationships between different computational problems and complexity classes. Solid arrows represent reductions or containment. (Here, we are abusing notation quite a bit and conflating complexity classes with individual computational problems.) Dashed arrows represent reductions or containments that only work in some parameter regimes. See Section 3. The problems marked in purple are problems whose hardness implies cryptography.

We also introduce two new total search problems that are closely related to Avoid, which we call $\mathsf{AffineAvoid}$ and $\mathsf{CollisionAvoid}$. In fact, both are at least as hard as Avoid, and we show analogues of Theorem 1.1 for both problems. (See also Section 3 for some basic results about these two problems.)

In $\mathsf{AffineAvoid}$, the input is again an expanding circuit $C : \{0,1\}^n \to \{0,1\}^{n+s}$ for $s > 0$. However, now instead of asking for a *single* point $y \in \{0,1\}^{n+s}$ that is not in the image of C, we ask for an entire d-dimensional affine subspace $S \subseteq \{0,1\}^{n+s}$ such that all points in S are not in the image of C. (Here, we define an affine subspace of $\{0,1\}^{n+s}$ via the natural identification of $\{0,1\}^{n+s}$ with the vector space $\mathbb{F}_2^{n+s}$. The affine subspace can of course be represented succinctly as the affine span of $d+1$ vectors in $\mathbb{F}_2^{n+s}$.) This problem is trivially at least as hard as Avoid, and for suitably chosen d we show (using Gowers norms) that the problem is total. Indeed, for the parameters that interest us most, we show that a random affine subspace of dimension $d = \log\log n$ is a valid solution to $\mathsf{AffineAvoid}$ with probability at least $1/(2n)$, which in particular implies that the problem is in $\mathsf{FZPP}^{\mathsf{NP}} \subseteq \mathsf{TF}\Sigma_2$ for these parameters, like Avoid.[3]

In $\mathsf{CollisionAvoid}$, the input is a *length-preserving* circuit $C : \{0,1\}^n \to \{0,1\}^n$. The goal is now *either* to output an element $y \in \{0,1\}^n$ that is not in the image of C *or* to output $x \in \{0,1\}^n$ such that the image $C(x)$ is unique (and to state whether the output is a non-image or an element with a unique image). Notice that $\mathsf{CollisionAvoid}$ is still a total problem, since either C is a bijection (in which case every $x \in \{0,1\}^n$ has a unique image), or it is not (in which case there is an element not in the image). Furthermore, we show that it is at least as hard as Avoid and that it is contained in $\mathsf{FZPP}^{\mathsf{NP}} \subseteq \mathsf{TF}\Sigma_2$. Indeed, we observe that, like for Avoid and $\mathsf{AffineAvoid}$, there is a simple input-independent distribution of outputs that yields a solution to $\mathsf{CollisionAvoid}$ on any circuit C with probability at least $1/3$. (The relationship between $\mathsf{CollisionAvoid}$ and Avoid seems at least superficially similar to the relationship between Papadimitriou's celebrated Pigeon and $\mathsf{WeakPigeon}$ problems [Pap94].)

Both of these new problems seem to be harder than Avoid. However, we show that even hardness of $\mathsf{CollisionAvoid}$ or $\mathsf{AffineAvoid}$ implies the existence of auxiliary-input one-way functions. (Theorem 1.1 can be viewed as a corollary of either Theorem 1.3 or Theorem 1.5, and similarly Theorem 1.2 can be viewed as a corollary of either Theorem 1.4 or Theorem 1.6. In fact, we do not bother to prove Theorems 1.1 and 1.2 directly, but instead prove the stronger theorems below.)

Theorem 1.3. *If there is a constant $c > 0$ such that no (randomized) polynomial-time algorithm solves* $\mathsf{AffineAvoid}$ *with probability larger than $1 - 1/n^c$, then auxiliary-input one-way functions exist.*

Theorem 1.4. *If there is some efficiently sampleable family of distributions of expanding circuits $\mathcal{D}_n$ and a constant $c > 0$ such that no (randomized) polynomial-time algorithm solves* $\mathsf{AffineAvoid}$ *with probability larger than $1-1/n^c$ on input distribution $\mathcal{D}_n$ with $d = \log\log n$, then one-way functions exist.*

Theorem 1.5. *If there is a constant $c > 0$ such that no (randomized) polynomial-time algorithm solves* $\mathsf{CollisionAvoid}$ *with probability larger than $1 - 1/n^c$, then auxiliary-input one-way functions exist.*

[3]One of the reasons that $\mathsf{AffineAvoid}$ interests us is the following. One can easily reduce the problem of finding a $(d+\ell)$-dimensional affine subspace outside of the image of a circuit $C : \{0,1\}^n \to \{0,1\}^{n+s+\ell}$ to the problem of finding a d-dimensional affine subspace outside of the image of a circuit $C' : \{0,1\}^n \to \{0,1\}^{n+s}$. This reduction is a bit more satisfying than the trivial reduction from Avoid with stretch $s+\ell$ to Avoid with stretch s. In particular, the reduction between $\mathsf{AffineAvoid}$ with different parameters essentially preserves the probability that a random subspace of the appropriate dimension is a valid output, while the reduction between Avoid instances does not preserve the probability that a random string is a valid output. This suggests that the complexity of $\mathsf{AffineAvoid}$ might not be too dependent on the stretch s. (Avoid with stretch s is known to be equivalent to Avoid with stretch s' for any $1 \le s, s' \le \mathrm{poly}(n)$ *under* $\mathsf{FP}^{\mathsf{NP}}$ *reductions.* But, it is unclear whether the same is true under reductions that are not given access to an NP oracle.) See Section 1.3.

Theorem 1.6. *If there is some efficiently sampleable family of distributions of length-preserving circuits $\mathcal{D}_n$ and a constant $c > 0$ such that no (randomized) polynomial-time algorithm solves* $\mathsf{CollisionAvoid}$ *with probability larger than $1-1/n^c$ on input distribution $\mathcal{D}_n$, then one-way functions exist.*

1.1.3 Some additional results concerning meta-complexity and hardness of hardness

Finally, we show some additional results regarding the randomized complexity of Avoid. These latter results are relatively minor and can be viewed as variants of known results from prior work, as we describe below.

First, in Appendix A, we show zero-error randomized reductions from Avoid to two variants of the MKtP problem. MKtP is a central problem in *meta-complexity*. These results can be viewed as variants of similar results in [RSW22]. Specifically, [RSW22, Theorem 6.6] shows that Avoid (with certain parameters) is in FNP if and only if a certain *conditional* version of MKtP is in FNP, while we show a zero-error randomized reduction from Avoid to a similar conditional version of MKtP. And, [RSW22, Theorems 6.7 and 6.10] show that a certain kind of "unitary" version of Avoid is in FP (resp. FNP) if and only if a different version of MKtP is in FP (resp. FNP), while we show a zero-error randomized reduction from Avoid itself to a similar version of MKtP.

Second, in Appendix B, we show a barrier to reducing hard problems in FNP to Avoid. The idea is that any reduction from a hard problem in FNP to Avoid can be converted into an algorithm that simply runs the reduction and simulates responses to the oracle queries with random strings. Since a random string is a valid solution to Avoid with probability $1 - 2^{-s}$, such an algorithm will be successful unless the number of oracle queries is large relative to 2^s. This yields a barrier against reducing hard problems in FNP to Avoid. (See Section 1.3 for more discussion.) This generalizes a result from [ILW23], which effectively ruled out deterministic one-query reductions when the stretch s is very large.

1.2 Related work

In the few years since Kleinberg, Korten, Mitropolsky, and Papadimitriou introduced Avoid in [KKMP21], there has been a flurry of exciting work about the problem. We list some of the relevant works below.

First, Korten showed that an efficient deterministic algorithm for Avoid would imply efficient deterministic constructions of *many* important objects whose existence follows from the probabilistic method [Kor22]. These include truth tables with nearly maximal circuit complexity, pseudorandom strings, nearly optimal two-source extractors and Ramsey graphs, rigid matrices, strings with large Kolmogorov complexity, hard communication problems, and hard data structure problems.

Then, Ren, Santhanam, and Wang showed more connections between algorithms for Avoid (perhaps with oracles), circuit lower bounds, and meta-complexity (such as the results that we described in Section 1.1.3); and they brought new attention to the problem of solving Avoid for very restricted classes of circuits (even in $\mathsf{FP}^{\mathsf{NP}}$) [RSW22]. Chen, Huang, Li, and Ren then showed how to use these ideas to essentially match the best known circuit lower bounds using (oracle) algorithms for Avoid on restricted classes of circuits [CHLR23]. Chen, Hirahara, and Ren then showed how to construct a *pseudodeterministic* algorithm for Avoid with access to an NP oracle, and showed that this algorithm implies novel circuit lower bounds [CHR24]. Li then improved on this work to show a pseudodeterministic algorithm for Avoid with access to an NP oracle, which implies essentially optimal circuit lower bounds for symmetric exponential time [Li24].

In a different line of work, Guruswami, Lyu, and Wang showed new (oracle) algorithms for range avoidance on restricted classes of circuits and showed that many objects of interest can be constructed if one can solve Avoid on even slightly less restricted classes of circuits [GLW22]. Gajulapalli, Golovnev, Nagargoje, and Saraogi continued this line of work by showing further algorithms and reductions of this flavor [GGNS23].

Finally, Ilango, Li, and Williams showed that if subexponentially secure indistinguishability obfuscation (a very strong cryptographic primitive) exists, then $\mathsf{Avoid} \notin \mathsf{FP}$ unless $\mathsf{NP} = \mathsf{coNP}$ [ILW23]. In a follow-up work, Chen and Li showed that $\mathsf{Avoid} \notin \mathsf{FNP}$ assuming hardness of certain lattice problems against non-deterministic adversaries [CL24]. To our knowledge, these are the only known examples of proofs that an efficient algorithm for Avoid would imply something that is thought not to be true. (Many of the results from prior work show that an efficient algorithm for Avoid would imply efficient algorithms that we expect to exist, even if we do not know how to unconditionally prove that such algorithms exist.)

1.3 Future directions

Our results suggest a number of interesting directions to explore further. We list some of them below.

Connections between Avoid and pseudorandomness. We show that if Avoid is suitably hard on average for randomized algorithms, then one-way functions exist. By well-known results in cryptography, this in turn implies the existence of cryptographic pseudorandom generators (PRGs) [HILL99]. So, if Avoid is hard on average, then cryptographic PRGs exist.

On the other hand, Korten [Kor22] showed that if Avoid is in FP, then there exists a different kind of PRG. (Cryptographic PRGs have polynomial stretch but satisfy a very strong notion of pseudorandomness, while Korten shows PRGs with exponential stretch that satisfy a much weaker notion of pseudorandomness.)

This might suggest a deeper connection between the complexity of Avoid and pseudorandomness. One might even hope (perhaps foolishly?) for a win-win result showing that, whether Avoid is easy or hard, one still obtains some kind of PRG. Our current results do not quite achieve this because (1) our construction of cryptographic PRGs from Avoid requires average-case hardness against randomized algorithms, while Korten's construction requires worst-case deterministic algorithms; and (2) the two results use different notions of PRGs. However, perhaps one can prove something in the spirit of such a result, or otherwise further explore this apparently deep connection between Avoid and pseudorandomness.

One possible direction towards better understanding this relationship would be to study possible worst-case to average-case reductions for Avoid. In particular, such a reduction would be a step towards removing the first issue described above.

Better understanding of CollisionAvoid and AffineAvoid. We introduce two new computational problems CollisionAvoid and AffineAvoid. We show some basic properties of these problems in Section 3, including reducing Avoid to both of them, and reducing both of them to Empty (for some parameter regimes in the case of AffineAvoid). See Figure 1.

However, there is much that we still do not know about these problems. So, we ask what more can be said. For example, we do not know whether they can be reduced back to Avoid (except, in the case of AffineAvoid, in a rather extreme setting of parameters). Even $\mathsf{FP}^{\mathsf{NP}}$ reductions would be interesting.

One of the things that makes AffineAvoid interesting is that the parameters d and s can be varied together, and this allows for non-trivial reductions between different regimes. In particular, Theorem 3.6 shows a reduction from $(s+\ell, d+\ell)$-AffineAvoid to (s,d)-AffineAvoid, which one can think of as a strengthening of the trivial result that $(s+\ell)$-Avoid reduces to s-Avoid. It would be particularly exciting to show a reduction in the other direction, e.g., to show a reduction from (s,d)-AffineAvoid to (s',d')-AffineAvoid for $s < s'$. Perhaps one can even show that $(s+\ell, d+\ell)$-AffineAvoid and (s,d)-AffineAvoid are equivalent.

Hardness of Avoid. In Appendix B, we show some barriers to reducing hard problems in FNP to Avoid. However, these barriers do not rule out such reductions; they simply show that these reductions must work in the low-stretch regime and make many oracle queries. It would therefore be very interesting to get around these barriers and actually reduce a plausibly hard problem to Avoid. (Prior work, such as [Kor22], showed reductions from problems that are either unlikely to be in FNP or are not truly thought to be hard. Sometimes we only know $\mathsf{FP}^{\mathsf{NP}}$ reductions.) One could potentially dream of proving NP-hardness of Avoid for small stretch $s = O(\log n)$, but a reduction from any plausibly hard problem in FNP would be interesting.

Our results in particular show that suitable hardness of Avoid implies the existence of cryptography. So, a suitable reduction from a plausibly hard problem $A \in \mathsf{FNP}$ to Avoid would show that hardness of A implies the same things.

2 Preliminaries

We denote $\{0, 1, \ldots, N-1\}$ by $[N]$. We denote probabilistic polynomial time algorithms by PPT.

2.1 Computational problems

Below we define some of the computational problems that interest us in this work. The first two problems, Empty and Avoid were originally defined in [KKMP21] (though what we call s-Avoid was called 2^s-Empty in [KKMP21]). [KKMP21] also observed that Empty is NP-hard and that Empty is equivalent to the variant of the problem in which the range of the circuit is $[N + \mathrm{poly}(\log N)]$ instead of $[N+1]$.

Definition 2.1. *The* Empty *problem is defined as follows: given as input the description of a circuit $C : [N] \to [N+1]$, find a $y \in [N+1]$ such that $\forall x \in [N] : C(x) \neq y$.*

Definition 2.2. *For any integer $s := s(n) > 0$, the s-*Avoid *problem is defined as follows: given as input the description of a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^{n+s}$, find a $y \in \{0,1\}^{n+s}$ such that $\forall x \in \{0,1\}^n : C(x) \neq y$.*

We call s the stretch *of an s-*Avoid *instance, and when the stretch is one we simply write* Avoid.

We also introduce the following two problems.

Definition 2.3. *For any $s := s(n) > 0$, $d := d(n)$, the (s, d)-$\mathsf{AffineAvoid}$ problem is defined as follows: given as input the description of a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^{n+s}$, and an integer d, find an affine subspace of dimension d outside the range of C, where we identify $\{0,1\}^{n+s}$ with $\mathbb{F}_2^{n+s}$ in the natural way for the purposes of defining an affine subspace.*

We call s the stretch *of an* $\mathsf{AffineAvoid}$ *instance.*

Definition 2.4. *The* $\mathsf{CollisionAvoid}$ *problem is defined as follows: given as input the description of a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^n$, output either:*

1. *$(\mathsf{NON\text{-}IMAGE}, y)$, where $y \in \{0,1\}^n$ is not in the image of C;*
2. *$(\mathsf{NON\text{-}COLLISION}, x)$, where $x \in \{0,1\}^n$ satisfies that that $C(x) \neq C(x')$ for all $x' \neq x$.*

2.2 Reductions

Definition 2.5. *A deterministic Karp reduction from a search problem A to a search problem B is a pair of deterministic polynomial-time algorithms R, S such that:*

1. *Given an instance I_A of A, $R(I_A)$ outputs an instance of I_B.*
2. *Given any solution s_B to $I_B := R(I_A)$, $S(I_A, s_B)$ outputs a solution s_A for I_A.*

We will need to be a little precise in our notion of a Karp reduction from a decision problem to a search problem. We adapt Papadimitriou's notion of reduction for our purposes.

Definition 2.6. *A Karp reduction from a decision problem A to a search problem B is a pair of polynomial time algorithms R, S such that:*

1. *If $x \in A$, it holds that for all y such that $\langle R(x), y\rangle \in B$, $S(x, y) = 1$.*
2. *If $x \notin A$, it holds that for all y, $S(x, y) = 0$.*

For convenience, we often describe our Karp reductions simply as algorithms for problem A that work with an oracle for problem B and make a single oracle call.

2.3 Kolmogorov complexity

Definition 2.7. *For a given string $x \in \{0,1\}^*$, the* Kolmogorov-Levin complexity *is defined as:*

$$\mathsf{Kt}(x) = \min_M \{d + \log t \mid U(\langle M\rangle, 1^t) = x, d = |\langle M\rangle|\},$$

where U is the Universal Turing machine that runs Turing machine M (here, $\langle M\rangle$ denotes the description of M in bits) for t steps and outputs what M outputs.

Correspondingly, the conditional Kolmogorov-Levin complexity *of a string y conditioned on x is defined as:*

$$\mathsf{Kt}(y|x) = \min_M \{d + \log t \mid U(\langle M\rangle, x, 1^t) = y, d = |\langle M\rangle|\},$$

where U is the Universal Turing machine that runs Turing machine M on input x for t steps and outputs what M outputs.

Definition 2.8 (Time-Bounded Kolmogorv Complexity)**.** *For a given string $x \in \{0,1\}^*$ and $t \in \mathbb{N} \cup \{\infty\}$, the* t-time bounded Kolmogorov-Levin complexity *is defined as:*

$$\mathsf{K}^t(x) = \min_M \{d \mid U(\langle M\rangle, 1^t) = x, d = |\langle M\rangle|\},$$

where U is the Universal Turing machine that runs Turing machine M (here, $\langle M\rangle$ denotes the description of M in bits) for t steps and outputs what M outputs.

Correspondingly, the t-time bounded conditional Kolmogorov complexity *of a string $y \in \{0,1\}^*$ conditioned on x is defined as:*

$$\mathsf{K}^t(y|x) = \min_M \{d \mid U(\langle M\rangle, x, 1^t) = y, d = |\langle M\rangle|\},$$

where U is the Universal Turing machine that runs Turing machine M on input x for t steps and outputs what M outputs.

Definition 2.9. *The MKtP problem is defined as follows: Given as input a string $x \in \{0,1\}^*$ and an integer k decide if $\mathsf{Kt}(x) \le k$.*

We will also be interested in the approximate version of the problem.

Definition 2.10. *For $\gamma : \mathbb{N} \to \mathbb{N}$, the γ-$\mathsf{GapMKtP}$ problem is defined as follows: Given as input a string $x \in \{0,1\}^*$ and an integer k,*

1. *if $\mathsf{Kt}(x) \le k$, output 1; and*
2. *if $\mathsf{Kt}(x) \ge k + \gamma(k)$, output 0.*

We also consider a non-standard problem that approximates the time-bounded conditional Kolmogorov complexity. (Notice that the fact that in the NO case we use K^∞ only makes the problem easier.)

Definition 2.11. *For $1 \le a < b$ and some time bound $t := t(n)$, (a,b)-$\mathsf{GapMcK^{t,\infty}P}$ is the promise problem defined as follows: Given as input strings $x, y \in \{0,1\}^*$,*

1. *if $\mathsf{K}^t(y \mid x) \le a$, output 1; and*
2. *if $\mathsf{K}^\infty(y \mid x) \ge b$, output 0.*

The following lemma will be useful for our purposes.

Lemma 2.12 (Chain Rule)**.** *For any two strings x, y, we have*

$$\mathsf{Kt}(x \circ y) \le \mathsf{Kt}(y|x) + \mathsf{Kt}(x) + O(\log(|x|)) ,$$

where $x \circ y$ is the concatenation of x and y.

Proof. To see this, note that to output $x \circ y$, it suffices for us to use two machines M_x (which outputs string x), and M_y (which outputs string y while using x as input). The algorithm is then defined as follows:

1. Run M_x to output x.
2. Copy x to the input tape of M_y.
3. Run $M_y(x)$ to append y to the output.

Now, letting t_x be the time it takes M_x to output string x, and t_y be the time it takes M_y to output string y (when given input x), the total time it takes to output $x \circ y$ is at most $t_x + t_y + |x|$. Furthermore, the description length is at most $\langle M_x \rangle + \langle M_y \rangle + O(\log(|x|))$.

Thus,

$$\begin{aligned}\mathsf{Kt}(x \circ y) &\leq \langle M_x \rangle + \langle M_y \rangle + O(\log(|x|)) + \log(t_x + t_y + |x|) \\ &\leq \langle M_x \rangle + \langle M_y \rangle + O(\log(|x|)) + \log(t_x) + \log(t_y) \\ &= Kt(x) + Kt(y|x) + O(\log(|x|)) .\end{aligned}$$

□

Lemma 2.13 (Incompressibility of random strings). *For any positive integer n and $0 \leq t \leq n$,*

$$\Pr_{x \sim \{0,1\}^n}[\mathsf{Kt}(x) \leq t] \leq 2^{-(n-t-1)} .$$

Proof. Let x be a randomly chosen string of length n. There are at most 2^{t+1} strings that can be output by some Turing machine whose description length is at most t. Thus, the probability that a randomly chosen string of length n has Kolmogorov-Levin complexity $\mathsf{Kt}(x) \leq t$ is at most:

$$\frac{2^{t+1}}{2^n} = 2^{-(n-t-1)} .$$

□

Lemma 2.14 (Weak Monotonicity). *Let $x \in \{0,1\}^{\ell_x}$ and $y \in \{0,1\}^{\ell_y}$. Then:*

$$\begin{aligned}\mathsf{Kt}(x) &\leq \mathsf{Kt}(x \circ y) + O(\log(\ell_x + \ell_y)) , \\ \mathsf{Kt}(y) &\leq \mathsf{Kt}(x \circ y) + O(\log(\ell_x + \ell_y)) .\end{aligned}$$

Proof. To see this, note that it suffices to design an algorithm that first computes $x \circ y$ and then outputs the first ℓ_x of $x \circ y$ to outputs x or the last ℓ_y bits to output y. □

Lemma 2.15 ([HIR23], Fact 2.13). *For any string $y \in \{0,1\}^*$, time bound t (including $t = \infty$), and positive integer α, we have*

$$\Pr_{x \sim \{0,1\}^n}[\mathsf{K}^t(x|y) \leq n - \alpha] \leq \frac{1}{2^{\alpha-1}} .$$

2.4 Average-case hardness and one-way circuits

We will need a notion of (weak) average-case hardness of a computational problem. Our notion of hardness corresponds to hardness on efficiently sampleable distributions. As in the cryptography literature, we call the sampling algorithm Gen.

Definition 2.16. *A computational problem* B *is* weakly hard on average with generator Gen *if* Gen *is a PPT algorithm that takes as input* 1^κ *and outputs an instance* x *of* B *with size* $n(\kappa) \le \mathrm{poly}(\kappa)$*, and for every PPT algorithm* $\mathcal{A}$ *there exists a* κ_0 *such that for all* $\kappa \ge \kappa_0$*,*

$$\Pr_{x \leftarrow \mathsf{Gen}(1^\kappa)}[\mathcal{A}(1^\kappa, x) \text{ is a solution to instance } x \text{ of } \mathsf{B}] \le 1 - 1/\kappa \; .$$

Note the close relationship between the above definition and the following definition of an efficiently sampleable (weak) one-way function. For convenience, we actually use a slightly non-standard circuit-based definition (which is equivalent to the standard definition).

Definition 2.17. *We say that a PPT algorithm* Gen samples a weak one-way circuit *if:*

1. *On input* 1^κ*,* Gen *outputs a circuit* $C : \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}$ *where* $n(\kappa)$ *and* $m(\kappa)$ *are some (fixed) polynomially bounded functions.*
2. *For any PPT algorithm* $\mathcal{A}$*, there exists a* κ_0 *such that for all* $\kappa \ge \kappa_0$*,*

$$\Pr_{C \leftarrow \mathsf{Gen}(1^\kappa), x \sim \{0,1\}^{n(\kappa)}}[x' \leftarrow \mathcal{A}(1^\kappa, C, C(x)),\ C(x') = C(x)] \le 1 - 1/\kappa \; .$$

Theorem 2.18 (Theorem 5.2.1 of [Zim04], Restated)**.** *If there exists a PPT algorithm that samples a weak one-way circuit, then one-way functions exist.*

We will also need the following simple lemma.

Lemma 2.19. *Suppose that* Gen *is an algorithm that takes as input* 1^κ *and outputs a circuit* $C : \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}$ *for some functions* $n(\kappa)$ *and* $m(\kappa)$*, and suppose that* $\mathcal{A}$ *is an algorithm with the property that*

$$\Pr_{C \leftarrow \mathsf{Gen}(1^\kappa), x \sim \{0,1\}^{n(\kappa)}}[x' \leftarrow \mathcal{A}(1^\kappa, C, C(x)),\ C(x') = C(x)] > 1 - 1/\kappa \; .$$

Then, for every $0 < \alpha < \kappa$*,*

$$\Pr_{C \leftarrow \mathsf{Gen}(1^\kappa)}[C \in S_\alpha] > 1 - \alpha/\kappa \; ,$$

where S_α *is the set of all circuits* C *such that*

$$\Pr_{x \sim \{0,1\}^{n(\kappa)}}[x' \leftarrow \mathcal{A}(1^\kappa, C, C(x)),\ C(x') = C(x)] > 1 - 1/\alpha \; .$$

Proof. Notice that

$$\begin{aligned}
1 - 1/\kappa &< \Pr_{C \leftarrow \mathsf{Gen}(1^\kappa), x \sim \{0,1\}^{n(\kappa)}}[x' \leftarrow \mathcal{A}(1^\kappa, C, C(x)),\ C(x') = C(x)] \\
&= \Pr[C \in S_\alpha] \cdot \Pr\big[C(x') = C(x) \mid C \in S_\alpha\big] + \Pr[C \notin S_{\mathcal{A}}] \cdot \Pr\big[C(x') = C(x) \mid C \notin S_\alpha\big] \\
&\le \Pr[C \in S_\alpha] + (1 - 1/\alpha) \cdot \Pr[C \notin S_\alpha] \\
&= \Pr[C \in S_\alpha]/\alpha + 1 - 1/\alpha \; .
\end{aligned}$$

Rearranging this, we see that $\Pr[C \in S_\alpha] > 1 - \alpha/\kappa$, as needed. □

Finally, we will need the following technical lemma, which bounds the probability of an algorithm encountering a certain bad event in terms of its failure probability in the one-way function game on a fixed circuit. The bad event is that a random element in the co-domain of the fixed circuit lands in the image of the circuit but the algorithm still fails to find an inverse.

Lemma 2.20. *For any circuit $C : \{0,1\}^n \to \{0,1\}^m$ and any algorithm $\mathcal{B}$, we have*

$$\Pr_{y\sim\{0,1\}^m,\mathcal{B}}[C(\mathcal{B}(y)) \neq y \wedge y \in \mathrm{Im}(C)] \leq 2^{n-m} \cdot \Pr_{x\sim\{0,1\}^n}[x' \leftarrow \mathcal{B}(C(x)),\ C(x') \neq C(x)]\ .$$

Proof. Notice that for any $y \in \mathrm{Im}(C)$, we must have $\Pr_{x\sim\{0,1\}^n}[C(x) = y] \geq 1/2^n$, since the domain of C has size 2^n. Therefore, we have

$$\begin{aligned}\Pr_{y,\mathcal{B}}[C(\mathcal{B}(y)) \neq y \wedge y \in \mathrm{Im}(C)] &= \sum_{y\in\mathrm{Im}(C)} \frac{1}{2^m} \Pr_{\mathcal{B}}[C(\mathcal{B}(y)) \neq y] \\ &\leq 2^{n-m} \cdot \sum_{y\in\mathrm{Im}(C)} \Pr_x[C(x) = y] \Pr_{\mathcal{B}}[C(\mathcal{B}(y)) \neq y] \\ &= 2^{n-m} \cdot \Pr_{x,\mathcal{B}}[C(\mathcal{B}(y)) \neq y \mid C(x) = y]\ . \end{aligned}$$ ☐

2.5 Auxiliary-input one-way function

We will use the notion of (weak) worst-case hardness of a computational problem against PPT algorithms.

Definition 2.21. *A computational problem* B *is* weakly hard in the worst case *if for every PPT algorithm $\mathcal{A}$ there exist infinitely many $\kappa \in \mathbb{N}$ such that there exists an instance x of B with size $n(\kappa) \leq \mathrm{poly}(\kappa)$ where*

$$\Pr[\mathcal{A}(1^\kappa, x) \text{ is a solution to instance } x \text{ of } \mathsf{B}] \leq 1 - 1/\kappa\ .$$

Analogously auxiliary-input one-way functions are weaker counterparts of standard one-way functions.

Definition 2.22. *A polynomial-time-computable auxiliary-input function $f = \{f_z : \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}\}_{z\in\{0,1\}^{s(\kappa)}}$ where $n(\kappa), m(\kappa), s(\kappa) \leq \mathrm{poly}(\kappa)$ is said to be an* auxiliary-input one-way function *secure against PPT adversaries if for every PPT adversary $\mathcal{A}$ and for infinitely many $\kappa \in \mathbb{N}$, there exists $z \in \{0,1\}^{s(\kappa)}$ such that*

$$\Pr_{x\sim\{0,1\}^n}[f_z(\mathcal{A}(1^\kappa, z, f_z(x))) = f_z(x)] < \mathrm{negl}(\kappa)\ .$$

We also define a weak version of auxiliary-input one-way function.

Definition 2.23. *A polynomial-time-computable auxiliary-input function $f = \{f_z : \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}\}_{z\in\{0,1\}^{s(\kappa)}}$ where $n(\kappa), m(\kappa), s(\kappa) \leq \mathrm{poly}(\kappa)$ is said to be a* weak auxiliary-input one-way function *secure against PPT adversaries if for every PPT adversary $\mathcal{A}$ and for infinitely many $\kappa \in \mathbb{N}$, there exists $z \in \{0,1\}^{s(\kappa)}$ such that*

$$\Pr_{x\sim\{0,1\}^n}[f_z(\mathcal{A}(1^\kappa, z, f_z(x))) = f_z(x)] < 1 - 1/\kappa\ .$$

Theorem 2.24. *If there exists a weak auxiliary-input one-way function, then auxiliary-input one-way functions exist.*

In Appendix C, we sketch a simple proof of this theorem for completeness. The proof is similar to the proof of Theorem 2.18.

2.6 Gowers norms

We review a useful definition in the literature of additive combinatorics: the Gowers norms.

Definition 2.25. *Let $f : \mathbb{F}_2^n \to \mathbb{R}$ be a function. For every non-negative integer d, we define the Gowers d-norm (U_d norm) of f to be:*

$$\|f\|_{U_d} = \left(\mathop{\mathbb{E}}_{s,v_1,\ldots,v_d \sim \mathbb{F}_2^n} \left[\prod_{S \in [d]} f(s + \sum_{j \in S} v_j) \right] \right)^{\frac{1}{2^d}} .$$

We remark that the Gowers 0-norm is simply the expectation of the function f, $\|f\|_{U_0} = \mathbb{E}_x[f(x)]$. We will use the Cauchy-Schwarz-Gowers inequality (see, e.g., [VW08, Lemma 2.4]) that shows that the Gowers norms are monotonically non-decreasing.

Theorem 2.26. *For every integer $n, k \geq 1$ and every $f : \mathbb{F}_2^n \to \mathbb{R}$,*

$$\|f\|_{U_{k-1}} \leq \|f\|_{U_k} .$$

We will now use the Gowers norms to prove that every large subset of $\mathbb{F}_2^n$ contains an affine subspace of non-trivial dimension.

Lemma 2.27. *Let $n \geq 2, d \geq 1$ be integers, and $S \subseteq \mathbb{F}_2^n$ be a set of size $|S| \geq \lambda 2^n$ for $\lambda \in (0, 1]$. Let $s, v_1, v_2, \ldots, v_d$ be $d+1$ vectors chosen uniformly and independently from $\mathbb{F}_2^n$. Then with probability at least $\lambda^{2^d} - 2^{d-n}$, $s + \mathrm{span}(v_1, \ldots, v_d)$ forms an affine subspace of dimension d which is contained in S.*

Proof. Let $f : \mathbb{F}_2^n \to \{0, 1\}$ be the indicator function of S: $f(x) = 1$ if and only if $x \in S$. Consider the Gowers d-norm of f, raised to the power of 2^d:

$$\|f\|_{U_d}^{2^d} = \left(\mathop{\mathbb{E}}_{s,v_1,\ldots,v_d \sim \mathbb{F}_2^n} \left[\prod_{T \in [d]} f(s + \sum_{j \in T} v_j) \right] \right) .$$

Notice that term $\prod_{T \in [d]} f(s + \sum_{j \in T} v_j) = 1$ if and only if $s + \mathrm{span}(v_1, \ldots, v_d)$ is fully contained in S. Hence, for uniform and independent $s, v_1, v_2, \ldots, v_d$, we have that $s + \mathrm{span}(v_1, \ldots, v_d)$ is contained in S with probability

$$\|f\|_{U_d}^{2^d} \geq \|f\|_{U_0}^{2^d} \geq (\lambda)^{2^d} ,$$

where the first inequality follows by Theorem 2.26, and the second inequality uses $\|f\|_{U_0} = \mathbb{E}_x[f(x)] \geq \lambda$.

Next, we consider the probability that $v_1, \ldots, v_d$ span a subspace of dimension d. The probability that d uniform and independent vectors are linearly independent is bounded from below by

$$\prod_{i=0}^{d-1}(1 - 2^{i-n}) \geq 1 - \sum_{i=0}^{d-1} 2^{i-n} \geq 1 - 2^{d-n} .$$

By the union bound, with probability at least $\lambda^{2^d} - 2^{d-n}$, $s + \mathrm{span}(v_1, \ldots, v_d)$ is a d-dimensional affine subspace contained in S. □

3 Basic results

In this section, we present a few simple reductions and observations about Avoid, CollisionAvoid, and AffineAvoid.

3.1 CollisionAvoid

We start by observing that Avoid is no harder than CollisionAvoid via a simple Karp reduction.

Theorem 3.1. *There is a deterministic Karp reduction from* Avoid *to* CollisionAvoid*.*

Proof. Given a circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$ for Avoid, the reduction constructs a CollisionAvoid instance $C' : \{0,1\}^n \to \{0,1\}^n$ as follows: $C'(x) = C(x)_{[n]}$. I.e., $C'(x)$ outputs the first n bits of $C(x)$.

The reduction then calls its CollisionAvoid oracle on input C', receiving as output either $(\mathsf{NON\text{-}IMAGE}, y)$ or $(\mathsf{NON\text{-}COLLISION}, x)$. If the oracle's output is $(\mathsf{NON\text{-}IMAGE}, y)$ for $y \in \{0,1\}^n$, then the reduction outputs $y \circ 0$. If the oracle's output is $(\mathsf{NON\text{-}COLLISION}, x)$ for $x \in \{0,1\}^n$, the reduction computes $y^* := C(x)$ and outputs $\overline{y}$, which is y^* with its last bit flipped.

Clearly the reduction is deterministic, runs in polynomial time, and makes a single oracle call.

To see that the reduction is correct, notice that if oracle's output is $(\mathsf{NON\text{-}IMAGE}, y)$, then by definition there is no $x' \in \{0,1\}^n$ such that $C'(x') = y$. By the definition of C' this means that there is no $x' \in \{0,1\}^n$ such that $C(x) = y \circ 0$, i.e., $y \circ 0$ is not in the image of C, as needed. On the other hand, if the oracle's output is $(\mathsf{NON\text{-}COLLISION}, x)$, then by definition the only input $x' \in \{0,1\}^n$ such that $C'(x') = C'(x)$ is x itself. In particular, y^* is the unique string in the image of C whose first n bits are $y^*_{[n]}$. Therefore, $\overline{y}$ is not in the image of C. □

Next, we show that CollisionAvoid is no harder than Empty.

Theorem 3.2. *There is a deterministic Karp reduction from* CollisionAvoid *to* Empty*.*

Proof. Given a circuit $C : \{0,1\}^n \to \{0,1\}^n$ for CollisionAvoid, the reduction constructs an Empty instance $C' : \{0,1\}^n \to [2^n + 1]$ to be

$$C'(x) = \begin{cases} 2^n & \text{if } x = 0^n \text{ ,} \\ C(x) & \text{otherwise .} \end{cases}$$

(In other words, C' maps 0^n to an element that is not in the range of C.)

The reduction then calls its Empty oracle on input C', receiving as output y. If the oracle's output satisfies that $y = C(0^n)$, the reduction outputs $(\mathsf{NON\text{-}COLLISION}, 0^n)$. Otherwise, the reduction outputs $(\mathsf{NON\text{-}IMAGE}, y)$.

Clearly the reduction is deterministic, runs in polynomial time and makes a single oracle call.

To see that the reduction is correct, consider a solution y for Empty with input C', clearly we have $y \in [2^n]$, or equivalently when written in binary $y \in \{0,1\}^n$. We consider two cases.

If $y = C(0^n)$, then, because y is a valid solution to Empty on input C, we know that $C(x) \neq C(0^n)$ for any $x \neq 0^n$ (since for $x \neq 0^n$, $C(x) = C'(x)$). Hence, $(\mathsf{NON\text{-}COLLISION}, 0^n)$ is a valid solution to the CollisionAvoid instance in this case.

Otherwise, $y \neq C(0^n)$ and since y is not in the image of C', it follows that y is not in the image of C. Hence, $(\mathsf{NON\text{-}IMAGE}, y)$ is a valid solution to the CollisionAvoid instance, as needed. □

Similar to Avoid, CollisionAvoid also admits an input-independent randomized algorithm that succeeds with constant probability. In particular, this implies that CollisionAvoid $\in \mathsf{FZPP}^{\mathsf{NP}}$ by using the NP oracle for verification of the guessed solutions.

Theorem 3.3. *There is an input-independent* BPP *algorithm for* CollisionAvoid *that succeeds with probability at least* $1/3$.

Proof. Regardless of any input circuit $C : \{0,1\}^n \to \{0,1\}^n$ for CollisionAvoid, the algorithm samples a uniform string $y \in \{0,1\}^n$. Then it outputs $(\mathsf{NON\text{-}IMAGE}, y)$ with probability $2/3$ and $(\mathsf{NON\text{-}COLLISION}, y)$ with probability $1/3$.

To see that the algorithm succeeds with probability at least $1/3$, let $\delta := 1 - \frac{|\mathrm{Im}(C)|}{2^n}$. We then have:

$$\Pr_y[y \notin \mathrm{Im}(C)] = \delta \ ,$$

$$\Pr_y[\forall x \neq y, C(x) \neq C(y)] \geq 1 - 2\delta \ .$$

The first inequality follows from the definition of δ. To see the second inequality, notice that there are at most 2δ fraction of inputs involved in collisions.

Therefore, the success probability is at least

$$\frac{2}{3} \cdot \delta + \frac{1}{3} \cdot (1 - 2\delta) \geq \frac{1}{3} \ .$$ □

3.2 AffineAvoid

We start by presenting some choices of parameters where AffineAvoid admits an $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm.

Theorem 3.4. *Let $c > 0$ be a constant, and $s := s(n) \geq 1$ and $d := d(n) \geq 1$ satisfy*

$$(1 - 2^{-s})^{2^d} - 2^{d-n-s} > \frac{1}{n^c}$$

*for all large enough n. Then (s,d)-*AffineAvoid $\in \mathsf{FZPP}^{\mathsf{NP}}$. *In particular, $(1, \log\log n)$-*AffineAvoid $\in \mathsf{FZPP}^{\mathsf{NP}}$.

Proof. Consider any circuit $C : \{0,1\}^n \to \{0,1\}^{n+s}$ as input of (s,d)-$\mathsf{AffineAvoid}$, and define S to be the set of all non-images of C. We have

$$|S| \geq 2^{n+s} - 2^n = (1-2^{-s})2^{n+s} .$$

The algorithm proceeds as follows. Sample $u, v_1 \dots, v_d \in \mathbb{F}_2^{n+s}$ uniformly at random. Then verify that $u + \mathrm{span}(v_1, \dots, v_d)$ has dimension d and use the NP oracle to verify that $\forall x \in \mathbb{F}_2^{n+s}$, $C(x) \notin u + \mathrm{span}(v_1, \dots, v_d)$. If all tests pass, output the affine subspace $u + \mathrm{span}(v_1, \dots, v_d)$. Otherwise, repeat the procedure.

It is easy to see that any affine subspace output by the algorithm is a correct solution. It remains to show that the algorithm terminates in expected $\mathrm{poly}(n)$ time.

By Lemma 2.27, with probability $(1-2^{-s})^{2^d} - 2^{d-n-s} > \frac{1}{n^c}$, $u + \mathrm{span}(v_1, \dots, v_d)$ sampled by the algorithm is a d-dimensional affine subspace contained in S. Hence the expected running time is bounded from above by n^c. ☐

Next, we present choices of parameters where $\mathsf{AffineAvoid}$ falls in $\mathsf{TF\Sigma_2P}$.

Theorem 3.5. *Let $s := s(n) \geq 1$ and $d := d(n) \geq 1$ satisfy*

$$(1-2^{-s})^{2^d} - 2^{d-n-s} > 0$$

for all large enough n. Then (s,d)-$\mathsf{AffineAvoid} \in \mathsf{TF\Sigma_2P}$. In particular, $(1, \log n - 1)$-$\mathsf{AffineAvoid} \in \mathsf{TF\Sigma_2P}$.

Proof. Let $C : \{0,1\}^n \to \{0,1\}^{n+s}$ be an input of (s,d)-$\mathsf{AffineAvoid}$, and S be the set of all non-images of C:

$$|S| \geq 2^{n+s} - 2^n = (1-2^{-s})2^{n+s} .$$

By Lemma 2.27, with probability $(1-2^{-s})^{2^d} - 2^{d-n-s} > 0$, we can sample a d-dimensional affine subspace contained in S. In other words, a solution to (s,d)-$\mathsf{AffineAvoid}$ is guaranteed to exist.

Hence, one could write down the $\mathsf{TF\Sigma_2P}$ statement as follows:

$$\exists u, v_1, \dots v_d \in \mathbb{F}_2^{n+s}, \forall x \in \mathbb{F}_2^n, C(x) \notin u + \mathrm{span}(v_1, \dots, v_d) ,$$

and the result follows. ☐

We conclude with a simple reduction between $\mathsf{AffineAvoid}$ with different parameters.

Theorem 3.6. *Let $\ell := \ell(n) \geq 1$ be an efficiently computable function. Then for every $d := d(n) \geq 1$ there is a deterministic Karp reduction from $(s+\ell, d+\ell)$-$\mathsf{AffineAvoid}$ to (s,d)-$\mathsf{AffineAvoid}$.*

In particular, by setting $d = 0$, we have that $(s+\ell, \ell)$-$\mathsf{AffineAvoid}$ reduces to s-Avoid.

Proof. Given a circuit $C : \{0,1\}^n \to \{0,1\}^{n+s+\ell}$ for $(s+\ell, d+\ell)$-$\mathsf{AffineAvoid}$, the reduction defines an (s,d)-$\mathsf{AffineAvoid}$ instance $C' : \{0,1\}^n \to \{0,1\}^{n+s}$ to be $C'(x) = C(x)_{[n+s]}$. I.e., $C'(x)$ outputs the first $n+s$ bits of $C(x)$.

The reduction then calls its (s,d)-$\mathsf{AffineAvoid}$ oracle on input C', receives as output $u, v_1, \dots, v_d \in \mathbb{F}_2^{n+s}$ where $u + \mathrm{span}(v_1, \dots, v_d)$ is an affine subspace that contains no images of C'.

Finally, the reduction outputs $u \circ 0^\ell, v_1 \circ 0^\ell, \ldots, v_d \circ 0^\ell, e_{n+s+1}, \ldots, e_{n+s+\ell}$ where $e_i \in \mathbb{F}_2^{n+s+\ell}$ is the ith standard basis vector.

Clearly the reduction is a deterministic Karp reduction with a single oracle call. Let $A := u + \mathrm{span}(v_1, \ldots, v_d) \subseteq \mathbb{F}_2^{n+s}$. Since no element in A is in the image of C', it follows that no point in $\mathbb{F}_2^{n+s+\ell}$, whose first $(n+s)$ coordinates lie in A, is in the image of C. In other words, the affine subspace $A' := u \circ 0^\ell + \mathrm{span}(v_1 \circ 0^\ell, \ldots, v_d \circ 0^\ell, e_{n+s+1}, \ldots, e_{n+s+\ell})$ contains no images of C, which finishes the proof. □

4 CollisionAvoid to Minicrypt

In this section we show that (worst-case/average-case) hardness of CollisionAvoid implies the existence of weak (auxiliary-input/standard) one-way function families.

In the following technical lemma, we show how to use an algorithm that inverts a circuit $\mathcal{C}$ with high probability to solve CollisionAvoid.

Lemma 4.1. *There is a PPT oracle algorithm $\mathcal{A}$ such that for every length preserving circuit $C : \{0,1\}^{n(\kappa)} \to \{0,1\}^{n(\kappa)}$ and every oracle $\mathcal{B}$, if*

$$\Pr_{x \sim \{0,1\}^n} \left[C\left(\mathcal{B}(1^\kappa, C, C(x))\right) = C(x) \right] \geq 1 - 1/\kappa \ ,$$

then for $\gamma := \kappa^{1/3}/10$,

$$\Pr\left[\mathcal{A}^{\mathcal{B}}(1^\gamma, C) \text{ solves } \mathsf{CollisionAvoid}\right] \geq 1 - 1/(2\gamma) \ .$$

Proof. We present in Algorithm 1 an algorithm that we claim solves CollisionAvoid on the input C with probability at least $1 - 1/(2\gamma)$.

Algorithm 1: $\mathcal{A}^{\mathcal{B}}(1^\gamma, C)$

```
Set $\kappa := 1000\gamma^3$;
Do $\Delta = 10\gamma + 1$ times
    Sample $y \sim \{0,1\}^n$;
    Set $x' \leftarrow \mathcal{B}(1^\kappa, C, y)$;
    if $C(x') \neq y$ then
        Output $(\mathsf{NON\text{-}IMAGE}, y)$;
    end
end
Sample $x \sim \{0,1\}^n$ and output $(\mathsf{NON\text{-}COLLISION}, x)$;
```

Clearly the algorithm runs in polynomial time.

To see that the algorithm is correct, recall that $\kappa := 1000\gamma^3$, and let $\Delta := 10\gamma + 1$. We will show that $\mathcal{A}$ solves CollisionAvoid on C with probability at least $1 - 1/(2\gamma)$.

Let

$$\varepsilon := |\operatorname{Im}(C)|/2^n = \Pr_{y \sim \{0,1\}^n} [y \in \operatorname{Im}(C)] \ ,$$

and define the event E to be the event that $\mathcal{A}$ outputs $(\mathsf{NON\text{-}IMAGE}, y)$ for some y.

Notice that E occurs if any sample y inside the loop of Algorithm 1 is not inverted. Therefore,

$$1 - \Pr[E] = \Pr_{y \sim \{0,1\}^n}[x' \leftarrow \mathcal{B}(1^\kappa, C, y) \ : \ C(x') = y]^\Delta = (\varepsilon - p)^\Delta \ ,$$

where

$$p := \Pr_{y \sim \{0,1\}^n}[x' \leftarrow \mathcal{B}(1^\kappa, C, y) \ : \ C(x') \neq y \text{ and } y \in \mathrm{Im}(C)]$$

is the probability that y lands in the image of C but $\mathcal{B}$ still fails to find a preimage of y. Of course, $p \geq 0$, and Lemma 2.20 tells us that $p < 1/\kappa$. Therefore,

$$1 - \varepsilon^\Delta \leq \Pr[E] < 1 - (\varepsilon - 1/\kappa)^\Delta \ .$$

(Here, we are using our choice of Δ as an odd integer to conclude that the inequality holds even if $1/\kappa > \varepsilon$.) It follows that

$$\begin{aligned} \Pr[\mathcal{A} \text{ fails}] &= \Pr[\mathcal{A} \text{ fails and } E] + \Pr[\mathcal{A} \text{ fails and not } E] \\ &< (1 - (\varepsilon - 1/\kappa)^\Delta) \cdot \Pr[\mathcal{A} \text{ fails} \mid E] + \varepsilon^\Delta \cdot \Pr[\mathcal{A} \text{ fails} \mid \text{not } E] \\ &\leq (1 - (\varepsilon - 1/\kappa)^\Delta) \cdot \Pr[\mathcal{A} \text{ fails} \mid E] + 2(1-\varepsilon) \cdot \varepsilon^\Delta \ , \end{aligned} \tag{1}$$

where the last line uses the fact that[4]

$$\Pr[\mathcal{A} \text{ fails} \mid \text{not } E] \leq 2(1 - \varepsilon) \ .$$

It remains to bound the probability that $\mathcal{A}$ fails conditioned on E, i.e., conditioned on outputting inside the loop. We have

$$\begin{aligned} \Pr[\mathcal{A} \text{ succeeds} \mid E] &= \Pr_{y \sim \{0,1\}^n}[y \notin \mathrm{Im}(C) \mid C(\mathcal{B}(1^\kappa, C, y)) \neq y] \\ &= \frac{1 - \varepsilon}{\Pr_{y \sim \{0,1\}^n}[C(\mathcal{B}(1^\kappa, C, y)) \neq y]} \\ &= \frac{1 - \varepsilon}{1 - \varepsilon + p} \\ &> \frac{1 - \varepsilon}{1 - \varepsilon + 1/\kappa} \ , \end{aligned}$$

where the last line again uses Lemma 2.20. So,

$$\Pr[\mathcal{A} \text{ fails} \mid E] < 1 - \frac{1 - \varepsilon}{1 - \varepsilon + 1/\kappa} = \frac{1}{1 + (1 - \varepsilon)\kappa} \ .$$

Plugging back in to Equation (1), we see that

$$\Pr[\mathcal{A} \text{ fails}] < \frac{1 - (\varepsilon - 1/\kappa)^\Delta}{1 + (1 - \varepsilon)\kappa} + 2(1 - \varepsilon) \cdot \varepsilon^\Delta \ .$$

[4]To see this, note that $\Pr[\mathcal{A} \text{ fails} \mid \text{not } E] = \Pr_{x \sim \{0,1\}^n}[\exists x' \colon x' \neq x, C(x) = C(x')]$, and let δ denote this quantity. For δ fraction of the inputs, they each have to collide with at least one other input, whereas the remaining $1 - \delta$ fraction of the inputs have unique images. Thus, the image size $|\mathrm{Im}(C)| = 2^n \varepsilon$ is at most $2^n \left(\frac{\delta}{2} + (1 - \delta)\right)$. Rearranging then yields the fact that $\delta \leq 2(1 - \varepsilon)$.

The result then follows by noting that each of the terms on the right-hand side above is bounded by $1/(4\gamma)$. In particular, for $\varepsilon \geq 1 - 1/(8\gamma\Delta) \geq 1 - 1/(4\gamma\Delta) + 1/\kappa$, the first term is bounded by

$$\frac{1-(\varepsilon - 1/\kappa)^{\Delta}}{1+(1-\varepsilon)\kappa} \leq 1 - \left(1 - \frac{1}{4\gamma\Delta}\right)^{\Delta} \leq \frac{1}{4\gamma} ,$$

while for $\varepsilon < 1 - 1/(8\gamma\Delta)$, we have

$$\frac{1-(\varepsilon - 1/\kappa)^{\Delta}}{1+(1-\varepsilon)\kappa} \leq \frac{1+1/\kappa^{\Delta}}{1+\kappa/(8\gamma\Delta)} < \frac{2}{1+1000\gamma^3/(8\gamma \cdot (10\gamma+1))} < \frac{1}{4\gamma} ,$$

as needed. On the other hand, the second term is maximized when $\varepsilon = \Delta/(\Delta+1)$, in which case it is equal to

$$2\left(1-\frac{\Delta}{\Delta+1}\right)\left(\frac{\Delta}{\Delta+1}\right)^{\Delta} \leq \frac{2}{\Delta+1} \leq \frac{1}{4\gamma} ,$$

as needed. ☐

Theorem 4.2. *Suppose* CollisionAvoid *is weakly hard in the worst case, then weak auxiliary-input one-way functions exist.*

As one could boost weak auxiliary-input one-way function using Theorem 2.24, the existence of auxiliary-input one-way function follows from the weak worst-case hardness of CollisionAvoid.

Corollary 4.3. *If* CollisionAvoid *is weakly hard in the worst case, then auxiliary-input one-way functions exist.*

Proof of Theorem 4.2. Define $\mathcal{C}_\kappa \subseteq \{0,1\}^{s(\kappa)}$ to be the set of all circuits $C : \{0,1\}^{n(\kappa)} \to \{0,1\}^{n(\kappa)}$ and $\mathcal{C} := \cup_{\kappa\in\mathbb{N}}\mathcal{C}_\kappa$. We claim that $\{f_C(x) := C(x)\}_{C\in\mathcal{C}}$ is a weak auxiliary-input one-way function.

Assume towards contradiction that there is a PPT algorithm $\mathcal{B}$ such that for all sufficiently large κ and $\forall C \in \mathcal{C}_\kappa$,

$$\Pr_{x\sim\{0,1\}^n}[C(\mathcal{B}(1^\kappa, C, C(x))) = C(x)] \geq 1 - 1/\kappa .$$

Then by Lemma 4.1, the PPT algorithm $\mathcal{A}^{\mathcal{B}}$ solves CollisionAvoid with probability at least $1 - 1/(2\gamma) > 1 - 1/\gamma$ for all circuits in $\mathcal{C}_\kappa$, contradicting the worst-case hardness of CollisionAvoid. ☐

Theorem 4.4. *Suppose* CollisionAvoid *is weakly hard on average with generator* Gen*, then there exists a generator* Gen$'$ *that samples weak one-way circuits.*

Combining the above with Theorem 2.18 immediately implies the following, which is the formal version of Theorem 1.6.

Corollary 4.5. *If* CollisionAvoid *is weakly hard on average with some generator, then (standard) one-way functions exist.*

Proof of Theorem 4.4. For any $\kappa > 0$, we define $\mathsf{Gen}'(1^\kappa) := \mathsf{Gen}(1^\gamma)$ for $\gamma = \gamma(\kappa) := \lfloor \kappa^{1/4}/100 \rfloor$. Since $\mathsf{Gen}(1^\gamma)$ outputs a circuit $C : \{0,1\}^{n(\gamma)} \to \{0,1\}^{n(\gamma)}$, which is an instance of CollisionAvoid, $\mathsf{Gen}'(1^\kappa)$ outputs a circuit $C : \{0,1\}^{n'(\kappa)} \to \{0,1\}^{n'(\kappa)}$, where $n'(\kappa) := n(\lfloor \kappa^{1/4}/100 \rfloor)$.

We claim that Gen' samples a weak one-way circuit. Assume towards contradiction that this is not true. Then, there exists a PPT algorithm $\mathcal{B}$ such that for infinitely many κ,

$$\Pr_{C \leftarrow \mathsf{Gen}'(1^\kappa), x \sim \{0,1\}^n}[x' \leftarrow \mathcal{B}(1^\kappa, C, C(x)),\ C(x') = C(x)] > 1 - 1/\kappa \ . \tag{2}$$

Notice that we may assume without loss of generality that this holds for infinitely many values of κ of the form $\kappa = (100\gamma)^4$ for integer γ.

We claim that Algorithm 1 solves average-case $\mathsf{CollisionAvoid}$ on the distribution produced by $\mathsf{Gen}(1^\gamma)$ with probability larger than $1 - 1/\gamma$ for arbitrarily large γ.

To see that the algorithm is correct, let $\kappa := (100\gamma)^4$, let $\alpha := 1000\gamma^3$, and let $\Delta := 10\gamma + 1$. We may assume that κ is such that Equation (2) holds. Let S be the set of circuits C that satisfy

$$\Pr_{x \sim \{0,1\}^n}[x' \leftarrow \mathcal{B}(1^\kappa, C, C(x)),\ C(x') = C(x)] > 1 - 1/\alpha \ .$$

Define an algorithm $\mathcal{B}'(1^\alpha, C) := \mathcal{B}(1^\kappa, C)$ and apply Lemma 2.19, we have that

$$\Pr_{C \sim \mathsf{Gen}(1^\gamma)}[\mathcal{A}^{\mathcal{B}'} \text{ solves } \mathsf{CollisionAvoid}] \geq (1 - \alpha/\kappa) \cdot \Pr\left[\mathcal{A}^{\mathcal{B}'} \text{ solves } \mathsf{CollisionAvoid} \mid C \in S\right] \ .$$

It therefore suffices to show that for each $C \in S$, $\mathcal{A}^{\mathcal{B}'}$ solves $\mathsf{CollisionAvoid}$ on C with probability at least $(1 - 1/\gamma)/(1 - \alpha/\kappa)$, and it follows from Lemma 4.1 that $\mathcal{A}^{\mathcal{B}'}$ solves $\mathsf{CollisionAvoid}$ on such C with probability at least $1 - 1/(2\gamma) > (1 - 1/\gamma)/(1 - \alpha/\kappa)$. □

5 $\mathsf{AffineAvoid}$ to Minicrypt

In this section we show that (worst-case/average-case) hardness of $\mathsf{AffineAvoid}$ implies the existence of weak (auxiliary-input/standard) one-way function families.

In the following technical lemma, we show how to use an algorithm that inverts a circuit $\mathcal{C}$ with high probability to solve $(1, \log\log n)$-$\mathsf{AffineAvoid}$.

Lemma 5.1. *There is a PPT oracle algorithm $\mathcal{A}$ such that for every expanding circuit $C : \{0,1\}^{n(\kappa)} \to \{0,1\}^{n(\kappa)+1}$ and every oracle $\mathcal{B}$, if*

$$\Pr_{x \sim \{0,1\}^n}[C(\mathcal{B}(1^\kappa, C, C(x))) = C(x)] \geq 1 - 1/\kappa \ ,$$

then for γ satisfying that $\kappa = 10n\log(n)\gamma\log(4\gamma)$,

$$\Pr\left[\mathcal{A}^{\mathcal{B}}(1^\gamma, C) \textit{ solves } (1, \log\log n)\text{-}\mathsf{AffineAvoid}\right] \geq 1 - 1/(2\gamma) \ .$$

Proof. We will show that Algorithm 2 satisfies the requirements of the lemma. Clearly, the algorithm runs in polynomial time. It remains to show that $\mathcal{A}$ solves $(1, \log\log n)$-$\mathsf{AffineAvoid}$ on C with probability at least $1 - 1/(2\gamma)$. Recall that $\kappa := 10n\log(n)\gamma\log(4\gamma)$ and let $\Delta := 2n\log(4\gamma)$.

For $1 \leq i \leq \Delta$, let E_i be the event where $\mathcal{A}$ outputs an affine subspace in the ith iteration of the loop. Then by the union bound,

$$\begin{aligned} \Pr[\mathcal{A} \text{ fails}] &\leq \sum_{i=1}^{\Delta} \Pr[\mathcal{A} \text{ fails and } E_i] + \Pr[\mathcal{A} \text{ fails and } \neg E_1, \ldots, \neg E_\Delta] \\ &\leq \Delta \Pr[\mathcal{A} \text{ fails and } E_1] + \Pr[\neg E_1, \ldots, \neg E_\Delta] \ . \end{aligned} \tag{3}$$

Algorithm 2: $\mathcal{A}^{\mathcal{B}}(1^\gamma, C)$

```
Set $d := \log\log n$;
Set $\kappa := 10n\log(n)\gamma\log(4\gamma)$;
Do $\Delta := 2n\log(4\gamma)$ times
    Sample $s, v_1, \dots, v_d \sim \{0,1\}^{n+1}$;
    Set $Q = s + \mathrm{span}(v_1, \dots, v_d)$;
    if $\dim(Q) = d$ and $\forall y \in Q, C(\mathcal{B}(1^\kappa, C, y)) \neq y$ then
        Output $s, v_1, \dots, v_d$;
    end
end
```

An affine subspace Q is a solution to AffineAvoid if $\dim(Q) = d$ and $Q \cap \mathrm{Im}(C) = \emptyset$. From Lemma 2.27, we know that for a random choice of $s, v_1, \dots, v_d$, the affine subspace $Q = s + \mathrm{span}(v_1, \dots, v_d)$ is a solution with probability at least $1/(2n)$. Moreover, if Q is a solution, then $\mathcal{A}$ outputs it. Therefore, from Lemma 2.27, the probability of not sampling such a Q in Δ trials is at most

$$\Pr[\neg E_1, \dots, \neg E_\Delta] \leq (1 - 1/(2n))^\Delta \ . \tag{4}$$

It remains to bound the probability $\Pr[\mathcal{A}$ fails and $E_1]$. This is the probability that $\mathcal{A}$ samples an affine subspace Q such that $\dim(Q) = d$, $Q \cap \mathrm{Im}(C) \neq \emptyset$, and the one-way function inverter $\mathcal{B}$ fails to invert on each point from $Q \cap \mathrm{Im}(C)$.

$$\begin{aligned}
\Pr[\mathcal{A} \text{ fails and } E_1] &\leq \Pr_{Q,\mathcal{A},\mathcal{B}}[Q \cap \mathrm{Im}(C) \neq \emptyset \text{ and } \forall y \in Q \cap \mathrm{Im}(C) : C(\mathcal{B}(1^\kappa, C, y)) \neq y] \\
&\leq \sum_{y \in \mathrm{Im}(C)} \Pr_Q[y \in Q] \cdot \Pr_{\mathcal{B}}[C(\mathcal{B}(1^\kappa, C, y)) \neq y] \\
&\leq \frac{2^d}{2^{n+1}} \cdot \sum_{y \in \mathrm{Im}(C)} \Pr_{\mathcal{B}}[C(\mathcal{B}(1^\kappa, C, y)) \neq y] \\
&= 2^d \cdot \Pr_{y,\mathcal{B}}[C(\mathcal{B}(1^\kappa, C, y)) \neq y \text{ and } y \in \mathrm{Im}(C)] \\
&\leq 2^{d-1} \cdot \Pr_x[x' \leftarrow \mathcal{B}(1^\kappa, C, C(x)),\ C(x') \neq C(x)] \\
&\leq \log n/(2\kappa) \ ,
\end{aligned} \tag{5}$$

where the penultimate inequality follows from Lemma 2.20, and the last inequality uses $d = \log\log n$ and that C satisfies Equation (7).

Using Equations (3) to (5), we can now bound the failure probability of $\mathcal{A}$ as follows,

$$\begin{aligned}
\Pr[\mathcal{A} \text{ fails}] &\leq \Delta \log n/(2\kappa) + (1 - 1/(2n))^\Delta \\
&\leq \Delta \log n/(2\kappa) + e^{-\Delta/2n} \\
&\leq 1/(10\gamma) + 1/(4\gamma) \\
&< 1/(2\gamma) \ ,
\end{aligned}$$

where the penultimate inequality follows from $\Delta = 2n\log(4\gamma)$ and $\kappa = 10n\log(n)\gamma\log(4\gamma)$. This finishes the proof of the lemma. ☐

Theorem 5.2. *Suppose* $(1, \log\log n)$-$\mathsf{AffineAvoid}$ *is weakly hard in the worst case, then weak auxiliary-input one-way functions exist.*

As one could boost weak auxiliary-input one-way function using Theorem 2.24, the existence of auxiliary-input one-way functions follows from the weak worst-case hardness of $\mathsf{AffineAvoid}$ (by Theorem 5.2 and Theorem 3.6).

Corollary 5.3. *For any efficiently computable* $\ell := \ell(n) \geq 1$, *if* $(\ell(n), \ell(n) + \log\log n)$-$\mathsf{AffineAvoid}$ *is weakly hard in the worst case, then auxiliary-input one-way functions exist.*

Proof of Theorem 5.2. Define $\mathcal{C}_\kappa \subseteq \{0,1\}^{s(\kappa)}$ to be the set of all circuits $C : \{0,1\}^{n(\kappa)} \to \{0,1\}^{n(\kappa)+1}$ and $\mathcal{C} := \cup_{\kappa\in\mathbb{N}}\mathcal{C}_\kappa$. We claim that $\{f_C(x) := C(x)\}_{C\in\mathcal{C}}$ is a weak auxiliary-input one-way function.

Assume towards contradiction that there is a PPT algorithm $\mathcal{B}$ such that for all sufficiently large κ and $\forall C \in \mathcal{C}_\kappa$,

$$\Pr_{x\sim\{0,1\}^n}[C(\mathcal{B}(1^\kappa, C, C(x))) = C(x)] \geq 1 - 1/\kappa\ .$$

Then by Lemma 5.1, the PPT algorithm $\mathcal{A}^{\mathcal{B}}$ solves $\mathsf{AffineAvoid}$ with probability at least $1 - 1/(2\gamma) > 1 - 1/\gamma$ for all circuits in $\mathcal{C}_\kappa$, contradicting the worst-case hardness of $(1, \log\log n)$-$\mathsf{AffineAvoid}$. ☐

Theorem 5.4. *Suppose* $(1, \log\log n)$-$\mathsf{AffineAvoid}$ *is weakly hard on average with generator* Gen, *then there exists a generator* Gen' *that samples weak one-way circuits.*

Combining the above with Theorems 2.18 and 3.6 immediately implies the following, which is the formal version of Theorem 1.4.

Corollary 5.5. *For any efficiently computable* $\ell := \ell(n) \geq 1$, *if* $(\ell(n), \ell(n) + \log\log n)$-$\mathsf{AffineAvoid}$ *is weakly hard on average with some generator, then (standard) one-way functions exist.*

Proof of Theorem 5.4. Let $\mathsf{Gen}(1^\gamma)$ output circuits $C\colon \{0,1\}^{n(\gamma)} \to \{0,1\}^{n(\gamma)+1}$. Since Gen is efficiently computable, $n(\gamma) \leq \gamma^c$ for some constant integer $c \geq 1$ and all large enough γ. For any $\kappa > 0$, we define $\mathsf{Gen}'(1^\kappa) := \mathsf{Gen}(1^\gamma)$ for $\gamma := \gamma(\kappa) = \lfloor \kappa^{1/(c+3)} \rfloor$. Notice that this choice of γ satisfies

$$1000n(\gamma)\log(n(\gamma))\gamma^2\log(4\gamma) \leq \kappa$$

for all sufficiently large κ.

Since Gen outputs a circuit $C : \{0,1\}^{n(\gamma)} \to \{0,1\}^{n(\gamma)+1}$ which is an instance of $(1, \log\log n(\gamma))$-$\mathsf{AffineAvoid}$, Gen' outputs a circuit $C : \{0,1\}^{n'(\kappa)} \to \{0,1\}^{n'(\kappa)+1}$ where $n'(\kappa) := n(\gamma(k))$.

We claim that Gen' samples a weak one-way circuit. If not, there is a PPT algorithm $\mathcal{B}$ such that for infinitely many κ,

$$\Pr_{C\leftarrow\mathsf{Gen}'(1^\kappa), x\sim\{0,1\}^n}[x' \leftarrow \mathcal{B}(1^\kappa, C, C(x)),\ C(x') = C(x)] > 1 - 1/\kappa\ . \tag{6}$$

We claim that Algorithm 2 solves average-case $(1, \log\log n(\gamma))$-$\mathsf{AffineAvoid}$ on the distribution produced by $\mathsf{Gen}(1^\gamma)$ with probability at least $1 - 1/\gamma$.

To see that the algorithm is correct, let κ be such that Equation (6) holds and let $\alpha := 10n\log(n)\gamma\log(4\gamma)$. Let S be the set of circuits C that satisfy

$$\Pr_{x\sim\{0,1\}^n}[x' \leftarrow \mathcal{B}(1^\kappa, C, C(x)),\ C(x') = C(x)] > 1 - 1/\alpha\ . \tag{7}$$

Define an algorithm $\mathcal{B}'(1^\alpha, C) := \mathcal{B}(1^\kappa, C)$ and apply Lemma 2.19, we have that

$$\Pr_{C\sim\mathsf{Gen}(1^\gamma)}[\mathcal{A}^{\mathcal{B}'} \text{ solves } \mathsf{AffineAvoid}] \geq (1 - \alpha/\kappa)\cdot \Pr\left[\mathcal{A}^{\mathcal{B}'} \text{ solves } \mathsf{AffineAvoid} \mid C \in S\right]\ .$$

By Lemma 2.19,

$$\Pr_{C\sim\mathsf{Gen}(1^\gamma)}[\mathcal{A} \text{ solves } \mathsf{AffineAvoid} \text{ on } C] \geq (1 - \alpha/\kappa)\cdot \Pr[\mathcal{A} \text{ solves } \mathsf{AffineAvoid} \text{ on } C \mid C \in S]\ .$$

It remains to show that for each $C \in S$, $\mathcal{A}^{\mathcal{B}'}$ solves $(1, \log\log n)$-$\mathsf{AffineAvoid}$ on C with probability at least $(1 - 1/\gamma)/(1 - \alpha/\kappa)$. Due to the choice of $\alpha/\kappa = 1/(100\gamma)$, it follows from Lemma 5.1 that $\mathcal{A}^{\mathcal{B}'}$ is successful on such C with probability at least $1 - 1/(2\gamma) \geq (1 - 1/\gamma)/(1 - \alpha/\kappa)$. □

Acknowledgments

We would like to thank Divesh Aggarwal and Siyao Guo for fruitful discussions.

ZL and SS are supported in part by the National Science Foundation CAREER award to Justin Thaler (grant CCF-1845125). AG is supported by the National Science Foundation CAREER award (grant CCF-2338730). NSD is supported in part by the NSF under Grants Nos. CCF-2122230 and CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google. AG, SS, and NSD would like to thank Divesh Aggarwal, the National University of Singapore, and the Centre for Quantum Technologies, at which some of this work was completed.

We are thankful to anonymous reviewers of this paper for pointing out relevant results from [RSW22, ILW23].

References

[CHLR23] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *STOC*, 2023. 1, 4

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *STOC*, 2024. 1, 4

[CL24] Yilei Chen and Jiatu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. In *STOC*, 2024. 1, 5

[GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In *RANDOM*, 2023. 1, 5

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In *RANDOM*, 2022. 1, 5

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 5

[HIR23] Yizhi Huang, Rahul Ilango, and Hanlin Ren. NP-hardness of approximating meta-complexity: A cryptographic approach. In *STOC*, 2023. 9

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *STOC*, 2023. 1, 4, 5, 22

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *CCC*, 1995. 2

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total functions in the polynomial hierarchy. In *ITCS*, 2021. 1, 4, 6

[Kor22] Oliver Korten. The hardest explicit construction. In *FOCS*, 2022. 1, 4, 5, 6

[KP24] Oliver Korten and Toniann Pitassi. Strong vs. weak range avoidance and the linear ordering principle. *ECCC*, 2024. Manuscript. 1

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *STOC*, 2024. 1, 5

[LORS24] Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. On the complexity of avoiding heavy elements. *ECCC*, 2024. Manuscript. 1

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. 3

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *FOCS*, 2022. 1, 4, 22

[VW08] Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for polynomials and protocols. *Theory of Computing*, 4(1):137–168, 2008. 12

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *FOCS*, 1982. 28

[Zim04] Marius Zimand. *Computational complexity: a quantitative perspective*. Elsevier, 2004. 10, 28

A Avoid and Kolmogorov Complexity

In this section, we prove Theorems A.1 and A.2. Specifically, in Appendix A.1, we provide an FZPP algorithm that solves Avoid using an (a, b)-$\mathsf{GapMcK}^{\mathsf{t},\infty}\mathsf{P}$ oracle for appropriate a, b, and t. Similarly, in Appendix A.2, we provide an FZPP algorithm that solves Avoid using a γ-$\mathsf{GapMKtP}$ oracle. (See Section 2.3 for the definitions of (a, b)-$\mathsf{GapMcK}^{\mathsf{t},\infty}\mathsf{P}$ and γ-$\mathsf{GapMKtP}$.)

A.1 Avoid to $\mathsf{GapMcK^{t,\infty}P}$

Theorem A.1. *For any $s := s(n) \geq 1$ and $g(n) > n + s(n)$, there is an* FZPP *reduction from s-*Avoid *on circuits of size $g(n)$ to (a,b)-*$\mathsf{GapMcK^{t,\infty}P}$ *for $a \leq n + O(1)$, $b \geq n + s(n) - 2$, and $t \leq \mathrm{poly}(g(n))$.*

Proof. Let $\mathcal{O}$ be an oracle for (a,b)-$\mathsf{GapMcK^{t,\infty}P}$. Notice that $\mathcal{O}(x,y)$ outputs 0 only if $\mathsf{K}^t(y|x) > a$.

Algorithm 3: $\mathsf{Avoid}(C)$

```
Sample $y \sim \{0,1\}^{n+s}$ ;
if $\mathcal{O}(C,y) = 0$ then
    Output $y$;
end
else
    Repeat the procedure;
end
```

We present the reduction in Algorithm 3. First, we prove the correctness of the reduction. For this, we show that for every $z \in \mathrm{Im}(C)$, $\mathsf{K}^t(z|C) \leq n + O(1)$. This is achieved by hard coding the n-bit input x and evaluating $C(x) = z$ in $\mathrm{poly}(|C|) = \mathrm{poly}(g(n))$ steps. Hence, any string y that is output by the reduction is not in the image of C.

It remains to prove that the reduction runs in expected polynomial time. For this, we show that the algorithm retries a constant number of times in expectation. Since y is chosen uniformly at random, Lemma 2.15 states that $\mathsf{K}^\infty(y|C) \geq n + s - 2$ with probability at least $1/2$. Thus, the reduction uses at most 2 attempts in expectation before finding a desired y. ☐

A.2 Avoid to $\mathsf{GapMKtP}$

Theorem A.2. *Let $s, g\colon \mathbb{Z}_{>0} \to \mathbb{Z}_{>0}$ be functions and $\gamma\colon \mathbb{Z}_{>0} \to \mathbb{Z}_{>0}$ be a non-decreasing function such that $s(n) = \omega(\log g(n))$ and*

$$\gamma(2g(n)^2) < s(n)/4 .$$

Then $s(n)$-Avoid *on circuits of size $g(n)$ is in* $\mathsf{FZPP}^{\gamma\text{-}\mathsf{GapMKtP}}$.

Note that we chose to present a relatively simple form of Theorem A.2, instead of attempting to present the strongest version of this theorem that we know how to prove. (E.g., a careful reading of our proof shows that it suffices to take $s \geq C \log g$ for some constant $C > 0$ that depends on the specific universal Turing machine used to define $\mathsf{GapMKtP}$, and that it suffices to take γ such that $\gamma((s+n) \cdot g + C \log g) < s - C \log g$.)

One interesting setting of the parameters of Theorem A.2 is where the approximation guarantee of the $\mathsf{GapMKtP}$ oracle is $\gamma(k) = O(\log k)$. In this case, for circuits of polynomial size $g(n) = \mathrm{poly}(n)$, the stretch in Theorem A.2 can be taken as low as $s = \omega(\log n)$. Also, for circuits of unrestricted size[5], the stretch in Theorem A.2 is only $s = \omega(n)$.

[5]Without loss of generality, we can assume that the circuit size g is at most 2^n, as otherwise Avoid can be trivially solved in time linear in the input size.

Corollary A.3. *Let $c > 0$ be an arbitrary constant, and $\gamma(k) = c \log k$. Then*

- *for any $s(n) = \omega(\log n)$, s-Avoid on circuits of polynomial size is in $\mathsf{FZPP}^{\gamma\text{-}\mathsf{GapMKtP}}$; and*
- *for any $s(n) = \omega(n)$, s-$\mathsf{Avoid} \in \mathsf{FZPP}^{\gamma\text{-}\mathsf{GapMKtP}}$.*

Proof of Theorem A.2. If the number of gates g is smaller than the number of outputs $n+s$, then some pair of the outputs computes the same function, and it is therefore trivial to solve Avoid on such circuits. Thus, in the following we assume that $g \geq n+s$, and, in particular, that $g(n)$ is unbounded.

Let $\mathcal{O}$ be an oracle for γ-$\mathsf{GapMKtP}$. For a string z of length ℓ, we will use $\mathcal{O}$ to compute $\mathsf{apxKt}(z)$ such that $\mathsf{Kt}(z) \leq \mathsf{apxKt}(z) < \mathsf{Kt}(z) + \gamma(\ell)$. To do this using $O(\ell)$ calls to $\mathcal{O}$, we simply output the highest value $i \in [\ell + O(\log \ell)]$ for which $\mathcal{O}(z,i)$ outputs 1. (Note that Kt of a string of length ℓ is bounded from above by $\ell + O(\log \ell)$.)

We describe our reduction's behavior on input a circuit $C : \{0,1\}^n \to \{0,1\}^{n+s}$ of size g in Algorithm 4 below.

Algorithm 4: $\mathsf{Avoid}(C)$

```
Sample $y_1, y_2, \ldots, y_g \sim \{0,1\}^{n+s}$;
$k_0 \leftarrow \mathsf{apxKt}(C)$;
for $i \in \{1, \ldots, g\}$ do
    $k_i \leftarrow \mathsf{apxKt}(C \circ y_1 \circ \cdots \circ y_i)$;
end
if $\exists i \in [g]$ s.t. $k_{i+1} - k_i \geq n + s/2$ then
    Output $y_{i+1}$;
end
else
    Repeat the procedure;
end
```

We first prove a basic fact about the k_i. To that end, note that for every $i \in [g+1]$,

$$|C \circ y_1 \circ \cdots \circ y_i| \leq g + i \cdot (n+s) \leq g + g \cdot (n+s) \leq 3g^2/2 ,$$

where we have used that $n+s \leq g$ and $g \leq g^2/2$ for large enough n. Thus, using the fact that $\gamma(2g(n)^2) < s(n)/4$ and that γ is non-decreasing, we have that

$$\begin{aligned}\gamma(\mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i)) &\leq \gamma\left(|C \circ y_1 \circ \cdots \circ y_i| + O(\log(|C \circ y_1 \circ \cdots \circ y_i|))\right) \\ &\leq \gamma(3g^2/2 + O(\log(3g^2/2))) \\ &\leq \gamma(2g^2) \\ &< s/4 .\end{aligned}$$

Therefore, for every $i \in [g]$,

$$\begin{aligned}k_{i+1} - k_i &\leq \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_{i+1}) + \gamma(\mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_{i+1})) - \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i) \\ &< \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_{i+1}) - \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i) + s/4 . \qquad (8)\end{aligned}$$

Next, we prove the correctness of the algorithm. Notice that given the description of C, every string y_{i+1} from the image of C can be described using n bits (namely, a preimage of y_{i+1}), and computed in time $\widetilde{O}(g)$. Thus, for every $y_{i+1} \in \mathrm{Im}(C)$,

$$\mathsf{Kt}(y_{i+1}|C) \leq n + O(\log g) \ . \tag{9}$$

In particular, if $y_{i+1} \in \mathrm{Im}(C)$, then

$$\begin{aligned} k_{i+1} - k_i &< \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_{i+1}) - \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i) + s/4 \\ &\leq \mathsf{Kt}(y_{i+1}|C \circ y_1 \circ \cdots \circ y_i) + \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i) - \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i) + s/4 + O(\log g) \\ &= \mathsf{Kt}(y_{i+1}|C \circ y_1 \circ \cdots \circ y_i) + s/4 + O(\log g) \\ &\leq \mathsf{Kt}(y_{i+1}|C) + s/4 + O(\log g) \\ &< n + s/4 + O(\log g) \\ &< n + s/2 \ , \end{aligned}$$

where the first inequality is by Equation (8), the second inequality uses Lemma 2.12, the penultimate inequality is by Equation (9), and the last inequality is due to $s = \omega(\log g)$. Therefore, any string y_{i+1} that is output by the algorithm is not in the image of C.

It remains to prove that the algorithm runs in expected polynomial time. For this, we show that the algorithm retries a constant number of times in expectation.

Since $y_1, \ldots, y_g$ are chosen randomly, Lemma 2.13 states that $\mathsf{Kt}(y_1 \circ \cdots \circ y_g) \geq g \cdot (n+s) - O(\log g)$ with probability at least $1 - 1/\mathrm{poly}(g) \geq 1/2$. It follows by Lemma 2.14 that with probability at least $1/2$,

$$\mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_g) \geq \mathsf{Kt}(y_1 \circ \cdots \circ y_g) - O(\log g) \geq g \cdot (n+s) - O(\log g) \ . \tag{10}$$

When Equation (10) holds, we have

$$\begin{aligned} \sum_{i=0}^{g-1} (\mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_{i+1}) - \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i)) &= \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_g) - \mathsf{Kt}(C) \\ &\geq g \cdot (n+s) - O(\log g) - \mathsf{Kt}(C) \ . \end{aligned}$$

It follows that when Equation (10) holds, there must exist some $i \in [g]$ such that

$$\begin{aligned} \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_{i+1}) - \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i) &\geq \frac{\mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_g) - \mathsf{Kt}(C)}{g} \\ &\geq (n+s) - \frac{O(\log g)}{g} - \frac{\mathsf{Kt}(C)}{g} \\ &\geq n + s - O(\log g) \\ &\geq n + 3s/4 \ , \end{aligned}$$

where the penultimate inequality holds due to the fact that $\mathsf{Kt}(C) \leq O(g \log g)$ (as a circuit with g gates can be described using $O(g \log g)$ bits), and the last inequality uses the fact that $s = \omega(\log g)$.

We conclude that with probability at least $1/2$, there exists an i such that

$$k_{i+1} - k_i \geq \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_{i+1}) - \mathsf{Kt}(C \circ y_1 \circ \cdots \circ y_i) - \gamma(2g^2) \geq n + 3s/4 - s/4 \geq n + s/2 \ .$$

Thus, the algorithm uses at most 2 attempts in expectation before finding a desired y_{i+1}. $\square$

B Hardness of hardness

Lastly, we show barriers to proving hardness of Avoid. Specifically, we show that if there is a randomized reduction from any problem $A \in \mathsf{FNP}$ to Avoid, then $A \in \mathsf{FZPP}$. (Here, we say $A \in \mathsf{FZPP}$ for a search problem A if there is a randomized algorithm $\mathcal{B}$ for A such that, whenever there exists a solution to an instance I_A, $\mathcal{B}$ will output a valid solution in expected polynomial time. We impose no restrictions on $\mathcal{B}$ when I_A has no solutions—e.g., it might not terminate.)

Theorem B.1. *For any constant $c > 0$, any $s := s(\ell) \geq 1$, and any $q := q(\ell) \geq 1$ satisfying $(1 - 2^{-s})^q \geq 1/\ell^c$ for all large enough ℓ, if there exists a (possibly randomized) polynomial-time reduction from a search problem $A \in \mathsf{FNP}$ to Avoid that on input an instance of A with size ℓ makes at most $q(\ell)$ calls to an Avoid oracle on circuits with stretch at least $s(\ell)$, then $A \in \mathsf{FZPP}$.*

Proof. Let $A \in \mathsf{FNP}$, and let $\mathcal{B}^{\mathsf{Avoid}}$ be the randomized polynomial-time reduction that makes at most q queries to its Avoid oracle. Let $p := p(\ell) \geq 1/\operatorname{poly}(\ell)$ be a lower bound on the success probability of $\mathcal{B}^{\mathsf{Avoid}}$ on inputs with length ℓ.

We now give an algorithm $\mathcal{D}$ that solves any instance I_A of A for which there exists a solution in expected polynomial time. The algorithm $\mathcal{D}$ on input I_A with size ℓ behaves as follows:

1. Simulate $\mathcal{B}^{\mathsf{Avoid}}(I_A)$.
2. Whenever $\mathcal{B}^{\mathsf{Avoid}}(I_A)$ makes a query to its Avoid oracle Avoid for a circuit $C : \{0,1\}^n \to \{0,1\}^{n+s'}$ for $s' \geq s$, $\mathcal{D}$ simulates the oracle's response with a uniformly random string of length $n + s'$.
3. When $\mathcal{B}^{\mathsf{Avoid}}(I_A)$ terminates and outputs a string w_A, $\mathcal{D}$ checks whether w_A is indeed a valid solution of I_A (which can be done efficiently because $A \in \mathsf{FNP}$).
4. If it is, $\mathcal{D}$ outputs w_A. Otherwise, it restarts.

It is clear that the algorithm $\mathcal{D}$ only ever outputs correct solutions w_A. Thus, it suffices to show that algorithm $\mathcal{D}$ runs in expected polynomial time on inputs I_A that have a valid solution. We do this by first bounding the probability that $\mathcal{D}$ succeeds in finding a valid witness string w_A in a single run of the loop above.

By definition, our simulation of $\mathcal{B}^{\mathsf{Avoid}}(I_A)$ must succeed with probability at least p if we solve all of the Avoid instances correctly. The probability that a random $(n + s')$-bit string is in the range of any fixed circuit $C : \{0,1\}^n \to \{0,1\}^{n+s'}$ is at most $2^{-s'} \leq 2^{-s}$. Thus, the probability that we succeed on all q instances is at least $(1 - 2^{-s})^q \geq 1/\ell^c$. And, the success probability of any given instance is therefore at least p/ℓ^c.

Thus, we run at most ℓ^c/p simulations in expectation. Since each simulation runs in polynomial time, $\mathcal{D}$ runs in expected polynomial time as claimed. □

There are two interesting choices of s and q that yield the following corollaries. By setting $q = 1$, then letting $s \geq 1$:

Corollary B.2. *If there exists a (randomized) Karp reduction from FSAT to Avoid (even with stretch 1), then $\mathsf{FZPP} = \mathsf{FNP}$.*

Furthermore, by letting $q = \text{poly}(\ell)$, we can then allow $s = \omega(\log(\ell)) = \omega(\log(n))$.

Corollary B.3. *If there exists a (randomized) polynomial-time reduction from* FSAT *to* s-Avoid *of stretch* $s(n) = \omega(\log(n))$*, then* $\mathsf{FZPP} = \mathsf{FNP}$*.*

In both cases, we remark that $\mathsf{FZPP} = \mathsf{FNP}$ would also imply that $\mathsf{NP} = \mathsf{RP}$.

Theorem B.4. *If* $\mathsf{FZPP} = \mathsf{FNP}$*, then* $\mathsf{NP} = \mathsf{RP}$*.*

Proof. Assume that FSAT has an algorithm B such that on satisfiable instances $\phi : \{0,1\}^n \to \{0,1\}$, $A(\phi)$ takes expected $p(n)$ time to output a satisfying assignment y for some polynomial $p(n)$.

We now give a randomized polynomial time algorithm D that has one-sided error for SAT. On input ϕ, we simulate $B(\phi)$ for $2p(n)$ steps, and if $B(\phi)$ outputs an assignment y, we output $\phi(y)$, otherwise we output 0. Note that this means our algorithm only outputs 1 if y is a satisfying assignment.

Assuming that ϕ is not satisfiable, we will never output 1. So, it suffices to lower bound the probability we output 1 assuming that ϕ is satisfiable. Note that by Markov, the probability that $B(\phi)$ uses more than $2p(n)$ is at most $1/2$. Thus the probability that we output 1 on satisfiable instances ϕ is at least $1/2$. $\square$

C Proof Sketch of Theorem 2.24

In this section, we sketch a simple proof of Theorem 2.24 which essentially follows from [Yao82]. Our presentation largely follows [Zim04, Theorem 5.2.1].

Proof Sketch. Let $f = \{f_z : \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}\}_{z\in\{0,1\}^{s(\kappa)}}$ be a weak auxiliary-input one-way function. Let $\ell := n \cdot \kappa$. We claim that $g = \{g_z : \{0,1\}^{\ell(\kappa)n(\kappa)} \to \{0,1\}^{\ell(\kappa)m(\kappa)}\}_{z\in\{0,1\}^{s(\kappa)}}$ where $g_z(x_1, \dots, x_\ell) := f_z(x_1)||\dots||f_z(x_\ell)$ is an auxiliary-input one-way function.

Set $\gamma := \kappa^c$ for some fixed constant c. Assume towards contradiction that there is an algorithm $\mathcal{B}$ such that for all sufficiently large κ and $\forall z \in \{0,1\}^{s(\kappa)}$,

$$\Pr_{X\sim\{0,1\}^{\ell n}}[g_z(\mathcal{B}(1^\gamma, z, g_z(X))) = g_z(X)] > \frac{1}{\gamma} + e^{-n} .$$

Consider Algorithm 5 for inverting f_z. Let $G \subseteq \{0,1\}^{\ell n}$ be the set of inputs of g_z that $\mathcal{B}$ successfully inverts. From the assumption above, one have that $|G| \geq 2^{\ell n}/\gamma$.

For any $x \in \{0,1\}^n$, let $N(x)$ be the set of ℓ-tuples $(x_1, x_2, \dots, x_\ell)$ containing x. Let V be the set of 'good' x satisfying that $\frac{N(x)\cap G}{N(x)} \geq \frac{1}{\gamma\ell}$. It suffices to show that $|V| \geq (1 - 1/\kappa)2^n$ as $\Delta = n\gamma\ell$ iterations of the loop would boost the success probability of inverting $f_z(x)$ to almost 1 for $x \in V$.

Algorithm 5: $\mathcal{A}^{\mathcal{B}}(1^{\gamma}, z, y)$

```
Do $\Delta := n\gamma\ell$ times
    Sample random $i \in \{1, \dots, \ell\}$;
    Sample $\ell - 1$ random strings from $\{0,1\}^n$ denoted by $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_\ell$;
    Set $Y := f_z(x_1)||\dots||f_z(x_{i-1})||y||f_z(x_{i+1})||\dots||f_z(x_\ell)$;
    Run $\mathcal{B}$ on input $Y$, let the output be $x'_1||x'_2||\dots||x'_i||\dots||x'_\ell$ ;
    if $f_z(x'_i) = y$ then
        Output $x'_i$;
    end
end
Output arbitrarily;
```

Let $\overline{V}$ be the complement of V. We have

$$\begin{aligned}|N(\overline{V}) \cap G| &\le \sum_{x \in \overline{V}} |N(x) \cap G| \\ &< 2^n \cdot \frac{1}{\gamma\ell} \cdot (\ell \cdot 2^{(\ell-1)n}) \\ &= \frac{1}{\gamma} 2^{\ell n} .\end{aligned}$$

This is only possible with $|\overline{V}| < \frac{1}{\kappa} 2^n$. Assume not, then a random $X \in \{0,1\}^{\ell n}$ avoids $N(\overline{V})$ with probability at most e^{-n}. Therefore,

$$\begin{aligned}|N(\overline{V}) \cap G| &= |G| - |G \cap \overline{N(\overline{V})}| \\ &\ge |G| - |\overline{N(\overline{V})}| \\ &> \frac{1}{\gamma} 2^{\ell n} .\end{aligned}$$

This completes the proof sketch. □