MINIMUM CIRCUIT SIZE, GRAPH ISOMORPHISM, AND RELATED PROBLEMS*

ERIC ALLENDER†, JOSHUA A. GROCHOW‡, DIETER VAN MELKEBEEK§, CRISTOPHER MOORE¶, AND ANDREW MORGAN§

Abstract. We study the computational power of deciding whether a given truth-table can be described by a circuit of a given size (the Minimum Circuit Size Problem, or MCSP for short), and of the variant denoted MKTP where circuit size is replaced by a polynomially-related Kolmogorov measure. Prior to our work, all reductions from supposedly-intractable problems to MCSP / MKTP hinged on the power of MCSP / MKTP to distinguish random distributions from distributions produced by hardness-based pseudorandom generator constructions. We develop a fundamentally different approach inspired by the well-known interactive proof system for the complement of Graph Isomorphism (GI). It yields a randomized reduction with zero-sided error from GI to MKTP. We generalize the result and show that GI can be replaced by any isomorphism problem for which the underlying group satisfies some elementary properties. Instantiations include Linear Code Equivalence, Permutation Group Conjugacy, and Matrix Subspace Conjugacy. Along the way we develop encodings of isomorphism classes that are efficiently decodable and achieve compression that is at or near the information-theoretic optimum; those encodings may be of independent interest.

Key words. Reductions between NP-intermediate problems, Graph Isomorphism, Minimum Circuit Size Problem, time-bounded Kolmogorov complexity

AMS subject classifications. 68Q15, 68Q17, 68Q30

1. Introduction.

Finding a circuit of minimum size that computes a given Boolean function constitutes the overarching goal in nonuniform complexity theory. It defines an interesting computational problem in its own right, the complexity of which depends on the way the Boolean function is specified. A generic and natural, albeit verbose, way to specify a Boolean function is via its truth-table. The corresponding decision problem is known as the Minimum Circuit Size Problem (MCSP): Given a truth-table and a threshold θ, does there exist a Boolean circuit of size at most θ that computes the Boolean function specified by the truth-table? The interest in MCSP dates back to the dawn of theoretical computer science [42]. It continues today partly due to the fundamental nature of the problem, and partly because of the work on natural proofs and the connections between pseudorandomness and computational hardness.

A closely related problem from Kolmogorov complexity theory is the Minimum KT Problem (MKTP), which deals with compression in the form of efficient programs instead of circuits. Rather than asking if the input has a small circuit when interpreted as the truth-table of a Boolean function, MKTP asks if the input has a small program that produces each individual bit of the input quickly. To be more specific, let us fix a universal Turing machine U. We consider descriptions of the input string x in the form of a program d such that, for every bit position i, U on input d and i outputs

*Submitted to the editors 21 November 2017. An extended abstract of this paper appeared in the *Proceedings of the 9th Innovations in Theoretical Computer Science Conference* (ITCS'18) [3].

†Rutgers University, Piscataway, NJ, USA (allender@cs.rutgers.edu). The research of this author was supported by NSF grants CCF-1555409 and CCF-1514164.

‡University of Colorado at Boulder, Boulder, CO, USA (jgrochow@colorado.edu). The research of this author was supported by an Omidyar Fellowship from the Santa Fe Institute and NSF grants DMS-1750319 and DMS-1622390.

§University of Wisconsin–Madison, Madison, WI, USA (dieter@cs.wisc.edu, amorgan@cs.wisc.edu). The research of these authors was supported by NSF grant CCF-1319822.

¶Santa Fe Institute, Santa Fe, NM, USA (moore@santafe.edu).

the i-th bit of x in T steps. The KT cost of such a description is defined as $|d| + T$, i.e., the bit-length of the program plus the running time. The KT complexity of x, denoted $\mathrm{KT}(x)$, is the minimum KT cost of a description of x. $\mathrm{KT}(x)$ is polynomially related to the circuit complexity of x when viewed as a truth-table (see Section 2.1 for a more formal treatment). On input a string x and an integer θ, MKTP asks whether $\mathrm{KT}(x) \leq \theta$.

Both MCSP and MKTP are in NP but are not known to be in P or NP-complete. As such, they are two prominent candidates for NP-intermediate status. Others include factoring integers, discrete log over prime fields, graph isomorphism (GI), and a number of similar isomorphism problems.

Whereas NP-complete problems all reduce one to another, even under fairly simple reductions, less is known about the relative difficulty of presumed NP-intermediate problems. Regarding MCSP and MKTP, despite their apparent similarity, it is not known whether one reduces to the other. Factoring integers and discrete log over prime fields are known to reduce to both MCSP and MKTP under randomized reductions with zero-sided error [1, 39]. Recently, Allender and Das [2] showed that GI and all of SZK (Statistical Zero Knowledge) reduce to both under randomized reductions with bounded error.

Those reductions and, in fact, all reductions of supposedly-intractable problems to MCSP / MKTP prior to our work proceed along the same well-trodden path. Namely, MCSP / MKTP is used as an efficient statistical test to distinguish random distributions from pseudorandom distributions, where the pseudorandom distribution arises from a hardness-based pseudorandom generator construction. In particular, [28] employs the construction based on the hardness of factoring Blum integers, [1, 2, 5, 39] use the construction from [24] based on the existence of one-way functions, and [1, 13] make use of the Nisan-Wigderson construction [35]. The property that MCSP / MKTP breaks the construction implies that the underlying hardness assumption fails relative to MCSP / MKTP, and thus that the supposedly hard problem reduces to MCSP / MKTP.

Contributions. The main conceptual contribution of our paper is a fundamentally different way of constructing reductions to MKTP based on a novel use of known interactive proof systems. Our approach applies to GI and a broad class of isomorphism problems. A common framework for those isomorphism problems is another conceptual contribution. In terms of results, our new approach allows us to eliminate the errors in the recent reductions from GI to MKTP, and more generally to establish *zero-sided error* randomized reductions to MKTP from many isomorphism problems within our framework. These include Linear Code Equivalence, Matrix Subspace Conjugacy, and Permutation Group Conjugacy (see Section 6 for the definitions). The technical contributions mainly consist of encodings of isomorphism classes that are efficiently decodable and achieve compression that is at or near the information-theoretic optimum.

Before describing the underlying ideas, we note that our techniques remain of interest even in light of the recent quasi-polynomial-time algorithm for GI [8]. For one, GI is still not known to be in P, and Group Isomorphism stands as a significant obstacle to this (as stated at the end of [8]). More importantly, our techniques also apply to the other isomorphism problems mentioned above, for which the current best algorithms are still exponential.

Let us also provide some evidence that our approach for constructing reductions to MKTP differs in an important way from the existing ones. We claim that the existing approach can only yield zero-sided error reductions to MKTP from problems

that are in $\mathsf{NP} \cap \mathsf{coNP}$, a class that neither GI nor any of the other isomorphism problems mentioned above are known to reside in. The reason for the claim is that the underlying hardness assumptions are fundamentally average-case,[1] which implies that the reduction can have both false positives and false negatives. For example, in the papers employing the construction from [24], MKTP is used in a subroutine to invert a polynomial-time-computable function (see Lemma 2.2 in Section 2.1), and the subroutine may fail to find an inverse. Given a reliable but imperfect subroutine, the traditional way to eliminate false positives is to use the subroutine for constructing an efficiently verifiable membership witness, and only accept after verifying its validity. As such, the existence of a traditional reduction without false positives from a language L to MKTP implies that $L \in \mathsf{NP}$. Similarly, a traditional reduction from L to MKTP without false negatives is only possible if $L \in \mathsf{coNP}$, and zero-sided error is only possible if $L \in \mathsf{NP} \cap \mathsf{coNP}$.

Main Idea. Instead of using the oracle for MKTP in the *construction* of a candidate witness and then verifying the validity of the candidate without the oracle, we use the power of the oracle in the *verification* process. This obviates the need for the language L to be in $\mathsf{NP} \cap \mathsf{coNP}$ in the case of reductions with zero-sided error.

Let us explain how to implement this idea for $L = \mathrm{GI}$. Recall that an instance of GI consists of a pair (G_0, G_1) of graphs on the vertex set $[n]$, and the question is whether $G_0 \equiv G_1$, i.e., whether there exists a permutation $\pi \in S_n$ such that $G_1 = \pi(G_0)$, where $\pi(G_0)$ denotes the result of applying the permutation π to the vertices of G_0. In order to develop a zero-sided error algorithm for GI, it suffices to develop one without false negatives. This is because the false positives can subsequently be eliminated using the known search-to-decision reduction for GI [32].

The crux for obtaining a reduction without false negatives from GI to MKTP is a witness system for the complement $\overline{\mathrm{GI}}$ inspired by the well-known two-round interactive proof system for $\overline{\mathrm{GI}}$ [19]. Consider the distribution $R_G(\pi) \doteq \pi(G)$ where $\pi \in S_n$ is chosen uniformly at random. By the Orbit–Stabilizer Theorem, for any fixed G, R_G is uniform over a set of size $N \doteq n!/|\operatorname{Aut}(G)|$ and thus has entropy $s = \log(N)$, where $\operatorname{Aut}(G) \doteq \{\pi \in S_n : \pi(G) = G\}$ denotes the set of automorphisms of G. For ease of exposition, let us assume that $|\operatorname{Aut}(G_0)| = |\operatorname{Aut}(G_1)|$ (which is actually the hardest case for GI), so both R_{G_0} and R_{G_1} have the same entropy s. Consider picking $r \in \{0,1\}$ uniformly at random, and setting $G = G_r$. If $(G_0, G_1) \in \mathrm{GI}$, the distributions R_{G_0}, R_{G_1}, and R_G are all identical, and therefore R_G also has entropy s. On the other hand, if $(G_0, G_1) \notin \mathrm{GI}$, the entropy of R_G equals $s+1$. The extra bit of information corresponds to the fact that in the nonisomorphic case each sample of R_G reveals the value of r that was used, whereas that bit gets lost in the reduction in the isomorphic case.

The difference in entropy suggests that a typical sample of R_G can be compressed more in the isomorphic case than in the nonisomorphic case. If we can compute some threshold such that $\mathrm{KT}(R_G)$ *never* exceeds the threshold in the isomorphic case, and exceeds it with nonnegligible probability in the nonisomorphic case, we have the witness system for $\overline{\mathrm{GI}}$ that we aimed for: Take a sample from R_G, and use the oracle for MKTP to check that it cannot be compressed at or below the threshold. The entropy difference of 1 may be too small to discern, but we can amplify the difference by taking multiple samples and concatenating them. Thus, we end up with a randomized mapping reduction of the following form, where t denotes the number

[1] In some settings worst-case to average-case reductions are known, but these reductions are themselves randomized with two-sided error.

of samples and θ the threshold:

(1.1) Pick $r \doteq r_1 \dots r_t \in \{0,1\}^t$ and $\pi_i \in S_n$ for $i \in [t]$, uniformly at random.
Output (y, θ) where $y \doteq y_1 \dots y_t$ and $y_i \doteq \pi_i(G_{r_i})$.

We need to analyze how to set the threshold θ and argue correctness for a value of t that is polynomially bounded. In order to do so, let us first consider the case where the graphs G_0 and G_1 are *rigid*, i.e., they have no nontrivial automorphisms, or equivalently, $s = \log(n!)$.

- If $G_0 \not\equiv G_1$, the string y contains all of the information about the random string r and the t random permutations $\pi_1, \dots, \pi_t$, which amounts to $ts + t = t(s+1)$ bits of information. This implies that y has KT-complexity close to $t(s+1)$ with high probability.
- If $G_0 \equiv G_1$, then we can efficiently produce each bit of y from the adjacency matrix representation of G_0 (n^2 bits) and the function table of permutations $\tau_i \in S_n$ (for $i \in [t]$) such that $y_i \doteq \pi_i(G_{r_i}) = \tau_i(G_0)$. Moreover, the set of all permutations S_n allows an efficiently decodable indexing, i.e., there exists an efficient algorithm that takes an index $k \in [n!]$ and outputs the function table of the k-th permutation in S_n according to some ordering. An example of such an indexing is the Lehmer code (see, e.g., [31, pp. 12-13] for specifics). This shows that

  $$\mathrm{KT}(y) \leq t\lceil s \rceil + (n + \log(t))^c \tag{1.2}$$

  for some constant c, where the first term represents the cost of the t indices of $\lceil s \rceil$ bits each, and the second term represents the cost of the n^2 bits for the adjacency matrix of G_0 and the polynomial running time of the decoding process.

If we ignore the difference between s and $\lceil s \rceil$, the right-hand side of (1.2) becomes $ts + n^c$, which is closer to ts than to $t(s+1)$ for t any sufficiently large polynomial in n, say $t = n^{c+1}$. Thus, setting θ halfway between ts and $t(s+1)$, i.e., $\theta \doteq t(s + \frac{1}{2})$, ensures that $\mathrm{KT}(y) > \theta$ holds with high probability if $G_0 \not\equiv G_1$, and never holds if $G_0 \equiv G_1$. This yields the desired randomized mapping reduction without false negatives, modulo the rounding issue of s to $\lceil s \rceil$. The latter can be handled by aggregating the permutations τ_i into blocks so as to make the amortized cost of rounding negligible. The details are captured in the Blocking Lemma of Section 3.1.

What changes in the case of non-rigid graphs? For ease of exposition, let us again assume that $|\operatorname{Aut}(G_0)| = |\operatorname{Aut}(G_1)|$. There are two complications:

(i) We no longer know how to efficiently compute the threshold $\theta \doteq t(s + \frac{1}{2})$ because $s \doteq \log(N)$ where $N \doteq n!/|\operatorname{Aut}(G_0)| = n!/|\operatorname{Aut}(G_1)|$ involves the size of the automorphism group.

(ii) The Lehmer code no longer provides sufficient compression in the isomorphic case as it requires $\log(n!)$ bits per permutation whereas we only have s to spend, which could be considerably less than $\log(n!)$.

In order to resolve (ii) we develop an efficiently decodable indexing of cosets for any subgroup of S_n given by a list of generators (see Lemma 3.4 in Section 3.2). In fact, our scheme even works for cosets of a subgroup within another subgroup of S_n, a generalization that may be of independent interest (see Lemma A.1 in the Appendix). Applying our scheme to $\operatorname{Aut}(G)$ and including a minimal list of generators for $\operatorname{Aut}(G)$ in the description of the program p allows us to maintain (1.2).

Regarding (i), we can deduce a good approximation to the threshold with high probability by taking, for both choices of $r \in \{0,1\}$, a polynomial number of samples

of R_{G_r} and using the oracle for MKTP to compute the exact KT-complexity of their concatenation. This leads to a randomized reduction from GI to MKTP with bounded error (from which one without false positives follows as mentioned before), reproving the earlier result of [2] using our new approach (see Remark 3.5 in Section 3.2 for more details).

In order to avoid false negatives, we need to improve the above approximation algorithm such that it never produces a value that is too small, while maintaining efficiency and the property that it outputs a good approximation with high probability. In order to do so, it suffices to develop a *probably-correct overestimator* for the quantity $n!/|\operatorname{Aut}(G)|$, i.e., a randomized algorithm that takes as input an n-vertex graph G, produces the correct quantity with high probability, and never produces a value that is too small; the algorithm should run in polynomial time with access to an oracle for MKTP. Equivalently, it suffices to develop a probably-correct *under*estimator of similar complexity for $|\operatorname{Aut}(G)|$.

The latter can be obtained from the known search-to-decision procedures for GI, and answering the oracle calls to GI using the above two-sided error reduction from GI to MKTP. There are a number of ways to implement this strategy; here is one that generalizes to a number of other isomorphism problems including Linear Code Equivalence.

1. Find a list of permutations that generates a subgroup of $\operatorname{Aut}(G)$ such that the subgroup equals $\operatorname{Aut}(G)$ with high probability.
   Finding a list of generators for $\operatorname{Aut}(G)$ deterministically reduces to GI (see, e. g., [32, implicit in sections 1.2–1.3]).[2] Substituting the oracle for GI by a two-sided error algorithm yields a list of permutations that generates $\operatorname{Aut}(G)$ with high probability, and always produces a subgroup of $\operatorname{Aut}(G)$. The latter property follows from the inner workings of the reduction, or can be imposed explicitly by checking every permutation produced and dropping it if it does not map G to itself. We use the above randomized reduction from GI to MKTP as the two-sided error algorithm for GI.
2. Compute the order of the subgroup generated by those permutations.
   There is a known deterministic polynomial-time algorithm to do this (see, e. g., [41]).

The resulting probably-correct underestimator for $|\operatorname{Aut}(G)|$ runs in polynomial time with access to an oracle for MKTP. Plugging it into our approach, we obtain a randomized reduction from GI to MKTP without false negatives. A reduction with zero-sided error follows as discussed earlier.

Before applying our approach to other isomorphism problems, let us point out the important role that the Orbit–Stabilizer Theorem plays. A randomized algorithm for finding generators for a graph's automorphism group yields a probably-correct underestimator for the size of the automorphism group, as well as a randomized algorithm for GI without false positives. The Orbit–Stabilizer Theorem allows us to turn a probably-correct underestimator for $|\operatorname{Aut}(G)|$ into a probably-correct overestimator for the size of the support of R_G, thereby switching the error from one side to the other, and allowing us to avoid false negatives instead of false positives.

[2] Briefly, suppose the vertices of G are $1, 2, \ldots, n$. By appropriately coloring vertices of two separate copies of G, one may query, for each $u = 1, 2, \ldots, n$ and each $v = u + 1, u + 2, \ldots, n$, whether there is an automorphism of G that fixes $1, 2, \ldots, u - 1$ and sends u to v. Moreover, whenever such an automorphism exists, it may be constructed efficiently through further refinement of the colors and use of oracle queries to GI, as in the standard search-to-decision reduction for GI. Such a collection of automorphisms generates $\operatorname{Aut}(G)$.

General Framework. Our approach extends to several other isomorphism problems. They can be cast in the following common framework, parameterized by an underlying family of group actions (Ω, H) where H is a group that acts on the universe Ω. We typically think of the elements of Ω as abstract objects, which need to be described in string format in order to be input to a computer; we let $\omega(z)$ denote the abstract object represented by the string z.

DEFINITION 1.1 (Isomorphism Problem). *An instance of an Isomorphism Problem consists of a pair $x = (x_0, x_1)$ that determines a universe Ω_x and a group H_x that acts on Ω_x such that $\omega_0(x) \doteq \omega(x_0)$ and $\omega_1(x) \doteq \omega(x_1)$ belong to Ω_x. Each $h \in H_x$ is identified with the permutation $h : \Omega_x \to \Omega_x$ induced by the action. The goal is to determine whether there exists $h \in H_x$ such that $h(\omega_0(x)) = \omega_1(x)$.*

When it causes no confusion, we drop the argument x and simply write H, Ω, ω_0, and ω_1. We often blur the—sometimes pedantic—distinction between z and $\omega(z)$. For example, in GI, each z is an $n \times n$ binary matrix (a string of length n^2), and represents the abstract object $\omega(z)$ of a graph with n labeled vertices; thus, in this case the correspondence between z and $\omega(z)$ is a bijection. The group H is the symmetric group S_n, and the action is by permuting the labels.

Table 1 summarizes how the problems we mentioned earlier can be cast in the framework (see Section 6 for details about the last three).

Problem	H	Ω
Graph Isomorphism	S_n	graphs with n labeled vertices
Linear Code Equivalence	S_n	subspaces of dimension d in $\mathbb{F}_q^n$
Permutation Group Conjugacy	S_n	subgroups of S_n
Matrix Subspace Conjugacy	$\mathrm{GL}_n(\mathbb{F}_q)$	subspaces of dimension d in $\mathbb{F}_q^{n\times n}$

TABLE 1
Instantiations of the Isomorphism Problem

We generalize our construction for GI to any Isomorphism Problem by replacing $R_G(\pi) \doteq \pi(G)$ where $\pi \in S_n$ is chosen uniformly at random, by $R_\omega(h) \doteq h(\omega)$ where $h \in H$ is chosen uniformly at random. The analysis that the construction yields a randomized reduction without false negatives from the Isomorphism Problem to MKTP carries over, provided that the Isomorphism Problem satisfies the following properties.

1. The underlying group H is *efficiently samplable*, and the action $(\omega, h) \mapsto h(\omega)$ is efficiently computable. We need this property in order to make sure the reduction is efficient.
2. There is an efficiently computable *normal form* for representing elements of Ω as strings. This property trivially holds in the setting of GI as there is a unique adjacency matrix that represents any given graph on the vertex set $[n]$. However, uniqueness of representation need not hold in general. Consider, for example, Permutation Group Conjugacy. An instance of this problem abstractly consists of two permutation groups (Γ_0, Γ_1), represented (as usual) by a sequence of elements of S_n generating each group. In that case there are many strings representing the same abstract object, i.e., a subgroup has many different sets of generators.
   For the correctness analysis in the isomorphic case it is important that H acts on the abstract objects, and *not* on the binary strings that represent them. In particular, the output of the reduction should only depend on the abstract

object $h(\omega)$, and not on the way ω was provided as input. This is because the latter may leak information about the value of the bit r that was picked. The desired independence can be guaranteed by applying a normal form to the representation before outputting the result. In the case of Permutation Group Conjugacy, this means transforming a set of permutations that generate a subgroup Γ into a canonical set of generators for Γ.
In fact, it suffices to have an efficiently computable *complete invariant* for Ω, i.e., a mapping from representations of objects from Ω to strings such that the image only depends on the abstract object, and is different for different abstract objects.

3. There exists a probably-correct overestimator for $N \doteq |H|/|\operatorname{Aut}(\omega)|$ that is computable efficiently with access to an oracle for MKTP. We need this property to set the threshold $\theta \doteq t(s + \frac{1}{2})$ with $s \doteq \log(N)$ correctly.
4. There exists an encoding for cosets of $\operatorname{Aut}(\omega)$ in H that achieves KT-complexity close to the information-theoretic optimum (see Section 2.2 for the definition of an encoding). This property ensures that in the isomorphic case the KT-complexity is never much larger than the entropy.

Properties 1 and 2 are fairly basic. Property 4 may seem to require an instantiation-dependent approach. However, in Section 4 we develop a *generic* hashing-based encoding scheme that meets the requirements. In fact, we give a nearly-optimal encoding scheme for any samplable distribution that is almost flat, without reference to isomorphism. Unlike the indexings from Lemma 3.4, the generic construction does not achieve the information-theoretic optimum, but it comes sufficiently close for our purposes.

The notion of a probably-correct overestimator in Property 3 can be further relaxed to that of a *probably-approximately-correct overestimator*, or *pac overestimator* for short. This is a randomized algorithm that with high probability outputs a value within an absolute deviation bound of Δ from the correct value, and never produces a value that is more than Δ below the correct value. More precisely, it suffices to efficiently compute with access to an oracle for MKTP a pac overestimator for $s \doteq \log(|H|/|\operatorname{Aut}(\omega)|)$ with deviation $\Delta = 1/4$. The relaxation suffices because of the difference of about $1/2$ between the threshold θ and the actual KT-values in both the isomorphic and the non-isomorphic case. As $s = \log|H| - \log|\operatorname{Aut}(\omega)|$, it suffices to have a pac overestimator for $\log|H|$ and a pac *under*estimator for $\log|\operatorname{Aut}(\omega)|$, both to within deviation $\Delta/2 = 1/8$ and of the required efficiency.

Generalizing our approach for GI, one way to obtain the desired underestimator for $\log|\operatorname{Aut}(\omega)|$ is by showing how to efficiently compute with access to an oracle for MKTP:

(a) a list L of elements of H that generates a subgroup $\langle L\rangle$ of $\operatorname{Aut}(\omega)$ such that $\langle L\rangle = \operatorname{Aut}(\omega)$ with high probability, and
(b) a pac underestimator for $\log|\langle L\rangle|$, the logarithm of the order of the subgroup generated by a given list L of elements of H.

Further mimicking our approach for GI, we know how to achieve (a) when the Isomorphism Problem allows a search-to-decision reduction. Such a reduction is known for Linear Code Equivalence, but remains open for problems like Matrix Subspace Conjugacy and Permutation Group Conjugacy. However, we show that (a) holds for a *generic* isomorphism problem provided that products and inverses in H can be computed efficiently (see Lemma 5.4 in Section 5.2). The proof hinges on the ability of MKTP to break the pseudo-random generator construction of [24] based on a purported one-way function (Lemma 2.2 in Section 2.1).

As for (b), we know how to efficiently compute the order of the subgroup *exactly* in the case of permutation groups ($H = S_n$), even without an oracle for MKTP, and in the case of many matrix groups over finite fields ($H = \mathrm{GL}_n(\mathbb{F}_q)$) with oracle access to MKTP, but some cases remain open (see footnote 6 in Section 5.2 for more details). Instead, we show how to *generically* construct a *pac underestimator* with small deviation given access to MKTP as long as products and inverses in H can be computed efficiently, and H allows an efficient complete invariant (see Lemma 5.5 in Section 5.2). The first two conditions are sufficient to efficiently generate a distribution $\widetilde{p}$ on $\langle L \rangle$ that is uniform to within a small relative deviation [7]. The entropy $\widetilde{s}$ of that distribution equals $\log |\langle L \rangle|$ to within a small additive deviation. As $\widetilde{p}$ is almost flat, our encoding scheme from Section 4 shows that $\widetilde{p}$ has an encoding whose length does not exceed $\widetilde{s}$ by much, and that can be decoded by small circuits. Given an efficient complete invariant for H, we can use an approach similar to the one we used to approximate the threshold θ to construct a pac underestimator for $\widetilde{s}$ with small additive deviation, namely the amortized KT-complexity of the concatenation of a polynomial number of samples from $\widetilde{p}$. With access to an oracle for MKTP we can efficiently evaluate KT. As a result, we obtain a pac underestimator for $\log |\langle L \rangle|$ with a small additive deviation that is efficiently computable with oracle access to MKTP.

The above ingredients allow us to conclude that all of the isomorphism problems in Table 1 reduce to MKTP under randomized reductions without false negatives. Moreover, we argue that Properties 1 and 2 are sufficient to generalize the construction of Allender and Das [2], which yields randomized reductions of the isomorphism problem to MKTP without false positives (irrespective of whether a search-to-decision reduction is known). By combining both reductions, we conclude that all of the isomorphism problems in Table 1 reduce to MKTP under randomized reductions with zero-sided error. See Sections 5 and 6 for more details.

Open Problems. The difference in compressibility between the isomorphic and non-isomorphic case is relatively small. As such, our approach is fairly delicate. Although we believe it yields zero-sided error reductions to MCSP as well, we currently do not know whether that is the case. An open problem in the other direction is to develop zero-error reductions from all of SZK to MKTP. We refer to Section 7 for further discussion and other future research directions.

2. Preliminaries. We assume familiarity with standard complexity theory, including the bounded-error randomized polynomial-time complexity classes BPP (two-sided error), RP (one-sided error, i.e., no false positives), and ZPP (zero-sided error, i.e., no false positives, no false negatives, and bounded probability of no output). In the remainder of this section we provide more details about KT-complexity, formally define the related notions of indexing and encoding, and review some background on graph isomorphism.

2.1. KT Complexity. The measure KT that we informally described in Section 1, was introduced and formally defined as follows in [1]. We refer to that paper for more background and motivation for the particular definition.

Definition 2.1 (KT). *Let U be a universal Turing machine. For each string x, define $\mathrm{KT}_U(x)$ to be*

$$\min\{\, |d| + T : \ (\forall \sigma \in \{0,1,*\})\ (\forall i \le |x|+1)\ U^d(i,\sigma) \text{ accepts in } T \text{ steps iff } x_i = \sigma \,\}.$$

We define $x_i = *$ if $i > |x|$; thus, for $i = |x|+1$ the machine accepts iff $\sigma = *$. The notation U^d indicates that the machine U has random access to the description d.

$\mathrm{KT}(x)$ is defined to be equal to $\mathrm{KT}_U(x)$ for a fixed choice of universal machine U with logarithmic simulation time overhead [1, Proposition 5]. In particular, if d consists of the description of a Turing machine M that runs in time $T_M(n)$ and some auxiliary information a such that $M^a(i) = x_i$ for $i \in [n]$, then $\mathrm{KT}(x) \leq |a| + c_M T_M(\log n) \log(T_M(\log n))$, where $n \doteq |x|$ and c_M is a constant depending on M. It follows that $(\mu/\log n)^{\Omega(1)} \leq \mathrm{KT}(x) \leq (\mu \cdot \log n)^{O(1)}$ where μ represents the circuit complexity of the mapping $i \mapsto x_i$ [1, Theorem 11].

The Minimum KT Problem is defined as $\mathrm{MKTP} \doteq \{(x, \theta) \mid \mathrm{KT}(x) \leq \theta\}$. [1] showed that an oracle for MKTP is sufficient to invert on average any function that can be computed efficiently. We use the following formulation:

LEMMA 2.2 (follows from Theorem 45 in [1]). *There exists a polynomial-time probabilistic Turing machine M using oracle access to* MKTP *so that the following holds. For any circuit C on n input bits,*

$$\Pr\left[C(\tau) = C(\sigma)\right] \geq 1/\operatorname{poly}(n) \text{ where } \tau \doteq M(C, C(\sigma)),$$

and the probability is over the uniform distribution of $\sigma \in \{0,1\}^n$ and the internal coin flips of M.

2.2. Random Variables, Samplers, Indexings and Encodings. A finite probability space consists of a finite sample space S, and a probability distribution p on S. Typical sample spaces include finite groups and finite sets of strings. The probability distributions underlying our probability spaces are always uniform.

A *random variable* R is a mapping from the sample space S to a set T, which typically is the universe Ω of a group action or a set of strings. The random variable R with the uniform distribution on S induces a distribution p on T. We sometimes use R to denote the induced distribution p as well.

The support of a distribution p on a set T is the set of elements $\tau \in T$ with positive probability $p(\tau)$. A distribution is *flat* if it is uniform on its support. The *entropy* of a distribution p is the expected value of $\log(1/p(\tau))$. The *min-entropy* of p is the largest real s such that $p(\tau) \leq 2^{-s}$ for every τ in the support of p. The *max-entropy* of p is the least real s such that $p(\tau) \geq 2^{-s}$ for every τ in the support of p. For a flat distribution, the min-, max-, and ordinary entropy coincide and equal the logarithm of the size of the support. For two distributions p and q on the same set T, we say that q approximates p within a factor $1+\delta$ if $q(\tau)/(1+\delta) \leq p(\tau) \leq (1+\delta) \cdot q(\tau)$ for all $\tau \in T$. In that case, p and q have the same support, and if p has min-entropy s, then q has min-entropy at least $s - \log(1+\delta)$, and if p has max-entropy s, then q has max-entropy at most $s + \log(1+\delta)$.

A *sampler* within a factor $1+\delta$ for a distribution p on a set T is a random variable $R : \{0,1\}^\ell \to T$ that induces a distribution that approximates p within a factor $1+\delta$. We say that R *samples T within a factor $1+\delta$ from length ℓ*. If $\delta = 0$ we call the sampler *exact*. The choice of $\{0,1\}^\ell$ reflects the fact that distributions need to be generated from a source of random bits. Factors $1+\delta$ with $\delta > 0$ are necessary in order to sample uniform distributions whose support is not a power of 2.

We consider ensembles of distributions $\{p_x\}$ where x ranges over $\{0,1\}^*$. We call the ensemble *samplable by polynomial-size circuits* if there exists an ensemble of random variables $\{R_{x,\delta}\}$ where δ ranges over the positive rationals such that $R_{x,\delta}$ samples p_x within a factor $1+\delta$ from length $\ell_{x,\delta}$ and $R_{x,\delta}$ can be computed by a circuit of size $\operatorname{poly}(|x|/\delta)$. We stress that the circuits can depend on the string x, not just on $|x|$. If in addition the mappings $(x, \delta) \mapsto \ell_{x,\delta}$ and $(x, \delta, \sigma) \mapsto R_{x,\delta}(\sigma)$ can be

computed in time $\mathrm{poly}(|x|/\delta)$, we call the ensemble *uniformly samplable in polynomial time*.

One way to obtain strings with high KT-complexity is as samples from distributions with high min-entropy.

PROPOSITION 2.3. *Let y be sampled from a distribution with min-entropy s. For all k, we have $\mathrm{KT}(y) \geq \lfloor s-k \rfloor$ except with probability at most 2^{-k}.*

Proof. There are only $\sum_{i=0}^{\lfloor s-k \rfloor -1} 2^i < 2^{s-k}$ descriptions of strings with complexity less than $\lfloor s-k \rfloor$. In a distribution with min-entropy s, every sample occurs with probability at most 2^{-s}. Thus the total probability mass on samples with complexity less than $\lfloor s-k \rfloor$ is at most $2^{s-k} \cdot 2^{-s} = 2^{-k}$. ∎

One way to establish upper bounds on KT-complexity is via efficiently decodable encodings into integers from a small range. Encodings with the minimum possible range are referred to as indexings. We use these notions in various settings. The following formal definition is for use with random variables and is general enough to capture all the settings we need. It defines an encoding via its decoder D; the range of the encoding corresponds to the domain of D.

DEFINITION 2.4 (encoding and indexing). *Let $R : S \to T$ be a random variable. An* encoding *of R is a mapping $D : [N] \to S$ such that for every $\tau \in R(S)$ there exists $i \in [N]$ such that $R(D(i)) = \tau$. We refer to $\lceil \log(N) \rceil$ as the* length *of the encoding. An* indexing *is an encoding with $N = |R(S)|$.*

Definition 2.4 applies to a set S by identifying S with the random variable that is the identity mapping on S. It applies to the cosets of a subgroup Γ of a group H by considering the random variable that maps $h \in H$ to the coset $h\Gamma$. It applies to a distribution induced by a random variable R by considering the random variable R itself.

We say that an ensemble of encodings $\{D_x\}$ is *decodable by polynomial-size circuits* if for each x there is a circuit of size $\mathrm{poly}(|x|)$ that computes $D_x(i)$ for every $i \in [N_x]$. If in addition the mapping $(x, i) \mapsto D_x(i)$ is computable in time $\mathrm{poly}(|x|)$, we call the ensemble *uniformly decodable in polynomial time*.

2.3. Graph Isomorphism and the Orbit-Stabilizer Theorem. Graph Isomorphism (GI) is the computational problem of deciding whether two graphs, given as input, are isomorphic. A *graph* for us is a simple, undirected graph, that is, a vertex set $V(G)$, and a set $E(G)$ of unordered pairs of vertices. An *isomorphism* between two graphs G_0, G_1 is a bijection $\pi\colon V(G_0) \to V(G_1)$ that preserves both edges and non-edges: $(v, w) \in E(G_0)$ if and only if $(\pi(v), \pi(w)) \in E(G_1)$. An isomorphism from a graph to itself is an *automorphism*; the automorphisms of a given graph G form a group under composition, denoted $\mathrm{Aut}(G)$. The Orbit–Stabilizer Theorem implies that the number of distinct graphs isomorphic to G equals $n!/|\mathrm{Aut}(G)|$. A graph G is *rigid* if $|\mathrm{Aut}(G)| = 1$, i.e., the only automorphism is the identity, or equivalently, all $n!$ permutations of G yield distinct graphs.

More generally, let H be a group acting on a universe Ω. For $\omega \in \Omega$, each $h \in H$ is an isomorphism from ω to $h(\omega)$. $\mathrm{Aut}(\omega)$ is the set of isomorphisms from ω to itself. By the Orbit–Stabilizer Theorem the number of distinct isomorphic copies of ω equals $|H|/|\mathrm{Aut}(\omega)|$.

3. Graph Isomorphism. In this section we show:

THEOREM 3.1. $\mathrm{GI} \in \mathsf{ZPP}^{\mathrm{MKTP}}$.

The crux is the randomized mapping reduction from deciding whether a given pair of n-vertex graphs (G_0, G_1) is in GI to deciding whether $(y, \theta) \in$ MKTP, as prescribed by (1.1). Recall that (1.1) involves picking a string $r \doteq r_1 \dots r_t \in \{0,1\}^t$ and permutations π_i at random, and constructing the string $y = y_1 \dots y_t$, where $y_i = \pi_i(G_{r_i})$. We show how to determine θ such that a sufficiently large polynomial t guarantees that the reduction has no false negatives. We follow the outline of Section 1, take the same four steps, and fill in the missing details.

3.1. Rigid Graphs. We first consider the simplest setting, in which both G_0 and G_1 are rigid. We argue that $\theta \doteq t(s + \frac{1}{2})$ works, where $s = \log(n!)$.

Nonisomorphic Case. If $G_0 \not\equiv G_1$, then (by rigidity), each choice of r and each distinct sequence of t permutations results in a different string y, and thus the distribution on the strings y has entropy $t(s+1)$ where $s \doteq \log(n!)$. Thus, by Proposition 2.3, $\mathrm{KT}(y) > \theta = t(s+1) - \frac{t}{2}$ with all but exponentially small probability in t. Thus with high probability the algorithm declares G_0 and G_1 nonisomorphic.

Isomorphic Case. If $G_0 \equiv G_1$, we need to show that $\mathrm{KT}(y) \leq \theta$ always holds. The key insight is that the information in y is precisely captured by the t permutations $\tau_1, \tau_2, \dots, \tau_t$ such that $\tau_i(G_0) = y_i$. These permutations exist because $G_0 \equiv G_1$; they are unique by the rigidity assumption. Thus, y contains at most ts bits of information. We show that its KT-complexity is not much larger than this. We rely on the following encoding, due to Lehmer (see, e.g., [31, pp. 12–33]):

PROPOSITION 3.2 (Lehmer code). *The symmetric groups S_n have indexings that are uniformly decodable in time* $\mathrm{poly}(n)$.

To bound $\mathrm{KT}(y)$, we consider a program d that has the following information hard-wired into it: n, the adjacency matrix of G_0, and the t integers $k_1, \dots, k_t \in [n!]$ encoding $\tau_1, \dots, \tau_t$. We use the decoder from Proposition 3.2 to compute the i-th bit of y on input i. This can be done in time $\mathrm{poly}(n, \log(t))$ given the hard-wired information.

As mentioned in Section 1, a naïve method for encoding the indices $k_1, \dots, k_t$ only gives the bound $t\lceil s \rceil + \mathrm{poly}(n, \log(t))$ on $\mathrm{KT}(y)$, which may exceed $t(s+1)$ and—*a fortiori*—the threshold θ, no matter how large a polynomial t is. We remedy this by aggregating multiple indices into blocks, and amortizing the encoding overhead across multiple samples. The following technical lemma captures the technique. For a set T of strings and $b \in \mathbb{N}$, the statement uses the notation T^b to denote the set of concatenations of b strings from T; we refer to Section 2.2 for the other terminology.

LEMMA 3.3 (Blocking Lemma). *Let $\{T_x\}$ be an ensemble of sets of strings such that all strings in T_x have the same length* $\mathrm{poly}(|x|)$. *Suppose that for each $x \in \{0,1\}^*$ and $b \in \mathbb{N}$, there is a random variable $R_{x,b}$ whose image contains $(T_x)^b$, and such that $R_{x,b}$ is computable by a circuit of size* $\mathrm{poly}(|x|, b)$ *and has an encoding of length $s'(x, b)$ decodable by a circuit of size* $\mathrm{poly}(|x|, b)$. *Then there are constants c_1 and c_2 so that, for every constant $\alpha > 0$, every $t \in \mathbb{N}$, every sufficiently large x, and every $y \in (T_x)^t$*

$$\mathrm{KT}(y) \leq t^{1-\alpha} \cdot s'(x, \lceil t^\alpha \rceil) \; + \; t^{\alpha \cdot c_1} \cdot |x|^{c_2}.$$

We first show how to apply the Blocking Lemma and then prove it. For a given rigid graph G, we let T_G be the image of the random variable R_G that maps $\pi \in S_n$ to $\pi(G)$ (an adjacency matrix viewed as a string of n^2 bits). We let $R_{G,b}$ be the b-fold Cartesian product of R_G, i.e., $R_{G,b}$ takes in b permutations $\tau_1, \dots, \tau_b \in S_n$, and maps them to $\tau_1(G)\tau_2(G) \cdots \tau_b(G)$. $R_{G,b}$ is computable by (uniform) circuits of size

poly(n, b). To encode an outcome $\tau_1(G)\tau_2(G)\cdots\tau_b(G)$, we use as index the number whose base-$(n!)$ representation is written $k_1k_2\cdots k_b$, where k_i is the index of τ_i from the Lehmer code. This indexing has length $s'(G,b) \doteq \lceil \log(n!^b) \rceil \leq bs + 1$. Given an index, the list of permutations $\tau_1, \ldots, \tau_b$ can be decoded by (uniform) circuits of size poly(n, b). By the Blocking Lemma, we have that

$$\mathrm{KT}(y) \leq t^{1-\alpha}(\lceil t^\alpha \rceil s + 1) + t^{\alpha c_1} \cdot n^{c_2} \leq ts + t^{1-\alpha} \cdot n^{c_0} + t^{\alpha c_1} \cdot n^{c_2} \tag{3.1}$$

for some constants c_0, c_1, c_2, every constant $\alpha > 0$, and all sufficiently large n, where we use the fact that $s = \log n! \leq n^{c_0}$. Setting $\alpha = \alpha_0 \doteq 1/(c_1 + 1)$, this becomes $\mathrm{KT}(y) \leq ts + t^{1-\alpha_0} n^{(c_0+c_2)}$. Taking $t = n^{1+(c_0+c_2)/\alpha_0}$, we see that for all sufficiently large n, $\mathrm{KT}(y) \leq t(s + \frac{1}{2}) \doteq \theta$.

Proof of Lemma 3.3. Let $R_{x,b}$ be the hypothesized random variables and $D_{x,b}$ their corresponding decoders. Fix x and t, let $m = \mathrm{poly}(|x|)$ denote the length of the strings in T_x, and let $b \in \mathbb{N}$ be a parameter to be set later.

To bound $\mathrm{KT}(y)$, we first break y into $\lceil t/b \rceil$ blocks $\widetilde{y}_1, \widetilde{y}_2, \ldots, \widetilde{y}_{\lceil t/b \rceil}$ where each $\widetilde{y}_i \in (T_x)^b$ (after padding $\widetilde{y}_{\lceil t/b \rceil}$ with arbitrary strings from T_x if needed). As the image of $R_{x,b}$ contains $(T_x)^b$, each $\widetilde{y}_j$ is encoded by some index k_j of length $s'(x, b)$.

Consider a program d that has x, t, m, b, the circuit for computing $R_{x,b}$, the circuit for computing $D_{x,b}$, and the indices $k_1, k_2, \ldots, k_{\lceil t/b \rceil}$ hardwired, takes an input $i \in \mathbb{N}$, and determines the i-th bit of y as follows. If $i > tm$, then the output is $*$. Otherwise, d first computes $j_0, j_1 \in \mathbb{N}$ so that i points to the j_1-th bit position in $\widetilde{y}_{j_0}$. Then, using $D_{x,b}$, k_{j_0}, and j_1, it finds σ such that $R_{x,b}(\sigma)$ equals $\widetilde{y}_{j_0}$. Finally, it computes $R_{x,b}(\sigma)$ and outputs the j_1-th bit, which is the i-th bit of y.

The bit-length of d is at most $\lceil t/b \rceil \cdot s'(x, b)$ for the indices, plus poly$(|x|, b, \log t)$ for the rest. The time needed by d is bounded by poly$(|x|, b, \log t)$. Thus

$$\begin{aligned} \mathrm{KT}(y) &\leq \lceil t/b \rceil \cdot s'(x, b) + \mathrm{poly}(|x|, b, \log t) \\ &\leq t/b \cdot s'(x, b) + \mathrm{poly}(|x|, b, \log t) \end{aligned}$$

where we used the fact that $s'(x, b) \leq \mathrm{poly}(|x|, b)$. The lemma follows by choosing $b = \lceil t^\alpha \rceil$. ∎

3.2. Known Number of Automorphisms. We generalize the case of rigid graphs to graphs for which we know the size of their automorphism groups. Specifically, in addition to the two input graphs G_0 and G_1, we are also given numbers N_0, N_1 where $N_i \doteq n!/|\mathrm{Aut}(G_i)|$. Note that if $N_0 \neq N_1$, we can right away conclude that $G_0 \not\equiv G_1$. Nevertheless, we do not assume that $N_0 = N_1$ as the analysis of the case $N_0 \neq N_1$ will be useful in Section 3.3.

The reduction is the same as in Section 3.1 with the correct interpretation of s. The main difference lies in the analysis, where we need to accommodate for the loss in entropy that comes from having multiple automorphisms.

Let $s_i \doteq \log(N_i)$ be the entropy in a random permutation of G_i. Set $s \doteq \min(s_0, s_1)$, and $\theta \doteq t(s + \frac{1}{2})$. In the nonisomorphic case the min-entropy of y is at least $t(s+1)$, so $\mathrm{KT}(y) > \theta$ with high probability. In the isomorphic case we upper bound $\mathrm{KT}(y)$ by about ts. Unlike the rigid case, we can no longer afford to encode an entire permutation for each permuted copy of G_0; we need a replacement for the Lehmer code. The following encoding, applied to $\Gamma = \mathrm{Aut}(G)$, suffices to complete the argument from Section 3.1.

LEMMA 3.4. *For every subgroup Γ of S_n there exists an indexing of the cosets*[3] *of Γ that is uniformly decodable in polynomial time when Γ is given by a list of generators.*

We prove Lemma 3.4 in the Appendix as a corollary to a more general lemma that gives, for each $\Gamma \leq H \leq S_n$, an efficiently computable indexing for the cosets of Γ in H.

Remark 3.5. Before we continue towards Theorem 3.1, we point out that the above ideas yield an alternate proof that GI $\in \mathsf{BPP}^{\mathrm{MKTP}}$ (and hence that GI $\in \mathsf{RP}^{\mathrm{MKTP}}$). This weaker result was already obtained in [2] along the well-trodden path discussed in Section 1; this remark shows how to obtain it using our new approach.

The key observation is that in both the isomorphic and the nonisomorphic case, with high probability $\mathrm{KT}(y)$ stays away from the threshold θ by a growing margin. Moreover, the above analysis allows us to efficiently obtain high-confidence approximations of θ to within any constant using sampling and queries to the MKTP oracle.

More specifically, for $i \in \{0,1\}$, let $\widetilde{y}_i$ denote the concatenation of $\widetilde{t}$ independent samples from R_{G_i}. Our analysis shows that $\mathrm{KT}(\widetilde{y}_i) \leq \widetilde{t}s_i + \widetilde{t}^{1-\alpha_0}n^c$ always holds, and that $\mathrm{KT}(\widetilde{y}_i) \geq \widetilde{t}s_i - \widetilde{t}^{1-\alpha_0}n^c$ holds with high probability. Thus, $\widetilde{s}_i \doteq \mathrm{KT}(\widetilde{y}_i)/\widetilde{t}$ approximates s_i with high confidence to within an additive deviation of $n^c/\widetilde{t}^{\alpha_0}$. Similarly, $\widetilde{s} \doteq \min(\widetilde{s}_0, \widetilde{s}_1)$ approximates s to within the same deviation margin, and $\widetilde{\theta} \doteq t(\widetilde{s} + \frac{1}{2})$ approximates θ to within an additive deviation of $tn^c/\widetilde{t}^{\alpha_0}$. The latter bound can be made less than 1 by setting $\widetilde{t}$ to a sufficiently large polynomial in n and t. Moreover, all these estimates can be computed in time $\mathrm{poly}(\widetilde{t}, n)$ with access to MKTP as MKTP enables us to evaluate KT efficiently.

3.3. Probably-Correct Underestimators for the Number of Automorphisms. The reason the $\mathsf{BPP}^{\mathrm{MKTP}}$-algorithm in Remark 3.5 can have false negatives is that the approximation $\widetilde{\theta}$ to θ may be too small. Knowing the quantities $N_i \doteq n!/|\operatorname{Aut}(G_i)|$ exactly allows us to compute θ exactly and thereby obviates the possibility of false negatives. In fact, it suffices to compute overestimates for the quantities N_i which are correct with non-negligible probability. We capture this notion formally as follows:

DEFINITION 3.6 (probably-correct overestimator). *Let $g : \Omega \to \mathbb{R}$ be a function, and M a randomized algorithm that, on input $\omega \in \Omega$, outputs a value $M(\omega) \in \mathbb{R}$. We say that M is a* probably-correct overestimator *for g if, for every $\omega \in \Omega$, $M(\omega) = g(\omega)$ holds with probability at least $1/\operatorname{poly}(|\omega|)$, and $M(\omega) > g(\omega)$ otherwise. A* probably-correct underestimator *for g is defined similarly by reversing the inequality.*

We point out that, for any probably-correct overestimator (underestimator), taking the minimum (maximum) among $\operatorname{poly}(|\omega|)$ independent runs yields the correct value with probability $1 - 2^{-\operatorname{poly}(|\omega|)}$.

We are interested in the case where $g(G) = n!/|\operatorname{Aut}(G)|$. Assuming this g on a given class of graphs Ω has a probably-correct overestimator M computable in randomized polynomial time with an MKTP oracle, we argue that GI on Ω reduces to MKTP in randomized polynomial time without false negatives.

To see this, consider the algorithm that, on input a pair (G_0, G_1) of n-vertex graphs, computes $\widetilde{N}_i = M(G_i)$ as estimates of the true values $N_i = n!/|\operatorname{Aut}(G_i)|$,

[3] The choice of left ($\pi\Gamma$) vs right ($\Gamma\pi$) cosets is irrelevant for us; all our results hold for both, and one can usually switch from one statement to the other by taking inverses. Related to this, there is an ambiguity regarding the order of application in the composition gh of two permutations: first apply g and then h, or vice versa. Both interpretations are fine. For concreteness, we assume the former.

and then runs the algorithm from Section 3.2 using the estimates $\widetilde{N}_i$.

- In the case where G_0 and G_1 are not isomorphic, if both estimates $\widetilde{N}_i$ are correct, then the algorithm detects $G_0 \not\equiv G_1$ with high probability.
- In the case where $G_0 \equiv G_1$, if $\widetilde{N}_i = N_i$ we showed in Section 3.2 that the algorithm always declares G_0 and G_1 to be isomorphic. Moreover, increasing θ can only decrease the probability of a false negative. As the computed threshold θ increases as a function of $\widetilde{N}_i$, and the estimate $\widetilde{N}_i$ is always at least as large as N_i, it follows that G_0 and G_1 are always declared isomorphic.

3.4. Arbitrary Graphs. A probably-correct overestimator for the function $G \mapsto n!/|\operatorname{Aut}(G)|$ on *any* graph G can be computed in randomized polynomial time with access to MKTP. The process is described in full detail in Section 1, based on a $\mathsf{BPP}^{\mathrm{MKTP}}$ algorithm for GI (taken from Remark 3.5 or from [2]). This means that the setting of Section 3.3 is actually the general one. The only difference is that we no longer obtain a mapping reduction from GI to MKTP, but an oracle reduction: We still make use of (1.1), but we need more queries to MKTP in order to set the threshold θ.

This shows that $\mathrm{GI} \in \mathsf{coRP}^{\mathrm{MKTP}}$. As $\mathrm{GI} \in \mathsf{RP}^{\mathrm{MKTP}}$ follows from the known search-to-decision reduction for GI, this concludes the proof of Theorem 3.1 that $\mathrm{GI} \in \mathsf{ZPP}^{\mathrm{MKTP}}$.

4. Estimating the Entropy of Flat Samplable Distributions. In this section we develop a key ingredient in extending Theorem 3.1 from GI to other isomorphism problems that fall within the framework presented in Section 1, namely efficient near-optimal encodings of cosets of automorphism groups. More generally, our encoding scheme works well for any samplable distribution that is flat or almost flat. It allows us to probably-approximately-correctly underestimate the entropy of such distributions with the help of an oracle for MKTP.

We first develop our encoding, which only requires the existence of a sampler from strings of polynomial length. The length of the encoding is roughly the max-entropy of the distribution, which is the information-theoretic optimum for flat distributions.

The lemma is stated in terms of a random variable that samples the distribution. Recall from Section 2 that a random variable R samples a distribution p from length ℓ when R has domain $\{0,1\}^\ell$, and p is identical to the distribution of $R(\sigma)$ with σ drawn uniformly at random from its domain.

LEMMA 4.1 (Encoding Lemma). *Consider an ensemble $\{R_x\}$ of random variables that sample distributions with max-entropy $s(x)$ from length $\operatorname{poly}(|x|)$. Each R_x has an encoding of length $s(x) + \log s(x) + O(1)$ that is decodable by polynomial-size circuits.*

To see how Lemma 4.1 performs, let us apply to the setting of GI. Consider the random variable R_G mapping a permutation $\pi \in S_n$ to $\pi(G)$. The induced distribution is flat and has entropy $s = \log(n!/|\operatorname{Aut}(G)|)$, and each $\pi \in S_n$ can be sampled from strings of length $O(n \log n)$. The Encoding Lemma thus yields an encoding of length $s + \log s + O(1)$ that is efficiently decodable. The bound on the length is worse than Lemma 3.4's bound of $\lceil s \rceil$, but will still be sufficient for the generalization of Theorem 3.1 and yield the result for GI.

We prove the Encoding Lemma using hashing. Here is the idea. Consider a random hash function $h : \{0,1\}^\ell \to \{0,1\}^m$ where ℓ denotes the length of the strings in the domain of R_x for a given x, and m is set slightly below $\ell - s$. For any fixed outcome y of R_x, there is a positive constant probability that no more than about

$2^\ell/2^m \approx 2^s$ of all samples $\sigma \in \{0,1\}^\ell$ have $h(\sigma) = 0^m$, and at least one of these also satisfies $R_x(\sigma) = y$. Let us say that h *works for* y when both those conditions hold. In that case—ignoring efficiency considerations—about s bits of information are sufficient to recover a sample σ_y satisfying $R_x(\sigma_y) = y$ from h.

Now a standard probabilistic argument shows that there is a sequence $h_1, h_2, \ldots$ of $O(s)$ hash functions such that for every possible outcome y, there is at least one h_i that works for y. Given such a sequence, we can encode each outcome y as the index i of a hash function h_i that works for y, and enough bits of information that allow us to *efficiently* recover σ_y given h_i. We show that $s + O(1)$ bits suffice for the standard linear-algebraic family of hash functions. The resulting encoding has length $s + \log(s) + O(1)$ and is decodable by circuits of polynomial size.

Proof of Lemma 4.1. Recall that a family $\mathcal{H}_{\ell,m}$ of functions $\{0,1\}^\ell \to \{0,1\}^m$ is *universal* if for any two distinct $\sigma_0, \sigma_1 \in \{0,1\}^\ell$, the distributions of $h(\sigma_0)$ and $h(\sigma_1)$ for a uniform choice of $h \in \mathcal{H}_{\ell,m}$ are independent and uniform over $\{0,1\}^m$. We make use of the specific universal family $\mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ that consists of all functions of the form $\sigma \mapsto U\sigma + v$, where U is a binary $(m \times \ell)$-matrix, v is a binary column vector of dimension ℓ, and σ is also viewed as a binary column vector of dimension ℓ [14]. Uniformly sampling from $\mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ means picking U and v uniformly at random.

Claim 4.2. *Let $\ell, m \in \mathbb{N}$ and $s \in \mathbb{R}$.*

1. *For every universal family $\mathcal{H}_{\ell,m}$ with $m = \ell - \lceil s \rceil - 2$, and for every $S \subseteq \{0,1\}^\ell$ with $|S| \geq 2^{\ell - s}$,*

$$\Pr[(\exists \sigma \in S)\, h(\sigma) = 0^m \text{ and } |h^{-1}(0^m)| \leq 2^{\lceil s \rceil + 3}] \geq \frac{1}{4},$$

   where the probability is over a uniformly random choice of $h \in \mathcal{H}_{\ell,m}$.
2. *The sets $h^{-1}(0^m)$ have indexings that are uniformly decodable in time polynomial in ℓ and m, where h ranges over $\mathcal{H}^{(\mathrm{lin})}_{\ell,m}$.*

Assume for now that the claim holds, and let us continue with the proof of the lemma.

Fix an input x, and let $\ell = \ell(x)$ and $s = s(x)$. Consider the family $\mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ with $m = \ell - \lceil s \rceil - 2$. For each outcome y of R_x, let S_y consist of the strings $\sigma \in \{0,1\}^\ell$ for which $R_x(\sigma) = y$. Since the distribution induced by R_x has max-entropy s, a fraction at least $1/2^s$ of the strings in the domain of R_x map to y. It follows that $|S_y| \geq 2^{\ell - s}$.

A hash function $h \in \mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ *works* for y if there is some $\sigma \in S_y$ with $h(\sigma) = 0^m$ and $|h^{-1}(0^m)| \leq 2^{\lceil s \rceil + 3}$. By the first part of Claim 4.2, the probability that a random $h \in \mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ works for a fixed y is at least $1/4$. If we now pick $3\lceil s \rceil$ hash functions independently at random, the probability that none of them work for y is at most $(3/4)^{3\lceil s \rceil} < 1/2^s$. Since there are at most 2^s distinct outcomes y, a union bound shows that there exists a sequence of hash functions $h_1, h_2, \ldots, h_{3\lceil s \rceil} \in \mathcal{H}^{(\mathrm{lin})}_{\ell,m}$ such that for every outcome y of R_x there exists $i_y \in [3\lceil s \rceil]$ such that h_{i_y} works for y.

The encoding works as follows. Let $D^{(\mathrm{lin})}$ denote the uniform decoding algorithm from part 2 of Claim 4.2 such that $D^{(\mathrm{lin})}(h, \cdot)$ decodes the set $h^{-1}(0^m)$. For each outcome y of R_x, let $j_y \in [2^{\lceil s \rceil + 3}]$ be such that $D^{(\mathrm{lin})}(h_{i_y}, j_y) = \sigma_y \in S_y$. Such a j_y exists since h_{i_y} works for y. Let $k_y = 2^{\lceil s \rceil + 3} i_y + j_y$. Given $h_1, h_2, \ldots, h_{3\lceil s \rceil}$ and ℓ and m as auxiliary information, we can decode σ_y from k_y by parsing out i_y and j_y, extracting h_{i_y} from the auxiliary information, and running $D^{(\mathrm{lin})}(h_{i_y}, j_y)$. This gives an encoding for R_x of length $\lceil s \rceil + 3 + \lceil \log(3\lceil s \rceil) \rceil = s + \log s + O(1)$ that can be

decoded in time poly($|x|$) with the hash functions as auxiliary information. As each hash function can be described using $(\ell+1)m$ bits and there are $3\lceil s\rceil \leq \text{poly}(|x|)$ many of them, the auxiliary information consists of no more than poly($|x|$) bits. Hard-wiring it yields a decoder circuit of size poly($|x|$). ∎

For completeness we argue Claim 4.2.

Proof of Claim 4.2. For part 1, let $m = \ell - \lceil s\rceil - 2$, and consider the random variables $X \doteq |h^{-1}(0^m) \cap S|$ and $Y \doteq |h^{-1}(0^m)|$. Because of universality we have that $\mathbb{V}(X) \leq \mathbb{E}(X) = |S|/2^m$, and by the choice of parameters $|S|/2^m \geq 4$. By Chebyshev's inequality

$$\Pr(X = 0) \leq \Pr(|X - \mathbb{E}(X)| \geq \mathbb{E}(X)) \leq \frac{\mathbb{V}(X)}{(\mathbb{E}(X))^2} \leq \frac{1}{\mathbb{E}(X)} \leq \frac{1}{4}.$$

We have that $\mathbb{E}(Y) = 2^\ell/2^m = 2^{\lceil s\rceil+2}$. By Markov's inequality

$$\Pr(Y \geq 2^{\lceil s\rceil+3}) = \Pr(Y \geq 2\,\mathbb{E}(Y)) \leq \frac{1}{2}.$$

A union bound shows that

$$\Pr(X = 0 \text{ or } Y \geq 2^{\lceil s\rceil+3}) \leq \frac{1}{4} + \frac{1}{2},$$

from which part 1 follows.

For part 2, note that if $|h^{-1}(0^m)| > 0$ then $|h^{-1}(0^m)| = 2^{\ell-r}$ where r denotes the rank of U. In that case, given U and v, we can use Gaussian elimination to find binary column vectors $\hat{\sigma}$ and $\sigma_1, \sigma_2, \ldots, \sigma_{\ell-r}$ such that $U\hat{\sigma} + v = 0^m$ and the σ_i's form a basis for the kernel of U. On input $j \in [2^{\ell-r}]$, the decoder outputs $\hat{\sigma} + \sum_{i=1}^{\ell-r} j_i\sigma_i$, where $\sum_{i=1}^{\ell-r} j_i 2^{i-1}$ is the binary expansion of $j-1$. The image of the decoder is exactly $h^{-1}(0^m)$. As the decoding process runs in time poly(ℓ, m) when given U and u, this gives the desired indexing. ∎

The first part of Claim 4.2 is commonly used, e.g., for randomness extraction in cryptography. The combination of the two parts of Claim 4.2 seems to have found fewer applications. [37] applies them in a similar way as we do (but with a single hash function), namely to boost the success probability of randomized circuits that decide CircuitSAT as a function of the number of input variables.[4]

Remark 4.3. The proof of the Encoding Lemma shows a somewhat more general result: For any ensemble $\{R_x\}$ of random variables whose domains consist of strings of length poly($|x|$), and for *any* bound $s(x)$, the set of outcomes of R_x with probability at least $1/2^{s(x)}$ has an encoding of length $s(x) + \log s(x) + O(1)$ that is decodable by a circuit of size poly($|x|$). In the case of flat distributions of entropy $s(x)$ that set contains all possible outcomes.

[4] More precisely, suppose there exists a randomized circuit family A of size $f(n, m)$ that decides CircuitSAT without false positives on instances consisting of circuits C with n input variables and of description length m such that the probability of success is at least $1/2^{\alpha n}$. Applying our encoding to the set of random bit sequences that make A accept on a positive instance C, and hard-wiring the input C into the circuit A, yields an equivalent instance C' on αn variables of size $f(n, m) + \mu(D)$, where $\mu(D)$ denotes the circuit size of D. Applying A to the description of this new circuit C' yields a randomized circuit A' to decide whether C is satisfiable without false positives. For the linear-algebraic family of hash functions, A' has size $O(f(n, m)\,\text{polylog}(f(n, m)))$. Its success probability is at least $1/2^{\alpha^2 n}$, which is larger than $1/2^{\alpha n}$ when $\alpha < 1$.

In combination with the Blocking Lemma, the Encoding Lemma yields upper bounds on KT-complexity in the case of distributions p that are samplable by polynomial-size circuits. More precisely, if y is the concatenation of t samples from p, we can essentially upper bound the amortized KT-complexity $\mathrm{KT}(y)/t$ by the max-entropy of p. On the other hand, Proposition 2.3 shows that if the samples are picked independently at random, with high probability $\mathrm{KT}(y)/t$ is not much less than the min-entropy of p. Thus, in the case of flat distributions, $\mathrm{KT}(y)/t$ is a good *probably-approximately-correct underestimator* for the entropy, a notion formally defined as follows.

Definition 4.4 (probably-approximately-correct underestimator). *Let $g : \Omega \to \mathbb{R}$ be a function, and M a randomized algorithm that, on input $\omega \in \Omega$, outputs a value $M(\omega) \in \mathbb{R}$. We say that M is a* probably-approximately-correct underestimator *(or* pac underestimator*) for g with deviation Δ if, for every $\omega \in \Omega$, $|M(\omega) - g(\omega)| \leq \Delta$ holds with probability at least $1/\operatorname{poly}(|\omega|)$, and $M(\omega) < g(\omega)$ otherwise. A* probably-approximately-correct overestimator *(or* pac overestimator*) for g is defined similarly, by reversing the last inequality.*

Similar to the case of probably-correct under-/overestimators, we can boost the confidence level of a pac under-/overestimator from $1/\operatorname{poly}(|\omega|)$ to $1-2^{-\operatorname{poly}(|\omega|)}$ by taking the max/min of $\operatorname{poly}(|\omega|)$ independent runs.

More generally, we argue that the amortized KT-complexity of samples yields a good pac underestimator for the entropy when the distribution is *almost* flat, i.e., the difference between the max- and min-entropy is small. As KT can be evaluated efficiently with oracle access to MKTP, pac underestimating the entropy of such distributions reduces to MKTP.

Corollary 4.5 (Entropy Estimator Corollary). *Let $\{p_x\}$ be an ensemble of distributions such that p_x is supported on strings of the same length $\operatorname{poly}(|x|)$. Consider a randomized process that on input x computes $\mathrm{KT}(y)/t$, where y is the concatenation of t independent samples from p_x. If p_x is samplable by circuits of polynomial size, then for t a sufficiently large polynomial in $|x|$, $\mathrm{KT}(y)/t$ is a pac underestimator for the entropy of p_x with deviation $\Delta(x)+o(1)$, where $\Delta(x)$ is the difference between the min- and max-entropies of p_x.*

Proof. Since the entropy lies between the min- and max-entropies, it suffices to show that $\mathrm{KT}(y)/t$ is at least the min-entropy of p_x with high probability, and is always at most the max-entropy of p_x (both up to $o(1)$ terms) when t is a sufficiently large polynomial. The lower bound follows from Proposition 2.3. It remains to establish the upper bound.

Let $\{R_{x,\delta}\}$ be the ensemble of random variables witnessing the samplability of $\{p_x\}$ by circuits of polynomial size, and let $s(x)$ denote the max-entropy of p_x. The Blocking Lemma allows us to bound $\mathrm{KT}(y)$ by giving an encoding for random variables whose support contains the b-tuples of samples from p_x. Let $R'_{x,b}$ denote the b-fold Cartesian product of $R_{x,1/b}$. $R'_{x,b}$ induces a distribution that approximates to within a factor of $(1+1/b)^b = O(1)$ the distribution of the b-fold Cartesian product of p_x, which is a distribution of max-entropy $bs(x)$. It follows that the distribution induced by $R'_{x,b}$ has min-entropy at most $bs(x) + O(1)$. Its support is exactly the b-tuples of samples from p_x. Moreover, the ensemble $\{R'_{x,b}\}$ is computable by circuits of size $\operatorname{poly}(n, b)$. By the Encoding Lemma there exists an encoding of $R'_{x,b}$ of length $bs(x) + \log b + \log s(x) + O(1)$ that is decodable by circuits of polynomial-size. The Blocking Lemma then says that there exist constants c_1 and c_2 so that for all $\alpha > 0$

and all sufficiently large n

$$\begin{aligned}\mathrm{KT}(y) &\le t^{1-\alpha}\cdot(\lceil t^\alpha\rceil\cdot s(x)+\log s(x)+\alpha\log t+O(1))+t^{\alpha c_1}\cdot n^{c_2}\\ &\le ts(x)+t^{1-\alpha}\cdot(n^{c_0}+c_0\log n+\alpha\log t+O(1))+t^{\alpha c_1}\cdot n^{c_2},\end{aligned}$$

where we use the fact that there exists a constant c_0 such that $s(x)\le n^{c_0}$. A similar calculation as the one following Equation (3.1) shows that $\mathrm{KT}(y)\le ts(x)+t^{1-\alpha_0}n^{c_0+c_2}$ for $t\ge n^c$ and n sufficiently large, where $\alpha_0=1/(1+c_1)$ and $c=1+(1+c_1)(c_0+c_2)$. Dividing both sides by t yields the claimed upper bound. ∎

5. Generic Isomorphism Problem. In Section 1 we presented a common framework for isomorphism problems and listed some instantiations in Table 1. In this section we state and prove a generalization of Theorem 3.1 that applies to many problems in this framework, including the ones from Table 1.

5.1. Generalization. The generalized reduction makes use of a complete invariant for the abstract universe Ω. For future reference, we define the notion with respect to a representation for an arbitrary ensemble of sets.

Definition 5.1 (representation and complete invariant). *Let $\{\Omega_x\}$ denote an ensemble of sets. A* representation *of the ensemble is a surjective mapping $\omega:\{0,1\}^*\to\cup_x\Omega_x$. A* complete invariant *for ω is a mapping $\nu:\{0,1\}^*\to\{0,1\}^*$ such that for all strings x,z_0,z_1 with $\omega(z_0),\omega(z_1)\in\Omega_x$*

$$\omega(z_0)=\omega(z_1)\Leftrightarrow\nu(z_0)=\nu(z_1).$$

$\omega(z)$ denotes the set element represented by the string z. The surjective property of a representation guarantees that every set element has at least one string representing it.

Note that for the function ν to represent a *normal form* (rather than just a complete invariant), it would need to be the case that $\omega(\nu(z))=\omega(z)$. Although this additional property holds for all the instantiations we consider, it is not a requirement. In our setting, all that matters is that $\nu(z)$ only depends on the element $\omega(z)$ that z represents, and is different for different elements.[5]

We are now ready to state the generalization of Theorem 3.1.

Theorem 5.2. *Let* Iso *denote an Isomorphism Problem as in Definition 1.1. Consider the following conditions:*

1. [action sampler] *The uniform distribution on H_x is uniformly samplable in polynomial time, and the mapping $(\omega,h)\mapsto h(\omega)$ underlying the action of H_x on Ω_x is computable in* ZPP.
2. [complete universe invariant] *There exists a complete invariant ν for the representation ω that is computable in* ZPP.
3. [entropy estimator] *There exists a probably-approximately-correct overestimator for $(x,\omega)\mapsto\log(|H_x|/|\operatorname{Aut}(\omega)|)$ with deviation $1/4$ that is computable in randomized time* $\mathrm{poly}(|x|)$ *with access to an oracle for* MKTP.

With these definitions:

(a) If conditions 1 and 2 hold, then $\mathrm{Iso}\in\mathsf{RP}^{\mathrm{MKTP}}$.

(b) If conditions 1, 2, and 3 hold, then $\mathrm{Iso}\in\mathsf{coRP}^{\mathrm{MKTP}}$.

[5] For complexity-theoretic investigations into the difference between complete invariants and normal forms, see, e.g., [11, 12, 17, 16].

In the case of GI, Ω denotes the universe of graphs on n vertices (represented as adjacency matrices viewed as strings of length n^2), and H the group of permutations on $[n]$ (represented as function tables). All conditions in the statement of Theorem 5.2 are met. The identity mapping can be used as the complete invariant ν in condition 2, and the probably-correct overestimator for $n!/|\operatorname{Aut}(G)|$ that we argued in Sections 1 and 3 immediately yields the pac overestimator for $\log(n!/|\operatorname{Aut}(G)|)$ required in condition 3.

Note that $\log(n!/|\operatorname{Aut}(G)|)$ equals the entropy of the distribution induced by the random variable R_G. In general, the quantity $\log(|H_x|/|\operatorname{Aut}(\omega)|)$ in condition 3 represents the entropy of $\nu(h(\omega))$ when $h \in H_x$ is picked uniformly at random.

Proof of Theorem 5.2. Let x denote an instance of length $n \doteq |x|$, defining a universe Ω, a group H that acts on Ω, and two elements $\omega_i = \omega_i(x)$ for $i \in \{0,1\}$. Both parts (a) and (b) make use of the random variables R_i for $i \in \{0,1\}$ where $R_i : H \to \{0,1\}^*$ maps $h \in H$ to $\nu(h(\omega_i))$.

Part (a). We follow the approach from [2]. Their argument uses Lemma 2.2, which states the existence of a randomized polynomial-time machine M with access to an MKTP oracle that, given a random sample y from the distribution induced by a circuit C, recovers with non-negligible probability of success an input σ so that $C(\sigma) = y$. If we can model the R_i as circuits of size $\operatorname{poly}(n)$ that take in an element h from H and output $R_i(h)$, this means that, with non-negligible probability over a random $h_0 \in H$, $M(R_0, R_0(h_0))$ outputs some h_1 so that $h_1(\omega_0) = h_0(\omega_0)$. The key observation is that when $\omega_0 \equiv \omega_1$, R_0 and R_1 induce the same distribution, and therefore, for a random element h_0, $M(R_1, R_0(h_0))$ outputs some h_1 so that $h_1(\omega_0) = h_0(\omega_0)$ with non-negligible probability probability of success. Thus Iso can be decided by trying the above a polynomial number of times, declaring $\omega_0 \equiv \omega_1$ if a trial succeeds, and declaring $\omega_0 \not\equiv \omega_1$ otherwise.

We do not know how to model the R_i exactly as circuits of size $\operatorname{poly}(n)$, but we can do so approximately. Condition 1 implies that we can construct circuits $C_{i,\delta}$ in time $\operatorname{poly}(n/\delta)$ that sample $h(\omega_i)$ within a factor $1+\delta$. Combined with the ZPP-computability of ν in condition 2 this means that we can construct a circuit C_ν in time $\operatorname{poly}(n)$ such that the composed circuit $C_\nu \circ C_{i,\delta}$ samples R_i within a factor $1+\delta$ from strings σ of length $\operatorname{poly}(n/\delta)$. We use the composed circuits in lieu of R_i in the arguments for M above. More precisely, we pick an input σ_0 for $C_{0,\delta}$ uniformly at random, and compute $\sigma_1 = M(C_{1,\delta}, C_{0,\delta}(\sigma_0))$. Success means that $h_1(\omega_0) = h_0(\omega_0)$, where $h_i = C_{i,\delta}(\sigma_i)$. The probability of success for an approximation factor of $1+\delta$ is at least $1/(1+\delta)^2$ times the probability of success in the exact setting, which is $1/\operatorname{poly}(n/\delta)$ in the isomorphic case. Fixing δ to any positive constant, a single trial runs in time $\operatorname{poly}(n)$, success can be determined in ZPP (by the second part of condition 1), and the probability of success is at least $1/\operatorname{poly}(n)$ in the isomorphic case. Completing the argument as in the exact setting above, we conclude that $\mathrm{Iso} \in \mathsf{RP}^{\mathrm{MKTP}}$.

Part (b). We extend the argument from Section 3. Let $s_i \doteq \log(|H|/|\operatorname{Aut}(\omega_i)|)$ for $i \in \{0,1\}$, and let M be the pac overestimator from condition 3. We assume that M has been amplified such that it outputs a good estimate with probability exponentially close to 1. Condition 1 and the ZPP-computability of ν imply that the distribution induced by R_i is uniformly samplable in polynomial time, i.e., for each $i \in \{0,1\}$ and $\delta > 0$, there is a random variable $R_{i,\delta}$ that samples R_i within a factor $1+\delta$ from length $\operatorname{poly}(|x|/\delta)$, and that is computable in time $\operatorname{poly}(|x|/\delta)$.

Let $t \in \mathbb{N}$ and δ be parameters to be determined. On input x, the algorithm be-

gins by computing the estimates $\widetilde{s}_i = M(x, \omega_i)$ for $i \in \{0,1\}$, and sets $\widetilde{s} \doteq \min(\widetilde{s}_0, \widetilde{s}_1)$ and $\widetilde{\theta} \doteq t(\widetilde{s} + \frac{1}{2})$. The algorithm then samples $r \in \{0,1\}^t$ uniformly, and constructs $y = (R_{r_i,\delta}(\sigma_i))_{i=1}^t$, where each σ_i is drawn independently and uniformly from $\{0,1\}^{\mathrm{poly}(n,1/\delta)}$. If $\mathrm{KT}(y) > \widetilde{\theta}$, the algorithm declares $\omega_0 \not\equiv \omega_1$; otherwise, the algorithm declares $\omega_0 \equiv \omega_1$.

Nonisomorphic Case. If $\omega_0 \not\equiv \omega_1$, we need to show $\mathrm{KT}(y) > \widetilde{\theta}$ with high probability. Since $R_{i,\delta}$ samples R_i within a factor of $1+\delta$, and R_i is flat with entropy s_i, it follows that $R_{i,\delta}$ has min-entropy at least $s_i - \log(1+\delta)$, and that y is sampled from a distribution with min-entropy at least

$$t(1 + \min(s_0, s_1) - \log(1+\delta)).$$

Since M is a pac overestimator with deviation $1/4$, $|\widetilde{s}_0 - s_0| \le 1/4$ and $|\widetilde{s}_1 - s_1| \le 1/4$ with high probability. When this happens, $\widetilde{s} \le \min(s_0, s_1) + 1/4$,

$$\widetilde{\theta} \le t(\min(s_0, s_1) + 3/4),$$

and Proposition 2.3 guarantees that $\mathrm{KT}(y) > \widetilde{\theta}$ except with probability exponentially small in t as long as δ is a constant such that $1 - \log(1+\delta) > 3/4$. Such a positive constant δ exists.

Isomorphic Case. If $\omega_0 \equiv \omega_1$, we need to show that $\mathrm{KT}(y) \le \widetilde{\theta}$ always holds for t a sufficiently large polynomial in n, and n sufficiently large. Recall that, since $\omega_0 \equiv \omega_1$, R_0 and R_1 induce the same distribution, so we can view y as the concatenation of t samples from R_0. Each R_0 is flat, hence has min-entropy equal to its max-entropy, and the ensemble of all R_0 (across all inputs x) is samplable by (uniform) polynomial-size circuits. The Entropy Estimator Corollary with $\Delta(x) \equiv 0$ then implies that $\mathrm{KT}(y) \le t(s_0 + o(1))$ holds whenever t is a sufficiently large polynomial in n, and n is sufficiently large. In that case, $\mathrm{KT}(y) \le t(\widetilde{s} + \frac{1}{4} + o(1)) < \widetilde{\theta}$ holds because $s_0 \le \widetilde{s} + 1/4$ follows from M being a pac overestimator for s_0 with deviation $1/4$. ∎

Remark 5.3. The notion of efficiency in conditions 1, and 2 can be relaxed to mean the underlying algorithm is implementable by a family of polynomial-size circuits that is constructible in $\mathsf{ZPP}^{\mathrm{MKTP}}$. It is important for our argument that the circuits themselves do not have oracle access to MKTP, but it is all right for them to be constructible in $\mathsf{ZPP}^{\mathrm{MKTP}}$ rather than P or ZPP. For example, a sampling procedure that requires knowing the factorization of some number (dependent on the input x) is fine because the factorization can be computed in $\mathsf{ZPP}^{\mathrm{MKTP}}$ [1] and then can be hard-wired into the circuit.

In particular, this observation yields an alternate way to show that integer factorization being in $\mathsf{ZPP}^{\mathrm{MKTP}}$ implies that the discrete log over prime fields is in $\mathsf{ZPP}^{\mathrm{MKTP}}$ [39]. Recall that an instance of the discrete log problem consists of a triple $x = (g, z, p)$, where g and z are integers, and p is a prime, and the goal is to find an integer y such that $g^y \equiv z \bmod p$, or report that no such integer exists. The search version is known to reduce to the decision version in randomized polynomial time, and the above observation shows that the decision version is in $\mathsf{ZPP}^{\mathrm{MKTP}}$. This is because computing the size of the subgroup of $\mathbb{F}_p^\times$ generated by g or z reduces to integer factorization, and can thus be computed in $\mathsf{ZPP}^{\mathrm{MKTP}}$.

5.2. Construction of Probably-Correct Overestimators. We now discuss some generic methods to satisfy condition 3 in Theorem 5.2, i.e., how to construct a probably-approximately-correct overestimator for the quantity $\log(|H|/|\operatorname{Aut}(\omega)|)$ that is computable in $\mathsf{ZPP}^{\mathrm{MKTP}}$.

Here is the generalization of the approach we used in Section 3.4 in the context of GI:

1. Find a list L of elements of H that generates a subgroup $\langle L\rangle$ of $\mathrm{Aut}(\omega)$ such that $\langle L\rangle = \mathrm{Aut}(\omega)$ with high probability.
2. Pac underestimate $\log|\langle L\rangle|$ with deviation 1/8. This yields a pac underestimator for $\log|\mathrm{Aut}(\omega)|$.
3. Pac overestimate $\log|H|$ with deviation 1/8.
4. Return the result of step 3 minus the result of step 2. This gives a pac overestimator for $\log(|H|/|\mathrm{Aut}(\omega)|)$ with deviation 1/4.

Although in the setting of GI we used the oracle for MKTP only in step 1, we could use it to facilitate steps 2 and 3 as well.

The first step for GI follows from the known search-to-decision reduction. It relies on the fact that *Colored Graph Isomorphism* reduces to GI, where Colored Graph Isomorphism allows one to assign colors to vertices with the understanding that the isomorphism needs to preserve the colors. For all of the isomorphism problems in Table 1, finding a set of generators for the automorphism group reduces to a natural colored version of the Isomorphism Problem, but it is not clear whether the colored version always reduces to the regular version. The latter reduction is known for Linear Code Equivalence, but remains open for problems like Permutation Group Conjugacy and Matrix Subspace Conjugacy.

However, there is a different, *generic* way to achieve step 1 above, namely based on Lemma 2.2, i.e., the power of MKTP to efficiently invert on average any efficiently computable function.

LEMMA 5.4. *Let* Iso *denote an Isomorphism Problem as in Definition 1.1 that satisfies conditions 1 and 2 of Theorem 5.2, and such that products and inverses in H_x are computable in $\mathsf{BPP}^{\mathrm{MKTP}}$. There exists a randomized polynomial-time algorithm using oracle access to* MKTP *with the following behavior: On input any instance x, and any $\omega \in \Omega_x$, the algorithm outputs a list of generators for a subgroup Γ of $\mathrm{Aut}(\omega)$ such that $\Gamma = \mathrm{Aut}(\omega)$ with probability $1 - 2^{-|x|}$.*

Proof. Consider an instance x of length $n \doteq |x|$, and $\omega \in \Omega_x$. We first argue that the uniform distribution on $\mathrm{Aut}(\omega)$ is uniformly samplable in polynomial time with oracle access to MKTP.

Let R_ω denote the random variable that maps $h \in H$ to $\nu(h(\omega))$, and let M be the machine from Lemma 2.2. As in the proof of Part 1 of Theorem 5.2, we can sample h from H uniformly (to within a small constant factor) and run $M(R_\omega, \nu(h(\omega)))$ to obtain, with nonnegligible probability, some $h' \in H$ such that $h'(\omega) = h(\omega)$. In that case, $h^{-1}h'$ is an automorphism of ω, and we say the process succeeds. The key observation is the following: Since $h' = M(R_\omega, \nu(h(\omega)))$, the distribution of h' conditioned on h only depends on the coset of $\mathrm{Aut}(G)$ that h belongs to. It follows that if h were sampled perfectly uniformly then, conditioned on success, the distribution of $h^{-1}h'$ is *uniform* over $\mathrm{Aut}(\omega)$. In truth, h is sampled uniformly to within a factor $1+\delta$; in that case $h^{-1}h'$ is (conditioned on success) likewise uniform on $\mathrm{Aut}(\omega)$ to within a factor $1+\delta$ and, as argued in the proof of Theorem 5.2, the probability of success is $1/\operatorname{poly}(n/\delta)$.

We run the process many times and retain the automorphism $h^{-1}h'$ from the first successful run (if any); $\operatorname{poly}(n/\delta)$ runs suffice to obtain, with probability $1 - 2^{-2n}$, an automorphism that is within a factor $1+\delta$ from uniform over $\mathrm{Aut}(\omega)$. By the computability parts of conditions 1 and 2, and by the condition that products and inverses in H can be computed in $\mathsf{BPP}^{\mathrm{MKTP}}$, each trial runs in time $\operatorname{poly}(n/\delta)$. Success

can be determined in ZPP as the group action is computable in ZPP. It follows that the uniform distribution on $\mathrm{Aut}(\omega)$ is uniformly samplable in polynomial time with oracle access to MKTP.

Finally, we argue that a small number of independent samples $h_1, h_2, \ldots, h_k$ for some constant $\delta > 0$ suffice to ensure that they generate all of $\mathrm{Aut}(\omega)$ with very high probability. Denote by Γ_i the subgroup of H_x generated by $h_1, \ldots, h_i$. Note that Γ_i always is a subgroup of $\mathrm{Aut}(\omega)$. For $i < k$, if Γ_i is not all of $\mathrm{Aut}(\omega)$, then $|\Gamma_i| \leq |\mathrm{Aut}(\omega)|/2$. Thus, with probability at least $\frac{1}{2} \cdot \frac{1}{1+\delta}$, $h_{i+1} \notin \Gamma_i$, in which case $|\Gamma_{i+1}| \geq 2|\Gamma_i|$. For any constant $\delta > 0$, it follows that $k \geq \Theta(n + \log |\mathrm{Aut}(\omega)|) = O(\mathrm{poly}(n))$ suffices to guarantee that $\Gamma_k = \mathrm{Aut}(\omega)$ with probability at least $1 - 2^{-2n}$. The lemma follows. ∎

The second step for GI followed from the ability to efficiently compute the order of permutation groups exactly. Efficient exact algorithms (possibly with access to an oracle for MKTP) are known for larger classes of groups, including most matrix groups over finite fields, but not for all.[6] We show how to *generically* pac underestimate $\log |\langle L \rangle|$ with small deviation (step 2), namely under the prior conditions that only involve H, and the additional condition of a ZPP-computable complete invariant ζ for H.

The construction hinges on the Entropy Estimator Corollary and viewing $\log |\langle L \rangle|$ as the entropy of the uniform distribution p_L on $\langle L \rangle$.

(α) Provided that p_L is samplable by circuits of polynomial size, the corollary allows us to pac underestimate $\log |\langle L \rangle|$ as $\mathrm{KT}(y)/t$, where y is the concatenation of t independent samples from p_L.

(β) If we are able to uniformly sample $\{p_L\}$ *exactly* in polynomial time (possibly with access to an oracle for MKTP), then we can evaluate the estimator $\mathrm{KT}(y)/t$ in polynomial time with access to MKTP. This is because the oracle for MKTP lets us evaluate KT in polynomial time.

Thus, if we were able to uniformly sample $\{p_L\}$ *exactly* in polynomial time, we'd be done. We do not know how to do that, but we can do it *approximately*, which we argue is sufficient.

The need for a ZPP-computable complete invariant comes in when representing the abstract group elements as strings. In order to formally state the requirement, we make the underlying representation of group elements explicit; we denote it by η.

LEMMA 5.5. *Let $\{H_x\}$ be an ensemble of groups. Suppose that the ensemble has a representation η such that the uniform distribution on H_x is uniformly samplable in polynomial-time, products and inverses in H_x are computable in* ZPP*, and there exists a* ZPP*-computable complete invariant for η. Then for any list L of elements of H_x, the logarithm of the order of the group generated by L, i.e., $\log |\langle L \rangle|$, can be pac underestimated with any constant deviation $\Delta > 0$ in randomized time $\mathrm{poly}(|x|, |L|)$ with oracle access to* MKTP*.*

Proof. Let ζ be the ZPP-computable complete invariant for η. For each list L of elements of H_x, let p_L denote the distribution of $\zeta(h)$ when h is picked uniformly at random from $\langle L \rangle$. Note that p_L is flat with entropy $s = \log |\langle L \rangle|$.

[6] For many cases where $L \subseteq \mathrm{GL}_n(\mathbb{F}_q)$, [9] shows how to compute the exact order of $\langle L \rangle$ in ZPP with oracles for integer factorization and the discrete log. Combined with follow-up results of [29, 33, 30], the only cases that remain open are those over a field of characteristic 2 where $\langle L \rangle$ contains at least one of the Ree groups ${}^2F_4(2^{2n+1})$ as a composition factor, and those over a field of characteristic 3 where $\langle L \rangle$ contains at least one of the Ree groups ${}^2G_2(3^{2n+1})$ as a composition factor. The claim follows as integer factorization and discrete log can be computed in $\mathsf{ZPP}^{\mathrm{MKTP}}$.

CLAIM 5.6. *The ensemble of distributions $\{p_L\}$ is uniformly samplable in polynomial time.*

For every constant $\delta > 0$, the claim yields a family of random variables $\{R_{L,\delta}\}$ computable uniformly in polynomial time such that $R_{L,\delta}$ induces a distribution $p_{L,\delta}$ that approximates p_L to within a factor $1+\delta$. Note that the min-entropy of $p_{L,\delta}$ is at least $s - \log(1+\delta)$, and the max-entropy of $p_{L,\delta}$ at most $s + \log(1+\delta)$, thus their difference is no more than $2\log(1+\delta)$.

Let $M_\delta(L)$ denote $\mathrm{KT}(y)/t$, where y is the concatenation of t independent samples from $p_{L,\delta}$.

(α) The Entropy Estimator Corollary guarantees that for any sufficiently large polynomial t, M_δ is a pac underestimator for the entropy of $p_{L,\delta}$ with deviation $2\log(1+\delta) + o(1)$, and thus a pac underestimator for $s = \log|\langle L\rangle|$ with deviation $3\log(1+\delta) + o(1)$.

(β) For any polynomial t, we can compute M_δ in polynomial time with access to an oracle for MKTP. This is because $R_{L,\delta}$ enables us to generate y in polynomial time. We then use the oracle for MKTP to compute $\mathrm{KT}(y)$ exactly, and divide by t.

Thus, M_δ meets all the requirements for our estimator as long as $3\log(1+\delta) < \Delta$, which holds for some positive constant δ.

This completes the proof of Lemma 5.5 modulo the proof of the claim. □

The proof of Claim 5.6 relies on the notion of *Erdős–Rényi generators*. A list of generators $L = (h_1, \ldots, h_k)$ is said to be Erdős–Rényi with factor $1+\delta$ if a random subproduct of L approximates the uniform distribution on $\langle L\rangle$ within a factor $1+\delta$, where a random subproduct is obtained by picking $r_i \in \{0,1\}$ for each $i \in [k]$ uniformly at random, and outputting $h_1^{r_1} h_2^{r_2} \cdots h_k^{r_k}$.

Proof of Claim 5.6. By definition, if L happens to be Erdős–Rényi with factor $1+\delta$, then p_L can be sampled to within a factor $1+\delta$ with fewer than $|L|$ products in H_x.

Erdős and Rényi [15] showed that, for any finite group Γ, with high probability, a list of $\mathrm{poly}(\log|\Gamma|, \log(1/\delta))$ random elements of Γ form an Erdős–Rényi list of generators with factor $1+\delta$. For $\Gamma = \langle L\rangle$, this gives a list L' for which we can sample $p_{L'} = p_L$. By hard-wiring the list L' into the sampler for $p_{L'}$, it follows that p_L is samplable by circuits of size $\mathrm{poly}(\log|\langle L\rangle|, \log(1/\delta)) \le \mathrm{poly}(|L|/\delta)$.

As for *uniformly* sampling $\{p_L\}$ in polynomial time, [7, Theorem 1.1] gives a randomized algorithm that generates out of L a list L' of elements from $\langle L\rangle$ that, with probability $1-\varepsilon$, are Erdős–Rényi with factor $1+\delta$. The algorithm runs in time $\mathrm{poly}(|x|, |L|, \log(1/\delta), \log(1/\varepsilon))$ assuming products and inverses in H_x can be computed in ZPP. For $\varepsilon = \delta/|\langle L\rangle|$, the overall distribution of a random subproduct of L' is within a factor $1+2\delta$ from p_L, and can be generated in time $\mathrm{poly}(|x|, |L|, \log(1/\delta)) \le \mathrm{poly}(|x|, |L|, 1/\delta)$. As δ can be an arbitrary positive constant, it follows that p_L is uniformly samplable in polynomial time. □

Following the four steps listed at the beginning of this section, we can replace condition 3 in Theorem 5.2 by the conditions of Lemma 5.4 (for step 1), those of Lemma 5.5 (for step 2), and the existence of an estimator for the size $|H|$ of the sample space as stated in step 3. This gives the following result:

THEOREM 5.7. *Let* Iso *denote an Isomorphism Problem as in Definition 1.1. Suppose that the ensemble $\{H_x\}$ has a representation η such that conditions 1 and 2 of Theorem 5.2 hold as well as the following additional conditions:*

4. [group operations] *Products and inverses in* H_x *are computable in* ZPP.
5. [sample space estimator] *The map* $x \mapsto |H_x|$ *has a pac overestimator with deviation* $1/8$ *computable in* $\mathsf{ZPP}^{\mathrm{MKTP}}$.
6. [complete group invariant] *There exists a complete invariant* ζ *for the representation* η *that is computable in* ZPP.

Then Iso $\in \mathsf{ZPP}^{\mathrm{MKTP}}$.

As was the case for Theorem 5.2, the conditions of Theorem 5.7 can be satisfied in a straightforward way for GI. The representation η of the symmetric groups S_n meets all the requirements that only involve the underlying group: uniform samplability as in the first part of condition 1, efficient group operations as in condition 4, the sample space size $|H| = |S_n| = n!$ can be computed efficiently (condition 5), and the identity mapping can be used as the complete group invariant ζ (condition 6). The efficiency of the action (the second part of condition 1) and condition 2 about a complete universe invariant are also met in the same way as before.

We point out that Claim 5.6 can be used to show that the uniform distribution on H_x is uniformly samplable in polynomial time (the first part of condition 1), provided a set of generators for H_x can be computed in ZPP. This constitutes another use of [7, Theorem 1.1].

On the other hand, the use of [7, Theorem 1.1] in the proof of Theorem 5.7 can be eliminated. Referring to parts (α) and (β) in the intuition and proof of Lemma 5.5, we note the following:

(α) The first part of the proof of Claim 5.6 relies on [15] but not on [7, Theorem 1.1]. It shows that p_L is samplable by polynomial-size circuits, which is sufficient for the Entropy Estimator Corollary to apply and show that $M_\delta(L) = \mathrm{KT}(y)/t$ is a pac underestimator for $\log|\langle L\rangle|$ with deviation $3\log(1+\delta)+o(1)$, where y is the concatenation of t independent samples from $p_{L,\delta}$ for a sufficiently large polynomial t.

(β) Specialized to the case where $\langle L\rangle = \mathrm{Aut}(\omega)$, the first part of the proof of Lemma 5.4 shows that, for any constant $\delta > 0$, $p_{L,\delta}$ is *uniformly* samplable in polynomial time with access to an oracle for MKTP. Once we have generated y with the help of MKTP, we use MKTP once more to evaluate $\mathrm{KT}(y)$ and output $M_\delta(L) = \mathrm{KT}(y)/t$.

This way, for any constant $\delta > 0$ we obtain a pac underestimator M_δ for $\log|\mathrm{Aut}(\omega)|$ with deviation $3\log(1+\delta) + o(1)$ that is computable in polynomial time with access to MKTP.

This alternate construction replaces steps 1 and 2 in the outline from the beginning of this section. The resulting alternate proof of Theorem 5.7 is more elementary (as it does not rely on [7, Theorem 1.1]) but does not entirely follow the approach we used for GI of first finding a list L of elements that likely generates $\mathrm{Aut}(\omega)$ (and never generates more) and then determining the size of the subgroup generated by L.

Remark 5.8. Remark 5.3 on relaxing the efficiency requirement in conditions 1 and 2 of Theorem 5.2 extends similarly to Theorem 5.7. For Theorem 5.7, it suffices that all the computations mentioned in conditions 1, 2, 4, and 6 be do-able by $\mathsf{ZPP}^{\mathrm{MKTP}}$-constructible ordinary circuits.

6. Instantiations of the Isomorphism Problem. In this section we argue that Theorem 5.7 applies to the instantiations of the Isomorphism Problem listed in Table 1 (other than GI, which we covered in Section 3). We describe each problem, provide some background, and show that the conditions of Theorem 5.7 hold, thus

proving that the problem is in $\mathsf{ZPP}^{\mathrm{MKTP}}$.

Linear code equivalence. A *linear code* over the finite field $\mathbb{F}_q$ is a d-dimensional linear subspace of $\mathbb{F}_q^n$ for some n. Two such codes are (permutationally) *equivalent* if there is a permutation of the n coordinates that makes them equal as subspaces.

Linear Code Equivalence is the problem of deciding whether two linear codes are equivalent, where the codes are specified as the row-span of a $d \times n$ matrix (of rank d), called a *generator matrix*. Note that two different inputs may represent the same code. There exists a mapping reduction from GI to Linear Code Equivalence over any field [38, 23]; Linear Code Equivalence is generally thought to be harder than GI.

In order to cast Code Equivalence in our framework, we consider the family of actions $(S_n, \Omega_{n,d,q})$ where $\Omega_{n,d,q}$ denotes the linear codes of length n and dimension d over $\mathbb{F}_q$, and S_n acts by permuting the coordinates. To apply Theorem 5.7, as the underlying group is S_n, we only need to check the efficiency of the action (second part of condition 1) and the complete universe invariant (condition 2). The former holds because the action only involves swapping columns in the generator matrix. For condition 2 we can define $\nu(z)$ to be the reduced row echelon form of z. This choice works because two generator matrices define the same code iff they have the same reduced row echelon form, and it can be computed in polynomial time.

COROLLARY 6.1. *Linear Code Equivalence is in* $\mathsf{ZPP}^{\mathrm{MKTP}}$.

Permutation Group Conjugacy. Two permutation groups $\Gamma_0, \Gamma_1 \leq S_n$ are *conjugate* (or permutationally isomorphic) if there exists a permutation $\pi \in S_n$ such that $\Gamma_1 = \pi\Gamma_0\pi^{-1}$; such a π is called a conjugacy.

The *Permutation Group Conjugacy* problem is to decide whether two subgroups of S_n are conjugate, where the subgroups are specified by a list of generators. The problem is known to be in $\mathsf{NP} \cap \mathsf{coAM}$, and is at least as hard as Linear Code Equivalence. Currently the best known algorithm runs in time $2^{O(n)}\,\mathrm{poly}(|\Gamma_1|)$ [10]—that is, the runtime depends not only on the input size (which is polynomially related to n), but also on the size of the groups generated by the input permutations, which can be exponentially larger.

Casting Permutation Group Conjugacy in the framework is similar to before: S_n acts on the subgroup by conjugacy. The action is computable in polynomial time (second part of condition 1) as it only involves inverting and composing permutations. It remains to check condition 2. Note that there are many different lists that generate the same subgroup. We make use of the normal form provided by the following lemma.

LEMMA 6.2. *There is a* $\mathrm{poly}(n)$*-time algorithm* ν *that takes as input a list* L *of elements of* S_n*, and outputs a list of generators for the subgroup generated by the elements in* L *such that for any two lists* L_0, L_1 *of elements of* S_n *that generate the same subgroup,* $\nu(L_0) = \nu(L_1)$.

The normal form from Lemma 6.2 was known to some experts (Babai, personal communication); for completeness we provide a proof in the Appendix. By Theorem 5.7 we conclude:

COROLLARY 6.3. *Permutation Group Conjugacy is in* $\mathsf{ZPP}^{\mathrm{MKTP}}$.

Matrix Subspace Conjugacy. A *linear matrix space* over $\mathbb{F}_q$ is a d-dimensional linear subspace of $n \times n$ matrices. Two such spaces V_0 and V_1 are *conjugate* if there is an invertible $n \times n$ matrix X such that $V_1 = XV_0X^{-1} \doteq \{X \cdot M \cdot X^{-1} : M \in V_0\}$, where "$\cdot$" represents matrix multiplication.

Matrix Subspace Conjugacy is the problem of deciding whether two linear matrix spaces are conjugate, where the spaces are specified as the linear span of d linearly

independent $n \times n$ matrices. There exist mapping reductions from GI and Linear Code Equivalence to Matrix Subspace Conjugacy [23]; Matrix Subspace Conjugacy is generally thought to be harder than Linear Code Equivalence.

In order to cast Matrix Subspace Conjugacy in our framework, we consider the family of actions $(\mathrm{GL}_n(\mathbb{F}_q), \Omega_{n,d,q})$ where $\mathrm{GL}_n(\mathbb{F}_q)$ denotes the n-by-n general linear group over $\mathbb{F}_q$ (consisting of all invertible n-by-n matrices over $\mathbb{F}_q$ with multiplication as the group operation), $\Omega_{n,d,q}$ represents the set of d-dimensional subspaces of $\mathbb{F}_q^{n\times n}$, and the action is by conjugation. As was the case with Linear Code Equivalence, two inputs may represent the same linear matrix space, and we use the reduced row echelon form of ω when viewed as a matrix in $\mathbb{F}_q^{d\times n^2}$ as the complete universe invariant. This satisfies condition 2 of Theorem 5.7. The action is computable in polynomial time (second part of condition 1) as it only involves inverting and multiplying matrices in $\mathrm{GL}_n(\mathbb{F}_q)$.

The remaining conditions only depend on the underlying group, which is different from before, namely $\mathrm{GL}_n(\mathbb{F}_q)$ instead of S_n. Products and inverses in $\mathrm{GL}_n(\mathbb{F}_q)$ can be computed in polynomial time (condition 4), and the identity mapping serves as the complete group invariant (condition 6). Thus, only the uniform sampler for $\mathrm{GL}_n(\mathbb{F}_q)$ (first part of condition 1) and the pac overestimator for $|\mathrm{GL}_n(\mathbb{F}_q)|$ (condition 5) remain to be argued.

The standard way of constructing the elements of $\mathrm{GL}_n(\mathbb{F}_q)$ consists of n steps, where the i-th step picks the i-th row as any row vector that is linearly independent of the $(i-1)$ prior ones. The number of choices in the i-th step is $q^n - q^{i-1}$. Thus, $|\mathrm{GL}_n(\mathbb{F}_q)| = \prod_{i=1}^{n}(q^n - q^{i-1})$, which can be computed in time $\mathrm{poly}(|x|)$ (condition 5). It also follows that the probability that a random $(n\times n)$-matrix over $\mathbb{F}_q$ is in $\mathrm{GL}_n(\mathbb{F}_q)$ is at least some positive constant (independent of n and q), which implies that $\{H_x\}$ can be uniformly sampled in time $\mathrm{poly}(|x|)$, satisfying the first part of condition 1.

COROLLARY 6.4. *Matrix Subspace Conjugacy is in* $\mathsf{ZPP}^{\mathrm{MKTP}}$.

Before closing, we note that there is an equivalent of the Lehmer code for $\mathrm{GL}_n(\mathbb{F}_q)$. We do not need it for our results, but it may be of interest in other contexts. In general, Lehmer's approach works for indexing objects that consist of multiple components where the set of possible values for the i-th component may depend on the values of the prior components, but the *number* of possible values for the i-th component is independent of the values of the prior components. An efficiently decodable indexing follows provided one can efficiently index the possible values for the i-th component given the values of the prior components. The latter is possible for $\mathrm{GL}_n(\mathbb{F}_q)$. We include a proof for completeness.

PROPOSITION 6.5. *For each n and prime power q, $\mathrm{GL}_n(\mathbb{F}_q)$ has an indexing that is uniformly decodable in time* $\mathrm{poly}(n, \log(q))$.

Proof. Consider the above process. In the i-th step, we need to index the complement of the subspace spanned by the $i-1$ row vectors picked thus far, which are linearly independent. This can be done by extending those $i-1$ row vectors by $n-i+1$ new row vectors to a full basis, and considering all q^{i-1} linear combinations of the $i-1$ row vectors already picked, and all $(q^{n-i+1}-1)$ *non-zero* linear combinations of the other basis vectors, and outputting the sum of the two components. More precisely, on input $k \in [q^n - q^{i-1}]$, write $k-1$ as $k_0 + k_1 q^{i-1}$ where k_0 and k_1 are nonnegative integers with $k_0 < q^{i-1}$, and output $v_0 + v_1$ where v_0 is the combination of the $i-1$ row vectors already picked with coefficients given by the binary expansion of k_0, and v_1 is linear combination of the other basis vectors with coefficients given

by the binary expansion of $k_1 + 1$. Using Gaussian elimination to construct the other basis vectors, the process runs in time $\text{poly}(n, \log(q))$. ∎

7. Future Directions. We end with a few directions for further research.

7.1. What about Minimum Circuit Size?. We suspect that our techniques also apply to MCSP in place of MKTP, but we have been unsuccessful in extending them to MCSP so far. To show our result for the complexity measure $\mu = \text{KT}$, we showed the following property for polynomial-time samplable flat distributions R: There exists an efficiently computable bound $\theta(s,t)$ and a polynomial t such that if y is the concatenation of t independent samples from R, then

(7.1) $\mu(y) > \theta(s,t)$ holds with high probability if R has entropy $s+1$, and

(7.2) $\mu(y) \leq \theta(s,t)$ always holds if R has entropy s.

We set $\theta(s,t)$ slightly below $\kappa(s+1,t)$ where $\kappa(s,t) \doteq st$. (7.1) followed from a counting argument, and (7.2) by showing that

$$\mu(y) \leq \kappa(s,t) \cdot \left(1 + \frac{n^c}{t^\alpha}\right) \tag{7.3}$$

always holds for some positive constants c and α. We concluded by observing that for a sufficiently large polynomial t the right-hand side of (7.3) is significantly below $\kappa(s+1,t)$.

Mimicking the approach with μ denoting circuit complexity, we set

$$\kappa(s,t) = \frac{st}{\log(st)} \cdot \left(1 + (2 - o(1)) \cdot \frac{\log\log(st)}{\log(st)}\right).$$

Then (7.1) follows from [43]. As for (7.2), the best counterpart to (7.3) we know of (see, e.g., [18]) is

$$\mu(y) \leq \frac{st}{\log(st)} \cdot \left(1 + (3 + o(1)) \cdot \frac{\log\log(st)}{\log(st)}\right).$$

However, in order to make the right-hand side of (7.3) smaller than $\kappa(s+1,t)$, t needs to be exponential in s.

One possible way around the issue is to boost the entropy gap between the two cases. This would not only show that all our results for MKTP apply to MCSP as well, but could also form the basis for reductions between different versions of MCSP (defined in terms of different circuit models, or in terms of different size parameters), and to clarify the relationship between MKTP and MCSP. Until now, all of these problems have been viewed as morally equivalent to each other, although no efficient reduction is known between *any* two of these, in either direction. Given the central role that MCSP occupies, it would be desirable to have a theorem that indicates that MCSP is fairly robust to minor changes to its definition. Currently, this is lacking.

On a related point, it would be good to know how the complexity of MKTP compares with the complexity of the KT-random strings: $R_{\text{KT}} = \{x : \text{KT}(x) \geq |x|\}$. Until our work, all prior reductions from natural problems to MCSP or MKTP carried over to R_{KT}—but this would seem to require even stronger gap amplification theorems. The relationship between MKTP and R_{KT} is analogous to the relationship between MCSP and the special case of MCSP that is denoted MCSP′ in [34]: MCSP′ consists of truth tables f of m-ary Boolean functions that have circuits of size at most $2^{m/2}$.

7.2. Statistical Zero Knowledge. Allender and Das [2] generalized their result that GI $\in \mathsf{RP}^{\mathrm{MKTP}}$ to $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathrm{MKTP}}$ by applying their approach to a known SZK-complete problem. Our proof that GI $\in \mathsf{coRP}^{\mathrm{MKTP}}$ similarly generalizes to $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathrm{MKTP}}$. We use the problem Entropy Approximation, which is complete for SZK under oracle reductions [20, Lemma 5.1]:[7] Given a circuit C and a threshold θ with the promise that the distribution induced by C has entropy either at most $\theta - 1$ or else at least $\theta + 1$, decide whether the former is the case. By combining the Flattening Lemma [21] with the Entropy Estimator Corollary, one can show that for any distribution of entropy s sampled by a circuit C, the concatenation of t random samples from C has, with high probability, KT complexity between $ts - t^{1-\alpha_0} \cdot \mathrm{poly}(|C|)$ and $ts + t^{1-\alpha_0} \cdot \mathrm{poly}(|C|)$ for some positive constant α_0. Along the lines of Remark 3.5, this allows us to show that Entropy Approximation, and hence all of SZK, is in $\mathsf{BPP}^{\mathrm{MKTP}}$.

We do not know how to eliminate the errors from those reductions: Is $\mathsf{SZK} \subseteq \mathsf{ZPP}^{\mathrm{MKTP}}$, or equivalently, is Entropy Approximation in $\mathsf{ZPP}^{\mathrm{MKTP}}$? Our approach yields that Entropy Approximation is in $\mathsf{coRP}^{\mathrm{MKTP}}$ (no false negatives) *when the input distributions are almost flat*, i.e., when the difference between the max- and min-entropy is small. However, it is not known whether that restriction of Entropy Approximation is complete for SZK[8] (Goldreich and Vadhan, personal communication). Moreover, we do not see how to eliminate the false positives.

Trying to go beyond SZK, recall that except for the possible use of the MKTP oracle in the construction of the probably-correct overestimator from condition 3 in Theorem 5.2 (or as discussed in Remark 5.3), the reduction in Theorem 5.2 makes only one query to the oracle. It was observed in [26] that the reduction also works for any relativized KT problem MKTP^A (where the universal machine for KT complexity has access to oracle A). More significantly, [26] shows that any problem that is accepted with negligible error probability by a probabilistic reduction that makes only one query, relative to *every* set MKTP^A, must lie in $\mathsf{AM} \cap \mathsf{coAM}$. Thus, without significant modification, our techniques cannot be used in order to reduce any class larger than $\mathsf{AM} \cap \mathsf{coAM}$ to MKTP.

The property that only one query is made to the oracle was subsequently used in order to show that MKTP is hard for the complexity class DET under mapping reductions computable in nonuniform NC^0 [4]. Similar hardness results (but for a more powerful class of reducibilities) hold also for MCSP [36]. This has led to unconditional lower bounds on the circuit complexity of MKTP [4, 25], showing that MKTP does not lie in the complexity class $\mathsf{AC}^0[p]$ for any prime p; it is still open whether similar circuit lower bounds hold for MCSP.

Appendix A. Coset Indexings and Normal Forms for Permutation Groups.

In this appendix we develop the efficiently decodable indexings for cosets of per-

[7] Entropy Approximation is complete for NISZK (Non-Interactive SZK) under *mapping* reductions [20]. Problems that are complete for SZK under mapping reductions include Statistical Difference [40] and Entropy Difference [21]. The mapping reduction from Statistical Difference to Entropy Difference in [22, Theorem 3] is similar to our reduction from Isomorphism Problems to MKTP.

[8] The Flattening Lemma [21] allows us to restrict to distributions that are almost flat in an *average-case* sense, but we need almost flatness in the above *worst-case* sense. For example, consider a circuit C that induces a distribution of entropy less than $\theta - 1$ whose support contains all strings of length n where $n \gg \theta$. In that case, there is no nontrivial worst-case bound on the KT complexity of samples from C; with positive probability, t samples from C may have KT-complexity close to $t \cdot n \gg t \cdot (\theta + 1)$.

mutation subgroups claimed in Lemma 3.4, and also use some of the underlying ideas to establish the normal form for permutation groups stated in Lemma 6.2.

Indexing Cosets. The indexings are not strictly needed for our main results as the generic encoding from the Encoding Lemma can be used as a substitute. However, the information-theoretic optimality of the indexings may be useful in other contexts. In fact, we present a further generalization that may be of independent interest, namely an efficiently decodable indexing for cosets of permutation subgroups within another permutation subgroup.

LEMMA A.1. *For all $\Gamma \leq H \leq S_n$, there exists an indexing of the cosets[9] of Γ within H that is uniformly decodable in polynomial time when Γ and H are given by a list of generators.*

Lemma 3.4 is just the instantiation of Lemma A.1 with $H = S_n$. The proof of Lemma A.1 requires some elements of the theory of permutation groups. Given a list of permutations $\pi_1, \ldots, \pi_k \in S_n$, we write $\Gamma = \langle \pi_1, \ldots, \pi_k \rangle \leq S_n$ for the subgroup they generate. Given a permutation group $\Gamma \leq S_n$ and a point $i \in [n]$, the Γ-orbit of i is the set $\{g(i) : g \in \Gamma\}$, and the Γ-stabilizer of i is the subgroup $\{g \in \Gamma : g(i) = i\} \leq \Gamma$.

We make use of the fact that (a) the number of cosets of a subgroup Γ of a group H equals $|H|/|\Gamma|$, and (b) the orbits of a subgroup Γ of H form a refinement of the orbits of H. We also need the following basic routines from computational group theory (see, for example, [27, 41]).

PROPOSITION A.2. *Given a set of permutations that generate a subgroup $\Gamma \leq S_n$, the following can be computed in time polynomial in n:*

(1) the cardinality $|\Gamma|$,
(2) a permutation in Γ that maps u to v for given $u, v \in [n]$, or report that no such permutation exists in Γ, and
(3) a list of generators for the subgroup Γ_v of Γ that stabilizes a given element $v \in [n]$.

The proof of Lemma A.1 makes implicit use of an efficient process for finding a *canonical representative* of $\pi\Gamma$ for a given permutation $\pi \in H$, where "canonical" means that the representative depends on the coset $\pi\Gamma$ only. The particular canonical representative the process produces can be specified as follows.

DEFINITION A.3. *For a permutation $\pi \in S_n$ and a subgroup $\Gamma \leq S_n$, the* canonical representative *of π modulo Γ, denoted $\pi \bmod \Gamma$, is the lexicographically least $\pi' \in \pi\Gamma$, where the lexicographic ordering is taken by viewing a permutation π' as the sequence $(\pi'(1), \pi'(2), \ldots, \pi'(n))$.*

The process is well-known. We spell it out in the proof of the following lemma as it provides intuition for the proof of Lemma A.1.

LEMMA A.4 (Theorem 10 in [6]). *There exists a polynomial-time algorithm that takes as input a generating set for a subgroup $\Gamma \leq S_n$ and a permutation $\pi \in S_n$, and outputs the canonical representative $\pi \bmod \Gamma$.*

Proof of Lemma A.4. Consider the element 1 of $[n]$. Permutations in $\pi\Gamma$ map 1 to an element v in the same Γ-orbit as $\pi(1)$, and for every element v in the Γ-orbit of $\pi(1)$ there exists a permutation in $\pi\Gamma$ that maps 1 to v. We can canonize the behavior of π on the element 1 by replacing π with a permutation $\pi_1 \in \pi\Gamma$ that maps 1 to the minimum element m in the Γ-orbit of $\pi(1)$. This can be achieved by multiplying π to

[9] Recall footnote 3 on page 13.

the right with a permutation in Γ that maps $\pi(1)$ to m.

Next we apply the same process to π_1 but consider the behavior on the element 2 of $[n]$. Since we are no longer allowed to change the value of $\pi_1(1)$, which equals m, the canonization of the behavior on 2 can only use multiplication on the right with permutations in Γ_m, i.e., permutations in Γ that stabilize the element m. Doing so results in a permutation $\pi_2 \in \pi_1\Gamma$.

We repeat this process for all elements $k \in [n]$ in order. In the k-th step, we canonize the behavior on the element k by multiplying on the right with permutations in $\Gamma_{\pi_{k-1}([k-1])}$, i.e., permutations in Γ that pointwise stabilize all of the elements $\pi_{k-1}(\ell)$ for $\ell \in [k-1]$. ∎

Proof of Lemma A.1. The number of canonical representatives modulo Γ in H equals the number of distinct (left) cosets of Γ in H, which is $|H|/|\Gamma|$. We construct an algorithm that takes as input a list of generators for Γ and H, and an index $i \in [|H|/|\Gamma|]$, and outputs the permutation σ that is the lexicographically i-th canonical representative modulo Γ in H.

The algorithm uses a prefix search to construct σ. In the k-th step, it knows the prefix $(\sigma(1), \sigma(2), \ldots, \sigma(k-1))$ of length $k-1$, and needs to figure out the correct value $v \in [n]$ to extend the prefix with. In order to do so, the algorithm needs to compute for each $v \in [n]$ the count c_v of canonical representatives modulo Γ in H that agree with σ on $[k-1]$ and take the value v at k. The following claims allow us to do that efficiently when given a permutation $\sigma_{k-1} \in H$ that agrees with σ on $[k-1]$. The claims use the notation $T_{k-1} \doteq \sigma_{k-1}([k-1])$, which also equals $\sigma([k-1])$.

Claim A.5. *The canonical representatives modulo Γ in H that agree with $\sigma \in H$ on $[k-1]$ are exactly the canonical representatives modulo $\Gamma_{T_{k-1}}$ in $\sigma_{k-1}H_{T_{k-1}}$.*

Proof. The following two observations imply Claim A.5.

(i) A permutation $\pi \in H$ agrees with $\sigma \in H$ on $[k-1]$
$\Leftrightarrow$ π agrees with σ_{k-1} on $[k-1]$
$\Leftrightarrow$ $\sigma_{k-1}^{-1}\pi \in H_{T_{k-1}}$
$\Leftrightarrow$ $\pi \in \sigma_{k-1}H_{T_{k-1}}$.

(ii) Two permutations in $\sigma_{k-1}H_{T_{k-1}}$, say $\pi \doteq \sigma_{k-1}g$ and $\pi' \doteq \sigma_{k-1}g'$ for $g, g' \in H_{T_{k-1}}$, belong to the same left coset of Γ iff they belong to the same left coset of $\Gamma_{T_{k-1}}$. This follows because if $\sigma_{k-1}g' = \sigma_{k-1}gh$ for some $h \in \Gamma$, then h equals $g^{-1}g' \in H_{T_{k-1}}$, so $h \in \Gamma \cap H_{T_{k-1}} = \Gamma_{T_{k-1}}$. ∎

Claim A.6. *The count c_v for $v \in [n]$ is nonzero iff v is the minimum of some $\Gamma_{T_{k-1}}$-orbit contained in the $H_{T_{k-1}}$-orbit of $\sigma_{k-1}(k)$.*

Proof. The set of values of $\pi(k)$ when π ranges over $\sigma_{k-1}H_{T_{k-1}}$ is the $H_{T_{k-1}}$-orbit of $\sigma_{k-1}(k)$. Since $\Gamma_{T_{k-1}}$ is a subgroup of $H_{T_{k-1}}$, this orbit is the union of some $\Gamma_{T_{k-1}}$-orbits. Combined with Claim A.5 and the construction of the canonical representatives modulo $\Gamma_{T_{k-1}}$, this implies Claim A.6. ∎

Claim A.7. *If a count c_v is nonzero then it equals $|H_{T_{k-1}\cup\{v\}}|/|\Gamma_{T_{k-1}\cup\{v\}}|$.*

Proof. Since the count is nonzero, there exists a permutation $\sigma' \in H$ that is a canonical representative modulo Γ that agrees with σ_{k-1} on $[k-1]$ and satisfies $\sigma'(k) = v$. Applying Claim A.5 with σ replaced by σ', k by $k' \doteq k+1$, T_{k-1} by $T'_k \doteq T_{k-1} \cup \{v\}$, and σ_{k-1} by any permutation $\sigma'_k \in H$ that agrees with σ' on $[k]$, yields Claim A.7. This is because the number of canonical representatives modulo $\Gamma_{T'_k}$ in $\sigma'_k H_{T'_k}$ equals the number of (left) cosets of $\Gamma_{T'_k}$ in $H_{T'_k}$, which is the quantity stated in Claim A.7. ∎

Algorithm A.1

Input: positive integer n, $\Gamma \leq H \leq S_n$, $i \in [|H|/|\Gamma|]$
Output: lexicographically i-th canonical representative modulo Γ in H

$\sigma_0 \leftarrow id$
for $k = 1$ to n **do**
  $O_1, O_2, \ldots \leftarrow \Gamma$-orbits contained in the H-orbit of $\sigma_{k-1}(k)$, in increasing order of $\min(O_i)$
  find integer ℓ such that $\sum_{j=1}^{\ell-1} c_{\min(O_j)} < i \leq \sum_{j=1}^{\ell} c_{\min(O_j)}$, where $c_v \doteq |H_v|/|\Gamma_v|$
  $i \leftarrow i - \sum_{i=1}^{\ell-1} c_{\min(O_j)}$
  $m \leftarrow \min(O_\ell)$
  find $\tau \in H$ such that $\tau(\sigma_{k-1}(k)) = m$
  $\sigma_k \leftarrow \sigma_{k-1}\tau$
  $H \leftarrow H_m$; $\Gamma \leftarrow \Gamma_m$
return σ_n

The algorithm builds a sequence of permutations $\sigma_0, \sigma_1, \ldots, \sigma_n \in H$ such that σ_k agrees with σ on $[k]$. It starts with the identity permutation $\sigma_0 = id$, builds σ_k out of σ_{k-1} for increasing values of $k \in [n]$, and outputs the permutation $\sigma_n = \sigma$.

Pseudocode for the algorithm is presented in Algorithm A.1. Note that the pseudocode modifies the arguments Γ, H, and i along the way. Whenever a group is referenced in the pseudocode, the actual reference is to a list of generators for that group.

The correctness of the algorithm follows from Claims A.6 and A.7. The fact that the algorithm runs in polynomial time follows from Proposition A.2. □

Normal Form. Finally, we use the canonization captured in Definition A.3 and Lemma A.4 to establish the normal form for permutation groups given by Lemma 6.2 (restated below):

LEMMA 6.2. *There is a polynomial-time algorithm ν that takes as input a list L of elements of S_n, and outputs a list of generators for the subgroup generated by the elements in L such that for any two lists L_0, L_1 of elements of S_n that generate the same subgroup, $\nu(L_0) = \nu(L_1)$.*

Proof. Let Γ denote the subgroup generated by L, and recall that $\Gamma_{[i]}$ denotes the subgroup of Γ that stabilizes each element in $[i]$, for $i \in \{0, 1, \ldots, n\}$. We have that $\Gamma_{[0]} = \Gamma$, and $\Gamma_{[n-1]}$ consists of the identity only.

We define $\nu(L)$ as follows. Start with ν being the empty list. For $i \in [n-1]$, in the i-th step we consider each $j \in [n]$ that is in the $\Gamma_{[i-1]}$-orbit of i in order. Note that for each such j, the permutations in $\Gamma_{[i-1]}$ that map i to j form a coset of $\Gamma_{[i-1]} \bmod \Gamma_{[i]}$. We append the canonical representative of this coset to ν. $\nu(L)$ is the value of ν after step $n-1$.

As we only include permutations from Γ, $\nu(L)$ generates a subgroup of Γ. By construction, for each $i \in [n-1]$, the permutations we add in the i-th step represent all cosets of $\Gamma_{[i-1]} \bmod \Gamma_{[i]}$. It follows by induction on $n-i$ that the permutations added to ν during and after the i-th step generate $\Gamma_{[i-1]}$ for $i \in [n]$. Thus, $\nu(L)$ generates $\Gamma_{[0]} = \Gamma$.

That $\nu(L)$ only depends on the subgroup Γ generated by L follows from its definition, which only refers to the abstract groups $\Gamma_{[i]}$, their cosets, and their canonical

representatives. That $\nu(L)$ can be computed in polynomial time follows by tracking a set of generators for the subgroups $\Gamma_{[i]}$ based on Proposition A.2. More specifically, we use item 2 to check whether a given j is in the $\Gamma_{[i-1]}$-orbit of i, and item 3 to obtain $\Gamma_{[i]}$ out of $\Gamma_{[i-1]}$ as $\Gamma_{[i]} = (\Gamma_{[i-1]})_i$. ∎

Acknowledgments. We thank V. Arvind for helpful comments about the graph automorphism problem and rigid graphs, Alex Russell and Yoav Kallus for helpful ideas on encoding and decoding graphs, Laci Babai and Peter Brooksbank for answering questions about computational group theory, and Oded Goldreich and Salil Vadhan for answering questions about SZK. We also thank the reviewers of the conference version [3] and the anonymous referees of SICOMP for their helpful suggestions.

REFERENCES

[1] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger, *Power from random strings*, SIAM Journal on Computing, 35 (2006), pp. 1467–1493, https://doi.org/10.1137/050628994.

[2] E. Allender and B. Das, *Zero knowledge and circuit minimization*, Information and Computation, 256 (2017), pp. 2–8, https://doi.org/10.1016/j.ic.2017.04.004.

[3] E. Allender, J. A. Grochow, D. van Melkebeek, C. Moore, and A. Morgan, *Minimum Circuit Size, Graph Isomorphism, and Related Problems*, in 9th Innovations in Theoretical Computer Science Conference (ITCS '18), vol. 94 of LIPIcs, 2018, pp. 20:1–20:20, https://doi.org/10.4230/LIPIcs.ITCS.2018.20.

[4] E. Allender and S. Hirahara, *New insights on the (non)-hardness of circuit minimization and related problems*, in Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS '17), 2017, https://doi.org/10.4230/LIPIcs.MFCS.2017.54.

[5] E. Allender, M. Koucký, D. Ronneburger, and S. Roy, *The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory*, Journal of Computer and System Sciences, 77 (2010), pp. 14–40, https://doi.org/10.1016/j.jcss.2010.06.004.

[6] V. Arvind and P. P. Kurur, *Graph isomorphism is in SPP*, Information and Computation, 204 (2006), pp. 835–852, https://doi.org/10.1016/j.ic.2006.02.002.

[7] L. Babai, *Local expansion of vertex-transitive graphs and random generation in finite groups*, in Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC '91), 1991, pp. 164–174, https://doi.org/10.1145/103418.103440.

[8] L. Babai, *Graph isomorphism in quasipolynomial time*, in Proceedings of the 48th annual ACM Symposium on Theory of Computing (STOC '16), 2016, pp. 684–697, https://doi.org/10.1145/2897518.2897542. See also arXiv:1512.03547 [cs.DS] and http://people.cs.uchicago.edu/~laci/upcc-fix.pdf.

[9] L. Babai, R. Beals, and Á. Seress, *Polynomial-time theory of matrix groups*, in Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09), 2009, pp. 55–64, https://doi.org/10.1145/1536414.1536425.

[10] L. Babai, P. Codenotti, and Y. Qiao, *Polynomial-time isomorphism test for groups with no abelian normal subgroups*, in Automata, Languages, and Programming (ICALP '12), 2012, pp. 51–62, https://doi.org/10.1007/978-3-642-31594-7_5.

[11] A. Blass and Y. Gurevich, *Equivalence relations, invariants, and normal forms*, SIAM Journal on Computing, 13 (1984), pp. 682–689, https://doi.org/10.1137/0213042.

[12] A. Blass and Y. Gurevich, *Equivalence relations, invariants, and normal forms. II*, in Logic and machines: decision problems and complexity (Münster, 1983), vol. 171 of Lecture Notes in Computer Science, Springer, Berlin, 1984, pp. 24–42, https://doi.org/10.1007/3-540-13331-3_31.

[13] M. Carmosino, R. Impagliazzo, V. Kabanets, and A. Kolokolova, *Learning algorithms from natural proofs*, in Proceedings of the 31st Computational Complexity Conference (CCC '16), 2016, pp. 10:1–10:24, https://doi.org/10.4230/LIPIcs.CCC.2016.10.

[14] J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences, 18 (1979), pp. 143–154, https://doi.org/10.1016/0022-0000(79)90044-8.

[15] P. Erdős and A. Rényi, *Probabilistic methods in group theory*, Journal d'Analyse Mathématique, 14 (1965), pp. 127–138, https://doi.org/10.1007/BF02806383.

[16] J. Finkelstein and B. Hescott, *Polynomial-time kernel reductions*. arXiv:1604.08558 [cs.CC], 2016.

[17] L. Fortnow and J. A. Grochow, *Complexity classes of equivalence problems revisited*, Information and Computation, 209 (2011), pp. 748–763, https://doi.org/10.1016/j.ic.2011.01.006.

[18] G. S. Frandsen and P. B. Miltersen, *Reviewing bounds on the circuit size of the hardest functions*, Information Processing Letters, 95 (2005), pp. 354–357, https://doi.org/10.1016/j.ipl.2005.03.009.

[19] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems*, Journal of the ACM, 38 (1991), pp. 691–729, https://doi.org/10.1145/116825.116852.

[20] O. Goldreich, A. Sahai, and S. Vadhan, *Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and NISZK*, in Advances in Cryptology — CRYPTO '99, 1999, pp. 467–484, https://doi.org/10.1007/3-540-48405-1_30.

[21] O. Goldreich and S. Vadhan, *Comparing entropies in statistical zero knowledge with applications to the structure of SZK*, in Proceedings of the 14th Annual IEEE Conference on Computational Complexity (CCC '99), 1999, pp. 54–73, https://doi.org/10.1109/CCC.1999.766262.

[22] O. Goldreich and S. Vadhan, *On the complexity of computational problems regarding distributions*, in Studies in Complexity and Cryptography – Miscellanea on the Interplay between Randomness and Computation, O. Goldreich, ed., vol. 6650 of Lecture Notes in Computer Science, Springer, 2011, pp. 13–29, https://doi.org/10.1007/978-3-642-22670-0_27.

[23] J. A. Grochow, *Matrix Lie algebra isomorphism*, in Proceedings of the 27th Annual IEEE Conference on Computational Complexity (CCC '12), 2012, pp. 203–213, https://doi.org/10.1109/CCC.2012.34.

[24] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby, *A pseudorandom generator from any one-way function*, SIAM Journal on Computing, 28 (1999), pp. 1364–1396, https://doi.org/10.1137/S0097539793244708.

[25] S. Hirahara and R. Santhanam, *On the average-case complexity of MCSP and its variants*, in Proceedings of the 32nd Computational Complexity Conference (CCC '17), 2017, pp. 7:1–7:20, https://doi.org/10.4230/LIPIcs.CCC.2017.7.

[26] S. Hirahara and O. Watanabe, *Limits of minimum circuit size problem as oracle*, in 31st Conference on Computational Complexity (CCC '16), vol. 50 of LIPIcs, 2016, pp. 18:1–18:20, https://doi.org/10.4230/LIPIcs.CCC.2016.18.

[27] D. F. Holt, B. Eick, and E. A. O'Brien, *Handbook of Computational Group Theory*, Discrete Mathematics and its Applications, Chapman & Hall/CRC, 2005.

[28] V. Kabanets and J.-Y. Cai, *Circuit minimization problem*, in Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC '00), 2000, pp. 73–79, https://doi.org/10.1145/335305.335314.

[29] W. M. Kantor and K. Magaard, *Black box exceptional groups of Lie type*, Transactions of the American Mathematical Society, 365 (2013), pp. 4895–4931, https://doi.org/10.1090/S0002-9947-2013-05822-9.

[30] W. M. Kantor and K. Magaard, *Black box exceptional groups of Lie type II*, Journal of Algebra, 421 (2015), pp. 524–540, https://doi.org/10.1016/j.jalgebra.2014.09.003.

[31] D. E. Knuth, *The Art of Computer Programming*, vol. 3: Sorting and Searching, Addison-Wesley, 1973.

[32] J. Köbler, U. Schöning, and J. Torán, *The Graph Isomorphism Problem: Its Structural Complexity*, Birkhauser Verlag, Basel, Switzerland, Switzerland, 1993, https://doi.org/10.1007/978-1-4612-0333-9.

[33] M. W. Liebeck and E. A. O'Brien, *Recognition of finite exceptional groups of Lie type*, Transactions of the American Mathematical Society, 368 (2016), pp. 6189–6226, https://doi.org/10.1090/tran/6534.

[34] C. D. Murray and R. R. Williams, *On the (non) NP-hardness of computing circuit complexity*, Theory of Computing, 13 (2017), pp. 1–22, https://doi.org/10.4086/toc.2017.v013a004.

[35] N. Nisan and A. Wigderson, *Hardness vs randomness*, Journal of Computer and System Sciences, 49 (1994), pp. 149–167, https://doi.org/10.1016/S0022-0000(05)80043-1.

[36] I. C. C. Oliveira and R. Santhanam, *Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness*, in 32nd Computational Complexity Conference (CCC '17), vol. 79 of LIPIcs, 2017, pp. 18:1–18:49, https://doi.org/10.4230/LIPIcs.CCC.2017.18.

[37] R. Paturi and P. Pudlák, *On the complexity of circuit satisfiability*, in Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC '10), 2010, pp. 241–250, https://doi.org/10.1145/1806689.1806724.

[38] E. Petrank and R. M. Roth, *Is code equivalence easy to decide?*, IEEE Transactions on Information Theory, 43 (1997), pp. 1602–1604, https://doi.org/10.1109/18.623157.

[39] M. Rudow, *Discrete logarithm and minimum circuit size*, Information Processing Letters, 128 (2017), pp. 1–4, https://doi.org/10.1016/j.ipl.2017.07.005.

[40] A. Sahai and S. Vadhan, *A complete problem for statistical zero knowledge*, Journal of the ACM, 50 (2003), pp. 196–249, https://doi.org/10.1145/636865.636868.

[41] A. Seress, *Permutation group algorithms*, vol. 152 of Cambridge Tracts in Mathematics, Cambridge University Press, Cambridge, 2003, https://doi.org/10.1017/CBO9780511546549.

[42] B. A. Trakhtenbrot, *A survey of Russian approaches to perebor (brute-force searches) algorithms*, IEEE Annals of the History of Computing, 6 (1984), pp. 384–400, https://doi.org/10.1109/MAHC.1984.10036.

[43] M. Yamamoto, *A tighter lower bound on the circuit size of the hardest Boolean functions*, Tech. Report TR11-086, Electronic Colloquium on Computational Complexity, 2011.