Electronic Colloquium on Computational Complexity, Report No. 46 (2023)

NP-Hardness of Approximating Meta-Complexity: A Cryptographic Approach

Yizhi Huang
Tsinghua University
huangyizhi01@gmail.com

Rahul Ilango
Massachusetts Institute of Technology
rilango@mit.edu

Hanlin Ren
University of Oxford
h4n1in.r3n@gmail.com

April 12, 2023

Abstract

It is a long-standing open problem whether the Minimum Circuit Size Problem (MCSP) and related meta-complexity problems are **NP**-complete. Even for the rare cases where the **NP**-hardness of meta-complexity problems are known, we only know very weak hardness of approximation.

In this work, we prove **NP**-hardness of approximating meta-complexity with nearly-optimal approximation gaps. Our key idea is to use *cryptographic constructions* in our reductions, where the security of the cryptographic construction implies the correctness of the reduction. We present both conditional and unconditional hardness of approximation results as follows.

- Assuming subexponentially-secure witness encryption exists, we prove essentially optimal **NP**-hardness of approximating conditional time-bounded Kolmogorov complexity ($\mathrm{K}^t(x \mid y)$) in the regime where $t \gg |y|$. Previously, the best hardness of approximation known was a $|x|^{1/\mathrm{poly}(\log\log|x|)}$ factor and only in the sublinear regime ($t \ll |y|$).
- Unconditionally, we show near-optimal **NP**-hardness of approximation for the Minimum Oracle Circuit Size Problem (MOCSP), where Yes instances have circuit complexity at most $2^{\epsilon n}$, and No instances are essentially as hard as random truth tables. Our reduction builds on a witness encryption construction proposed by Garg, Gentry, Sahai, and Waters (STOC'13). Previously, it was unknown whether it is **NP**-hard to distinguish between oracle circuit complexity s versus $10s\log N$.
- Finally, we define a "multi-valued" version of MCSP, called mvMCSP, and show that w.p. 1 over a random oracle O, mvMCSP^O is **NP**-hard to approximate under quasi-polynomial-time reductions with O oracle access. Intriguingly, this result follows almost directly from the security of Micali's CS proofs (Micali, SICOMP'00).

In conclusion, we give three results convincingly demonstrating the power of cryptographic techniques in proving **NP**-hardness of approximating meta-complexity.

ISSN 1433-8092

Contents

1 Introduction

Given an object (such as a string or a Boolean function), how hard is it to compute the "computational complexity" of this object? Such questions can be formalised by *meta-complexity* problems which aim to capture the "complexity of complexity" [All17]. A prominent example of a meta-complexity problem is the *Minimum Circuit Size Problem* (MCSP) [KC00]. In MCSP, one is given the length-2^n truth table of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ as well as a size parameter s, and the goal is to determine whether f can be computed by a circuit of size at most s.

Characterising the precise computational complexity of many meta-complexity problems, especially MCSP, remains elusive. It is easy to see that MCSP is in **NP** (simply guess a circuit of size at most s and check, by brute force,[1] that it computes the given truth table). On the other hand, building on the natural proofs framework [RR97, HILL99, GGM86], Kabanets and Cai [KC00] showed that if one-way functions exist, then MCSP is not in **P**. Therefore, MCSP is an intractable problem in **NP** under standard cryptographic assumptions. However, the question of whether MCSP is **NP**-complete remains wide open. Indeed, Levin is reported to have delayed publishing his theory of **NP**-completeness [Lev73] in hopes of showing MCSP is **NP**-complete.[2] Since then, there have been many works investigating whether MCSP and related problems are **NP**-complete (e.g., [KC00, ABK+06, AHM+08, Fel09, KS08, HP15, HW16, AHK17, MW17, HOS18, AIV19, AD17, IKV18, AH19, Ila20a, ILO20, Ila20b, SS20, Hir20, ACM+21, LP22, Hir22a, Hir22b]).

1.1 Why Care About NP-Hardness of Meta-Complexity?

Since we already know that MCSP and other meta-complexity problems are intractable under standard cryptographic assumptions, one may wonder what the motivation is for showing these problems are **NP**-hard. Perhaps surprisingly, researchers have discovered a growing number of important motivations for showing meta-complexity problems are actually **NP**-hard. We list some that we find compelling:

Eliminating Heuristica. *Heuristica* is the name Impagliazzo [Imp95] gives to a world where $\mathbf{P} \neq \mathbf{NP}$ but **NP** is easy on average. Unlike other complexity classes such as **EXP**, **PSPACE**, or $\mathbf{NC}^1$ [Bar89, FF93, BFNW93, TV07], there is no known worst-case to average-case reduction for **NP**. Indeed, there are barrier results against any **NP**-complete problem having a "black-box" worst-case to average-case reduction [FF93, BT06]. In a breakthrough result, Hirahara [Hir18] overcomes this barrier by giving a non-black-box worst-case to average-case reduction for approximating MCSP. If one could show this approximation version of MCSP is **NP**-hard, then this would imply that **NP** *does have* a worst-case to average-case reduction and thus rule out Heuristica.

Later work of Hirahara [Hir22b] further extends this result by showing that, to eliminate Heuristica, it suffices to show that a certain additive approximation to GapMINcKT (roughly speaking, a "conditional" version of meta-complexity) is **NP**-hard.

Basing one-way functions on $\mathbf{P} \neq \mathbf{NP}$. A longstanding goal in cryptography is to base the existence of one-way functions on worst-case assumptions such as $\mathbf{P} \neq \mathbf{NP}$ (or rather $\mathbf{NP} \not\subseteq \mathbf{BPP}$). Recently, an approach to showing this has emerged using meta-complexity [LP20, RS21, ACM+21, LP21b, LP21a, IRS22, LP22]. In a breakthrough paper, Liu and Pass [LP20] show that one-way functions exist if and only if time-bounded Kolmogorov complexity is mildly

[1]This guess and check are non-deterministically efficient since every Boolean function on n-bits has a trivial circuit of size $O(n2^n)$, which is polynomial size since we are given the length-2^n truth table as input.

[2]Allender and Das [AD17] cite a personal communication from Levin regarding this and a discussion can be found on Levin's webpage [Lev].

hard on average over the uniform distribution. As mentioned previously, Hirahara's worst-case to average-case reduction [Hir18] also holds for approximating time-bounded Kolmogorov complexity. Thus, if one "just" combines these two results and also shows that approximating time-bounded Kolmogorov complexity is **NP**-hard, then we would have that one-way functions exist if and only if $\mathbf{P} \neq \mathbf{NP}$. Unfortunately, the results of [Hir18] and [LP20] do not yet compose, as the types of average-case hardness that they consider are different ([Hir18] considered errorless heuristics while [LP20] considered error-prone heuristics).

Proving circuit lower bounds. Any reduction from SAT to MCSP needs to generate No instances of MCSP, which is equivalent to circuit lower bounds; therefore, **NP**-hardness of meta-complexity has a strong connection to circuit lower bounds. This argument was formalized by Kabanets and Cai [KC00], who show that if MCSP is **NP**-complete under "natural" reductions[3], then **E** does not have polynomial-size circuits. Note that this is a consequence that we believe but seems hard to show with current techniques.

We also mention an instance where new circuit lower bounds are proved along the way of pursuing the **NP**-hardness of meta-complexity. Ilango [Ila20b] showed that for every constant d there is a constant $\epsilon > 0$ and a function whose depth-d and depth-$(d+1)$ *formula* complexity are $2^{\epsilon n}$ apart. This follows from the techniques used to prove the **NP**-hardness of MCSP for constant-depth formulas; note that the standard switching lemma arguments [Hås86] are unable to prove such strongly-exponential ($2^{\Omega(n)}$) size lower bounds.

Curiosity. MCSP and its time-bounded Kolmogorov complexity variants are simple and important computational problems that have been studied since at least the 1960s [Tra84]. It is remarkable that despite this long history of study, these problems (unlike thousands of other problems) have thoroughly eluded attempts at classifying their complexity (in particular, completeness for some natural complexity class). Indeed, we lack compelling evidence either for or against the existence of a polynomial-time mapping reduction from SAT to MCSP or many other meta-complexity problems. The situation is especially lacklustre when considering hardness of approximation. Essentially no **NP**-hardness is known for any even moderately strong model (e.g. depth-3 formulas) beyond logarithmic factors[4] in the truth table [KS08, Ila20b, Hir22b]. Are these problems **NP**-complete or not? Are they **NP**-hard to approximate or not?

1.2 Can Cryptography Help?

The starting point of our work is the following question:

Can *cryptography* be useful in showing the **NP**-*hardness* of meta-complexity?

In some sense, prior work already shows that the answer to this question is yes. For example, a trivial corollary of Kabanets and Cai [KC00] is that if one-way functions exist, then MCSP $\in \mathbf{P}$ if and only if $\mathbf{P} = \mathbf{NP}$. One can view this as a kind of **NP**-completeness result, but the proof is somewhat unsatisfying: if one-way functions exist, then both $\mathbf{P} \neq \mathbf{NP}$ and MCSP $\notin \mathbf{P}$.

Another (more satisfying) example is a result by Impagliazzo, Kabanets, and Volkovich [IKV18], who show that if indistinguishability obfuscation ($i\mathcal{O}$) exists, then MCSP $\in \mathbf{ZPP}$ if and only if $\mathbf{NP} = \mathbf{ZPP}$. Their proof can be viewed as a *non-black box* reduction from SAT to MCSP. However, one drawback is that assuming $i\mathcal{O}$ exists is very close to assuming that

[3]That is, deterministic reductions whose output length and numerical parameters only depend on the input length (instead of the particular input), and the sizes of the inputs and the outputs are polynomially related. Almost all known **NP**-complete problems are **NP**-hard under "natural" reductions.

[4]The only exception to this that we are aware of is Hirahara's recent result that it is **NP**-hard to compute an $n^{1/\text{poly}\log\log n}$ factor approximation to the conditional time-bounded Kolmogorov complexity. But even this is in a weaker sublinear-time model.

one-way functions exist. In particular, if $i\mathcal{O}$ exists and **NP** is not in **BPP** infinitely often, then one-way functions exist [KMN+14].

Thus, while these results are interesting, in both cases it is somewhat unclear what the takeaway should be. Do these results really suggest that MCSP is **NP**-hard, or rather perhaps just that MCSP is intractable based on plausible cryptographic and complexity-theoretic assumptions?

To address this, one can refine the original question.

Can *cryptography* be useful in showing *black-box* **NP**-*hardness* of meta-complexity?

Here by a black-box reduction, we mean showing, for example, that one can solve SAT in polynomial time given an oracle to MCSP. Such a result would constitute perhaps the strongest evidence yet that MCSP is indeed **NP**-complete under the usual definition of **NP**-completeness.

It may seem counter-intuitive that cryptography could be helpful in proving black-box **NP**-completeness results. While the existence of one-way functions implies that problems like MCSP are intractable [RR97, HILL99, KC00], it is not at all clear how to turn this into a black-box reduction from say SAT to MCSP.[5]

Intriguingly, a recent breakthrough result by Hirahara [Hir22a] uses tools from information-theoretic cryptography, such as secret sharing schemes and one-time encryption schemes, to show the **NP**-hardness of many important meta-complexity problems. Indeed, Hirahara's result convincingly demonstrates the power of information-theoretic cryptography for proving **NP**-hardness of meta-complexity problems.

In this paper, we focus on notions from *computational* cryptography, instead of information-theoretic cryptography. There is a natural intuition for why such cryptography could be useful: it gives *structured computational hardness* one could hope to exploit. In more detail, one potential reason it is difficult to prove the **NP**-hardness of MCSP is that we lack strong enough circuit lower bounds. Indeed, just deterministically generating a No instance of MCSP requires proving circuit lower bounds! It is hard to imagine proving an **NP**-hardness result when we cannot even generate an explicit No instance. Moreover, this argument is made formal by several works [KC00, MW17, HP15, SS20], who showed that **NP**-hardness of MCSP under certain types of reductions would imply breakthrough separations in complexity theory such as $\mathbf{EXP} \not\subseteq \mathbf{P}_{/\mathrm{poly}}$.

Thus, since **NP**-hardness of MCSP (at least in some settings) implies circuit lower bounds, it is natural to wonder whether we can go in the opposite direction: are circuit lower bounds *sufficient* for the **NP**-hardness of meta-complexity problems? So far the answer appears to be *no*. For example, we have subexponential-size lower bounds against $\mathbf{AC}^0$ [Ajt83, FSS84, Yao85, Hås86] and $\mathbf{AC}^0[p]$ where p is a prime [Raz87, Smo87, Smo93], but the **NP**-hardness of $\mathbf{AC}^0$-MCSP and $\mathbf{AC}^0[p]$-MCSP remain important open problems.[6] Apparently, to show that MCSP is **NP**-complete, one needs hardness with some "structure." Can cryptography give such structured hardness?

1.3 Our Results

We show three main results, each one using a cryptographic construction (i.e. JLS's indistinguishability obfuscation[7] [JLS21], GGSW's witness encryption [GGSW13], or Micali's CS proofs [Mic00]) to get either a conditional or an unconditional **NP**-hardness result in meta-complexity.

[5]One potential way of doing this is to show that there is a one-way function that is **NP**-hard to invert. But, as discussed earlier, constructing such a one-way function remains a major open question.

[6]Ilango [Ila20b] showed that the *formula* version of $\mathbf{AC}^0$-MCSP is **NP**-hard under quasi-polynomial-time randomised Turing reductions, but the *circuit* versions of $\mathbf{AC}^0$-MCSP is not known to be **NP**-hard [CHI+21]. Prior to these results, the largest circuit class $\mathscr{C}$ for which **NP**-hardness of $\mathscr{C}$-MCSP was known is only $\mathbf{DNF} \circ \mathbf{XOR}$ [HOS18].

[7]More specifically, we use that the JLS construction implies the existence of witness encryption from well-founded assumptions.

Moreover, our results imply **NP**-hardness of approximation with *near-optimal approximation gaps.* In our view, the central conceptual takeaway from our results is a strongly positive answer to the question above:

> Cryptography is indeed a powerful tool for showing black-box **NP**-hardness of meta-complexity!

1.3.1 Witness Encryption and Conditional Time-Bounded Kolmogorov Complexity

The *t-time-bounded Kolmogorov complexity* of a string $x \in \{0,1\}^n$, denoted $\mathrm{K}^t(x)$, is the minimum length of any program that outputs x in time at most t [Kol65, Sip83, Ko86]. Similarly, the *conditional t-time-bounded Kolmogorov complexity* of a string $x \in \{0,1\}^n$ given a string $y \in \{0,1\}^m$, denoted $\mathrm{K}^t(x \mid y)$, is the minimum length of any program that outputs x in time t when given oracle access to y. (See Section 2.4 for formal definitions of $\mathrm{K}^t(\cdot)$ and $\mathrm{K}^t(\cdot \mid \cdot)$.)

In a recent work, Hirahara [Hir22b] shows that it is **NP**-hard to approximate $\mathrm{K}^t(x \mid y)$ to a factor of $n^{1/\mathrm{poly}\log\log n}$. This improves on prior work, which could only show an $O(\log n)$ factor hardness of approximation for conditional time-bounded Kolmogorov complexity and related problems [Ila20a, ACM$^+$21, LP22].

In all of the above **NP**-hardness results, the instances of $\mathrm{K}^t(x \mid y)$ are in the *sublinear* time regime, i.e. where $t \ll |y|$ and thus one does not even have enough time to read all the bits of y. Intriguingly, Hirahara [Hir22b] shows that if one could improve these **NP**-hardness results to show a certain additive hardness of approximation in the *superlinear* regime where $t \gg |y|$ (so one has time to read all of y), then this would eliminate Heuristica!

This strongly motivates understanding the complexity of conditional time-bounded Kolmogorov complexity in the superlinear regime. Should we expect this problem to be **NP**-hard? Even if it is, is it **NP**-hard in the rather specific approximation regime Hirahara needs?

We show that, conditioned on a widely believed cryptographic assumption, this problem is indeed **NP**-hard with essentially optimal hardness of approximation.

Theorem 1.1 (Informal version of Theorem 3.1)**.** *Assume subexponentially-secure witness encryption exists. Then the following promise problem is* **NP***-hard under randomized polynomial-time (black-box) reductions: given strings* (x, y) *where* $|x| = n$ *and* $|y| = \mathrm{poly}(n)$*, output*

- YES *if* $\mathrm{K}^{\mathrm{poly}(n)}(x \mid y) \leq n^{.01}$*;*
- NO *if* $\mathrm{K}^{2^{n^2}}(x \mid y) \geq n - O(1)$*.*

We will discuss the notion of witness encryption and its plausibility in a few paragraphs, but before we do that we make some remarks about this theorem. First, we emphasize that, under the assumption, we get a standard, black-box, randomized many-one reduction from **NP** to the promise problem stated above.

Next, we note that the gap in Theorem 1.1 is essentially maximal **NP**-hardness of approximation. The complexity of the YES instances is at most $n^{.01}$ and the constant .01 can be made arbitrarily small (one cannot hope for YES instances with complexity subpolynomial in n without giving a subexponential time algorithm for SAT). On the other hand, the NO instances have complexity at least $n - O(1)$, which is an additive constant away from the maximum complexity of any n-bit string. Moreover, the gap in the time bound is extremely large: $\mathrm{poly}(n)$ in the YES case versus 2^{n^2} in the NO case. (In fact, the n^2 in the exponent can be made into an arbitrary polynomial in n; see Theorem 3.1 for details.)

Finally, we return to Hirahara's approach to eliminating Heuristica. Despite the strong hardness of approximation Theorem 1.1 gives, it does not give the hardness of approximation needed to eliminate Heuristica. The precise reason is somewhat technical (we refer a curious reader to Section 3 for the details). At a high level, the reason is that the specific additive hardness of approximation Hirahara needs has a somewhat non-standard dependence on the instance $(x \mid y)$, in particular on the "computational depth" of y. The upshot of this is that one needs to give hardness of approximation on instances $(x \mid y)$, where y has low computational depth. It is unclear whether the instances produced by the reduction in Theorem 3.1 have this property.

Nevertheless, we overcome this, under a further assumption. Assuming the existence of subexponentially secure injective one-way functions, we can modify the reduction in Theorem 1.1 so that with high probability an output $(x \mid y)$ of the reduction will have a y with low computational depth.

Theorem 1.2 (Informal version of Corollary 3.7)**.** *Assume subexponentially secure injective one-way functions and subexponentially secure witness encryption exists. Then the promise problem, whose* **NP***-hardness was shown to exclude Heuristica by Hirahara [Hir22b], is in fact* **NP***-hard (under randomized polynomial-time many-one reductions).*

We find Theorem 1.2 rather surprising. One can interpret this result as saying that, under widely believed assumptions in cryptography, Hirahara's approach to eliminating Heuristica *provably* works! Of course, for the purpose of eliminating Heuristica this Theorem 1.2 by itself is not so interesting since if subexponentially secure one-way functions exist, then **NP** is (automatically) hard on average. Even so, we find this result enlightening, especially because the "ground truth" of whether this problem was in fact **NP**-hard was not at all clear.

Before we continue, we discuss the notion of witness encryption informally (see Section 2 for a formal definition). Introduced by Garg, Gentry, Sahai and Waters [GGSW13], witness encryption is a cryptographic primitive that encrypts a message using a (public) instance of some **NP**-complete language. Let φ be a formula (for example, any satisfying assignment of φ is a proof of Riemann Hypothesis of at most 10,000 pages long). We can encrypt a secret message m (e.g., a Bitcoin address for a prize awarded to whoever proves Riemann Hypothesis) using φ such that:

1. If φ is satisfiable, then any party with a satisfying assignment of φ (e.g., any mathematician with a valid proof of Riemann Hypothesis) can decrypt the message in polynomial time, and

2. if φ is unsatisfiable, then the encryption of two different messages should be computationally indistinguishable.

Witness encryption turns out to be a very powerful primitive. It was shown in [GGSW13] that witness encryption can be used to build public-key encryption [DH76, GM84], Identity-Based Encryption [Sha84, BF03], and Attribute-Based Encryption [SW05] for circuits. The witness encryption in [GGSW13] also yielded the first candidate for Rudich-type secret sharing scheme [Bei11, KNY17]. In this work, we show an unexpected application of witness encryption in complexity theory: it implies the **NP**-hardness of meta-complexity problems!

Finally, we discuss the plausibility of witness encryption. Subexponentially secure witness encryption is implied [GGH$^+$16] as a special case by the existence of subexponentially secure indistinguishability obfuscators ($i\mathcal{O}$) [BGI$^+$12]. In a recent breakthrough paper, Jain, Lin, and Sahai [JLS21] show that subexponentially secure[8] $i\mathcal{O}$ exists assuming four standard "well-founded" assumptions (later work of Jain, Lin, and Sahai reduced this to three assumptions [JLS22b]). As a result, the existence of the witness encryption used in Theorem 1.1 has now become a widely-believed assumption.

[8]We note that the definition of subexponentially secure that we need is slightly different from the one explicitly used by Jain, Lin, and Sahai [JLS22b], although their result readily generalizes to our definition [JLS22a]. See Remark 2.5 for a detailed discussion of this.

We also remark that witness encryption is a plausibly weaker assumption than $i\mathcal{O}$. However, the only known constructions (from well-founded assumptions) of subexponentially-secure witness encryption are from $i\mathcal{O}$ (see the recent paper of Vaikuntanathan, Wee, and Wichs [VWW22] for a discussion of this).

1.3.2 Oracle Witness Encryption and MOCSP

Our second result is about MOCSP, the conditional variant of MCSP. In MOCSP, we are given a truth table of a function $f : \{0,1\}^n \to \{0,1\}$ as well as a truth table of an oracle $O : \{0,1\}^{O(n)} \to \{0,1\}$, and we are asked to compute the minimum size of any oracle circuit $C : \{0,1\}^n \to \{0,1\}$ that computes f with oracle access to O. Ilango [Ila20a] showed that MOCSP is **NP**-hard to approximate to roughly a logarithmic factor in the input length. Uncertain as to whether or not current techniques could prove stronger hardness of approximation, Ilango left as an open question to either show an N^ϵ factor hardness of approximation for MOCSP for some constant $\epsilon > 0$, where N is the length of the input to MOCSP, or to show a barrier against proving such strong inapproximability results [Ila20a, Open Question 1.5].

We resolve this open question by *unconditionally* showing that MOCSP is **NP**-hard to approximate with a very large approximation factor. In what follows, we denote by $\mathsf{CC}^O(f)$[9] the size of the minimum O-oracle circuit that computes f.

Theorem 1.3 (Informal version of Corollary 4.8)**.** *For any $\epsilon > 0$, the following promise problem is* **NP***-hard under polynomial-time randomised mapping reductions: given a truth table f of length ℓ and an oracle truth table O of length* $\mathrm{poly}(\ell)$*, distinguish between the following two cases:*

(Yes instances) $\mathsf{CC}^O(f) \le \ell^\epsilon$*;*

(No instances) $\mathsf{CC}^O(f) \ge \ell^{1-\epsilon}$*.*

A similar result also holds for GapMINcKT, the problem of approximating conditional time-bounded Kolmogorov complexity; see Theorem 4.2 for details. (In contrast to Theorem 1.1, this result only holds in the sublinear time-bounded regime). In addition, we also proved the **NP**-hardness of MOCSP where Yes instances are *exactly* computable by small circuits, but No instances are *average-case hard* against larger circuits; see Theorem 4.1 for details.

Before we discuss the proof techniques, we comment a bit more on the problem MOCSP. As Ilango [Ila20a] suggested, MOCSP is a nice "testing ground" for hardness results we conjecture for MCSP. Similar to MCSP, MOCSP is also in **NP**; it is easy to see that MOCSP is no easier than MCSP. And it is also pointed out by [Ila20a] that, essentially the same proof as in [MW17] shows that if MOCSP is **NP**-hard under *deterministic* polynomial-time reductions, then **EXP** $\neq$ **ZPP**. We will see another example of MOCSP being a "testing ground" for MCSP later (Theorem 1.6). We hope that our results shed some light on the complexity of MCSP.

Perhaps surprisingly, the key idea underlying our proof is again *witness encryption*, despite our proof being *unconditional*. In more detail, our proof utilises the notion of witness encryption *in oracle worlds*, where both the encryption and decryption algorithms have access to an oracle. We show that exponentially-secure witness encryption exists, *unconditionally*, in a carefully constructed oracle world. We also show that such a secure oracle witness encryption scheme implies Theorem 1.3: roughly speaking, we map a formula φ to a function f and an oracle O, where O contains the oracle world as well as a lot of ciphertexts encrypted using φ. If φ is satisfiable, then a small circuit with a satisfying assignment hardcoded can compute f from these ciphertexts easily; if φ is unsatisfiable, then any small circuit computing f would violate the security of witness encryption.

[9] CC stands for *circuit complexity*.

We now discuss how we construct a witness encryption scheme in oracle worlds. A natural approach is to consider candidate witness encryption schemes in the literature and build oracles that make them secure. Fortunately, the original candidate proposed by [GGSW13] already suffices. As this candidate uses multilinear maps [BS03, GGH12], we replace it with an oracle implementing the *generic multilinear map model* (which is the multilinear map version of the generic group model [Sho97]). It turns out that the security of [GGSW13] construction is provable in the generic multilinear map model! See Sections 4.3 and 4.4 for details.

A lesson from this result is that unconditional security results in idealised models are not only heuristic arguments that certain cryptographic protocols "seem secure"; they also have (rigorous) implications in complexity theory.

One last aspect we find interesting and worth noting is that, unlike previous results (e.g., [Ila20a, Hir22a]), our proof of Theorem 1.3 does not rely on the PCP theorem [AS98, ALM+98]. Nevertheless, we obtain much stronger hardness of approximation results! The construction in [GGSW13] works directly for the **NP**-complete language Exact-Cover [Kar72], so our result is also a direct reduction from Exact-Cover to GapMOCSP. This is in contrast to previous results (e.g., [Ila20a, Hir22a]) that need to start with a hardness-of-approximation result (e.g., set cover [DS14] or the Minimum Monotone Satisfying Assignment problem [DS04, ABMP01]), which relies on the PCP theorem.

1.3.3 CS Proofs and A Multi-Valued Version of MCSP with Random Oracles

Our third result is about a "multi-valued" version of MCSP, which we denote as mvMCSP. In mvMCSP, we are given the truth table of a "multi-valued" function $f \subseteq \{0,1\}^n \times \{0,1\}^m$, where for each input $x \in \{0,1\}^n$, any $y \in \{0,1\}^m$ such that $(x, y) \in f$ is a valid output. The goal is to compute the size of the smallest circuit $C : \{0,1\}^n \to \{0,1\}^m$ that computes f, i.e.,

$$\forall x \in \{0,1\}^n, (x, C(x)) \in f.$$

Let $k \geq 1$ be a constant, Gap_k-mvMCSP denotes the following promise problem: given the length-2^{n+m} truth table of a "multi-valued" function $f \subseteq \{0,1\}^n \times \{0,1\}^m$ and a parameter s, distinguish between the following two cases:

(Yes instances) there is a circuit C of size s such that

$$\Pr_{x \leftarrow \{0,1\}^n}[(x, C(x)) \in f] = 1;$$

(No instances) for any circuit C of size s^k,

$$\Pr_{x \leftarrow \{0,1\}^n}[(x, C(x)) \in f] \leq 1/s^k.$$

Theorem 1.4. *For every constant $k > 1$, with probability 1 over a random oracle O, the problem $\mathrm{Gap}_k\text{-mvMCSP}^O$ is* **NP***-hard under* $\mathbf{TIME}[2^{\mathrm{polylog}(n)}]^O$ *(deterministic quasi-polynomial time with an O oracle) mapping reductions.*

Perhaps intriguingly, Theorem 1.4 essentially follows from the security of Micali's CS proofs [Mic00] in the random oracle model. We think this is the interesting aspect of Theorem 1.4, as it illustrates the connection between cryptography and **NP**-hardness of meta-complexity in a *direct and straightforward* way.

We now describe this connection in more detail. Let $L \in \mathbf{NP}$. An *argument system* for L involves a prover and a verifier, where both parties know an instance $x \overset{?}{\in} L$ and the prover wants to convince the verifier that $x \in L$. If x is indeed in L, then an efficient prover (with a witness of $x \in L$) could convince the verifier with certainty; if $x \notin L$, then any prover of a certain size could only convince the verifier with small probability.

If one looks carefully at this definition, one realises that this is nothing but a reduction from L to a "meta-complexity" problem! In particular, this is a "meta-complexity" problem about the complexity of convincing the verifier. If $x \in L$, then this complexity should be small, while if $x \notin L$, then this complexity should be large. Therefore, if every language in **NP** admits an argument system (of some kind), then some meta-complexity problem (related to this argument system) is **NP**-complete. This is exactly what happens in Theorem 1.4: since every problem in **NP** has a SNARG (succinct non-interactive argument) in the random oracle model [Mic00], a certain meta-complexity problem should be **NP**-complete. When we work out the definition of this meta-complexity problem, it becomes exactly mvMCSP.

Moreover, this approach gives us **NP**-hardness of approximation with "the largest gap possible". If $x \in L$, then the complexity of "convincing the verifier" is a fixed polynomial of $|x|$, since the prover essentially needs to hardwire a witness for x; if $x \notin L$, then by the security of the argument system, the complexity of "convincing the verifier" can be made arbitrarily large (by adjusting the security parameter).

This idea also shows that if (subexponentially-secure) SNARGs exist (in the unrelativised world), then mvMCSP is **NP**-hard to approximate.

Corollary 1.5. *Suppose that subexponentially-secure SNARGs exist. Then for every $k \in \mathbb{N}$,* Gap_k-mvMCSP *is* **NP***-hard under deterministic quasi-polynomial time reductions.*

1.3.4 Applications

Using the ideas developed in this paper, we also make progress on two other problems: pseudorandom self-reductions for **NP**-complete languages and heuristics for Complexity.

Pseudorandom self-reductions for NP-complete languages. In 2017, Hirahara and Santhanam [HS17] observed that if exponentially-hard one-way functions exist, then MCSP admits a *pseudorandom self-reduction*: a self-reduction that maps a worst-case instance to a distribution that is indistinguishable from the uniform distribution. In contrast, if **PH** does not collapse, then **NP**-complete problems do not admit (non-adaptive) *random* self-reductions [BT06]. Hirahara and Santhanam viewed this result as a property that "distinguishes the MCSP problem from natural **NP**-complete problems" [HS17].

Perhaps surprisingly, Elrazik, Robere, Schuster, and Yehuda [ERSY22] recently gave evidence *against* this. They show natural **NP**-complete problems that also plausibly admit pseudorandom self-reductions. In particular, under a non-uniform version of the Planted Clique Conjecture, the Clique problem admits a non-adaptive pseudorandom self-reduction. There might be some property that distinguishes MCSP from natural **NP**-complete problems, but having pseudorandom self-reductions is not one of them!

One weakness of the results in [ERSY22] is that they need to assume the Planted Clique Conjecture, which is much stronger than the existence of one-way functions. Moreover, the Planted Clique problem can be solved in $n^{O(\log n)}$ time, which means their distributions are not pseudorandom against adversaries of quasi-polynomial size.

Our **NP**-hardness results on MOCSP allow us to achieve the best of both worlds: assuming the existence of one-way functions, there is an **NP**-complete problem with pseudorandom self-reductions.

Theorem 1.6 (Informal Version of Theorem 6.2)**.** *If one-way functions exist, then there is an* **NP***-complete problem (namely* GapMOCSP*) that admits pseudorandom self-reductions.*

We remark that **NP**-hardness of *approximation* is crucial for this application, as our self-reduction blows up the circuit complexity of the input truth tables in MOCSP by a factor of

$N^{\Omega(1)}$, where $N = 2^n$ is the length of the input truth table. (The **NP**-hardness of approximating Clique [Hås96, Zuc07] is also crucial to the results in [ERSY22].)

Heuristics for Complexity. The Complexity problem [KKMP21] asks the following: given the truth table of an oracle O, find a truth table f such that the O-oracle circuit complexity of f is large. A random truth table is always hard (with high probability) even in the presence of a fixed oracle O, so there is a trivial randomised algorithm solving Complexity. On the other hand, a deterministic algorithm for Complexity, even only for the case that O is the all-zero truth table, implies a circuit lower bound for **E**. Thus deterministic algorithms solving Complexity are of great interest.

We consider *deterministic heuristics* for this problem. We say a deterministic algorithm $\mathcal{A}$ is a *heuristic* for Complexity under the uniform distribution if

$$\Pr_O[\mathsf{CC}^O(f) > 2^n/10n \mid f = \mathcal{A}(O)] \geq 1 - o(1),$$

where f is a truth table of length 2^n.

Inspired by the **NP**-hardness of Gap-mvMCSP^O for a random oracle O, we design an *unconditional* heuristic for Complexity:

Theorem 1.7 (Informal Version of Theorem 6.4)**.** *There is,* unconditionally*, a deterministic heuristic for* Complexity *in certain parameter regimes.*

The idea is simple: if O is a uniformly random input, then solving Complexity means proving circuit lower bounds *in the random oracle model.* Therefore, we can take any proof that **E** requires large circuits relative to a random oracle, and turn it into a heuristic for Complexity. In fact, our construction is extremely simple: Suppose $O : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is a random oracle over $2n$ bits, then the function $f : \{0,1\}^n \to \{0,1\}$ is defined as

$$f(x) = \bigoplus_{y \in \{0,1\}^n} O(x, y).$$

It is not hard to show that for a random oracle O, the O-oracle circuit complexity of f is exponential.

1.4 Related Work

For a general survey of meta-complexity, we point the reader to Allender's recent surveys [All17, All21] and the references therein. Below we discuss the prior works that are mostly related to our results.

NP-hardness of meta-complexity problems. Related to Theorems 1.1 and 1.3, several **NP**-hardness results have been shown for conditional meta-complexity problems. Ilango [Ila20a] introduced the problem MOCSP and proved that MOCSP is **NP**-hard. Allender, Cheraghchi, Myrisiotis, Tirumula, and Volkovich [ACM$^+$21] proved the **NP**-hardness of McKTP, the problem of computing conditional KT-complexity,[10] and Liu and Pass [LP22] showed that MINcKT, the problem of computing conditional time-bounded Kolmogorov complexity, is **NP**-complete. In all three aforementioned results, the hardness of approximation given is relatively weak, namely at most a logarithmic factor. This logarithmic factor arises because the reductions begin from set cover, where a logarithmic factor is optimal [Fei98, DS14]. A recent exciting work by Hirahara [Hir22b] greatly improves the hardness of approximation known for MINcKT, showing it is **NP**-hard to approximate to a $n^{1/\text{poly}\log\log n}$ factor.

[10]The KT-complexity is a notion of resource-bounded Kolmogorov complexity defined in [All01, ABK$^+$06].

We now discuss work related to Theorem 1.4. Ilango, Loff, and Oliveira [ILO20] showed that Multi-MCSP is **NP**-hard under randomised reductions. Here, Multi-MCSP is the problem of computing the circuit complexity of a multi-output function. It is easy to see that Multi-MCSP reduces to mvMCSP. In [ILO20], the number of output bits of the function is *exponential* in the number of input bits, but the hard function is fixed (i.e., any input corresponds to a unique output). On the other hand, in mvMCSP, the number of output bits is only polynomially larger than the number of input bits, but there might be many valid outputs for each input. Thus the two results are not directly comparable.

Hirahara [Hir22a] proved that MCSP* is **NP**-hard under randomised reductions. Here, MCSP* is the problem of computing the circuit complexity of a *partial* truth table. Since MCSP* reduces to mvMCSP, it follows that mvMCSP is also **NP**-hard under randomised reductions.

However, we emphasise that our **NP**-hardness results hold for *very large* approximation gaps: the Yes instances are computable in size s, while the No instances are inapproximable by size $2^{\mathrm{polylog}(s)}$. The results in [ILO20] only proved the **NP**-hardness of approximating Multi-MCSP within a small additive factor, and the results in [Hir22a] only proved the **NP**-hardness of approximating MCSP* within a multiplicative factor of n^α for some constant $\alpha < 1$.

Using cryptography to prove hardness of meta-complexity. It is already known from Kabanets and Cai [KC00] that we can use an MCSP oracle to invert any candidate one-way function. By building concrete (auxiliary-input) one-way function candidates, it was shown that MCSP is hard for discrete logarithm [ABK+06, Rud17], graph isomorphism [AGvM+18], and actually the whole class **SZK** [AD17].

Impagliazzo, Kabanets, and Volkovich [IKV18] show that, assuming indistinguishability obfuscation exists, we have that **NP** = **ZPP** if and only if MCSP $\in$ **ZPP**. We stress that this is a logical equivalence, not a black-box reduction. We also note that assuming strong cryptographic objects like indistinguishability obfuscation exist is very close to assuming MCSP $\notin$ **ZPP** (since if one-way functions do exist, then MCSP is not in **ZPP**).

As we mentioned previously, Hirahara's recent **NP**-hardness results for MCSP* [Hir22a] and MINcKT [Hir22b] utilizes secure secret sharing schemes and one-time encryption schemes, tools from *information theoretic* cryptography. In contrast, our results utilize cryptographic objects, such as witness encryption, that are *computationally secure* either based on computational assumptions or relative to a specifically designed oracle. Interestingly, witness encryption is equivalent to a computational version of secret sharing [KNY17], but it is unclear if there is a unifying framework behind Hirahara's results and our results. We leave this intriguing question for future research.

A concurrent work by Hirahara [Hir23] presents a meta-complexity problem called GapMdKP whose **NP**-hardness characterizes one-way functions, giving further connection between cryptography and the **NP**-hardness of meta-complexity.

Finally, Allender and Hirahara [AH19] showed that under cryptographic assumptions, a gap version of MCSP is **NP**-intermediate (i.e., neither in **P** nor **NP**-hard). However, the gap they consider is so large that if their version of GapMCSP were **NP**-hard, then SAT would be in subexponential time.

1.5 Discussions on Barriers Results

There are mainly two barriers to showing **NP**-hardness of meta-complexity problems: relativisation [Ko91] and oracle independence [HW16].

Ko [Ko91] showed that any **NP**-hardness result for MINLT (which is some meta-complexity problem that we do not define here) must be non-relativising. This relativisation barrier was overcome by [Hir22a] using non-relativising techniques such as the PCP theorem. Our results are also non-relativising:

- Due to the use of *cryptographic assumptions*, Theorem 1.1 could show consequences that might be impossible to prove unconditionally in a relativising way. However, the proof of Theorem 1.1 (that witness encryption implies **NP**-hardness of MINcKT) is relativising.

- Theorem 1.3 uses the non-relativising fact that Exact-Cover is **NP**-complete. Indeed, the main technical ingredient of Theorem 1.3 is a witness encryption scheme for Exact-Cover.

- Theorem 1.4 uses the PCP theorem, which is non-relativising. We also note that Theorem 1.4 does *not* show that mvMCSP^O is $\mathbf{NP}^O$-complete, as we could only reduce **NP** (instead of $\mathbf{NP}^O$) to mvMCSP^O.

It was observed in [HW16] that most reductions (at their time) to MCSP are *oracle-independent*, i.e., they also work for MCSP^A for *every* oracle A. Then, [HW16] showed that under plausible assumptions, **NP**-hardness of MCSP cannot be established via oracle-independent reductions. Hirahara's results [Hir22a] are subject to this barrier since they showed the **NP**-hardness of $(\text{MKTP}^\star)^A$ for every oracle A.

Unfortunately, Theorems 1.3 and 1.4 are also subject to this barrier.[11] In particular, to prove the soundness of our reduction (i.e., the No instances we generated are indeed No instances), we proved strong circuit lower bounds in certain oracle worlds $\mathcal{O}$. These lower bounds hold for not only $\mathcal{O}$-oracle circuits, but also *programs* of bounded query complexity to $\mathcal{O}$ (and possibly unbounded time). Therefore, for every fixed (additional) oracle A, these lower bounds also extend to A-oracle circuits. It is a very intriguing question to obtain **NP**-hardness of (approximating) meta-complexity problems via reductions that are not oracle-independent.

2 Preliminaries

We assume the reader is familiar with basic complexity-theoretic and cryptographic notions, which can be found in e.g. Arora and Barak's textbook [AB09] and Goldreich's textbook [Gol01] respectively.

We use $\mathcal{U}_n$ to denote the uniform distribution over $\{0,1\}^n$.

Oracle circuits. The size of an (oracle) circuit is the number of wires in the circuit. It is a well-known fact that every oracle circuit of size S (with a single type of oracle gate) can be described in $3S \log S$ bits, thus there are at most $S^{O(S)}$ many circuits of size S.

For an oracle O and a Boolean function or relation f, $\mathsf{CC}^O(f)$ is the size of the minimum O-oracle circuit that computes f, and $\mathsf{CC}^O_\delta(f)$ the size of the minimum O-oracle circuit that computes f correctly on $1-\delta$ fraction of inputs.

2.1 Witness Encryption

Definition 2.1 (Witness Encryption [GGSW13])**.** Let $L \in \mathbf{NP}$, R be the witness relation for L. A *witness encryption scheme* for L consists of polynomial-time algorithms $(\mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Encrypt}(1^\lambda, x, b; r)$ takes as input a security parameter 1^λ, an instance x, a message bit $b \in \{0,1\}$, and some randomness r; it outputs a ciphertext c.

- $\mathsf{Decrypt}(1^\lambda, c, x, w)$ takes as input a security parameter 1^λ, a ciphertext c, an instance x, and a witness w such that $(x,w) \in R$; it operates *deterministically* and outputs a message bit b.

[11] This barrier does not apply to Theorem 1.1 due to the use of cryptographic assumptions.

These algorithms satisfy the following two conditions:

(Correctness) For any security parameter λ, any $b \in \{0,1\}$, any $x \in L$ and any x, w such that $(x,w) \in R$, we have that

$$\Pr_r[\mathsf{Decrypt}(\mathsf{Encrypt}(1^\lambda, x, b; r), x, w) = b] = 1.$$

(Security) Let $S : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to (0,1)$ be functions. We say the witness encryption scheme is secure against size-S adversaries with advantage ϵ, if for every security parameter λ, every size-$S(\lambda)$ circuit A and any $x \notin L$,

$$\left|\Pr_r[A(\mathsf{Encrypt}(1^\lambda, x, 0; r)) = 1] - \Pr_r[A(\mathsf{Encrypt}(1^\lambda, x, 1; r)) = 1]\right| < \epsilon(\lambda).$$

We say a witness encryption scheme is *subexponentially secure* if it secure against $S(\lambda) = O(2^{\lambda^\delta})$-sized circuits with advantage $\epsilon(\lambda) = O(2^{-\lambda^\delta})$ for some $\delta > 0$.

We will also consider the case where one wants to witness-encrypt strings.

Definition 2.2 (Semantically Secure Multi-bit Witness Encryption)**.** Let $L \in \mathbf{NP}$ and R be the witness relation for L. A *semantically secure multi-bit witness encryption scheme* for L consists of polynomial-time algorithms $(\mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Encrypt}(1^\lambda, x, m; r)$ takes as input a security parameter 1^λ, an instance x, a message $m \in \{0,1\}^n$, and some randomness r; it outputs a ciphertext c.
- $\mathsf{Decrypt}(1^\lambda, c, x, w)$ takes as input a security parameter 1^λ, a ciphertext c, an instance x, and a witness w such that $(x,w) \in R$; it operates *deterministically* and outputs a message m.

These algorithms satisfy the following two conditions:

(Correctness) For any security parameter λ, any $b \in \{0,1\}$, any $x \in L$ and any x, w such that $(x,w) \in R$, we have that

$$\Pr_r[\mathsf{Decrypt}(\mathsf{Encrypt}(1^\lambda, x, b; r), x, w) = b] = 1.$$

(Security) Let $S : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to (0,1)$ be functions. We say the witness encryption scheme is semantically secure against size-S adversaries with advantage ϵ, if for every security parameter λ, every size-$S(\lambda)$ circuit A, any messages $m, m' \in \{0,1\}^n$ and any $x \notin L$,

$$\left|\Pr_r[A(\mathsf{Encrypt}(1^\lambda, x, m; r)) = 1] - \Pr_r[A(\mathsf{Encrypt}(1^\lambda, x, m'; r)) = 1]\right| < \epsilon(\lambda).$$

One can construct witness encryption for strings from witness encryption on bits by simply encrypting "bit-by-bit."

Proposition 2.3. *If a (one-bit) witness encryption scheme for L exists against $S(\lambda)$-sized adversaries with advantage ϵ, then a semantically secure multi-bit witness encryption scheme for L exists against $(S(\lambda)/\mathrm{poly}(\lambda))$-sized adversaries with advantage $\epsilon \cdot \mathrm{poly}(\lambda)$.*

Since the multi-bit and single-bit variants of witness encryption are equivalent, for the remainder of the paper we do not distinguish between the two objects.

It is known that indistinguishability obfuscation $i\mathcal{O}$ implies witness encryption as a special case [GGH+16]. Thus, by Jain, Lin, and Sahai's breakthrough construction of $i\mathcal{O}$ [JLS21, JLS22b], subexponentially secure witness encryption exists conditioned on three well-founded assumptions.

Theorem 2.4 (Informal version of [JLS22b, JLS22a])**.** *Assuming three "well-founded" subexponential security assumptions, subexponentially secure witness encryption exists.*

Remark 2.5. We note that Jain-Lin-Sahai [JLS22b] uses a slightly different notion of subexponentially security than us. Their notion of subexponentially secure says S-sized circuits can get advantage at most ϵ where S is any polynomial and ϵ is subexponentially small. In contrast, for our results, we use the (also standard) definition of subexponential security where S and ϵ are both subexponential. We stress that the Jain-Lin-Sahai [JLS22b] result extends to our notion of subexponential security when one correspondingly strengthens the three assumptions [JLS22a] and that these strengthened assumptions are still reasonable and well-founded.

2.2 Cryptographic Commitments

We will make use of the following slightly non-standard notion of a commitment scheme.

Definition 2.6 (Subexponentially Secure Injective Commitment Scheme)**.** A *subexponentially secure, injective, computationally hiding commitment scheme* is a probabilistic polynomial-time algorithm Commit with the following properties:

- **Functionality:** Commit takes as input a security parameter 1^λ, a message $m \in \{0,1\}^\star$, and randomness $r \in \{0,1\}^\star$ and outputs a string in $\{0,1\}^\star$. The output is denoted $\mathsf{Commit}(1^\lambda, m; r)$. We write $\mathsf{Commit}(1^\lambda, m)$ when the randomness used is implicit.
- **Injectivity:** For any pair of distinct valid inputs $(1^\lambda, m; r) \neq (1^{\lambda'}, m'; r')$ we have that $\mathsf{Commit}(1^\lambda, m; r) \neq \mathsf{Commit}(1^{\lambda'}, m'; r')$.
- **Security:** There exists an $\epsilon > 0$ such that for any n, λ and m and $m' \in \{0,1\}^n$ and every size $O(2^{\lambda^\epsilon})$ circuit adversary A, we have that
$$\left|\Pr_r[A(\mathsf{Commit}(1^\lambda, m; r)) = 1] - \Pr_r[A(\mathsf{Commit}(1^\lambda, m'; r)) = 1]\right| \leq O(2^{-\lambda^\epsilon}).$$

We note that the above notion is slightly stronger than the usual notion of a *statistically binding* and computationally hiding commitment scheme as it requires commitments to the same message to be different when different randomness is used.

The classical construction of a bit commitment scheme from an injective one-way function yields an injective, computationally hiding commitment scheme. Because the construction is simple, we sketch the proof below.

Theorem 2.7 (Construction 4.4.2 and Proposition 4.4.3 in Goldreich [Gol01])**.** *If there is a subexponentially secure injective one-way function, then there is a subexponentially secure, injective, computationally hiding commitment scheme.*

Proof Sketch. For each bit b one wants to commit to, pick a uniformly random $r \in \{0,1\}^\lambda$ and output $(1^\lambda, h, f(r), h(r) \oplus b)$ where f is a subexponentially secure injective one-way function and h is a hardcore predicate given by Goldreich-Levin [GL89]. The security of this scheme follows from the security of the hardcore predicate. Furthermore, observe that this construction is injective. $\square$

2.3 SNARGs

Definition 2.8 (Succinct Non-interactive Arguments, SNARGs)**.** Let $L \in \mathbf{NP}$, R be the witness relation for L. A SNARG for L consists of polynomial-time algorithms (P, V) with the following syntax:

- $P(1^\lambda, \mathsf{crs}, x, w)$ takes as input a security parameter 1^λ, a common random string crs, an instance x and a witness w such that $(x, w) \in R$; it outputs a short proof π.

- $V(1^\lambda, \mathsf{crs}, x, \pi)$ takes as input a security parameter 1^λ, a common random string crs, an instance x, and a proof π; it either accepts or rejects.

Let r denote the length of crs. These algorithms satisfy the following conditions:

(Succinctness) $r \leq \mathrm{poly}(\lambda)$ and $|\pi| \leq \mathrm{poly}(\lambda, \log n)$.

(Correctness) For any security parameter λ, any $x \in L$ and any w such that $(x, w) \in R$, we have that

$$\Pr_{\mathsf{crs} \leftarrow \{0,1\}^r}[V(1^\lambda, \mathsf{crs}, x, \pi) \text{ accepts} \mid \pi \leftarrow P(1^\lambda, \mathsf{crs}, x, w)] = 1.$$

(Security) Let $S : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to (0, 1)$ be functions. We say the SNARG is secure against size-S adversaries with advantage ϵ, if for every security parameter λ, every $S(\lambda)$-size circuit A (which implements a malicious prover) and any $x \notin L$,

$$\Pr_{\mathsf{crs} \leftarrow \{0,1\}^r}[V(1^\lambda, \mathsf{crs}, x, \pi) \text{ accepts} \mid \pi \leftarrow A(1^\lambda, \mathsf{crs})] \leq \epsilon(\lambda).$$

There are some additional properties that we want a SNARG to have:

(Laconism) We say the SNARG is *laconic* if the prover only sends one bit to the verifier, i.e., it always holds that $|\pi| = 1$.

(Predictability) We say the SNARG is *predictable* if the following holds. There is a generator Gen that receives as input a security parameter 1^λ and some randomness, and outputs a common random string crs and a trapdoor τ. The distribution of crs is uniform over $\{0, 1\}^r$; the trapdoor τ is known to the verifier but not to the prover (and it is hard to infer τ from crs). Upon receiving the proof π, the verifier simply checks whether $\pi = \tau$.

It is known that any predictable SNARG can be made laconic [FNV17]:

Fact 2.9 ([FNV17, GL89])**.** *If there exists a predictable SNARG for a language $L \in$ **NP**, then there exists a predictable and* laconic *SNARG for L.*

2.4 Kolmogorov Complexity

Informally speaking, the *Kolmogorov complexity* of a string x is the length of the shortest program M that prints x in a finite amount of time [LV08].

To formally define Kolmogorov complexity, we need to fix a universal Turing machine U. Let M be the description of a Turing machine, x be an input, and t be a time bound. The universal Turing machine $U(\langle M \rangle, x, 1^t)$ simulates M on input x for t steps, and outputs the output of M. Moreover, $U(\langle M \rangle, x, 1^t)$ runs in time $\mathrm{poly}(|\langle M \rangle|, |x|, t)$.

Definition 2.10 (Kolmogorov Complexity)**.** Fix a universal Turing machine U. Let $x, y \in \{0, 1\}^\star$ be strings and $t \in \mathbb{N} \cup \{\infty\}$ be a time bound. The *t-time-bounded Kolmogorov complexity* of x conditioned on y, denoted as $\mathrm{K}^t_U(x \mid y)$, is the minimum length of any program M that given y outputs x in t steps. Formally:

$$\mathrm{K}^t_U(x \mid y) := \min\{|M| : U(\langle M \rangle, y, 1^t) = x\}.$$

When $t = \infty$, we arrive at the definition of (unbounded) Kolmogorov complexity $\mathrm{K}_U(x \mid y) := \mathrm{K}^\infty_U(x \mid y)$. When y is the empty string ε, we denote $\mathrm{K}^t_U(x) := \mathrm{K}^t_U(x \mid \varepsilon)$.

Our results hold for any universal Turing machine U, therefore we drop the subscript U and write $\mathrm{K}^t(x \mid y)$ instead.

In Section 5, we also make use of *probabilistic* Kolmogorov complexity [GKLO22] to implement certain compression arguments.

Definition 2.11 (Probabilistic Kolmogorov Complexity)**.** Let $x, y \in \{0,1\}^\star$ be strings, $\delta > 0$, and $t \in \mathbb{N} \cup \{\infty\}$ be a time bound. The *t-time-bounded probabilistic Kolmogorov complexity* of x conditioned on y, denoted as $\mathrm{pK}^t_\delta(x \mid y)$, is the minimum program length k such that for a δ fraction of random strings w, there is a program of length k which outputs x given (w, y) in t steps. Formally:

$$\mathrm{pK}^t_\delta(x \mid y) := \min\left\{k : \Pr_{w \leftarrow \{0,1\}^t}\left[\exists M \in \{0,1\}^k, U(\langle M\rangle, (w,y), 1^t) = x\right] \geq \delta\right\}.$$

The following facts will be useful.

Fact 2.12 ([GKLO22, Lemma 18])**.** *Let $t \in \mathbb{N}$, $x \in \{0,1\}^n$ be a string, then*

$$\mathrm{K}(x \mid t) \leq \mathrm{pK}^t_{2/3}(x) + O(\log n).$$

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

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

Fact 2.14 ([GKLO22, Lemma 20])**.** *For any string $y \in \{0,1\}^\star$, time bound t (including $t = \infty$), $\delta \in (0,1]$, and positive integer α, we have*

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathrm{pK}^t_\delta(x \mid y) \leq n - \alpha] \leq \frac{1}{2^{\alpha-1} \cdot \delta}.$$

3 Conditional NP-Hardness of Approximating Meta-Complexity

3.1 Witness Encryption Implies NP-Hardness of Approximating $\mathrm{K}^t(x \mid y)$

We show that subexponentially-secure witness encryption implies essentially optimal **NP**-hardness of approximating conditional time-bounded Kolmogorov complexity.

Theorem 3.1. *Assume there exists subexponentially-secure witness encryption for* **NP***. Then for every $c \geq 1$, let $t_2(n) := 2^{n^c}$, there are polynomials p and t_1 such that the following promise problem is* **NP***-hard under randomised reductions: given strings (x, y) where $|x| = n$ and $|y| = p(n)$, output*

- Yes *if $\mathrm{K}^{t_1(n)}(x \mid y) \leq n^{1/c}$;*
- No *if $\mathrm{K}^{t_2(n)}(x \mid y) \geq n - O(1)$.*

In order to prove Theorem 3.1, we will use an additional security property of witness encryption that is implied by subexponential semantic security. Essentially, this security property says that if one encrypts a uniformly random message using a No instance of the language, then no algorithm running in subexponential time can, given the ciphertext, output a list significantly smaller than the message domain that contains the plaintext with constant probability.

Definition 3.2 (List Security)**.** We say a witness encryption scheme (Encrypt, Decrypt) for L is *S-list-secure* if for all sufficiently large λ, all $x \notin L$, all positive integers q and all size-$S(\lambda)$ adversaries *List* that output a set of at most $2^q/10^4$ binary strings, we have[12]

$$\Pr_{m \leftarrow \{0,1\}^q, r}[m \in \mathit{List}(\mathsf{Encrypt}(1^\lambda, x, m; r))] < 1/3.$$

[12]Note that the security condition is vacuous when q becomes greater than $S(\lambda)$, as in that case the adversary does not even have time to produce m.

We show that list security follows from subexponentially-secure witness encryption.

Lemma 3.3. *If a semantically secure witness encryption scheme for L is secure against size-$2^{O(\lambda^\delta)}$ adversaries with advantage $\epsilon = o(1)$, then it is $(2^{O(\lambda^\delta)})$-list-secure.*

We defer the proof of Lemma 3.3 to the end of this section. We now show how to use Lemma 3.3 to prove the **NP**-hardness of approximating conditional time-bounded Kolmogorov complexity.

Proof of Theorem 3.1. Let $(\mathsf{Encrypt}, \mathsf{Decrypt})$ be a 2^{λ^ϵ}-list-secure witness encryption scheme for the language SAT, where $\epsilon > 0$ is a constant. The reduction from SAT to the above promise problem is as follows. Given a formula φ of length N, we set $\lambda := \lceil N^{4c^2/\epsilon} \rceil$ and $n := \lceil \lambda^{\epsilon/2c} \rceil$. Let $m \leftarrow \{0,1\}^n$ and r be some uniform randomness, we output the instance $(x = m \mid y = \mathsf{Encrypt}(1^\lambda, \varphi, m; r))$. This completes our description of the reduction.

Clearly, this runs in polynomial time. It remains to show that the reduction is correct.

If φ is satisfiable with witness $w \in \{0,1\}^N$, then x can be recovered from y by running $\mathsf{Decrypt}(1^\lambda, y, \varphi, w)$. This shows that

$$\mathrm{K}^{\mathrm{poly}(\lambda)}(x \mid \mathsf{Encrypt}(1^\lambda, \varphi, x; r)) \leq O(N) = O(n^{1/2c}) < n^{1/c}.$$

Now, suppose φ is unsatisfiable. Let *Low* denote the set of strings with low conditional $t_2(n)$-time-bounded Kolmogorov complexity. That is:

$$\mathit{Low} = \{z \in \{0,1\}^n : \mathrm{K}^{t_2(n)}(z \mid \mathsf{Encrypt}(1^\lambda, \varphi, x; r)) \leq n - 10^5\}.$$

Observe that given $\mathsf{Encrypt}(1^\lambda, \varphi, x; r)$, the brute-force algorithm can print the elements of *Low* in time $\mathrm{poly}(t_2(n) \cdot 2^n \cdot \lambda) \leq 2^{\lambda^\epsilon}$. By the list security of the witness encryption and Fact 2.13, the probability that $x \in \mathit{Low}$ is at most $1/3$. Thus, with probability at least $2/3$, we have that

$$\mathrm{K}^{t_2(n)}(m \mid \mathsf{Encrypt}(1^\lambda, \varphi, m; r)) > n - 10^5. \qquad \square$$

We note that, on its own, Theorem 3.1 does not give the **NP**-hardness needed to eliminate Heuristica using the approach of Hirahara [Hir22b]. This is because for Hirahara's approach one needs to show hardness on instances $(x \mid y)$ where y has *low computational depth.*

Definition 3.4 (Computational Depth [AFvMV06])**.** The *t-time computational depth* of a string x, denote $\mathrm{cd}^t(x)$ is defined as

$$\mathrm{cd}^t(x) := \mathrm{K}^t(x) - \mathrm{K}(x).$$

Formally, in [Hir22b], one needs to show it is **NP**-hard to approximate $\mathrm{K}^t(x \mid y)$ up to an additive term[13] of $\mathrm{cd}^t(y) + \log^2(|x| \cdot |y| \cdot t)$.

We show how to adapt our proof to handle this case, assuming (in addition) the existence of subexponentially secure injective one-way functions. We do this in a two-step approach. First, we show that if the $\mathsf{Encrypt}(\cdot,\cdot,\cdot,\cdot)$ is an injective function, then with high probability the y in our reduction will have small computational depth.

Lemma 3.5. *Let R be the reduction in the proof of Theorem 3.1 and assume the* $\mathsf{Encrypt}(\cdot,\cdot,\cdot,\cdot)$ *function is injective. Then for any formula φ, with probability at least $1 - o(1)$, the instance $(x \mid y)$ outputted by $R(\varphi)$ will satisfy*

$$\mathrm{cd}^{\mathrm{poly}(\lambda)}(y) \leq n^{1/c}.$$

[13]The additive term stated here is sufficient, but Hirahara [Hir22b] actually shows one can get away with showing **NP**-hardness of a somewhat smaller additive approximation. We use this bound because it is easier to state.

Proof. Fix any formula φ. Recall that when R is run on φ, the y output is $\mathsf{Encrypt}(1^\lambda, \varphi, x; r)$ were r and x are chosen uniformly at random. By the efficiency of the Encrypt algorithm, it is easy to see that

$$\mathrm{K}^{\mathrm{poly}(\lambda)}(y) \leq \log \lambda + |\varphi| + |r| + |x| + O(\log n) \leq |r| + |x| + O(n^{1/2c}).$$

On the other hand, since Encrypt is injective and r and x are chosen uniformly at random, a standard counting argument shows that

$$\mathrm{K}(y) \geq |r| + |x| - O(\log n)$$

with probability at least $1 - o(1)$.

Putting these together, we get that

$$\mathrm{cd}^{\mathrm{poly}(\lambda)}(y) = \mathrm{K}^{\mathrm{poly}(\lambda)}(y) - \mathrm{K}(y) \leq O(n^{1/2c}) < n^{1/c}. \qquad \square$$

Next, we observe that the Encrypt function can always be made injective assuming the existence of a subexponentially-secure injective one-way functions.

Lemma 3.6. *Assume there exists a subexponentially-secure injective one-way function and a subexponentially-secure witness encryption for* **NP**. *Then there is a subexponentially-secure witness encryption scheme for* **NP** *where the* Encrypt *function is injective.*

Proof. Let (Encrypt, Decrypt) be the assumed subexponentially-secure witness encryption scheme for SAT. Our goal is to append outputs to Encrypt to create a new encryption function Encrypt′ that is injective, while maintaining its security. In one sentence, the idea is to append to the outputs of Encrypt an injective commitment of the inputs to Encrypt.

In more detail, recall the notion of a subexponentially secure, injective commitment scheme Commit (Definition 2.6), which exists assuming subexponentially secure injective one-way functions exist (Theorem 2.7).

We consider the function Encrypt′ that given a security parameter λ, a formula φ, a message m and randomness r and r', outputs the concatenation of the following two strings:

- $\mathsf{Encrypt}(1^\lambda, \varphi, m, r)$ (i.e., the witness encryption of m using φ with randomness r) and
- $\mathsf{Commit}(1^\lambda, (\varphi, m, r), r')$ (i.e, a commitment to the string (φ, m, r) using randomness r').

Since the Commit function is injective, it is easy to see that the Encrypt′ function is also injective.

It remains to show that Encrypt′ has subexponential security. We do this by a hybrid argument. The first hybrid we consider is when one encrypts using the output of the Encrypt′ algorithm. That is, one outputs

- $\mathsf{Encrypt}(1^\lambda, \varphi, m, r)$ (i.e., the witness encryption of m using φ with randomness r) and
- $\mathsf{Commit}(1^\lambda, (\varphi, m, r), r')$ (i.e, a commitment to the string (φ, m, r) using randomness r').

By the subexponential security of Commit, the output in the first hybrid is subexponentially-indistinguishable from the hybrid distribution (which we refer to as hybrid two) where one outputs:

- $\mathsf{Encrypt}(1^\lambda, \varphi, m, r)$ (same as before) and
- $\mathsf{Commit}(1^\lambda, 0^{|m|+|r|+|\varphi|}, r')$ (i.e, swapping (φ, m, r) with the all zeroes string).

Note that in hybrid two, the second output (i.e. $\mathsf{Commit}(1^\lambda, 0^{|m|+|r|+|\varphi|}, r'))$ does not depend on m, r, or φ at all (except on their length). Thus, we can appeal to the subexponential security of $\mathsf{Encrypt}$ and conclude that the output of hybrid two is subexponentially secure. Therefore, the encryption algorithm in the first hybrid (i.e. $\mathsf{Encrypt}'$) is also subexponentially secure (with some loss in the subexponential, which comes from going from the first hybrid to the second). Hence, we have that $\mathsf{Encrypt}'$ is subexponentially secure. □

Putting together Theorem 1.1, Lemma 3.5, and Lemma 3.6, we can show that the promise problem that Hirahara [Hir22b] shows would eliminate Heuristica if it is in **NP**-hard, is indeed **NP**-hard under widely believed assumptions.

Corollary 3.7. *Assume there exists subexponentially-secure witness encryption for* **NP** *and subexponentially-secure injective one-way functions. Then for every $c \geq 1$, let $t_2(n) := 2^{n^c}$, there are polynomials p and t_1 such that the following promise problem is* **NP***-hard under randomised reductions: given strings (x, y) where $|x| = n$ and $|y| = p(n)$, output*

- YES *if $\mathrm{K}^{t_1(n)}(x \mid y) \leq n^{1/c}$ and $\mathrm{cd}^{t_1}(y) \leq n^{1/c}$;*
- NO *if $\mathrm{K}^{t_2(n)}(x \mid y) \geq n - O(1)$ and $\mathrm{cd}^{t_1}(y) \leq n^{1/c}$.*

We end this subsection by proving Lemma 3.3:

Proof of Lemma 3.3. Let $(\mathsf{Encrypt}, \mathsf{Decrypt})$ be a witness encryption scheme for L secure against size $2^{\Omega(\lambda^\delta)}$ with advantage ϵ. We will show that $(\mathsf{Encrypt}, \mathsf{Decrypt})$ is (S')-list-secure for $S' := 2^{c\lambda^\delta}$, where c is a small enough universal constant.

Fix any $x \notin L$, any positive integer t, any sufficiently large λ, and any size-$S'(\lambda)$ adversary *List*. For contradiction, suppose that

$$\Pr_{m \leftarrow \{0,1\}^q, r}[m \in \mathit{List}(\mathsf{Encrypt}(1^\lambda, x, m; r))] \geq 1/3.$$

We will show that there exists $m_0, m_1 \in \{0,1\}^q$ and an S-sized adversary A such that

$$\left| \Pr_r[A(\mathsf{Encrypt}(1^\lambda, x, m_0; r)) = 1] - \Pr_r[A(\mathsf{Encrypt}(1^\lambda, x, m_1; r)) = 1] \right| \geq \Omega(1).$$

This would violate semantic security, which contradicts the security of the witness encryption scheme.

It remains to show that there exists such m_0, m_1, and A. We begin with the following claim. (We delay its proof, which is by probabilistic method, until later.)

Claim 3.8. *There exists $m_0, m_1 \in \{0,1\}^q$ such that all of the following properties hold:*

1. *for all $i \in \{0,1\}$,*
$$\Pr_r[m_i \in \mathit{List}(\mathsf{Encrypt}(1^\lambda, x, m_i; r))] \geq 1/6;$$
2. *for all $i \neq j \in \{0,1\}$,*
$$\Pr_r[m_i \in \mathit{List}(\mathsf{Encrypt}(1^\lambda, x, m_j; r))] \leq 1/20.$$

Fix any m_0 and m_1 as in the claim. We now construct the adversary A that works as follows given a ciphertext e:

- Let L be the set given by $\mathit{List}(e)$.
- If L contains m_0 but not m_1, then output 0.
- If L contains m_1 but not m_0, then output 1.

- Otherwise output a uniformly random element of $\{0,1\}$.

This completes the description of A.

We observe that for every $b \in \{0,1\}$, the probability that A outputs b when e is a random encryption of m_b is at least:

$$\begin{aligned}
&\Pr[m_b \in L \wedge m_{1-b} \notin L] + \frac{1}{2}(\Pr[m_b \in L \wedge m_{1-b} \in L] + \Pr[m_b \notin L \wedge m_{1-b} \notin L]) \\
&= \frac{1}{2}\Pr[m_b \in L] + \frac{1}{2}\Pr[m_{1-b} \notin L] \\
&\geq \frac{1}{2}(1/6 + 19/20) = 1/2 + \Omega(1).
\end{aligned}$$

Thus, we have that

$$\left|\Pr_r[A(\mathsf{Encrypt}(1^\lambda, x, m_0; r)) = 1] - \Pr_r[A(\mathsf{Encrypt}(1^\lambda, x, m_1; r)) = 1]\right| \geq \Omega(1),$$

as desired.

Since $S' = 2^{c\lambda^\delta}$ for a small enough constant c, and the size of *List* is at most S', we know that A has size at most $2^{O(\lambda^\delta)}$. Thus the adversary A breaks semantic security.

It remains to prove Claim 3.8. We argue by the probabilistic method that if two messages m_0 and m_1 are chosen uniformly at random, then they have the desired properties with positive probability.

We begin by making the following subclaim.

Claim 3.9. *When m is chosen uniformly at random from $\{0,1\}^q$, with probability at least $1/6$ we have that*

$$\Pr_r[m \in \mathit{List}(\mathsf{Encrypt}(1^\lambda, \varphi, m; r))] \geq 1/6.$$

Proof. If the claim is false, we have that

$$\begin{aligned}
1/3 &\leq \Pr_{m \leftarrow \{0,1\}^q, r}[m \in \mathit{List}(\mathsf{Encrypt}(1^\lambda, \varphi, m; r))] \\
&\leq 5/6 \cdot 1/6 + 1/6 \cdot 1 \\
&< 1/3,
\end{aligned}$$

which is a contradiction. ◇

The subclaim shows that m_0 and m_1 will satisfy the first property in Claim 3.8 with probability at least $1/36$ (since the two conditions are independent).

We show another subclaim.

Claim 3.10. *Fix any $m \in \{0,1\}^q$. If m' is chosen uniformly at random from $\{0,1\}^q$, then with probability at least .99,*

$$\Pr_r[m' \in \mathit{List}(\mathsf{Encrypt}(1^\lambda, \varphi, m; r))] \leq 1/20.$$

Proof. Since *List* contains at most $2^q/10^4$ elements, we have

$$\mathop{\mathbb{E}}_{m'}[\Pr_r[m' \in \mathit{List}(\mathsf{Encrypt}(1^\lambda, \varphi, m; r))]] \leq 10^{-4}.$$

Thus the claim follows by Markov's inequality. ◇

This subclaim shows that the probability that m_0 and m_1 do not satisfy the second property in Claim 3.8 is at most $.01 \cdot 2 = .02$.

Putting this together, the probability that m_0 and m_1 satisfy both properties in Claim 3.8 is at least $1/36 - .02 > 0$. This concludes the proof of Claim 3.8, and also concludes the proof of Lemma 3.3. □

3.2 SNARGs Imply NP-Hardness of Approximating mvMCSP

In this subsection, we show that, conditioned on SNARGs existing, mvMCSP is **NP**-hard to approximate.

Theorem 3.11. *Let $L \in$ **NP**, R be the witness relation for L. Let $\delta > 0$ be a constant, $S(\lambda) = 2^{\lambda^\delta}$ and $\epsilon(\lambda) = 2^{-\lambda^\delta}$. Suppose there is a SNARG for L that is secure against size-S adversaries with advantage ϵ. Then L reduces to* mvMCSP *via a quasi-polynomial time deterministic mapping reduction.*

Moreover, for any constant $k \geq 1$, there is some constant $\delta > 0$ such that L reduces to the following promise problem $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ via a quasi-polynomial time deterministic mapping reduction:

$$\Pi_{\mathrm{YES}} := \{(T, s) : \mathsf{CC}(T) \leq s\},$$
$$\Pi_{\mathrm{NO}} := \{(T, s) : \mathsf{CC}_{1/2+1/2^{-\log^k s}}(T) \geq 2^{\log^k s}\}.$$

Proof. Let (Gen, P, V) be the SNARG for L. Let x be an instance of L, $\tilde{n} := |x|$, and $\lambda := \log^{(k+1)/\delta} \tilde{n}$. Let $n := \mathrm{poly}(\lambda)$ be the length of crs that $\mathsf{Gen}(1^\lambda; -)$ outputs, and $\ell := \mathrm{poly}(\lambda, \log \tilde{n})$ be the length of the proof that $P(1^\lambda, \mathsf{crs}, x, -)$ outputs. Given x, we compute the following table $T : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}$: For every $\mathsf{crs} \in \{0,1\}^n$ and every $\pi \in \{0,1\}^\ell$,

$$T(\mathsf{crs}, \pi) = 1 \text{ if and only if } V(1^\lambda, \mathsf{crs}, x, \pi) \text{ accepts.}$$

Also, let $s := \mathrm{poly}(s') \leq \mathrm{poly}(\tilde{n})$, where s' is the size of P in the SNARG. It is easy to see that our reduction runs in time

$$2^{n+\ell} \cdot \mathrm{poly}(\tilde{n}) \leq 2^{\mathrm{polylog}(\tilde{n})}.$$

Now we prove the correctness of this reduction. Suppose $x \in L$, then there is a string w such that $(x, w) \in R$. Let C be the following circuit

$$C(\mathsf{crs}) = P(1^\lambda, \mathsf{crs}, x, w)$$

with x and w hardwired. The size of C is at most s. For every crs, it follows from the correctness of SNARG that $T(\mathsf{crs}, C(\mathsf{crs})) = 1$. Therefore, $(T, s) \in \Pi_{\mathrm{YES}}$.

On the other hand, suppose $x \notin L$. Let C be any circuit of size at most $2^{\log^k s} \leq S(\lambda)$. By the security of SNARGs, it follows that

$$\Pr_{\mathsf{crs} \leftarrow \{0,1\}^n}[T(\mathsf{crs}, \pi) = 1 \mid \pi \leftarrow C(\mathsf{crs})] \leq \epsilon(\lambda) \leq 2^{-\log^k s}.$$

Therefore, $(T, s) \in \Pi_{\mathrm{NO}}$. □

If SNARGs with stronger properties exist[14], then we show **NP**-hardness of approximating MCSP* and MCSP.

Theorem 3.12. *In Theorem 3.11:*

- *If the SNARG is laconic, then L reduces to* MCSP* *via a quasi-polynomial time deterministic reduction.*
- *If the SNARG is predictable, then L reduces to* MCSP *via a quasi-polynomial time deterministic reduction.*

[14] Unfortunately, we are not aware of any candidate SNARGs yet with these properties.

Proof Sketch. If the SNARG is laconic, then the proof only consists of one bit. Now, the table T that we construct becomes a partial truth table tt:

$$tt(\mathsf{crs}) = \begin{cases} 0 & \text{if } T(\mathsf{crs}, 0) = 1 \text{ but } T(\mathsf{crs}, 1) = 0; \\ 1 & \text{if } T(\mathsf{crs}, 0) = 0 \text{ but } T(\mathsf{crs}, 1) = 1; \\ \star & \text{if } T(\mathsf{crs}, 0) = T(\mathsf{crs}, 1) = 1. \end{cases} \tag{1}$$

Of course, if for some crs we have both $T(\mathsf{crs}, 0) = T(\mathsf{crs}, 1) = 0$, then we know that $x \notin L$. Thus $x \in L$ if and only if there is a small circuit consistent with tt.

If the SNARG is predictable, then by Fact 2.9, it can be made laconic while maintaining predictability. As the SNARG is predictable, the third case in Eq. (1) would never happen, and thus tt is a total truth table and we can reduce L to MCSP. □

4 Unconditional NP-Hardness of GapMOCSP

In this section, we show *unconditionally* that GapMOCSP is **NP**-hard even with the "largest" possible approximation factor: The Yes instances are computable by a size-$2^{\epsilon n}$ circuit for an arbitrarily small constant[15] $\epsilon > 0$, while the complexities of the No instances essentially match the complexity of random truth tables. We also show a similar result for GapMINcKT: it is **NP**-hard to distinguish between strings of conditional time-bounded Kolmogorov complexity at most N^ϵ and at least $N^{1-\epsilon}$.

Theorem 4.1. GapMOCSP *is* **NP***-hard under polynomial-time randomised mapping reductions. More precisely, for every constant $\epsilon > 0$, there is a polynomial-time randomised mapping reduction from an* **NP***-complete language to the following promise problem:*

$$\Pi_{\mathrm{Yes}} := \{(f, O) : \mathsf{CC}^O(f) \le 2^{\epsilon n}\},$$
$$\Pi_{\mathrm{No}} := \{(f, O) : \mathsf{CC}^O_{1/2-2^{-0.3n}}(f) > 2^{0.3n}\}.$$

Here f is the truth table of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and oracle O has size $2^{\Theta(n)}$.

Theorem 4.2. GapMINcKT *is* **NP***-hard under polynomial-time randomised mapping reductions. More precisely, for every constant $\epsilon \in (0, 1)$ and $\gamma > 1$, there is a polynomial-time randomised mapping reduction from an* **NP***-complete language to the following promise problem:*

$$\Pi_{\mathrm{Yes}} := \{(x, y) : \mathrm{K}^{N^{1+\epsilon}}(x \mid y) \le N^\epsilon\},$$
$$\Pi_{\mathrm{No}} := \{(x, y) : \mathrm{K}^{N^\gamma}(x \mid y) > N^{1-\epsilon}\}.$$

Here x has length N and y has length $\mathrm{poly}(N^\gamma)$.

Overview of this section. In short, our proof consists of two parts. First, we implement the witness encryption construction of Garg, Gentry, Sahai, and Waters [GGSW13] in a certain oracle world, and prove its security (unconditionally) in this oracle world; then, we show that such oracle witness encryption schemes imply the **NP**-hardness of GapMOCSP and GapMINcKT.

In Section 4.1, we define witness encryption in oracle worlds, and prove that it implies **NP**-hardness of GapMOCSP and GapMINcKT. A subtlety is that our **NP**-hardness results require "strong" security of oracle witness encryption, while we could only prove "weak" security for

[15] If one proves **NP**-hardness in the regime where $\epsilon = o(1)$, then this gives a subexponential-time algorithm for SAT (which is believed to be unlikely). This is why we say our approximation factor is the "largest" possible.

the construction in [GGSW13], therefore we bridge this gap in Section 4.2 by showing "weakly-secure" oracle witness encryption implies "strongly-secure" ones. In Section 4.3, we describe an oracle witness encryption scheme based on the candidate in [GGSW13], and then in Section 4.4, we prove the security of this scheme. Finally, in Section 4.5, we put the pieces together and prove the **NP**-hardness of GapMOCSP and GapMINcKT.

4.1 From Witness Encryption to NP-Hardness of GapMOCSP

We first define witness encryption in oracle worlds. Roughly speaking, it is a witness encryption scheme where the encryption and decryption algorithms have access to an oracle. We define this notion of witness encryption since in a carefully constructed oracle world, we can construct an *unconditionally secure* witness encryption scheme, and obtain unconditional **NP**-hardness of MOCSP. Also note that we require the oracle to have fan-in at most $O(\lambda)$, where $\lambda \ll N$ is the security parameter.[16]

Definition 4.3 (Witness Encryption in Oracle Worlds)**.** Let $L \in \mathbf{NP}$, R be the witness relation for L, $\mathcal{O} = \{\mathcal{O}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of distributions where each $\mathcal{O}_\lambda$ is a distribution over oracles $O_\lambda : \{0,1\}^{O(\lambda)} \to \{0,1\}$. A *witness encryption scheme* for L w.r.t. $\mathcal{O}$ consists of polynomial-time oracle algorithms $(\mathsf{Encrypt}^{(-)}, \mathsf{Decrypt}^{(-)})$ with the following syntax:

- $\mathsf{Encrypt}^{O_\lambda}(1^\lambda, x, b; r)$ takes as input a security parameter 1^λ, an instance $x \in \{0,1\}^n$, a message bit $b \in \{0,1\}$, and some randomness r; it has oracle access to the oracle O_λ, and outputs a ciphertext c.
- $\mathsf{Decrypt}^{O_\lambda}(1^\lambda, c, x, w)$ takes as input a security parameter 1^λ, a ciphertext c, an instance $x \in \{0,1\}^n$, and a witness w such that $(x,w) \in R$; it has oracle access to the oracle O_λ, operates *deterministically*, and outputs a message bit b.

These algorithms satisfy the following *correctness* condition: For any security parameter λ, any oracle O_λ in the support of $\mathcal{O}_\lambda$, any $b \in \{0,1\}$, any $x \in L$ and any w such that $(x,w) \in R$, it holds that

$$\Pr_r[\mathsf{Decrypt}^{O_\lambda}(1^\lambda, \mathsf{Encrypt}^{O_\lambda}(1^\lambda, x, b; r), x, w) = b] = 1.$$

We also define the security of witness encryptions in oracle worlds. We consider security against *programs*: every program of a certain description length and query complexity fails to break our witness encryption except with negligible probability. We consider two types of security:

- *weak* security, where for each fixed program A, security holds for a random oracle $O_\lambda \leftarrow \mathcal{O}_\lambda$; and
- *strong* security, where w.h.p. over a random oracle $O_\lambda \leftarrow \mathcal{O}_\lambda$, security holds for every resource-bounded program A.

Definition 4.4 (Security of Oracle Witness Encryption)**.** Let $L, R, \mathcal{O} = \{O_\lambda\}_{\lambda\in\mathbb{N}}$, and the witness encryption scheme $(\mathsf{Encrypt}^{(-)}, \mathsf{Decrypt}^{(-)})$ be defined as in Definition 4.3. Let $Q, S : \mathbb{N} \to \mathbb{N}$ denote the query complexity and size respectively, and let $\epsilon : \mathbb{N} \to (0,1)$. We say a (randomised) program A and an instance $x \notin L$ *breaks* the oracle O_λ with advantage $\epsilon(\lambda)$, if

$$\left|\Pr_{r,A}[A^{O_\lambda}(\mathsf{Encrypt}^{O_\lambda}(1^\lambda, x, 0; r)) = 1] - \Pr_{r,A}[A^{O_\lambda}(\mathsf{Encrypt}^{O_\lambda}(1^\lambda, x, 1; r)) = 1]\right| \geq \epsilon(\lambda). \quad (2)$$

[16] It is trivial to construct an oracle witness encryption scheme where the oracle has fan-in $\Omega(N)$ where N is the size of an instance of L.

We say the witness encryption scheme is *weakly* secure against programs of query complexity $Q(\lambda)$ with advantage $\epsilon(\lambda)$, if for every $x \notin L$ and every program A of query complexity at most $Q(\lambda)$, w.p. at most $\epsilon(\lambda)$ over the oracle $O_\lambda \leftarrow \mathcal{O}_\lambda$, (A, x) breaks O_λ with advantage $\epsilon(\lambda)$.

We say the witness encryption scheme is *strongly* secure against programs of query complexity $Q(\lambda)$ and size $S(\lambda)$ with advantage $\epsilon(\lambda)$, if w.p. at most $\epsilon(\lambda)$ over the oracle $O_\lambda \leftarrow \mathcal{O}_\lambda$, there is some instance $x \notin L$ and some adversary program A of query complexity at most $Q(\lambda)$ and size at most $S(\lambda)$ such that (A, x) breaks O_λ with advantage $\epsilon(\lambda)$.

We show that the existence of an oracle witness encryption scheme for a language $L \in \mathbf{NP}$ that is strongly secure implies a reduction from L to GapMOCSP.

Theorem 4.5. *Let $L \in \mathbf{NP}$. Suppose there is a distribution $\mathcal{O}_\lambda$ over oracles $O : \{0,1\}^{O(\lambda)} \to \{0,1\}$, whose truth tables are sampleable in* $\mathrm{poly}(2^\lambda)$ *time, and a witness encryption scheme for L w.r.t. $\mathcal{O}$ that is correct and strongly secure against size-$2^{\Omega(\lambda)}$ adversaries with advantage $2^{-\Omega(\lambda)}$.*

Then, for any constant $\epsilon > 0$, there is a polynomial-time randomised mapping reduction from L to the following promise problem:

$$\Pi_{\text{Yes}} := \{(f, O) : \mathsf{CC}^O(f) \le 2^{\epsilon n}\},$$
$$\Pi_{\text{No}} := \{(f, O) : \mathsf{CC}^O_{1/2-2^{-0.3n}}(f) > 2^{0.3n}\}.$$

Proof. Let $(\mathsf{Encrypt}^{(-)}, \mathsf{Decrypt}^{(-)})$ be the witness encryption scheme for L. Let x be an instance of L and $N := |x|$. Let $k \in \mathbb{N}$ be a large enough constant such that the following holds:

- O_λ is an oracle that receives $k\lambda$ inputs;
- $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ runs in N^k time;
- The witness length for L is at most N^k.
- The security of our witness encryption scheme is against adversaries that make $2^{\lambda/k}$ queries and its advantage is $2^{-\lambda/k}$.

Let $n := \lceil (3k \log N)/\epsilon \rceil$ and $\lambda := 10kn$.

The reduction. The truth table $f : \{0,1\}^n \to \{0,1\}$ will be a uniformly random string. To construct our oracle truth table O, we first construct an oracle world with two oracles O_λ and O_{ct} (ct for "ciphertext"), and then concatenate them into a single oracle:

- First, we draw an oracle O_λ from $\mathcal{O}_\lambda$ and include it in our oracle world.
- For every $i \in \{0,1\}^n$, let $c_i \in \{0,1\}^{N^k}$ be an encryption of $f(i)$. In particular, pick some fresh random bits r_i and let $c_i := \mathsf{Encrypt}^{O_\lambda}(1^\lambda, x, f(i); r_i)$. We create an oracle O_{ct} that stores every c_i: For $i \in \{0,1\}^n$ and $j \in \{0,1\}^{\lceil k \log N \rceil}$, $O_{\mathsf{ct}}(i, j)$ is the j-th bit of c_i.
- Finally, our oracle O is the concatenation of O_λ and O_{ct}. More precisely, O takes as input a string y of length $1 + k\lambda$. Let y_0 be the first bit of y.
  - If $y_0 = 0$, then we parse $y = y_0 \circ y_1$ where $|y_1| = k\lambda$, and return $O(y) = O_\lambda(y_1)$.
  - If $y_0 = 1$, then we parse $y = y_0 \circ i \circ j \circ y_{\mathsf{pad}}$ where $|i| = n$ and $|j| = \lceil k \log N \rceil$, and return $O(y) = O_{\mathsf{ct}}(i, j)$. (Note that $n + \lceil k \log N \rceil < k\lambda$.)

The truth tables of f and O_λ have length at most $2^{O(k\lambda)} \le \mathrm{poly}(N)$. It is easy to see that our reduction runs in (randomised) polynomial time. Now we prove its correctness.

Completeness. If $x \in L$, then the following circuit C^O computes f:

- The circuit C^O receives an advice w such that $(x, w) \in R$, as well as an input $i \in \{0,1\}^n$.
- The circuit makes N^k queries to O_{ct} to determine the ciphertext c_i.
- Then it outputs $\mathsf{Decrypt}^{O_\lambda}(1^\lambda, c_i, x, w)$.

We can see that the size of C^O is at most $N^{2k} \le 2^{\epsilon n}$, therefore $\mathsf{CC}^O(f) \le 2^{\epsilon n}$ and $(f, O) \in \Pi_{\mathrm{YES}}$.

Soundness. On the other hand, suppose that $x \notin L$, we want to show that w.h.p. $(f, O) \in \Pi_{\mathrm{NO}}$, i.e., for every size-$2^{0.3n}$ oracle circuit C,

$$\Pr_{i \leftarrow \{0,1\}^n}[C^O(i) = f(i)] \le 1/2 + 2^{-0.3n}.$$

We achieve this by a hybrid argument. We fix the oracle $O_\lambda \leftarrow \mathcal{O}_\lambda$, and w.p. $1 - o(1)$, our witness encryption scheme is secure in this oracle world. We also fix a random truth table $f : \{0,1\}^n \to \{0,1\}$, and w.p. $1 - o(1)$ it is "hard" in a sense that will be described in Claim 4.6. We fix a one-to-one correspondence between strings of length n and numbers in $\{0, 1, \dots, 2^n - 1\}$. We consider $2^n + 1$ hybrid games $\mathsf{Hyb}_0, \mathsf{Hyb}_1, \dots, \mathsf{Hyb}_{2^n}$, where we also abuse the notation $O \leftarrow \mathsf{Hyb}_g$ to denote that oracle O is drawn from the g-th hybrid game. In Hyb_g:

- For each $i \in \{0,1\}^n$, if $i < g$, then c_i is an encryption of a random bit, i.e., $c_i := \mathsf{Encrypt}^{O_\lambda}(1^\lambda, x, b; r_i)$ for fresh random bits b and r_i; if $i \ge g$, then c_i is an encryption of $f(i)$, i.e., $c_i := \mathsf{Encrypt}^{O_\lambda}(1^\lambda, x, f(i); r_i)$ for fresh random bits r_i.
- The oracle O is defined from O_λ and $\{c_i\}$ as above. That is, $O(0, y_1) = O_\lambda(y_1)$ and $O(1, i, j, y_{\mathsf{pad}})$ is the j-th bit of c_i.

We can see that Hyb_0 corresponds to our reduction, and (if f is "hard" then) Hyb_{2^n} is a secure game. Define

$$\mathsf{adv}_g := \max_C \left\{ \Pr_{O \leftarrow \mathsf{Hyb}_g}[\mathsf{correct}(C^O, f) \ge 1/2 + 2^{-0.3n}] \right\},$$

where C ranges over all oracle circuits of size at most $2^{0.3n}$, and

$$\mathsf{correct}(C^O, f) := \Pr_{i \leftarrow \{0,1\}^n}[C^O(i) = f(i)].$$

Think of the oracle circuit C winning the g-th hybrid game if C^O and f have a non-trivial correlation for $O \leftarrow \mathsf{Hyb}_g$. Then, adv_g is the probability that there is an oracle C of size $2^{0.3n}$ that wins the g-th hybrid game.

Claim 4.6. *With probability* $\ge 1 - o(1)$ *over the choice of* f*, we have that* $\mathsf{adv}_{2^n} \le 2^{-0.4n}$.

Proof. Note that in Hyb_{2^n}, the oracle O is independent of f, as each c_i is an encryption of a fresh random bit. Now, let $\gamma := 2^{-0.4n}$ and $\kappa := 2^{-0.3n}$. Suppose there is an oracle circuit C of size at most $2^{0.3n}$ such that

$$\Pr_{O \leftarrow \mathsf{Hyb}_{2^n}}[\mathsf{correct}(C^O, f) \ge 1/2 + \kappa] > \gamma.$$

For any oracle O such that $\mathsf{correct}(C^O, f) \ge 1/2 + \kappa$, given the oracle circuit C and the set of $(1/2 - \kappa)2^n$ indices where C^O is wrong, we can recover the entire truth table f. In other words, given O, we can describe f in

$$|C| + \log \binom{2^n}{(1/2 - \kappa)2^n} + O(1) \le |C| + 2^n H(1/2 - \kappa) + O(1)$$

bits, where $H(p) = -p \log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function. Since $H(1/2 - \epsilon) = 1 - (2/\ln 2)\epsilon^2 + O(\epsilon^4)$, the description of f has length at most $2^n - D$ where $D := \frac{2}{\ln 2}\kappa^2 2^n - O(\kappa^4 2^n) - |C| \geq \Omega(2^{0.4n})$.

It follows that w.p. at least γ over $O \leftarrow \mathsf{Hyb}_{2^n}$, there is a machine M of description length $2^n - D$ that outputs f. This means that $\mathrm{pK}_\gamma(f \mid O_\lambda, x) \leq 2^n - D + O(1)$. By Fact 2.14, there are at most $\frac{O(1)}{2^D \cdot \gamma} \ll 2^{-2^{0.35n}}$ fraction of such f.

Therefore, by a union bound over all oracle circuits of size at most $2^{0.3n}$, we have that $\mathsf{adv}_{2^n} \leq 2^{-0.4n}$ w.p. $1 - o(1)$ over the choice of f. ◇

We say a truth table f is *hard* if it satisfies Claim 4.6.

Claim 4.7. *For every* $0 \leq g < 2^n$, $\mathsf{adv}_g \leq \mathsf{adv}_{g+1} + 2^{-\lambda/k}$.

Proof. Let C be the oracle circuit of size at most $2^{0.3n}$ that maximises

$$\Pr_{O \leftarrow \mathsf{Hyb}_g}[\mathsf{correct}(C^O, f) \geq 1/2 + 2^{-0.3n}].$$

We want to show that the above quantity is at most $\mathsf{adv}_{g+1} + 2^{-\lambda/k}$.

Consider the following (probabilistic) adversary A. Given a ciphertext c, $A(c)$ attempts to figure out whether c is an encryption of 0 or 1.

- First, we define some ciphertexts $\{c_i\}$. For each $i \in \{0,1\}^n$, if $i < g$ then define c_i to be an encryption of a fresh random bit; if $i > g$ then define c_i to be an encryption of $f(i)$; if $i = g$ then $c_i = c$.
- Let O be the oracle defined from O_λ and $\{c_i\}$ as above.
- Use at most $O(2^n \cdot |C|)$ oracle queries to compute $\mathsf{correct}(C^O, f)$. If $\mathsf{correct}(C^O, f) \geq 1/2 + 2^{-0.3n}$, return 1; otherwise return 0.

Let $b \in \{0,1\}$ be a bit, p_b be the probability that $A(c) = 1$ when c is an encryption of b. More precisely:

$$p_b := \Pr[A(c) = 1 \mid A, r \leftarrow \{0,1\}^{N^k}, c \leftarrow \mathsf{Encrypt}^{O_\lambda}(1^\lambda, x, b; r)].$$

Since A only needs to hardcode f, the program length of A is only $2^n + O(1) \leq 2^{\lambda/k}$. The query complexity of A (to the oracle O_λ) is also at most $2^{\lambda/k}$, since it computes all c_i with at most $2^n \cdot N^k < 2^{9n}$ queries, and then computes $\mathsf{correct}(C^O, f)$ with at most $2^n \cdot 2^{0.3n} = 2^{1.3n}$ oracle queries. Thus, by the security of our witness encryption scheme, we have $|p_0 - p_1| \leq 2^{-\lambda/k}$.

Note that $\mathsf{adv}_g = p_{f(g)}$ and $\mathsf{adv}_{g+1} \geq (p_0 + p_1)/2$. Therefore

$$\mathsf{adv}_g \leq \mathsf{adv}_{g+1} + |(p_0 - p_1)/2| \leq \mathsf{adv}_{g+1} + 2^{-\lambda/k}. \qquad ◇$$

Let f be a "hard" function in the sense of Claim 4.6. It follows from Claim 4.7 that $\mathsf{adv}_0 \leq 2^{-0.4n} + 2^{n-\lambda/k} \leq 2^{-0.3n}$. Since the distribution of the oracle O in Hyb_0 is exactly the distribution produced by our reduction, we have that $\mathsf{CC}^O_{1/2-2^{-0.3n}}(f) > 2^{0.3n}$ w.h.p. The soundness of our reduction is now established. □

Inspecting the above proof, we obtain the following stronger corollary:

Corollary 4.8. *Let* $0 < \delta_1, \delta_2 < 1$ *be two constants such that* $\delta_1 + 2\delta_2 < 1$. *Under the same hypothesis of Theorem 4.5, there is a randomised mapping reduction from* L *to the following promise problem:*

$$\Pi_{\text{YES}} := \{(f, O) : \mathsf{CC}^O(f) \leq 2^{\epsilon n}\},$$
$$\Pi_{\text{NO}} := \{(f, O) : \mathsf{CC}^O_{1/2-2^{-\delta_2 n}}(f) > 2^{\delta_1 n}\}.$$

The above corollary is optimal, as for any constants $0 < \delta_1, \delta_2 < 1$ such that $\delta_1 + 2\delta_2 > 1$ and any function $f : \{0,1\}^n \to \{0,1\}$, if n is large enough, then it holds that $\mathsf{CC}_{1/2-2^{-\delta_2 n}}(f) \le 2^{\delta_1 n}$. See Appendix A for details.

Here we state another version of Theorem 4.5, but for Turing machines instead of circuits. The proof is very similar to that of Theorem 4.5, and we will only give a proof sketch here.

Theorem 4.9. *Let $L \in \mathbf{NP}$. Suppose there is a distribution $\mathcal{O}_\lambda$ over oracles $O : \{0,1\}^{O(\lambda)} \to \{0,1\}$, whose truth tables are sampleable in $\mathrm{poly}(2^\lambda)$ time, and a witness encryption scheme for L that is correct and strongly secure against $2^{\Omega(\lambda)}$-query adversaries with advantage $2^{-\Omega(\lambda)}$.*

Then, for any constants $\epsilon \in (0,1)$ and $\gamma > 1$, there is a polynomial-time randomised mapping reduction from L to the following promise problem:

$$\Pi_{\text{YES}} := \{(f, O) : \mathrm{K}^{\ell^{1+\epsilon}}(f \mid O) \le \ell^\epsilon\},$$
$$\Pi_{\text{NO}} := \{(f, O) : \mathrm{K}^{\ell^{\gamma}}(f \mid O) > \ell^{1-\epsilon}\},$$

where $|f| = \ell$ and $|O| = \mathrm{poly}(\ell^\gamma)$.

Proof Sketch. Here we will only cover the difference from the proof of Theorem 4.5. Let x be an instance of L, $N := |x|$, k, n be defined in the same way as Theorem 4.5, and $\lambda := (10+\gamma)kn$. We also define $\ell := 2^n$.

The truth table f and the oracle truth table O are sampled in the same way as Theorem 4.5.

Completeness. If $x \in L$, then the following Turing machine M^O computes f:

- The machine M^O receives an advice w such that $(x, w) \in R$.
- The machine enumerates $i \in \{0,1\}^n$.
- For every i, the machine makes N^k queries to O_{ct} to determine the ciphertext c_i, and then outputs $\mathsf{Decrypt}^{\mathcal{O}_\lambda}(1^\lambda, c_i, x, w)$.

We can see that the description size of M^O is at most $N^{2k} < 2^{\epsilon n} = \ell^\epsilon$ and the running time is at most $N^{2k} \cdot 2^n < \ell^{1+\epsilon}$.

Soundness. Suppose that $x \notin L$, we want to show that w.h.p., for every oracle Turing machine M that has description length at most $2^{(1-\epsilon)n}$ and makes at most $2^{\gamma n}$ oracle queries, M^O does not output f. We use the same proof as for Theorem 4.5, but with C^O replaced by M^O.

The definitions of hybrid games Hyb_g and ciphertexts c_i in each Hyb_g are the same as in Theorem 4.5. The advantage of Hyb_g is

$$\mathsf{adv}_g := \max_M \left\{ \Pr_{O \leftarrow \mathsf{Hyb}_g}[M^O \text{ outputs } f \text{ in time } 2^{\gamma n}] \right\},$$

where M ranges over all programs of length $2^{(1-\epsilon)n}$ and query complexity $2^{\gamma n}$. Since the oracle in Hyb_{2^n} does not depend on f, the following claim analogous to Claim 4.6 holds:

Claim 4.10. *With probability $1 - o(1)$ over the choice of f, we have that $\mathsf{adv}_{2^n} \le 2^{-n}$.*

Proof Sketch. Fix M, then $\Pr_{O,f}[M^O \text{ outputs } f \text{ in time } 2^{\gamma n}] \le 1/2^\ell$. By Markov's inequality,

$$\Pr_f\left[\Pr_O[M^O \text{ outputs } f \text{ in time } 2^{\gamma n}] \le 2^{-n}\right] \ge 1 - 2^{n-2^n}.$$

Now by a union bound over all machines M with description length at most $2^{(1-\epsilon)n}$,

$$\Pr_f[\mathsf{adv}_{2^n} \le 2^{-n}] \ge 1 - 2^{2^{(1-\epsilon)n}} \cdot 2^{n-2^n} = 1 - o(1). \qquad \diamond$$

The following claim analogous to Claim 4.7 also holds:

Claim 4.11. *For every* $0 \le g < 2^n$, $\mathsf{adv}_g \le \mathsf{adv}_{g+1} + 2^{-\lambda/k}$.

Proof Sketch. The proof intuition is similar to that of Claim 4.7, namely that any machine M such that $\Pr_O[M^O$ outputs $f]$ for $O \leftarrow \mathsf{Hyb}_g$ and $O \leftarrow \mathsf{Hyb}_{g+1}$ differs by at least $2^{-\lambda/k}$ can be used to break the security of witness encryption.

We still construct an adversary A as in Claim 4.7, and the construction is the same except for the third step: instead of computing $\mathsf{correct}(C^O, f)$, here A uses at most $2^{\gamma n}$ queries to simulate M^O; if M^O outputs f in time $2^{\gamma n}$, A returns 1, otherwise A returns 0.

By similar arguments as in Claim 4.7, A can be implemented by a machine of description length $2^{\lambda/k}$ that makes at most $2^{\lambda/k}$ queries, and by the security of our witness encryption scheme we have

$$\mathsf{adv}_g \le \mathsf{adv}_{g+1} + 2^{-\lambda/k}. \qquad \diamond$$

Now let f be a "hard" function in the sense of Claim 4.10. It follows from Claim 4.11 that $\mathsf{adv}_0 \le 2^{-n} + 2^{n-\lambda/k} = o(1)$. Since the distribution of the oracle O in Hyb_0 is exactly the distribution produced by our reduction, we have that $\mathrm{K}^{\ell^\gamma}(f \mid O) > \ell^{1-\epsilon}$ w.h.p. The soundness is therefore established. $\square$

4.2 From Weak Security to Strong Security

Unfortunately, we can only prove that the GGSW construction is weakly secure (Theorem 4.25), but Theorem 4.5 requires strong security to conclude **NP**-hardness of MOCSP. In this section, we show that the "salted" version of any weakly secure oracle witness encryption is strongly secure.

AI-ROM, BF-ROM, and salting. Our proof is crucially based on the results in [CDGS18], so it may be helpful to discuss their ideas before proceeding with our proof. The main motivation of [CDGS18] was to bridge the gap between the *random oracle model* (ROM) and the *auxiliary-input random oracle model* (AI-ROM). The *bit-fixing random oracle model* (BF-ROM) will be a useful proxy between these two models.

- In ROM, a protocol is secure if for every fixed adversary $\mathcal{A}$, w.h.p. over a random oracle $\mathcal{O}$, $\mathcal{A}^{\mathcal{O}}$ cannot break the protocol; clearly, this corresponds to weak security in our paper.
- In AI-ROM, the adversary receives (a limited amount of) auxiliary information $\alpha := \alpha(\mathcal{O})$ (which can be thought of as advice), and the security requirement becomes that for every fixed adversary $\mathcal{A}$, w.h.p. over a random oracle $\mathcal{O}$, $\mathcal{A}^{\mathcal{O}}$ with advice $\alpha(\mathcal{O})$ cannot break the protocol. We can simply think of $\alpha(\mathcal{O})$ as the best adversary circuit for breaking the protocol w.r.t. $\mathcal{O}$, and $\mathcal{A}$ as a universal machine that evaluates the oracle circuit $\alpha(\mathcal{O})$. Therefore, security in AI-ROM corresponds to strong security in our paper.
- In BF-ROM, before attacking, the adversary chooses a small number of positions in the random oracle and fixes (i.e., overrides) them arbitrarily. The protocol is secure if w.h.p. over the rest parts of the random oracle $\mathcal{O}$, $\mathcal{A}^{\mathcal{O}}$ cannot break the protocol.

An important technical lemma ([CDGS18, Lemma 1]) allows us to replace the AI-ROM model with the easier-to-analyze BF-ROM. The lemma states the following: Consider the distribution $\mathcal{O}$ of the random oracle in the AI-ROM model conditioned on the advice string $\alpha(\mathcal{O})$, then this distribution can always be approximated by a convex combination of bit-fixing random oracles. It follows that security in BF-ROM implies security in AI-ROM.

Finally, [CDGS18] proposed *salting* as a generic way of deriving the security in BF-ROM from the security in the usual ROM. Let $\mathcal{O}$ be a distribution of oracles under which a certain protocol is weakly secure, i.e., for every adversary $\mathcal{A}$, w.h.p. over $\mathcal{O}$, $\mathcal{A}^{\mathcal{O}}$ cannot break the protocol. Let K be a large enough number, $\mathcal{O}' = (\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_K)$ be K independent copies of $\mathcal{O}$. (That is, the interface of $\mathcal{O}'$ is as follows: on input $salt \in [K]$ and x, the value $\mathcal{O}_{salt}(x)$ is returned.) Consider the BF-ROM over $\mathcal{O}'$. If the adversary is allowed to fix P bits of $\mathcal{O}'$ where $P \ll K$, then for a random $salt \leftarrow [K]$, w.h.p. $\mathcal{O}_{salt}$ is a uniformly random oracle (without any fixed bit), and we can invoke the weak security of $\mathcal{O}_{salt}$ to argue the security of $\mathcal{O}'$ in BF-ROM. By the above lemma, this also implies the (strong) security of $\mathcal{O}'$ in AI-ROM.

The formal proof. In what follows, we assume every possible oracle in the support of $\mathcal{O}_\lambda$ is drawn with equal probability. Abusing notation, we also use $\mathcal{O}_\lambda$ to denote the support of $\mathcal{O}_\lambda$ (which is a set of oracles).[17] Let $K = 2^{O(\lambda)}$ denote the number of salts, we consider random variables over the support $(\mathcal{O}_\lambda)^K$.

Definition 4.12. We say a random variable $\mathcal{O}'$ over the range $(\mathcal{O}_\lambda)^K$ is *P-fixing*[18] if it is fixed on at most P coordinates and uniform on the rest.

Lemma 4.13 ([CDGS18, Lemma 1])**.** *Let X be distributed uniformly over $(\mathcal{O}_\lambda)^K$ and $Z := f(X)$, where $f : (\mathcal{O}_\lambda)^K \to \{0,1\}^S$ is an arbitrary function. For every $\gamma > 0$ and $P \in \mathbb{N}$, there exists a family $\{Y_z\}_{z\in\{0,1\}^S}$ such that:*

- *Each Y_z is the convex combination of P-fixing random variables over $(\mathcal{O}_\lambda)^K$.*
- *For every distinguisher D taking an S-bit input and querying at most $T < P$ coordinates of its oracle,*

$$\left|\Pr[D^X(f(X)) = 1] - \Pr[D^{Y_{f(X)}}(f(X)) = 1]\right| \le \frac{(S + \log(1/\gamma)) \cdot T}{P} + \gamma.$$

In the lemma above, it would be helpful to think of D as a universal Turing machine, and f as the function that maps the oracle world to the best adversary of description length S for breaking the protocol. (There is no requirement that f needs to be "efficient".) The lemma states that even after the S-bit non-uniformity is fixed, the conditional distribution of the oracle can be approximated by bit-fixing distributions, thus allowing us to transform security proofs in BF-ROM into security proofs in AI-ROM.

Theorem 4.14. *Let $L \in$ **NP**, R be the witness relation for L.*

Let $\mathcal{O}_\lambda$ be a distribution over oracles $O_\lambda : \{0,1\}^{O(\lambda)} \to \{0,1\}$ whose truth tables are sampleable in $\mathrm{poly}(2^\lambda)$ time, such that there is a witness encryption scheme for L w.r.t. $\mathcal{O}$ that is weakly secure against programs of query complexity $2^{\Omega(\lambda)}$ with advantage $2^{-\Omega(\lambda)}$.

Then there is a distributions $\mathcal{O}'_\lambda$ over oracles $O'_\lambda : \{0,1\}^{O(\lambda)} \to \{0,1\}$ whose truth tables are also sampleable in $\mathrm{poly}(2^\lambda)$ time, and a witness encryption scheme for L w.r.t. $\mathcal{O}'$ that is strongly secure against programs of query complexity $2^{\Omega(\lambda)}$ and size $2^{\Omega(\lambda)}$ with advantage $2^{-\Omega(\lambda)}$.

Proof. Let $c \in \mathbb{N}$ be a large enough constant such that the following holds:

- The input length of $\mathcal{O}_\lambda$ is $c \cdot \lambda$.
- For every program A of query complexity $2^{\lambda/c}$, w.p. at most $2^{-\lambda/c}$ over a random oracle $O \leftarrow \mathcal{O}_\lambda$,

$$\left|\Pr_r[A^O(\mathsf{Encrypt}^O(1^\lambda, x, 0; r)) = 1] - \Pr_r[A^O(\mathsf{Encrypt}^O(1^\lambda, x, 1; r)) = 1]\right| \ge 2^{-\lambda/c}.$$

[17]This is because the statement of [CDGS18, Lemma 1] requires one to start with the uniform distribution over a set of oracles.

[18]Such random variables are called *P-bit-fixing* in [CDGS18]. We removed the word "bit" because each coordinate of $\mathcal{O}'$ is not a bit, but an oracle of $2^{O(\lambda)}$ size.

Without loss of generality, we assume that $\lambda \geq 6c \cdot \log n$, where $n = |x|$ is the input length of L.

Let $k := \lambda/(2c)$. We construct a new family of oracles $\mathcal{O}' = \{\mathcal{O}'_\lambda\}_{\lambda\in\mathbb{N}}$. To sample an oracle O' from $\mathcal{O}'_\lambda$, we sample 2^k oracles $O_{0^k}, \dots, O_{salt}, \dots, O_{1^k}$ independently from $\mathcal{O}_\lambda$, indexed by strings $salt \in \{0,1\}^k$, and then define

$$O'(salt, x) = O_{salt}(x) \quad \forall salt \in \{0,1\}^k, x \in \{0,1\}^{c\lambda}.$$

The new witness encryption scheme works as follows:

- $\mathsf{Encrypt}_{\mathsf{new}}^{O'}(1^\lambda, x, b)$ samples a random $salt \leftarrow \{0,1\}^k$, calculates the ciphertext $ct \leftarrow \mathsf{Encrypt}^{O_{salt}}(1^\lambda, x, b)$, and outputs $ct_{\mathsf{new}} := (salt, ct)$ as the final ciphertext.
- $\mathsf{Decrypt}_{\mathsf{new}}^{O'}(1^\lambda, ct_{\mathsf{new}}, x, w)$ parses ct_{new} as $(salt, ct)$ where $salt \in \{0,1\}^k$ and ct is the ciphertext for the old witness encryption, and then outputs $\mathsf{Decrypt}^{O_{salt}}(1^\lambda, ct, x, w)$.

The correctness of $(\mathsf{Encrypt}_{\mathsf{new}}, \mathsf{Decrypt}_{\mathsf{new}})$ is easy to see, so it remains to prove its (strong) security. To this end, we invoke Lemma 4.13 where:

- $X := \mathcal{O}_\lambda$ and $K := 2^k$ (note that here, $\mathcal{O}_\lambda$ is indeed the uniform distribution over its support);
- $f : (\mathcal{O}_\lambda)^K \to \{0,1\}^{2^{\lambda/(6c)}+n}$ is the function mapping an oracle to its best adversary; that is, given an oracle $O' \in (\mathcal{O}_\lambda)^K$, $f(O')$ is the pair (x, A) that maximizes
$$\left|\Pr_r[A^{O'}(\mathsf{Encrypt}_{\mathsf{new}}^{O'}(1^\lambda, x, 0; r)) = 1] - \Pr_r[A^{O'}(\mathsf{Encrypt}_{\mathsf{new}}^{O'}(1^\lambda, x, 1; r)) = 1]\right|,$$
where $x \in \{0,1\}^n \setminus L$ and A is a program of length $2^{\lambda/(6c)}$ and query complexity $2^{\lambda/(6c)}$; if there are ties, we let $f(O')$ be the lexicographically first maximiser (x, A);
- $\gamma := 2^{-\lambda/(6c)}$ and $P := 2^{\lambda/(2c)}$.

It follows from Lemma 4.13 that for every $x \in \{0,1\}^n \setminus L$ and every adversary program A of length $2^{\lambda/(6c)}$ and query complexity $2^{\lambda/(6c)}$, there is a distribution $Y_{(x,A)}$ that is a convex combination of P-fixing random variables over $(\mathcal{O}_\lambda)^K$, such that the following holds. For every bit $b \in \{0,1\}$, let $D_b^{O'}(x, A)$ be the distinguisher that receives (x, A) as auxiliary inputs, and outputs $A^{O'}(\mathsf{Encrypt}_{\mathsf{new}}^{O'}(1^\lambda, x, b; r))$. Then

$$\begin{aligned}&\left|\Pr_{O',D_b}\left[D_b^{O'}(f(O')) = 1\right] - \Pr_{O,D_b}\left[D_b^{Y_{f(O')}}(f(O')) = 1\right]\right| \\ \leq\ & \frac{(2^{\lambda/(6c)} + n + \log(1/\gamma)) \cdot 2^{\lambda/(6c)}}{P} + \gamma \leq 4 \cdot 2^{-\lambda/(6c)}.\end{aligned}$$

(Recall that $n \leq 2^{\lambda/(6c)}$.) Now, we have:

$$\begin{aligned}&\left|\Pr_{O',D_0}\left[D_0^{O'}(f(O')) = 1\right] - \Pr_{O',D_1}\left[D_1^{O'}(f(O')) = 1\right]\right| \\ \leq\ & 8 \cdot 2^{-\lambda/(6c)} + \left|\Pr_{O',D_0}\left[D_0^{Y_{f(O')}}(f(O')) = 1\right] - \Pr_{O',D_1}\left[D_1^{Y_{f(O')}}(f(O')) = 1\right]\right| \\ \leq\ & 8 \cdot 2^{-\lambda/(6c)} + \sum_{(x,A)} \Pr_{O'}[f(O') = (x, A)] \cdot \\ & \qquad \left|\Pr_{O' \leftarrow \mathcal{O}'(x,A), D_0}\left[D_0^{Y_{(x,A)}}(x, A) = 1\right] - \Pr_{O' \leftarrow \mathcal{O}'(x,A), D_1}\left[D_1^{Y_{(x,A)}}(x, A) = 1\right]\right|, \qquad (3)\end{aligned}$$

where $\mathcal{O}'(x, A)$ denotes the distribution of O' conditioned on $f(O') = (x, A)$.

Since each $Y_{(x,A)}$ is a convex combination of P-fixing random variables, it suffices to show:

Claim 4.15. *Fix x, A and let Z be a P-fixing source, then*

$$\left|\Pr_{Z,D_0}\left[D_0^Z(x,A)=1\right]-\Pr_{Z,D_1}\left[D_1^Z(x,A)=1\right]\right| \le 2^{-\lambda/(3c)}.$$

Proof. Let $(O_{0^k},\dots,O_{1^k})$ be the random variable representing an oracle sampled from Z. Since Z is P-fixing, there is an index set $\mathcal{I}\subseteq\{0,1\}^k$ of size at least 2^k-P such that the marginal distribution of $\{O_i\}_{i\in\mathcal{I}}$ is the uniform distribution over $\mathcal{O}_\lambda^{\mathcal{I}}$.

Recall that $D_b^Z(x,A)$ represents the following experiment:

1. First, sample an oracle $Z := (O_{0^k},\dots,O_{1^k})$.
2. Then, sample $salt \leftarrow \{0,1\}^k$.
3. Then, encrypt the message b using the witness encryption w.r.t. oracle O_{salt}, and obtain $ct \leftarrow \mathsf{Encrypt}^{O_{salt}}(1^\lambda, x, b)$.
4. Finally, output $A^Z(salt, ct)$.

For every $salt' \in \mathcal{I}$, conditioned on $salt = salt'$, A^Z should fail to distinguish an encryption of 0 from an encryption of 1. This essentially follows from the weak security of our old witness encryption scheme: Consider a new adversary A' that hardcodes $salt'$ and samples $O_{salt''}$ for every $salt'' \ne salt'$. (If $salt'' \in \mathcal{I}$ then $O_{salt''}$ is sampled independently from $\mathcal{O}_\lambda$; otherwise $O_{salt''}$ is a fixed oracle according to Z.) It receives an oracle $O \leftarrow \mathcal{O}_\lambda$ and it treats O as the oracle $O_{salt'}$. Note that the adversary A' is fixed in advance and does not depend on $O = O_{salt'}$; also, the query complexity of the adversary is at most $2^{\lambda/(6c)} < 2^{\lambda/c}$. Let $p_{salt',b}$ denote the probability that A' outputs 1 when given the encryption of b w.r.t. oracle $O_{salt'}$, then w.p. at least $2^{-\lambda/c}$ over $O_{salt'}$, we have $|p_{salt',0}-p_{salt',1}| \le 2^{-\lambda/c}$. It follows that

$$\begin{aligned}&\left|\Pr_{Z,D_0}\left[D_0^Z(x,A)=1\right]-\Pr_{Z,D_1}\left[D_1^Z(x,A)=1\right]\right|\\ \le\ &\mathop{\mathbb{E}}_{salt}\left[|p_{salt,0}-p_{salt,1}|\right]\\ \le\ &(P/2^k)+2\cdot 2^{-\lambda/c} \le 2^{-\lambda/(3c)}.\end{aligned}$$

◇

Now we can see that

$$(3) \le 4\cdot 2^{-\lambda/(6c)} + 2^{-\lambda/(3c)} \le 2^{-\lambda/(7c)}.$$

It follows from a standard Markov bound that w.p. at most $2^{-\lambda/(14c)}$ over the oracle $O' \leftarrow \mathcal{O}'_\lambda$, there exists $x \in \{0,1\}^n \setminus L$ and an adversary program A of length $2^{\lambda/(6c)}$ and query complexity $2^{\lambda/(6c)}$ such that (A, x) breaks O' with advantage $2^{-\lambda/(14c)}$. □

4.3 Description of GGSW

In this subsection, we describe an oracle world with a weakly secure witness encryption scheme (unconditionally). Roughly speaking, our witness encryption scheme is the one in [GGSW13] and our oracle world implements the generic multilinear map model [Sho97]. Like in [GGSW13], the witness encryption scheme works for the **NP**-complete language Exact-Cover [Kar72], defined as follows:

Definition 4.16 (Exact-Cover). An Exact-Cover instance consists of a number n and a collection of subsets $X_1, X_2, \dots, X_m \subset [n]$. It is a Yes instance if and only if there is a sub-collection in which every element in $[n]$ appears exactly once. More formally:

- There exists an index set $I \subset [m]$, such that $\bigcup_{i\in I} X_i = [n]$ and for all $i, j \in I$, $i \ne j$, we have $X_i \cap X_j = \varnothing$.

Such an index set I is a *witness* for the instance.

The generic multilinear map model. The oracle world implements a cryptographic multilinear map [BS03]. In an n-multilinear group family, we have a sequence of cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_n$ of the same order p. We use addition for the group operation and 0 for the identity. We also have a set of multilinear maps $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j}$ for every i, j such that $i + j \le n$. The map is multilinear in the sense that $e_{i,j}(g_i, g_j) = g_i + g_j$, where g_i, g_j, and $e_{i,j}(g_i, g_j)$ are interpreted as elements in $\mathbb{G}_i$, $\mathbb{G}_j$, and $\mathbb{G}_{i+j}$ respectively.

In the generic multilinear map model, we do not have access to the actual group elements, but receive their *labels* instead. For each group $\mathbb{G}_i$, there is a bijective label function $\sigma_i : [p] \to \mathbb{G}_i$ mapping labels to group elements.[19] The adversary operates on labels instead of group elements and its only access to the group structure is via the multilinear map:

$$e_{i,j} : [p] \times [p] \to [p], \quad e_{i,j}(\sigma_i^{-1}(g_i), \sigma_j^{-1}(g_j)) = \sigma_{i+j}^{-1}(g_i + g_j). \tag{4}$$

(That is, the map $e_{i,j}$ receives two labels s_i, s_j, treats s_i as the labelling of $g_i \in \mathbb{G}_i$, treats s_j as the labelling of $g_j \in \mathbb{G}_j$, and returns the labelling of $g_i + g_j \in \mathbb{G}_{i+j}$.) Intuitively, suppose the label functions are random, then the adversary has no idea about which actual group elements are represented by which label.

Construction of the oracle world. Let $p \in (2^\lambda, 2^{\lambda+1})$ be a prime number, $(\mathbb{G}, +)$ be the cyclic group of order p. We independently sample $n + 1$ bijections $\sigma_1, \sigma_2, \dots, \sigma_{n+1} : [p] \to \mathbb{G}$ uniformly at random. We identify $[p]$ with the lexicographically smallest p strings of length $\lambda+1$. The groups $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_{n+1}$ are isomorphic to $\mathbb{G}$, but we use the subscript i to emphasize that this group is paired with the bijection σ_i.

We now implement Eq. (4) as a (Boolean) oracle. Let $\mathsf{Add} : \{0,1\}^{\lceil \log_2 (n+1) \rceil} \times \{0,1\}^{\lambda+1} \times \{0,1\}^{\lceil \log_2 (n+1) \rceil} \times \{0,1\}^{\lambda+1} \to \{0,1\}^{\lambda+1}$ be the following oracle: on input (i, s_i, j, s_j), if $s_i, s_j \in [p]$, $i + j \le n + 1$, and neither i nor j are zero, then we return

$$\mathsf{Add}(i, s_i, j, s_j) = \sigma_{i+j}^{-1}(\sigma_i(s_i) + \sigma_j(s_j)); \tag{5}$$

otherwise we return $0^{\lambda+1}$ (denoting the query is invalid).

Let $\mathcal{O}_\lambda$ be the set consisting of all Add that will be sampled as above. We also abuse notation and denote $\mathcal{O}_\lambda$ as the distribution of Add defined from uniformly and independently random bijections $\sigma_1, \sigma_2, \dots, \sigma_{n+1}$.

It is easy to see that the oracle world is efficiently sampleable.

Claim 4.17. *The oracle world can be sampled in* $\mathrm{poly}(2^\lambda)$ *time.*

Proof. Note that sampling a bijection $[p] \to \mathbb{G}$ can be done in $\mathrm{poly}(2^\lambda)$ time, and computing each entry of Add with that bijection takes $\mathrm{poly}(2^\lambda)$ time. Since there are $\mathrm{poly}(2^\lambda)$ entries, the total time is $\mathrm{poly}(2^\lambda)$. $\square$

Construction of the witness encryption scheme. Now we construct the witness encryption scheme. Let $x = (X_1, X_2, \dots, X_m)$ be an Exact-Cover instance. The witness encryption scheme consists of oracle algorithms $(\mathsf{Encrypt}^{(-)}, \mathsf{Decrypt}^{(-)})$ defined as the following:

- $\mathsf{Encrypt}^{\mathsf{Add}}(1^\lambda, x, b; r)$ first uniformly samples $s_1, s_2, \dots, s_n \leftarrow [p]$ using the randomness r. Let $g_i := \sigma_1(s_i)$ be the element in $\mathbb{G}_1$ under label s_i. For every set $X \subseteq [n]$, define $g_X := \sum_{j \in X} g_j$, and treat g_X as an element in $\mathbb{G}_{|X|}$. Then, let $s_X := \sigma_{|X|}^{-1}(g_X)$.

[19] In the literature, it is more common to write $\sigma_i : \mathbb{G}_i \to [p]$, so σ_i gives every element in $\mathbb{G}_i$ a label in $[p]$. However, in the security proof later, we will construct hybrids where the map $[p] \to \mathbb{G}_i$ is not injective and thus $\mathbb{G}_i \to [p]$ is not a map (though this happens with negligible probability), so we choose to define σ_i as $[p] \to \mathbb{G}_i$.

We sample[20] $\mathsf{msg}_0 \leftarrow [p] \setminus \{s_{X_i} : |X_i| = 1\}$ and then $\mathsf{msg}_1 \leftarrow [p] \setminus (\{\mathsf{msg}_0\} \cup \{s_{X_i} : |X_i| = 1\})$ uniformly using the randomness r. As we are encrypting $b \in \{0,1\}$, define $g_{n+1} := \sigma_1(\mathsf{msg}_b)$, and extend the above definition of g_X to $X \subseteq [n+1]$. Finally we output the ciphertext

$$\big(\{s_{X_i}\}_{i\in[m]}, s_{[n+1]}, \mathsf{msg}_0, \mathsf{msg}_1\big). \tag{6}$$

Of course, we cannot decode each s_i in order to compute the ciphertext; we need to use the oracle Add instead. In particular, for every set $X \subseteq [n]$ where $|X| > 1$, let x be any element in X, we have $s_X = \mathsf{Add}(|X|-1, s_{X\setminus\{x\}}, 1, s_x)$. Therefore we can compute each s_{X_i} and $s_{[n+1]}$ in polynomial time, only using oracle access to Add.

- $\mathsf{Decrypt}^{\mathsf{Add}}(1^\lambda, c, x, I)$ first parses the ciphertext c into the form Eq. (6). Suppose the witness $I = \{i_1, i_2, \dots, i_{|I|}\}$. Then we compute $s_{[n]} = s_{X_{i_1}\cup X_{i_2}\cup\cdots\cup X_{i_{|I|}}}$ with oracle Add by recursively computing

$$s_{X_{i_1}\cup X_{i_2}\cup\cdots\cup X_{i_j}} = \mathsf{Add}\Big(|X_{i_1}\cup X_{i_2}\cup\cdots\cup X_{i_{j-1}}|, s_{X_{i_1}\cup X_{i_2}\cup\cdots\cup X_{i_{j-1}}}, |X_{i_j}|, s_{X_{i_j}}\Big).$$

  Then we use Add again to compute $\mathsf{Add}(n, s_{[n]}, 1, \mathsf{msg}_0)$. If $\mathsf{Add}(n, s_{[n]}, 1, \mathsf{msg}_0) = s_{[n+1]}$, then we return 0; otherwise we return 1.

Correctness. Recall that $g_i = \sigma_1(s_i)$ for each i, and $g_X = \sum_{x\in X} g_i$ for each $X \subseteq [n]$. Let s_X be computed as in the encryption algorithm, then it is easy to prove by induction that for every non-empty $X \subset [n]$, it is indeed true that $s_X = \sigma_{|X|}^{-1}(g_X)$. Therefore, for every non-empty $X, Y \subset [n]$ such that $X \cap Y = \varnothing$, we have

$$s_{X\cup Y} = \sigma_{|X|+|Y|}^{-1}(g_X + g_Y) = \mathsf{Add}(|X|, s_X, |Y|, s_Y).$$

It follows that if $x \in$ EXACT-COVER and I is a correct witness for x, then the algorithm $\mathsf{Decrypt}$ computes $s_{[n]}$ correctly. Then, there are two cases.

- Suppose $b = 0$, then $\mathsf{Add}(n, s_{[n]}, 1, \mathsf{msg}_0) = \mathsf{Add}(n, s_{[n]}, 1, \mathsf{msg}_b) = s_{[n+1]}$, thus the algorithm outputs 0.

- Suppose $b = 1$. Note that $\mathsf{msg}_0 \neq \mathsf{msg}_1$ and σ_i is a bijection, thus $\mathsf{Add}(n, s_{[n]}, 1, \mathsf{msg}_0) \neq \mathsf{Add}(n, s_{[n]}, 1, \mathsf{msg}_1) = \mathsf{Add}(n, s_{[n]}, 1, \mathsf{msg}_b)$, and the algorithm outputs 1.

4.4 Security of GGSW

In this subsection, we prove the security of the witness encryption scheme: if $x \notin$ EXACT-COVER, then the encryption of 0 and the encryption of 1 are computationally indistinguishable.

4.4.1 Hybrid Games

We use a hybrid argument. In this subsection, we define all hybrid games involved in the security proof. Each hybrid game has an initialisation phase Init and an interactive phase. The challenger runs the initialisation phase at the beginning, which generates a secret bit b and a public encryption of b; then the challenger and the adversary run the interactive phase, during which the challenger responds to the adversary's Add oracle queries. Finally, the adversary succeeds if he guesses b correctly.

In what follows, if a function f is not injective, we will use $f^{-1}(x)$ to denote an *arbitrary* element in the preimage (say, the lexicographically smallest one). This definition is not crucial,

[20]Here we require msg_0 and msg_1 to be different from any s_{X_i} such that $|X_i| = 1$, so that security proof later becomes more convenient.

since every function in the hybrid games will be injections *with high probability.* Also, for an oracle call $\mathsf{Add}(i, s_i, j, s_j)$, we always implicitly assume that $i, j \neq 0$, $i+j \leq n+1$, and $s_i, s_j \in [p]$, since otherwise the oracle query is "invalid" and Add always returns $0^{\lambda+1}$.

To describe the hybrid games, we will need random tapes S, $T^{(1)}$, $T^{(2)}$, $\dots$, $T^{(n+1)}$. Here:

- Let Q be an upper bound on the number of oracle queries of the adversary, we sample $S \leftarrow \mathbb{G}^{2Q+n+2}$ uniformly at random.
- For each $i \in [n+1]$, let $T^{(i)} \in [p]^p$ be a random permutation. (That is, $\forall i \in [n+1], \forall j_1 \neq j_2, T^{(i)}_{j_1} \neq T^{(i)}_{j_2}$.)

We can treat each S and $T^{(i)}$ as a *stream* of elements: we access the elements in the order from the first to the last, and we never access the j-th element before seeing the $(j-1)$-st one. Also note that S and T do not appear in Hyb_0, but appear in every subsequent game.

Definition of Hyb_0. The game Hyb_0 is exactly the security game of our witness encryption scheme. In the Init phase:

- The challenger samples bijections $\sigma_1, \sigma_2, \dots, \sigma_{n+1} : [p] \to \mathbb{G}$ uniformly at random.
- Then, she samples $g_1, g_2, \dots, g_n \leftarrow \mathbb{G}$ uniformly at random and computes $g_{X_1}, g_{X_2}, \dots, g_{X_m}$.
- She also uniformly samples $\mathsf{msg}_0 \leftarrow [p] \setminus \{\sigma_1^{-1}(g_{X_i}) : |X_i| = 1\}$ and then $\mathsf{msg}_1 \leftarrow [p] \setminus (\{\mathsf{msg}_0\} \cup \{\sigma_1^{-1}(g_{X_i}) : |X_i| = 1\})$.
- Finally, she outputs the ciphertext

$$\left(\{\sigma^{-1}_{|X_i|}(g_{X_i})\}_{i\in[m]}, \sigma^{-1}_{n+1}(g_{[n]} + \sigma_1(\mathsf{msg}_b)), \mathsf{msg}_0, \mathsf{msg}_1\right).$$

When the adversary queries $\mathsf{Add}(i, s_i, j, s_j)$, the challenger computes $g_i \leftarrow \sigma_i(s_i)$ and $g_j \leftarrow \sigma_j(s_j)$, and then replies $\sigma^{-1}_{i+j}(g_i + g_j)$.

Definition of Hyb_0'. The game Hyb_0' is *equivalent* to Hyb_0, but it is defined in a way using the random tapes S and $T^{(i)}$. This will make it easier to compare Hyb_0' with later hybrid games. In Hyb_0', the challenger maintains partial functions $\sigma_1, \sigma_2, \dots, \sigma_{n+1} : [p] \to \mathbb{G}$. In the Init phase:

- The challenger reads $S_1, \dots, S_{n+2}$ from the tape S and sets $g_i := S_i$ for every $i \in [n+2]$.
- Then, for every $i = 1, 2, \dots, m$ in ascending order, if there is no $i' < i$ such that $|X_{i'}| = |X_i|$ and $g_{X_{i'}} = g_{X_i}$,[21] the challenger reads the next entry $T^{(|X_i|)}_{\mathsf{new}}$ in $T^{(|X_i|)}$, and sets $\sigma_{|X_i|}(T^{(|X_i|)}_{\mathsf{new}}) := g_{X_i}$.
- Then, let msg_0 and msg_1 be the next two entries in $T^{(1)}$; the challenger sets $\sigma_1(\mathsf{msg}_0) := g_{n+1}$ and $\sigma_1(\mathsf{msg}_1) := g_{n+2}$.
- The challenger also reads the first entry of $T^{(n+1)}$, namely $T^{(n+1)}_1$, and sets $\sigma_{n+1}(T^{(n+1)}_1) := g_{[n]} + \sigma_1(\mathsf{msg}_b)$.
- Finally, the challenger outputs the ciphertext

$$\left(\{\sigma^{-1}_{|X_i|}(g_{X_i})\}_{i\in[m]}, \sigma^{-1}_{n+1}(g_{[n]} + \sigma_1(\mathsf{msg}_b)), \mathsf{msg}_0, \mathsf{msg}_1\right).$$

When the adversary queries $\mathsf{Add}(i, s_i, j, s_j)$, the challenger performs the following.

- If $\sigma_i(s_i)$ is not defined, then the challenger reads the next entry S_{new} in S. If S_{new} is not in the image of σ_i, she sets $\sigma_i(s_i) := S_{\mathsf{new}}$; otherwise, she samples R from $\mathbb{G} \setminus \mathsf{Image}(\sigma_i)$ uniformly at random and sets $\sigma_i(s_i) := R$.
- If $\sigma_j(s_j)$ is not defined, then the challenger reads the next entry S_{new} in S. If S_{new} is not in the image of σ_j, she sets $\sigma_j(s_j) := S_{\mathsf{new}}$; otherwise, she samples R from $\mathbb{G} \setminus \mathsf{Image}(\sigma_j)$ uniformly at random and sets $\sigma_j(s_j) := R$.

[21]For this moment, the reader can safely ignore the phrase "if there is no $i' < i$ such that $|X_{i'}| = |X_i|$ and $g_{X_{i'}} = g_{X_i}$", as the probability that this does not happen is negligible.

- Let $g^1 := \sigma_i(s_i)$ and $g^2 := \sigma_j(s_j)$. If $\sigma_{i+j}^{-1}(g^1 + g^2)$ is not defined, then the challenger reads the next entry $T_{\mathsf{new}}^{(i+j)}$ in $T^{(i+j)}$. If $T_{\mathsf{new}}^{(i+j)}$ is already in the domain of σ_{i+j}, then she disposes it and reads a new $T_{\mathsf{new}}^{(i+j)}$; she repeats this until getting a $T_{\mathsf{new}}^{(i+j)}$ that is not in the domain of σ_{i+j}. Now she sets $\sigma_{i+j}(T_{\mathsf{new}}^{(i+j)}) := g^1 + g^2$.
- Finally, the challenger returns $\sigma_{i+j}^{-1}(g^1 + g^2)$.

After the interactive phase, for every i, σ_i restricted to its domain is a bijection. To extend σ_i to a full bijection between $[p]$ and $\mathbb{G}$, for each $s \in [p]$ not yet in the domain of σ_i, we uniformly sample g from $\mathbb{G} \setminus \mathsf{Image}(\sigma_i)$ and sets $\sigma_i(s) := g$. This step does not affect the game; its only purpose is to assist our proof that Hyb_0 and Hyb_0' are equivalent.

Definition of Hyb_1. Next, we define a game Hyb_1, which is almost the same as Hyb_0' but with the following difference. When the challenger answers $\mathsf{Add}(-)$ queries, she samples S_{new} from S. In Hyb_0', if S_{new} is in the range of σ_i (or σ_j), she re-samples an element R and sets $\sigma_i(s_i)$ (or $\sigma_j(s_j)$) to be R. In Hyb_1, the challenger always sets $\sigma_i(s_i)$ (or $\sigma_j(s_j)$) to be S_{new}, regardless of whether this would introduce a collision (making σ_i or σ_j not injective any more). However, we will show that the collision probability is negligible, thus Hyb_1 will be close to Hyb_0'.

We present a complete description of Hyb_1. In this game, the challenger maintains partial functions $\sigma_1, \sigma_2, \ldots, \sigma_{n+1} : [p] \to \mathbb{G}$. In the Init phase (this is exactly the same as Hyb_0'):

- The challenger reads $S_1, \ldots, S_{n+2}$ from the tape S and sets $g_i := S_i$ for every $i \in [n+2]$.
- Then, for every $i = 1, 2, \ldots, m$ in ascending order, if there is no $i' < i$ such that $|X_{i'}| = |X_i|$ and $g_{X_{i'}} = g_{X_i}$, the challenger reads the next entry $T_{\mathsf{new}}^{(|X_i|)}$ in $T^{(|X_i|)}$, and sets $\sigma_{|X_i|}(T_{\mathsf{new}}^{(|X_i|)}) := g_{X_i}$.
- Then, let msg_0 and msg_1 be the next two entries in $T^{(1)}$; the challenger sets $\sigma_1(\mathsf{msg}_0) := g_{n+1}$ and $\sigma_1(\mathsf{msg}_1) := g_{n+2}$.
- The challenger also reads the first entry of $T^{(n+1)}$, namely $T_1^{(n+1)}$, and sets $\sigma_{n+1}(T_1^{(n+1)}) := g_{[n]} + \sigma_1(\mathsf{msg}_b)$.
- Finally, the challenger outputs the ciphertext

$$\Big(\{\sigma_{|X_i|}^{-1}(g_{X_i})\}_{i\in[m]}, \sigma_{n+1}^{-1}(g_{[n]} + \sigma_1(\mathsf{msg}_b)), \mathsf{msg}_0, \mathsf{msg}_1\Big).$$

When the adversary queries $\mathsf{Add}(i, s_i, j, s_j)$, the challenger performs the following.

- If $\sigma_i(s_i)$ is not defined, then the challenger reads the next entry S_{new} in S and sets $\sigma_i(s_i) := S_{\mathsf{new}}$.
- If $\sigma_j(s_j)$ is not defined, then the challenger reads the next entry S_{new} in S and sets $\sigma_j(s_j) := S_{\mathsf{new}}$.
- Let $g^1 := \sigma_i(s_i)$ and $g^2 := \sigma_j(s_j)$. If $\sigma_{i+j}^{-1}(g^1 + g^2)$ is not defined, then the challenger reads the next entry $T_{\mathsf{new}}^{(i+j)}$ in $T^{(i+j)}$. If $T_{\mathsf{new}}^{(i+j)}$ is already in the domain of σ_{i+j}, then she disposes it and reads a new $T_{\mathsf{new}}^{(i+j)}$; she repeats this until getting a $T_{\mathsf{new}}^{(i+j)}$ that is not in the domain of σ_{i+j}. Now she sets $\sigma_{i+j}(T_{\mathsf{new}}^{(i+j)}) := g^1 + g^2$.
- Finally, the challenger returns $\sigma_{i+j}^{-1}(g^1 + g^2)$.

Definition of Hyb_2. In Hyb_2, instead of assigning values to each g_i, we treat them as formal variables. Let $\mathcal{V} := \{\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_{n+2}, \hat{v}_1, \hat{v}_2, \ldots\}$ be the set of formal variables. (Here, $\hat{g}_1, \ldots, \hat{g}_{n+2}$ corresponds to $g_1, \ldots, g_{n+2}$ in Hyb_1, and each $\hat{v}_i$ is a new formal variable that might be created during $\mathsf{Add}(-)$.) Let $\mathbb{Z}\mathcal{V}$ denote the space of $\mathbb{Z}$-linear combinations over $\mathcal{V}$, i.e.,

$$\mathbb{Z}\mathcal{V} := \left\{\sum_{i=1}^{n+2} \alpha_i \hat{g}_i + \sum_{i=1}^{n'} \beta_i \hat{v}_i : n' \in \mathbb{N}, \alpha_i, \beta_i \in \mathbb{Z}\right\}.$$

For every $i \in [n+2]$, the challenger maintains a partial function $\Sigma_i : [p] \to \mathbb{Z}\mathcal{V}$. We still use the notation that for a set $S \subseteq [n+1]$, $\hat{g}_S := \sum_{i\in S} \hat{g}_i$. (Thus, $\hat{g}_S$ is an element in $\mathbb{Z}\mathcal{V}$.)

In the Init phase:

- For every $i = 1, 2, \ldots, m$ in ascending order, the challenger reads the next entry $T^{(|X_i|)}_{\mathsf{new}}$ in $T^{(|X_i|)}$, and sets $\Sigma_{|X_i|}(T^{(|X_i|)}_{\mathsf{new}}) := \hat{g}_{X_i}$.
- Then, let msg_0 and msg_1 be the next two entries in $T^{(1)}$; the challenger sets $\Sigma_1(\mathsf{msg}_0) := \hat{g}_{n+1}$ and $\Sigma_1(\mathsf{msg}_1) := \hat{g}_{n+2}$.
- The challenger also reads the first entry of $T^{(n+1)}$, namely $T^{(n+1)}_1$, and sets $\Sigma_{n+1}(T^{(n+1)}_1) := \hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_b)$.
- Finally, she outputs the ciphertext

$$\Big(\{\Sigma^{-1}_{|X_i|}(\hat{g}_{X_i})\}_{i\in[m]}, \Sigma^{-1}_{n+1}(\hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_b)), \mathsf{msg}_0, \mathsf{msg}_1\Big).$$

When the adversary queries $\mathsf{Add}(i, s_i, j, s_j)$, the challenger performs the following.

- If $\Sigma_i(s_i)$ is not defined, then let $\hat{v}_{\mathsf{new}}$ denote the first unused formal variable among $\{\hat{v}_k\}_{k\in\mathbb{N}}$ so far, and set $\Sigma_i(s_i) := \hat{v}_{\mathsf{new}}$.
- Then, if $\Sigma_j(s_j)$ is not defined, then let $\hat{v}'_{\mathsf{new}}$ denote the first unused formal variable among $\{v_k\}_{k\in\mathbb{N}}$ so far, and set $\Sigma_j(s_j) := \hat{v}'_{\mathsf{new}}$.
- Let $\hat{\ell}_i := \Sigma_i(s_i)$ and $\hat{\ell}_j := \Sigma_j(s_j)$. Then $\hat{\ell}_i + \hat{\ell}_j$ is also an element in $\mathbb{Z}\mathcal{V}$. If $\Sigma^{-1}_{i+j}(\hat{\ell}_i + \hat{\ell}_j)$ is not defined, then the challenger reads the next entry $T^{(i+j)}_{\mathsf{new}}$ in $T^{(i+j)}$. If $T^{(i+j)}_{\mathsf{new}}$ is already in the domain of σ_{i+j}, then she disposes it and reads a new $T^{(i+j)}_{\mathsf{new}}$; she repeats this until getting a $T^{(i+j)}_{\mathsf{new}}$ that is not in the domain of σ_{i+j}. Now she sets $\sigma_{i+j}(T^{(i+j)}_{\mathsf{new}}) := \hat{\ell}_i + \hat{\ell}_j$.
- Finally, the challenger returns $\Sigma^{-1}_{i+j}(\hat{\ell}_i + \hat{\ell}_j)$.

4.4.2 Proof of Security

We first show that Hyb_0 and Hyb'_0 are equivalent.

Lemma 4.18. *Suppose $S \leftarrow \mathbb{G}^{2Q+n+2}$ and random permutations $T^{(1)}, T^{(2)}, \ldots, T^{(n+1)} \in [p]^p$ are uniformly sampled. Then in Hyb'_0, the distribution of $(\sigma_1, \sigma_2, \ldots, \sigma_{n+1}, g_1, g_2, \ldots, g_n)$ is the uniform distribution over $(\mathsf{Bij}([p], \mathbb{G}))^{n+1} \times \mathbb{G}^n$, where $\mathsf{Bij}([p], \mathbb{G})$ is the set of all bijections from $[p]$ to $\mathbb{G}$. Moreover, conditioned on fixed $(\sigma_1, \sigma_2, \ldots, \sigma_{n+1}, g_1, g_2, \ldots, g_n)$, the distribution of $(\mathsf{msg}_0, \mathsf{msg}_1)$ is the uniform distribution over $\{(\mu, \nu) \in ([p] \setminus \{\sigma_1^{-1}(g_{X_i}) : |X_i| = 1\})^2 : \mu \neq \nu\}$.*

Proof. We fix arbitrary bijections $\sigma'_1, \sigma'_2, \ldots, \sigma'_{n+1} : [p] \to \mathbb{G}$ and $g'_1, g'_2, \ldots, g'_n \in \mathbb{G}$. Define $g'_X = \sum_{j\in X} g'_j$. Let $H := \{\sigma_1^{-1}(g_{X_i}) : |X_i| = 1\}$, and we fix arbitrary $\mathsf{msg}'_0, \mathsf{msg}'_1 \in [p] \setminus H$ such that $\mathsf{msg}'_0 \neq \mathsf{msg}'_1$. It suffices to prove that

$$\Pr\left[\begin{array}{l}(\sigma_1, \sigma_2, \ldots, \sigma_{n+1}, g_1, g_2, \ldots, g_n, \mathsf{msg}_0, \mathsf{msg}_1)\\ = (\sigma'_1, \sigma'_2, \ldots, \sigma'_{n+1}, g'_1, g'_2, \ldots, g'_n, \mathsf{msg}'_0, \mathsf{msg}'_1)\end{array}\right] = \frac{1}{(p!)^{n+1} p^n (p - |H|)(p - |H| - 1)}.$$

After the challenger sets g_i, since each entry of S is uniformly sampled from $\mathbb{G}$, we have

$$\Pr[(g_1, g_2, \ldots, g_n) = (g'_1, g'_2, \ldots, g'_n)] = \frac{1}{p^n}.$$

Then, after we assign a preimage of g_{X_i} in $\sigma_{|X_i|}$ for every $i \in [m]$ (unless already assigned), suppose the number of (preimage, image) pairs in σ_i is r_i for each $i \in [n+1]$. By the definition of $T^{(1)}, T^{(2)}, \ldots, T^{(n+1)}$, whenever we assign a preimage in σ_i, the preimage is uniformly sampled from all element from $[p]$ not yet in the domain of σ_i, so we have

$$\Pr\left[\begin{array}{l}(\sigma_1, \sigma_2, \ldots, \sigma_{n+1}, g_1, g_2, \ldots, g_n)\\ = \Big(\sigma'_1|_{\mathsf{Domain}(\sigma_1)}, \sigma'_2|_{\mathsf{Domain}(\sigma_2)}, \ldots, \sigma'_{n+1}|_{\mathsf{Domain}(\sigma_{n+1})}, g'_1, g'_2, \ldots, g'_n\Big)\end{array}\right] = \frac{\prod_{i=1}^{n+1}(p - r_i)!}{(p!)^{n+1} p^n}.$$

Next, we sample msg_0 and msg_1. Conditioned on

$$\begin{aligned}&(\sigma_1, \sigma_2, \dots, \sigma_{n+1}, g_1, g_2, \dots, g_n)\\ &= \left(\sigma_1'|_{\mathsf{Domain}(\sigma_1)}, \sigma_2'|_{\mathsf{Domain}(\sigma_2)}, \dots, \sigma_{n+1}'|_{\mathsf{Domain}(\sigma_{n+1})}, g_1', g_2', \dots, g_n'\right),\end{aligned}$$

msg_0 is sampled from $[p] \setminus H$ and msg_1 from $[p] \setminus (\{\mathsf{msg}_1\} \cup H)$, so after sampling them, we have

$$\Pr\left[\begin{array}{l}(\sigma_1, \sigma_2, \dots, \sigma_{n+1}, g_1, g_2, \dots, g_n, \mathsf{msg}_0, \mathsf{msg}_1)\\ = \left(\sigma_1'|_{\mathsf{Domain}(\sigma_1)}, \sigma_2'|_{\mathsf{Domain}(\sigma_2)}, \dots, \sigma_{n+1}'|_{\mathsf{Domain}(\sigma_{n+1})}, g_1', g_2', \dots, g_n', \mathsf{msg}_0', \mathsf{msg}_1'\right)\end{array}\right]$$
$$= \frac{\prod_{i=1}^{n+1}(p - r_i)!}{(p!)^{n+1}p^n(p - |H|)(p - |H| - 1)}.$$

Later, we are concerned only with the process we add (preimage, image) pairs to σ_i, and when we add such a pair, we call it one step. Let $\sigma_{i,j}$ denote the σ_i when there's j (preimage, image) pairs in it. We will prove by induction on the number k of steps that

$$\Pr\left[\begin{array}{l}(\sigma_{1,j_1}, \sigma_{2,j_2}, \dots, \sigma_{n+1,j_{n+1}}, g_1, g_2, \dots, g_n, \mathsf{msg}_0, \mathsf{msg}_1)\\ = \left(\sigma_1'|_{\mathsf{Domain}(\sigma_{1,j_1})}, \sigma_2'|_{\mathsf{Domain}(\sigma_{2,j_2})}, \dots, \sigma_{n+1}'|_{\mathsf{Domain}(\sigma_{n+1,j_{n+1}})}, g_1', g_2', \dots, g_n', \mathsf{msg}_0', \mathsf{msg}_1'\right)\end{array}\right]$$
$$= \frac{\prod_{i=1}^{n+1}(p - j_i)!}{(p!)^{n+1}p^n(p - |H|)(p - |H| - 1)},$$

where for each i, j_i denotes the number of (preimage, image) pairs in σ_i after k steps.

The base case where $k = 0$ is that $j_i = r_i$ for every i, and is proved above. Suppose the statement is true for k, and we prove it for $k + 1$. Suppose in the $(k + 1)$-st step, we add a new (preimage, image) pair to σ_u. *Conditioned on*

$$\begin{aligned}&(\sigma_{1,j_1}, \sigma_{2,j_2}, \dots, \sigma_{n+1,j_{n+1}}, g_1, g_2, \dots, g_n, \mathsf{msg}_0, \mathsf{msg}_1)\\ &= \left(\sigma_1'|_{\mathsf{Domain}(\sigma_{1,j_1})}, \sigma_2'|_{\mathsf{Domain}(\sigma_{2,j_2})}, \dots, \sigma_{n+1}'|_{\mathsf{Domain}(\sigma_{n+1,j_{n+1}})}, g_1', g_2', \dots, g_n', \mathsf{msg}_0', \mathsf{msg}_1'\right),\end{aligned}$$

the $(k + 1)$-st step has the following two possibilities:

- The challenger determines $s \in [p] \setminus \mathsf{Domain}(\sigma_u)$, uniformly samples $g \in \mathbb{G} \setminus \mathsf{Image}(\sigma_{u,j_u})$, and sets $\sigma_{u,j_u+1}(s) = \sigma_u(s) := g$. Therefore, since $\sigma_u'(s) \notin \mathsf{Image}(\sigma_{u,j_u})$,
$$\Pr\left[\sigma_{u,j_u+1} = \sigma_u'|_{\mathsf{Domain}(\sigma_{u,j_u+1})}\right] = \frac{1}{p - j_u}.$$
- The challenger determines $g \in \mathbb{G} \setminus \mathsf{Image}(\sigma_{u,j_u})$, uniformly samples $s \in [p] \setminus \mathsf{Domain}(\sigma_{u,j_u})$, and sets $\sigma_{u,j_u+1}(s) = \sigma_u(s) := g$. Therefore, since $(\sigma_u')^{-1}(g) \notin \mathsf{Domain}(\sigma_{u,j_u})$,
$$\Pr\left[\sigma_{u,j_u+1} = \sigma_u'|_{\mathsf{Domain}(\sigma_{u,j_u+1})}\right] = \frac{1}{p - j_u}.$$

So this equation holds in either possibility, and thus

$$\Pr\left[\begin{array}{l}(\sigma_{1,j_1}, \dots, \sigma_{u,j_u+1}, \dots, \sigma_{n+1,j_{n+1}}, g_1, \dots, g_n, \mathsf{msg}_0, \mathsf{msg}_1) =\\ \left(\sigma_1'|_{\mathsf{Domain}(\sigma_{1,j_1})}, \dots, \sigma_u'|_{\mathsf{Domain}(\sigma_{u,j_u+1})}, \dots, \sigma_{n+1}'|_{\mathsf{Domain}(\sigma_{n+1,j_{n+1}})}, g_1', \dots, g_n', \mathsf{msg}_0', \mathsf{msg}_1'\right)\end{array}\right]$$
$$= \frac{\prod_{i=1}^{n+1}(p - j_i)!}{(p!)^{n+1}p^n(p - |H|)(p - |H| - 1)} \cdot \frac{1}{p - j_u} = \frac{(p - j_1)! \cdots (p - j_u - 1)! \cdots (p - j_{n+1})!}{(p!)^{n+1}p^n(p - |H|)(p - |H| - 1)}.$$

After the whole process ends, $j_1 = j_2 = \cdots = j_{n+1} = p$, and hence

$$\Pr\left[\begin{array}{l}(\sigma_1, \sigma_2, \dots, \sigma_{n+1}, g_1, g_2, \dots, g_n, \mathsf{msg}_0, \mathsf{msg}_1)\\ = (\sigma_1', \sigma_2', \dots, \sigma_{n+1}', g_1', g_2', \dots, g_n', \mathsf{msg}_0', \mathsf{msg}_1')\end{array}\right] = \frac{1}{(p!)^{n+1}p^n(p - |H|)(p - |H| - 1)}. \quad \square$$

Next, we show that Hyb_0' and Hyb_1 are indistinguishable by adversaries with limited query complexity.

Lemma 4.19. *Let Q be an upper bound on the number of oracle queries made by the adversary. Let $T^{(1)}, T^{(2)}, \ldots, T^{(n+1)} \in [p]^p$ be arbitrary permutations as defined before, and let S be uniformly sampled from $\mathbb{G}^{2Q+n+2}$. Then, with probability at least $1 - 2Q(m+3+3Q)/|\mathbb{G}|$ over S, the permutations $\{\sigma_i\}$ defined in Hyb_0' and Hyb_1 are equal, when the random tapes are S and T.*

Proof. Note that for fixed S and T, Hyb_0' and Hyb_1 are the same if the following never happens whenever the adversary queries $\mathsf{Add}(i, s_i, j, s_j)$:

- s_i is not yet in the domain of σ_i, and the next entry in S is already in the image of σ_i;
- s_j is not yet in the domain of σ_j, and the next entry in S is already in the image of σ_j.

The size of the image of σ_i is at most $m + 3 + 3Q$, and there are at most Q rounds, so the probability over S that the above never happens is at least $1 - 2Q(m+3+3Q)/p$. □

In what follows, we will show that Hyb_1 and Hyb_2 are indistinguishable; this is the most important part of the proof.

For each formal variable $\hat{g}_i$ and $\hat{v}_i$, we assign a value $\mathsf{val}(\hat{g}_i) = S_i$ and $\mathsf{val}(\hat{v}_i) = S_{n+2+i}$, where S is the random tape. For every linear combination $\hat{\ell} = \sum_i \alpha_i \hat{g}_i + \sum_i \beta_i \hat{v}_i$, this induces a value of $\hat{\ell}$, namely $\mathsf{val}(\hat{\ell}) = \sum_i \alpha_i \mathsf{val}(\hat{g}_i) + \sum_i \beta_i \mathsf{val}(\hat{v}_i)$.

Let $\Sigma_1, \Sigma_2, \ldots, \Sigma_{n+1} : [p] \to \mathbb{Z}\mathcal{V}$ be partial functions, we call $\{\mathsf{val} \circ \Sigma_i\}$ the *realisation* of $\{\Sigma_i\}$. In other words, $\sigma_i : [p] \to \mathbb{G}$ is the realisation of $\Sigma_i : [p] \to \mathbb{Z}\mathcal{V}$, if $\mathsf{Domain}(\sigma_i) = \mathsf{Domain}(\Sigma_i)$ and for every $s \in \mathsf{Domain}(\sigma_i)$, $\sigma_i(s) = \mathsf{val}(\Sigma_i(s))$.

Lemma 4.20. *Let $\hat{\ell}, \hat{\ell}' \in \mathbb{Z}\mathcal{V}$, $\hat{\ell} \neq \ell'$. For every $\hat{x} \in \mathcal{V}$, we assign a value $\mathsf{val}(\hat{x}) \leftarrow \mathbb{G}$ independently and uniformly at random. Then* $\Pr[\mathsf{val}(\hat{\ell}) = \mathsf{val}(\hat{\ell}')] = 1/p$.

Proof. Suppose $\hat{\ell} = \sum_{i=1}^{n+2} \alpha_i \hat{g}_i + \sum_{i=1}^{n'} \beta_i \hat{v}_i$, $\hat{\ell}' = \sum_{i=1}^{n+2} \alpha_i' \hat{g}_i + \sum_{i=1}^{n'} \beta_i' \hat{v}_i$. Since $\hat{\ell} \neq \hat{\ell}'$, without loss of generality, suppose $\alpha_1 \neq \alpha_1'$. Then for arbitrary $\mathsf{val}(\hat{g}_2), \ldots, \mathsf{val}(\hat{g}_{n+2}), \mathsf{val}(\hat{v}_1), \ldots, \mathsf{val}(\hat{v}_{n'})$,

$$\Pr[\mathsf{val}(\hat{\ell}) = \mathsf{val}(\hat{\ell}')] = \Pr_{\mathsf{val}(\hat{g}_1)}\left[(\alpha_1 - \alpha_1')\mathsf{val}(\hat{g}_1) = \sum_{i=2}^{n+2} (\alpha_i' - \alpha_i)\mathsf{val}(\hat{g}_i) + \sum_{i=1}^{n'} (\beta_i' - \beta_i)\mathsf{val}(\hat{v}_i)\right] = \frac{1}{p}.$$

□

We call $\hat{\ell}, \hat{\ell}' \in \mathbb{Z}\mathcal{V}$ *collide* if $\hat{\ell} \neq \hat{\ell}'$ but $\mathsf{val}(\hat{\ell}) = \mathsf{val}(\hat{\ell}')$. The following is an immediate corollary of Lemma 4.20.

Corollary 4.21. *Let $\hat{\ell}, \hat{\ell}' \in \mathbb{Z}\mathcal{V}$, $\hat{\ell} \neq \hat{\ell}'$, then* $\Pr[\hat{\ell}, \hat{\ell}'$ *collide*$] \leq 1/p$.

The following lemma shows that Hyb_1 and Hyb_2 are indistinguishable by adversaries with limited query complexity. This is shown by induction over the number of queries:

Lemma 4.22. *Let $q \in \{0\} \cup [Q]$ where Q is an upper bound on the number of oracle queries made by the adversary. Let $T^{(1)}, T^{(2)}, \ldots, T^{(n+1)} \in [p]^p$ be arbitrary permutations as defined before, and let S be uniformly sampled from $\mathbb{G}^{2Q+n+2}$. Suppose after q rounds, the partial functions as defined in Hyb_1 and Hyb_2 are $\{\sigma_i\}$ and $\{\Sigma_i\}$ respectively, when the random tapes are S and T.*

Then with probability at least $1 - (m+3+3q)^2/p$ over S, the event $P_q = P_q^0 \wedge P_q^1 \wedge P^2 \wedge P_q^3 \wedge P_q^4 \wedge P_q^5$ happens, where $P_q^0, P_q^1, P^2, P_q^3, P_q^4, P_q^5$ are defined as follows.

- *P_q^0: for every i, there is no collision among $\mathsf{Image}(\Sigma_i)$ after the q-th round.*
- *P_q^1: $\{\sigma_i\}$ is the realisation of $\{\Sigma_i\}$ after the q-th round.*
- *P^2: the inputs of the adversary in Hyb_1 and Hyb_2 are equal.*

- P_q^3: *in the first q rounds, the oracle queries and answers of the adversary in* Hyb_1 *and* Hyb_2 *are equal respectively.*
- P_q^4: *for every i, σ_i and Σ_i are injective (so that σ_i^{-1} and Σ_i^{-1} are well-defined).*
- P_q^5: *in* Hyb_1 *and* Hyb_2*, the number of entries in S that have been read are the same.*

Proof. We prove the statement by induction on q.

Base case. The base case is when $q = 0$ (after the Init phase). First note that after the Init phase, Σ_i is always an injection; this is because we assumed there are no two sets $i \neq j$ such that $X_i = X_j$ in our EXACT-COVER instance. By the definitions of $\mathsf{Hyb}_1, \mathsf{Hyb}_2$, and val, we have $g_i = S_i = \mathsf{val}(\hat{g}_i)$ for $i \in [n+2]$. It follows that $g_{X_i} = \mathsf{val}(\hat{g}_{X_i})$ for $i \in [m]$.

Next, we show that if P_0^0 happens, then the whole event P_0 happens. Assume P_0^0, i.e. there is no collision among the set $L = \{\hat{g}_{X_1}, \ldots, \hat{g}_{X_m}, \hat{g}_{n+1}, \hat{g}_{n+2}, \hat{g}_{[n]\cup\{n+1+b\}}\}$.

1. Since the preimages in $\{\Sigma_i\}$ and $\{\sigma_i\}$ of L and $\mathsf{val}(L)$ are both assigned according to the random tapes $T^1, \ldots, T^{n+1}$ in the same order, it follows that for every $\hat{\ell} \in L$, if $\hat{\ell} = \hat{g}_S$, then $\sigma_{|S|}(\Sigma_{|S|}^{-1}(\hat{\ell})) = \mathsf{val}(\hat{\ell})$. In other words, $\{\sigma_i\}$ is the realisation of $\{\Sigma_i\}$, i.e. P_0^1 happens.

2. It is easy to see that when P_0^1 happens, P_2 always happens.

3. P_0^3 always happens trivially.

4. Since Σ_i is injective and there is no collision among L, it follows that σ_i is also injective, i.e. P_0^4 happens.

5. Both hybrid game uses $n + 2$ elements from the random tape S, so P_0^5 happens.

Therefore, the whole event P_0 happens.

By definition of S, the elements $g_1, \ldots, g_{n+2}$ are uniformly distributed in $\mathbb{G}$. So by Corollary 4.21, for any pair in L, the probability that they collide is at most $1/p$, so the probability that there is a collision in L is at most $(m+3)^2/p$. It follows that

$$\Pr[P_0] = \Pr[P_0^0] \geq 1 - (m+3)^2/p,$$

thus the statement is true for $q = 0$.

Induction step. Suppose the statement is true for q oracle queries, and we now prove the statement for $q + 1$ oracle queries. In what follows, we assume that P_q happens.

Note that if P_q^4 and P_q^1 happen, then for every i, there is no collision among $\mathsf{Image}(\Sigma_i)$ after the q-th round: if ℓ and ℓ' collide, then by P_q^1, $\Sigma_i^{-1}(\ell) = \sigma_i^{-1}(\mathsf{val}(\ell)) = \sigma_i^{-1}(\mathsf{val}(\ell')) = \Sigma_i^{-1}(\ell')$, which is a contradiction to P_q^4. Also, since P_q^1 happens, for each $i \in [n+1]$, $\mathsf{Domain}(\sigma_i) = \mathsf{Domain}(\Sigma_i)$.

Since P^2 and P_q^3 happen, after the q-th round, the adversaries in Hyb_1 and Hyb_2 are in the same state, so they will make the same query in the $(q+1)$-st round. Suppose the query is $\mathsf{Add}(j_1, s_{j_1}, j_2, s_{j_2})$. Next, the challenger in Hyb_1 obtains values $\sigma_{j_1}(s_{j_1})$ and $\sigma_{j_2}(s_{j_2})$, and the challenger in Hyb_2 obtains values $\Sigma_{j_1}(s_{j_1})$ and $\Sigma_{j_2}(s_{j_2})$. We can see that $\sigma_{j_1}(s_{j_1}) = \mathsf{val}(\Sigma_{j_1}(s_{j_1}))$ and $\sigma_{j_2}(s_{j_2}) = \mathsf{val}(\Sigma_{j_2}(s_{j_2}))$, since:

- If $s_{j_1} \notin \mathsf{Domain}(\sigma_{j_1})$, then we also have $s_{j_1} \notin \mathsf{Domain}(\Sigma_{j_1})$. The challenger will set $\sigma_{j_1}(s_{j_1})$ to be the next unused entry in S in Hyb_1, and set $\mathsf{val}(\Sigma_{j_1}(s_{j_1}))$ to be the next unused entry in S in Hyb_2. Since P_q^5 holds, these two entries are the same.

- If $s_{j_1} \in \mathsf{Domain}(\sigma_{j_1})$, then it is also true that $s_{j_1} \in \mathsf{Domain}(\Sigma_{j_1})$. Since P_q^1 happens, we also have $\sigma_{j_1}(s_{j_1}) = \mathsf{val}(\Sigma_{j_1}(s_{j_1}))$.

- The same argument applies to (j_2, s_{j_2}).

So it still holds at this time that $\{\sigma_i\}$ is the realisation of $\{\Sigma_i\}$. Also, we have $\sigma_{j_1}(s_{j_1}) + \sigma_{j_2}(s_{j_2}) = \mathsf{val}(\Sigma_{j_1}(s_{j_1})+\Sigma_{j_2}(s_{j_2}))$. In the next step, the challenger in Hyb_1 adds $\sigma_{j_1}(s_{j_1})+\sigma_{j_2}(s_{j_2})$ to the image of $\sigma_{j_1+j_2}$ if it was not in the image before, and the challenger in Hyb_2 adds $\Sigma_{j_1}(s_{j_1}) + \Sigma_{j_2}(s_{j_2})$ to the image of $\Sigma_{j_1+j_2}$ if it was not in the image before.

We show that if (P_q and) P^0_{q+1} happens, then the whole event P_{q+1} happens. To this end, assume P^0_{q+1} happens, i.e. for every i, there is no collision among the image of Σ_i (after the $(q+1)$-st round).

1. If $\Sigma_{j_1}(s_{j_1}) + \Sigma_{j_2}(s_{j_2})$ is newly added to $\mathsf{Image}(\Sigma_{j_1+j_2})$ in the Add query, then by P^0_{q+1}, $\mathsf{val}(\Sigma_{j_1}(s_{j_1}) + \Sigma_{j_2}(s_{j_2})) = \sigma_{j_1}(s_{j_1}) + \sigma_{j_2}(s_{j_2})$ is also newly added to the image of $\sigma_{j_1+j_2}$. Moreover, they will have the same preimage (namely the next unused element of $T^{(j_1+j_2)}$).

   If $\Sigma_{j_1}(s_{j_1}) + \Sigma_{j_2}(s_{j_2})$ is not newly added to the image of $\Sigma_{j_1+j_2}$, then clearly, $\sigma_{j_1}(s_{j_1}) + \sigma_{j_2}(s_{j_2})$ is also not newly added to the image of $\sigma_{j_1+j_2}$.

   It follows that $\{\sigma_i\}$ is still the realisation of $\{\Sigma_i\}$, i.e. P^1_{q+1} happens.

2. P^2 happens by the induction hypothesis.

3. The queries made by the adversary in the $(q+1)$-st round are the same in both hybrid games. Since P^1_{q+1} happens, $\sigma_{j_1+j_2}$ is the realisation of $\Sigma_{j_1+j_2}$, thus the answers are also the same in both hybrid games. It follows that P^3_{q+1} happens.

4. Since Σ_i is injective, there is no collision among the image of Σ_i, and $\{\sigma_i\}$ is the realisation of $\{\Sigma_i\}$, it follows that σ_i is also injective, i.e. P^4_{q+1} happens.

5. As we have seen before, (if P_q happens then) P_5^{q+1} always happens.

Therefore, the whole event P_{q+1} happens.

Now we bound the probability that P_{q+1} happens. Note that if P_q happens but P^0_{q+1} does not, then the at most 3 new linear combinations in the $(q+1)$-st round collide with the image of $\{\Sigma_i\}$ in the q-th round. Since the image in the q-th round has at most $m+3+3q$ elements, by Corollary 4.21,

$$\begin{aligned}\Pr_S[P_{q+1}] &= \Pr_S[P_q] \cdot \Pr_S[P^0_{q+1} \mid P_q] \\ &\geq \left(1 - \frac{(m+3+3q)^2}{p}\right) \cdot \prod_{i=0}^{2}\left(1 - \frac{(m+3+3q+i)}{p}\right) \\ &\geq 1 - \frac{(m+3+3q+3)^2}{p}.\end{aligned}$$ □

Finally, we show that if our Exact-Cover instance is a No instance, then Hyb_2 is unconditionally hard:

Lemma 4.23. *Suppose that $(1^n, \{X_i\}_{i\in[m]}) \notin$ Exact-Cover. Fix S and T, in Hyb_2, the input, oracle queries, and oracle answers of the adversary are the same for $b = 0$ and $b = 1$.*

Proof. Without loss of generality, we first assume A never queries (i, s_i, j, s_j) where i or j equals $n+1$, since such a query gives A no information. Then, the answers of the oracle queries of A are labels of linear combinations of $\hat{g}_{X_i}, \Sigma_1(\mathsf{msg}_0), \Sigma_1(\mathsf{msg}_1), \hat{v}_j$. Since $(1^n, \{X_i\}_{i\in[m]}) \notin$ Exact-Cover, we have that for every $b_1, \ldots, b_m \in \mathbb{Z}_{\geq 0}$,

$$\sum_{i=1}^{m} b_i \hat{g}_{X_i} \neq \hat{g}_{[n]}.$$

Therefore, the oracle answers are never of the form $\Sigma_{n+1}^{-1}(\hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_{b'}))$ for $b' = 0, 1$.

Now we prove by induction on q that in the first q rounds, the following are the same for $b = 0$ and $b = 1$:

- the inputs of the adversary;
- the oracle queries and answers;
- the number of used elements in $T^{(i)}$ for each i;
- the domain of Σ_i, and $\Sigma_i(s)$ for any s in the domain of Σ_i, except when $i = n+1$ and $s = T_1^{(n+1)}$. (Recall that when $i = n+1$ and $s = T_1^{(n+1)}$, we have $\Sigma_i(s) = \hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_b)$.)

Suppose $q = 0$, it suffices to prove that the above are the same after the Init phase for $b = 0$ or $b = 1$. Note that in these two games, the only difference in the Init phase is when the challenger computes $\hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_b)$. Since $\hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_b)$ is always a new linear combination not yet in the image of Σ_{n+1}, the challenger will always set $\Sigma_1(T_1^{(n+1)}) = \hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_b)$, regardless of b. So the input to the adversary, as well as every $\Sigma_i(s)$ except for $i = n+1$ and $s = T_1^{(n+1)}$, are the same for $b = 0$ or $b = 1$. It is easy to check that for each i, the number of used elements in $T^{(i)}$ are the same for $b = 0$ or $b = 1$.

Now suppose the statement is true for q rounds, we prove it for $q+1$ rounds. By the induction hypothesis, the adversary is in the same status after the q-th round for $b = 0$ and $b = 1$, and thus the query in the $(q+1)$-st round are the same. Suppose this query is $\mathsf{Add}(i, s_i, j, s_j)$, then the challenger will return the label of $\Sigma_i(s_i) + \Sigma_j(s_j)$. Note that $\Sigma_i(s_i) + \Sigma_j(s_j)$ cannot be of the form of $\hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_{b'})$ as shown above. Therefore, before the challenger computes $\Sigma_i(s_i) + \Sigma_j(s_j)$:

- if $\exists s$ such that $\Sigma_i(s_i) + \Sigma_j(s_j) = \Sigma_{i+j}(s)$ for either b, then it cannot be the case that $i + j = n+1$ and $s = T_1^{(n+1)}$. By our induction hypothesis, for the other b, it also holds that $\Sigma_i(s_i) + \Sigma_j(s_j) = \Sigma_{i+j}(s)$;
- if the preimage of $\Sigma_i(s_i) + \Sigma_j(s_j)$ in Σ_{i+j} is not yet defined (for both b), then in both games, $\Sigma_{i+j}^{-1}(\Sigma_i(s_i) + \Sigma_j(s_j))$ will be assigned as the next element of $T^{(i+j)}$. By our induction hypothesis, this element is the same for $b = 0$ or $b = 1$.

Therefore, the oracle answers, the number of used elements in $T^{(i+j)}$, and the changes made to Σ_{i+j} will be the same in the $(q+1)$-st round for $b = 0$ or $b = 1$. □

A direct corollary is the following.

Corollary 4.24. *Suppose that $(1^n, \{X_i\}_{i\in[m]}) \notin$ Exact-Cover. Then any adversary A with Q queries cannot distinguish between $b = 0$ and $b = 1$. That is, for arbitrary $S \in \mathbb{G}^{2Q+n+2}$, $T^{(1)}, T^{(2)}, \ldots, T^{(n+1)} \in [p]^p$,*

$$\begin{aligned}&A^{\mathsf{Add}}\Big(\{\Sigma_{|X_i|}^{-1}(\hat{g}_{X_i})\}_{i\in[m]}, \Sigma_{n+1}^{-1}(\hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_0)), \mathsf{msg}_0, \mathsf{msg}_1\Big)\\ &= A^{\mathsf{Add}}\Big(\{\Sigma_{|X_i|}^{-1}(\hat{g}_{X_i})\}_{i\in[m]}, \Sigma_{n+1}^{-1}(\hat{g}_{[n]} + \Sigma_1(\mathsf{msg}_1)), \mathsf{msg}_0, \mathsf{msg}_1\Big).\end{aligned}$$

We now prove the security of the witness encryption scheme in the oracle world.

Theorem 4.25. *Let $\mathcal{O} = \{\mathcal{O}_\lambda\}_\lambda$ be the oracle world and $\mathsf{Encrypt}^{(-)}$ be the encryption algorithm of the witness encryption scheme constructed in Section 4.3. Suppose that $x := (1^n, \{X_i\}_{i\in[m]}) \notin$ Exact-Cover. Then, for any adversary A that makes at most Q queries, with probability at least $1 - 100m^2/2^{\lambda/4}$ over $\mathsf{Add} \leftarrow \mathcal{O}_\lambda$,*

$$\left|\Pr_r[A^{\mathsf{Add}}(\mathsf{Encrypt}^{\mathsf{Add}}(1^\lambda, x, 0; r)) = 1] - \Pr_r[A^{\mathsf{Add}}(\mathsf{Encrypt}^{\mathsf{Add}}(1^\lambda, x, 1; r)) = 1]\right| < \frac{Q^2}{2^{3\lambda/4}}. \quad (7)$$

Proof. For $\mathsf{Hyb} \in \{\mathsf{Hyb}_0, \mathsf{Hyb}_0', \mathsf{Hyb}_1, \mathsf{Hyb}_2\}$, abusing notation, we denote by $\mathsf{Hyb}(A; b)$ the output of A as adversary in the game Hyb with message b. Note that $\mathsf{Hyb}(A; b)$ is a random variable that depends on Add, r[22] (for $\mathsf{Hyb} = \mathsf{Hyb}_0$) or S, T (for other games Hyb).

The adversary A makes at most Q queries, so by Lemma 4.19, Lemma 4.22, and Corollary 4.24, for any permutations $T^{(1)}, T^{(2)}, \ldots, T^{(n+1)} \in [p]^p$, we have

$$\begin{aligned}
&\Pr_S[\mathsf{Hyb}_0'(A;0) \neq \mathsf{Hyb}_0'(A;1)] \\
\leq\ &\Pr_S[\mathsf{Hyb}_0'(A;0) \neq \mathsf{Hyb}_1(A;0)] + \Pr_S[\mathsf{Hyb}_1(A;0) \neq \mathsf{Hyb}_2(A;0)] + \Pr_S[\mathsf{Hyb}_2(A;0) \neq \mathsf{Hyb}_2(A;1)] + \\
&\Pr_S[\mathsf{Hyb}_2(A;1) \neq \mathsf{Hyb}_1(A;1)] + \Pr_S[\mathsf{Hyb}_1(A;1) \neq \mathsf{Hyb}_0'(A;1)] \\
\leq\ &\frac{2Q(m+3+3Q)}{p} + \frac{(m+3+3Q)^2}{p} + 0 + \frac{(m+3+3Q)^2}{p} + \frac{2Q(m+3+3Q)}{p} \\
\leq\ &\frac{2(m+3+5Q)(m+3+3Q)}{p} \leq \frac{100Q^2m^2}{2^\lambda}.
\end{aligned}$$

By Lemma 4.18, it follows that

$$\Pr_{\mathsf{Add}, r}[\mathsf{Hyb}_0(A;0) \neq \mathsf{Hyb}_0(A;1)] = \Pr_{S,T}[\mathsf{Hyb}_0'(A;0) \neq \mathsf{Hyb}_0'(A;1)] \leq \frac{100Q^2m^2}{2^\lambda}.$$

By Markov's inequality,

$$\Pr_{\mathsf{Add}}\left[\Pr_r[\mathsf{Hyb}_0(A;0) \neq \mathsf{Hyb}_0(A;1)] \leq \frac{Q^2}{2^{3\lambda/4}}\right] \geq 1 - \frac{100m^2}{2^{\lambda/4}},$$

which implies the theorem statement. □

4.5 NP-Hardness of GapMOCSP and GapMINcKT

We now prove the **NP**-hardness of GapMOCSP, using our construction of witness encryption in oracle world and Theorem 4.5.

Theorem 4.1. GapMOCSP *is* **NP***-hard under polynomial-time randomised mapping reductions. More precisely, for every constant* $\epsilon > 0$*, there is a polynomial-time randomised mapping reduction from an* **NP***-complete language to the following promise problem:*

$$\begin{aligned}
\Pi_{\text{YES}} &:= \{(f, O) : \mathsf{CC}^O(f) \leq 2^{\epsilon n}\}, \\
\Pi_{\text{NO}} &:= \{(f, O) : \mathsf{CC}^O_{1/2 - 2^{-0.3n}}(f) > 2^{0.3n}\}.
\end{aligned}$$

Here f is the truth table of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and oracle O has size $2^{\Theta(n)}$.

Proof. First, by Theorem 4.25, there is a distribution $\mathcal{O}_{\mathsf{GGSW}}$ over oracles $O : \{0,1\}^{O(\lambda)} \to \{0,1\}$ with a witness encryption scheme that is weakly secure against programs of query complexity $2^{\lambda/4}$ with advantage $2^{-\lambda/4}$. By Claim 4.17, the oracle world is sampleable in $\mathrm{poly}(2^\lambda)$ time.

By Theorem 4.14, there is a distribution $\mathcal{O}'_{\mathsf{GGSW}}$ over oracles $O' : \{0,1\}^{O(\lambda)} \to \{0,1\}$ with a witness encryption scheme that is *strongly* secure against programs of size $2^{\Omega(\lambda)}$ and query complexity $2^{\Omega(\lambda)}$ with advantage $2^{-\Omega(\lambda)}$. This oracle world is also sampleable in $\mathrm{poly}(2^\lambda)$ time.

Therefore, it follows from Theorem 4.5 that GapMOCSP is **NP**-hard under polynomial-time randomised mapping reduction. □

[22] Recall that Add contains information about $\{\sigma_i\}$ and r is the random string used for sampling $\{g_i\}, \mathsf{msg}_0, \mathsf{msg}_1$.

Theorem 4.2. GapMINcKT *is* **NP***-hard under polynomial-time randomised mapping reductions. More precisely, for every constant* $\epsilon \in (0,1)$ *and* $\gamma > 1$*, there is a polynomial-time randomised mapping reduction from an* **NP***-complete language to the following promise problem:*

$$\Pi_{\text{Yes}} := \{(x,y) : \mathrm{K}^{N^{1+\epsilon}}(x \mid y) \le N^{\epsilon}\},$$
$$\Pi_{\text{No}} := \{(x,y) : \mathrm{K}^{N^{\gamma}}(x \mid y) > N^{1-\epsilon}\}.$$

Here x has length N and y has length $\mathrm{poly}(N^{\gamma})$*.*

Proof Sketch. Same as above, except that we replace GapMOCSP with GapMINcKT and Theorem 4.5 with Theorem 4.9. □

5 Unconditional NP-Hardness of Gap-mvMCSP^O

In this section, we show that with probability 1 over a random oracle O, Gap-mvMCSP^O is **NP**-hard to approximate under **QuasiP**^{O} reductions. Perhaps curiously, the core of our reduction is the CS proofs protocol by Micali [Mic00].

Theorem 5.1. *With probability* 1 *over a random oracle O,* Gap-mvMCSP^O *is* **NP***-hard under deterministic* **TIME**$[2^{\mathrm{polylog}(n)}]^O$ *reductions.*

Moreover, for any language $L \in$ **NP** *and any constant $t \ge 1$, there is a constant c_1 and a mapping computable in deterministic $2^{\mathrm{polylog}(n)}$ time with an O oracle, that maps an instance $x \in \{0,1\}^n$ to a table T satisfying the following:*

$$x \in L \implies \mathsf{CC}^O(T) \le s := O(n^{c_1});$$
$$x \notin L \implies \mathsf{CC}^O_{2^{-\log^t s}}(T) \ge 2^{\log^t s}.$$

Roadmap of this section. A large part of this section is devoted to the exposition of CS proofs and its security; namely, Section 5.1 provides an introduction to CS proofs, Section 5.2 and 5.3 proves the security of CS proofs, and finally we prove Theorem 5.1 in Section 5.4. Like in Section 4, here we also need the "strong" security of CS proofs; in this section we call it "security in the *common random string* model".[23] The security proof for the plain random oracle model is provided in Section 5.2, and the security proof for the common random string model is provided in Section 5.3.

5.1 Description of CS Proofs

We first provide a self-contained description of CS proofs. We begin by introducing the two necessary ingredients: PCPs and Merkle trees.

5.1.1 Probabilistically Checkable Proofs

Theorem 5.2 (PCP theorem [AS98, ALM+98])**.** *For any language $L \in$* **NP***, there exists a PCP verifier for L, denoted as V_{PCP}, such that the following holds. The verifier takes two inputs x and $r \in \{0,1\}^{O(\log|x|)}$, runs in* $\mathrm{poly}(|x|)$ *time, makes* $O(1)$ *oracle queries to a proof string $\pi \in \{0,1\}^{\mathrm{poly}(|x|)}$, and either accepts or rejects. The verifier satisfies the following property:*

[23]The common random string model is essentially equivalent to salting (see Section 4.2). Unfortunately, we became aware of the beautiful work on [CDGS18] only after this section is completed, thus we prove the security of CS proofs in the common random string model by a somewhat sophisticated compression argument. We believe that it is possible to invoke [CDGS18] and obtain a simpler proof.

- *Completeness: For any $x \in L$, there is a proof π such that*

$$\Pr_{r \leftarrow \{0,1\}^{O(\log |x|)}} [V_{\mathrm{PCP}}^{\pi}(x, r) \text{ accepts}] = 1.$$

- *Soundness: Let r be uniformly random in $\{0,1\}^{O(\log |x|)}$. For any $x \notin L$ and any purported proof $\pi \in \{0,1\}^{\mathrm{poly}(|x|)}$,*

$$\Pr_{r \leftarrow \{0,1\}^{O(\log |x|)}} [V_{\mathrm{PCP}}^{\pi}(x, r) \text{ accepts}] \le 1/2.$$

By parallel repetition (i.e., repeatedly executing V_{PCP} on fresh random bits), the soundness error can be reduced to arbitrarily small. Let $c \ge 1$ be a parameter, we obtain the following PCP by repeating the verifier c times over independent random seeds:

Corollary 5.3. *For any language $L \in \mathbf{NP}$ and any parameter $c \ge 1$, there exists a PCP verifier for L, denoted as V_{PCP}, with the following parameters:*

$$\begin{aligned}
r_{\mathrm{PCP}} &:= O(c \log |x|), && \textit{(randomness complexity)}\\
q_{\mathrm{PCP}} &:= O(c), && \textit{(query complexity)}\\
\ell_{\mathrm{PCP}} &:= \mathrm{poly}(|x|), && \textit{(proof length)}\\
s_{\mathrm{PCP}} &:= 2^{-c}. && \textit{(soundness parameter)}
\end{aligned}$$

The verifier takes two inputs x and $r \in \{0,1\}^{r_{\mathrm{PCP}}}$, runs in $\mathrm{poly}(|x|)$ time, makes q_{PCP} oracle queries to a proof string $\pi \in \{0,1\}^{\ell_{\mathrm{PCP}}}$, and either accepts or rejects. The verifier satisfies the following property:

- *Completeness: For any $x \in L$, there is a proof $\pi \in \{0,1\}^{\ell_{\mathrm{PCP}}}$ such that*

$$\Pr_{r \leftarrow \{0,1\}^{r_{\mathrm{PCP}}}} [V_{\mathrm{PCP}}^{\pi}(x, r) \text{ accepts}] = 1.$$

- *Soundness: Let r be uniformly random in $\{0,1\}^{r_{\mathrm{PCP}}}$. For any $x \notin L$ and any purported proof $\pi \in \{0,1\}^{\ell_{\mathrm{PCP}}}$,*

$$\Pr_{r \leftarrow \{0,1\}^{r_{\mathrm{PCP}}}} [V_{\mathrm{PCP}}^{\pi}(x, r) \text{ accepts}] \le s_{\mathrm{PCP}}.$$

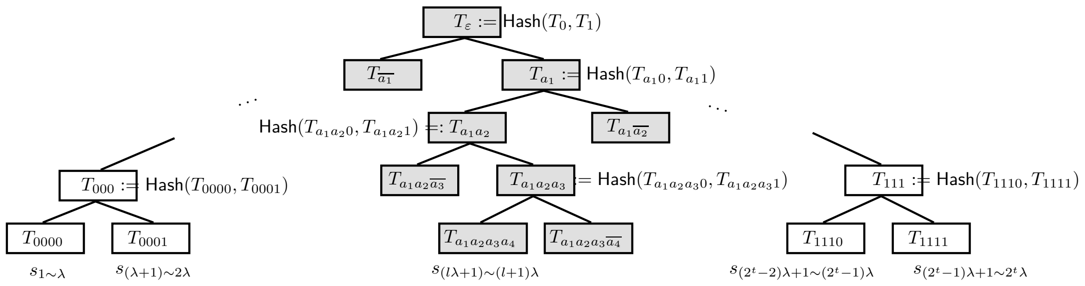
5.1.2 Merkle Trees

Another important component of the CS proofs is Merkle trees [Mer89], which allows the prover to *commit* to a long string s (e.g., the PCP proof); later, for every index i and $b = s_i$, the prover can generate a short proof that s_i is indeed equal to b.

The Merkle tree will use a hash function $\mathsf{Hash} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$, and it is computationally binding as long as Hash is *collision resistant*, which roughly means it is computationally hard to find two different strings x, y such that $\mathsf{Hash}(x) = \mathsf{Hash}(y)$.

Construction 5.4 (Merkle Tree)**.** Let $\mathsf{Hash} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ be an oracle (think of Hash as a collision-resistant hash function). Let s be a string of length $2^t\lambda$; if the length of s is not of the form $2^t\lambda$, we can pad zeros after s. The Merkle tree for s is a depth-t binary tree T defined as follows.

We index the nodes of T as follows: the root is indexed by the empty string ε and the children of the node with index α are indexed by $\alpha 0$ and $\alpha 1$ respectively. Each node α is associated with a value T_α. For a t-bit number $l = \overline{a_t a_{t-1} \cdots a_1}$ (which corresponds to the $(l+1)$-st leaf node), $T_{a_1 a_2 \cdots a_t}$ is the λ-bit string $s_{l\lambda+1} s_{l\lambda+2} \cdots s_{(l+1)\lambda}$. (That is, the $(l+1)$-st part of s if we divide it into portions of length λ.) For each non-leaf node α, we define T_α to be the hash of the concatenation of the values associated with its two children, i.e. $T_\alpha = \mathsf{Hash}(T_{\alpha 0} \circ T_{\alpha 1})$. We call T_ε the *root* of the Merkle tree.



$T_\varepsilon :=$ Hash(T_0, T_1)
$T_{\overline{a_1}}$
$T_{a_1} :=$ Hash(T_{a_10}, T_{a_11})
Hash$(T_{a_1a_20}, T_{a_1a_21}) =: T_{a_1a_2}$
$T_{a_1\overline{a_2}}$
$T_{000} :=$ Hash(T_{0000}, T_{0001})
$T_{a_1a_2\overline{a_3}}$
$T_{a_1a_2a_3} :=$ Hash$(T_{a_1a_2a_30}, T_{a_1a_2a_31})$
$T_{111} :=$ Hash(T_{1110}, T_{1111})
T_{0000}
T_{0001}
$T_{a_1a_2a_3a_4}$
$T_{a_1a_2a_3\overline{a_4}}$
T_{1110}
T_{1111}
$s_{1\sim\lambda}$
$s_{(\lambda+1)\sim 2\lambda}$
$s_{(l\lambda+1)\sim(l+1)\lambda}$
$s_{(2^t-2)\lambda+1\sim(2^t-1)\lambda}$
$s_{(2^t-1)\lambda+1\sim 2^t\lambda}$

Figure 1: Example of a Merkle tree where $t = 4$. Grey nodes are nodes in the authentication path of the leaf node $T_{a_1a_2a_3a_4}$.

The prover commits to s by announcing the root value T_ε to public. Suppose that the prover wants to convince the verifier what the value of s_i is. The prover first finds the corresponding leaf $a_1a_2 \cdots a_t$, where $\lfloor (i-1)/\lambda \rfloor = \overline{a_t a_{t-1} \cdots a_1}$ (that is, s_i is in the $(\overline{a_t a_{t-1} \cdots a_1} + 1)$-st part of s). Then the prover sends the verifier all the nodes on the path from the leaf to the root

$$T_{a_1a_2\cdots a_t}, T_{a_1a_2\cdots a_{t-1}}, \ldots, T_{a_1},$$

along with their siblings

$$T_{a_1a_2\cdots a_{t-1}(1-a_t)}, T_{a_1a_2\cdots a_{t-2}(1-a_{t-1})}, \ldots, T_{(1-a_1)};$$

we call this set of Merkle tree nodes the *authentication path*. The verifier then checks for any $0 \le j \le t-1$ whether $T_{a_1a_2\cdots a_j} = \mathsf{Hash}(T_{a_1a_2\cdots a_{j-1}0} \circ T_{a_1a_2\cdots a_{j-1}1})$. If all of these equations hold and the claimed value of s_i is consistent with $T_{a_1a_2\cdots a_t}$, then the verifier accepts; otherwise, the verifier rejects.

The following proposition shows that the Merkle tree is computationally binding. Roughly speaking, if the prover could find two different strings s^1 and s^2 with the same root value, then the prover could also find a collision of Hash.

Proposition 5.5. *There exists an algorithm* FindCol *that satisfies the following.*

- *The input consists of two strings $s^1, s^2 \in \{0,1\}^\lambda$ that are two leaves at the same location of two Merkle trees, along with their respective authentication path, under the promise that the authentication paths are valid under an oracle* Hash *and the two Merkle tree roots are equal but $s^1 \neq s^2$.*
- *The output consists of two strings $r^1, r^2 \in \{0,1\}^{2\lambda}$ such that $r^1 \neq r^2$ but* $\mathsf{Hash}(r^1) = \mathsf{Hash}(r^2)$.
- FindCol *runs in deterministic* $\mathrm{poly}(t, \lambda)$ *time.*

Proof. Let T^1 and T^2 be the Merkle trees of s^1 and s^2 under the oracle Hash. Let $a_1a_2 \cdots a_t$ be the indices of the leaves in the two Merkle trees. From the promise we have $T^1_\varepsilon = T^2_\varepsilon$ but $T^1_{a_1a_2\cdots a_t} \neq T^2_{a_1a_2\cdots a_t}$.

Then, FindCol computes the smallest positive integer j such that $T^1_{a_1a_2\cdots a_j} \neq T^2_{a_1a_2\cdots a_j}$. Since $T^1_{a_1a_2\cdots a_t} \neq T^2_{a_1a_2\cdots a_t}$ but $T^1_\varepsilon = T^2_\varepsilon$, j is well-defined. Also note that the authentication paths for $a_1a_2 \cdots a_t$ in both Merkle trees are provided as input, so we can easily compute j. Because of the minimality of j, $T^1_{a_1a_2\cdots a_{j-1}} = T^2_{a_1a_2\cdots a_{j-1}}$.

FindCol just outputs $r^1 := T^1_{a_1a_2\cdots a_{j-1}0} \circ T^1_{a_1a_2\cdots a_{j-1}1}$ and $r^2 := T^2_{a_1a_2\cdots a_{j-1}0} \circ T^2_{a_1a_2\cdots a_{j-1}1}$. It is easy to see that $r^1 \neq r^2$, $\mathsf{Hash}(r^1) = \mathsf{Hash}(r^2)$, and that FindCol runs in $\mathrm{poly}(2^t, \lambda)$ time. □

5.1.3 Kilian's Protocol

This subsection provides an exposition of Kilian's protocol [Kil92], a four-message (interactive) succinct argument for **NP**. Kilian's protocol is a crucial ingredient of CS proofs.

Consider a language $L \in \mathbf{NP}$ and $x \in L \cap \{0,1\}^n$. A prover can convince a verifier that $x \in L$ by providing an **NP** witness $w \in \{0,1\}^{\mathrm{poly}(n)}$, but this requires $\mathrm{poly}(n)$ bits of communication. Using PCPs, an honest prover can convince a verifier that $x \in L$ within communication complexity q_{PCP} (which is typically $O(\log n)$): the verifier runs the PCP verifier VPCP to obtain some query indices $i[1], i[2], \dots, i[q_{\mathrm{PCP}}]$, and sends these indices to the prover. Let Π be the PCP proof, the prover sends $\Pi[i[1]], \Pi[i[2]], \dots, \Pi[i[q_{\mathrm{PCP}}]]$ back to the verifier, which the verifier checks using the PCP verifier V_{PCP}. However, this naïve protocol cannot prevent malicious provers from cheating: a malicious prover, after seeing the indices $i[1], i[2], \dots, i[q_{\mathrm{PCP}}]$, can simply make up some answer bits that make V_{PCP} accept.

To deal with this problem, Kilian's protocol [Kil92] requires the prover to *commit* to a proof before the prover sees $i[1], i[2], \dots, i[q_{\mathrm{PCP}}]$, using a Merkle tree. In the first round, the prover builds a Merkle tree over the PCP proof π, and sends its root to the verifier (as a commitment to π). The protocol then continues as above, except that the prover also sends the authentication paths of each $\Pi[i[j]]$ to the verifier, and the verifier will also check whether the authentication path is consistent with both Hash (the oracle used in Merkle tree) and the bits of PCP proof that the verifier queries.

Formal description. We define a function $\mathsf{Verify}^O(\mathsf{Root}, \mathsf{seed}, \pi)$ as follows. The function receives three inputs and an oracle:

- $\mathsf{Root} \in \{0,1\}^\lambda$ is the root of the Merkle tree, committed by the prover;
- $\mathsf{seed} \in \{0,1\}^{r_{\mathrm{PCP}}}$ is the PCP randomness;
- π is the succinct proof that the prover produces, which consists of the authentication paths $(\mathsf{auth}_1, \dots, \mathsf{auth}_{q_{\mathrm{PCP}}})$ for the answers of the PCP queries; and
- O is the Hash oracle used in the Merkle tree.

For each $j \in [q_{\mathrm{PCP}}]$, let $i[j]$ denote the index of the PCP proof such that auth_j is an authentication path for the $i[j]$-th bit, and let $\Pi[i[j]]$ denote the $i[j]$-th bit of the PCP proof as claimed in auth_j. The function returns 1 if all of the following are true:

- on PCP randomness seed, the PCP verifier indeed reads the $(i[1], i[2], \dots, i[q_{\mathrm{PCP}}])$-th bit of the PCP proof, and accepts if it receives $(\Pi[i_1], \Pi[i_2], \dots, \Pi[i[q_{\mathrm{PCP}}]])$;
- each authentication path is consistent with O and has Root as the Merkle tree root.

Formally, Kilian's protocol proceeds as follows. Let $O : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ be a (hash) oracle. Assuming $x \in L$ and $\Pi \in \{0,1\}^{\ell_{\mathrm{PCP}}}$ is the PCP proof for x, we describe the behaviours of both the verifier and the honest prover. (However, keep in mind that when $x \notin L$, the malicious prover could behave arbitrarily.)

1. The prover computes a Merkle tree over Π, using O as the Hash function. Let $\mathsf{Root} \in \{0,1\}^\lambda$ be the root of this Merkle tree, the prover commits to π by sending Root to the verifier.
2. Then, the verifier randomly samples $\mathsf{seed} \in \{0,1\}^{r_{\mathrm{PCP}}}$ and sends it to the prover.
3. The prover executes the PCP verifier to obtain the indices $(i[1], i[2], \dots, i[q_{\mathrm{PCP}}])$. For each $j \in [q_{\mathrm{PCP}}]$, the prover sends the authentication path auth_j to the verifier, which consists of all the nodes on the path from the leaf containing $\Pi[i[j]]$ to the root, as well as the siblings of these nodes. Let $\pi := (\mathsf{auth}_1, \mathsf{auth}_2, \dots, \mathsf{auth}_{q_{\mathrm{PCP}}})$.
4. Finally, the verifier accepts if and only if $\mathsf{Verify}^O(\mathsf{Root}, \mathsf{seed}, \pi) = 1$.

5.1.4 Micali's CS Proofs

By combining PCPs and Merkle trees, Kilian's protocol allows a potentially malicious prover to convince the verifier of the membership of an instance x in a language $L \in \mathbf{NP}$. However, this protocol is interactive, and it requires 3 messages between the prover and the verifier.

Fortunately, it is possible to convert it to a non-interactive protocol by applying the Fiat-Shamir heuristic [FS86]. Recall that Kilian's protocol is *public-coin*, i.e., the verifier's message is simply a uniformly random string. To apply the Fiat-Shamir heuristic, we replace this random string with the hash value of the previous prover message, which in our case is Root. Note that we need to use a different hash function from the one used for the Merkle tree; here we use another oracle to provide the hash function. If the hash function is "random" enough, the hash value can be regarded as uniformly random bits (this intuition will be formalised in Theorem 5.9). Now the prover could simulate the verifier and compute the indices $(i[1], i[2], \ldots, i[q])$. The prover then continues as in Kilian's protocol, sending the verifier the corresponding bits of the PCP proof Π and their authentication paths, and in addition the Merkle tree root and the query indices. The verifier does the same checks in Kilian's protocol, and in addition, checks whether the query indices are consistent with the hash value of the Merkle tree root.

The protocol above is called Micali's CS proofs [Mic00]. Below is a formal description.

Formal description. As mentioned before, the CS proofs protocol requires two oracles. The first oracle is used for the **M**erkle **T**ree, so we call it $O^{\mathsf{MT}} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$; the second oracle is used for the **F**iat-**S**hamir heuristic, mapping the first prover message (Root) to the verifier message (seed), thus we call it $O^{\mathsf{FS}} : \{0,1\}^{\lambda} \to \{0,1\}^{r_{\text{PCP}}}$.

We assume $x \in L$ and $\Pi \in \{0,1\}^{\ell_{\text{PCP}}}$ is the PCP proof of x. We describe the behaviours of both the verifier and the honest prover. (Again, keep in mind that if $x \notin L$, then the malicious prover could behave arbitrarily.)

1. The prover computes a Merkle tree over Π, using O^{MT} as the Hash function. Let $\mathsf{Root} \in \{0,1\}^{\lambda}$ be the root of this Merkle tree.

2. The prover computes $\mathsf{seed} := O^{\mathsf{FS}}(\mathsf{Root})$ as the PCP randomness.

3. The prover executes the PCP verifier to obtain the indices $(i[1], i[2], \ldots, i[q_{\text{PCP}}])$. Let auth_j denote the authentication path corresponding to $\Pi[i[j]]$, the prover obtains a proof string $\pi := (\mathsf{auth}_1, \mathsf{auth}_2, \ldots, \mathsf{auth}_{q_{\text{PCP}}})$.

4. The prover sends (Root, π) to the verifier. The verifier computes $\mathsf{seed} := O^{\mathsf{FS}}(\mathsf{Root})$ and accepts if and only if $\mathsf{Verify}^{O^{\mathsf{MT}}}(\mathsf{Root}, \mathsf{seed}, \pi) = 1$.

5.1.5 CS Proofs in the Common Random String Model

Section 5.1.4 describes the CS proofs protocol in the plain random oracle model. The security of CS proofs in this model states that if $x \notin L$, then for every size-S circuit C that implements a malicious prover, w.h.p. over random oracles O^{MT} and O^{FS}, C cannot convince the verifier that $x \in L$. However, to reduce L to Gap-mvMCSP, we need security in the *common random string* (CRS) model.

In the CRS model, before the protocol starts, a common random string $\mathsf{crs} \leftarrow \{0,1\}^{O(\lambda)}$ is announced to the public; then the prover sends a proof (Root, π) and the verifier verifies it, as usual. We aim at the following stronger security requirement: if $x \notin L$, then the following is true w.h.p. over random oracles O^{MT} and O^{FS}. For *every* size-S circuit C that implements a malicious prover, w.h.p. over crs, C cannot convince the verifier that $x \in L$.

Notice the change of the quantifier here: in the plain model we only require that any malicious prover fails w.r.t. a random oracle, but for each random oracle there could be a malicious prover that cheats successfully. (For example, this prover could hardcode a pair of collisions of O^{MT}.) In the CRS model, we require that for most oracles O^{MT} and O^{FS}, *every* size-S circuit that implements a malicious prover cannot cheat, where "cheat" here means cheating w.h.p. over crs.

Formal description. Let $k = O(\lambda)$ denote the length of the crs. In one sentence, the CRS model consists of 2^k copies of the plain model, one for each possible crs. More formally, we have two random oracles $O^{\mathsf{MT}} : \{0,1\}^k \times \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ and $O^{\mathsf{FS}} : \{0,1\}^k \times \{0,1\}^\lambda \to \{0,1\}^{r_{\mathrm{PCP}}}$. Let $x \in L$ and $\Pi \in \{0,1\}^{\ell_{\mathrm{PCP}}}$ be the PCP proof for x, the behaviours of the honest prover and the verifier are as follows:

1. The verifier chooses $\mathsf{crs} \leftarrow \{0,1\}^k$ uniformly at random and sends it to the prover.
2. The prover computes a Merkle tree over Π, using $O^{\mathsf{MT}}(\mathsf{crs}, -)$ as the Hash function. Let $\mathsf{Root} \in \{0,1\}^\lambda$ be the root of this Merkle tree.
3. The prover computes $\mathsf{seed} := O^{\mathsf{FS}}(\mathsf{crs}, \mathsf{Root})$ as the PCP randomness.
4. The prover executes the PCP verifier to obtain $(i[1], i[2], \dots, i[q_{\mathrm{PCP}}])$, computes the authentication paths auth_j (still under the oracle $O^{\mathsf{MT}}(\mathsf{crs}, -)$), and obtains the proof string $\pi := (\mathsf{auth}_1, \mathsf{auth}_2, \dots, \mathsf{auth}_{q_{\mathrm{PCP}}})$.
5. The prover sends (Root, π) to the verifier. The verifier computes $\mathsf{seed} := O^{\mathsf{FS}}(\mathsf{crs}, \mathsf{Root})$ and accepts if and only if $\mathsf{Verify}^{O^{\mathsf{MT}}(\mathsf{crs},-)}(\mathsf{Root}, \mathsf{seed}, \pi) = 1$.

From the above description, it is easy to see that if $x \in L$, there is always a small-size prover that convinces the verifier:

Theorem 5.6. *Suppose $x \in L$. For every oracles O^{MT} and O^{FS}, there is an oracle circuit A of size $\mathrm{poly}(n, k, \lambda, \ell_{\mathrm{PCP}}, r_{\mathrm{PCP}})$ such that*

$$\Pr_{\mathsf{crs} \leftarrow \{0,1\}^k}\left[\mathsf{Verify}^{O^{\mathsf{MT}}(\mathsf{crs},-)}(\mathsf{Root}, \mathsf{seed}, \pi) = 1 \;\middle|\; \begin{array}{c} (\mathsf{Root}, \pi) \leftarrow A^{O^{\mathsf{MT}}, O^{\mathsf{FS}}}(\mathsf{crs}) \\ \mathsf{seed} \leftarrow O^{\mathsf{FS}}(\mathsf{crs}, \mathsf{Root}) \end{array}\right] = 1.$$

It remains to prove the soundness of CS proofs, i.e., if $x \notin L$, then any prover of size $2^{o(\lambda)}$ fails to convince the verifier that $x \in L$. It turns out that the CRS model creates some technical difficulties for proving soundness. If the adversary, on input crs, cannot access oracles $O^{\mathsf{MT}}(\mathsf{crs}', -)$ or $O^{\mathsf{FS}}(\mathsf{crs}', -)$ for $\mathsf{crs}' \neq \mathsf{crs}$, then the 2^k plain-model CS proofs will be "independent". Thus, we can use Chernoff bound to show that there is only a tiny fraction of crs for which the malicious prover cheats successfully. However, these plain-model CS proofs are *not* independent, and we need to handle the case that the malicious prover, on input crs, could query arbitrary $O^{\mathsf{MT}}(\mathsf{crs}', -)$ or $O^{\mathsf{FS}}(\mathsf{crs}', -)$.

Intuitively, we need to find a large subset of crs that is "independent". However, it is unclear how to find these crs even when given the malicious prover (implemented as a circuit). Our solution is to use a *compression* argument: fix O^{MT} and O^{FS}, if the protocol is *not* secure, then we can compress O^{MT} and O^{FS} in much less than $|O^{\mathsf{MT}}| + |O^{\mathsf{FS}}|$ bits. Therefore, if O^{MT} and O^{FS} are random oracles, then the protocol should be secure with high probability. After we fixed O^{MT} and O^{FS}, we could run the prover on each crs and analyse its computational history. Then we find a significant fraction of crs such that their computational histories are "independent", and such that the malicious prover cheats successfully on each crs. If the malicious prover is successful on crs, then we can compress $O^{\mathsf{MT}}(\mathsf{crs}, -)$ and $O^{\mathsf{FS}}(\mathsf{crs}, -)$; since these crs we pick are "independent", compressing one crs does not influence the other strings crs'.

To implement this strategy one needs to re-prove the security of CS proofs in the plain model, using a compression argument. This is executed in Section 5.2; the proof is equivalent to the one in [Mic00], but this compression argument will make the later proofs in the CRS model cleaner. Then, in Section 5.3, we prove the security in the CRS model.

5.2 Security of CS Proofs in the Plain Model

Collisions. Let O be an oracle and $A^{(-)}(x)$ be an oracle circuit with input x. We say $A^O(x)$ *finds a collision* for O if there are two queries $O(a)$ and $O(b)$ made by A on input x such that $a \neq b$ and $O(a) = O(b)$. A collision of $A^O(x)$ is the pair of integers (i, j), where $i < j$, the i-th gate of $A^O(x)$ is an oracle gate that queries $O(a)$, the j-th gate queries $O(b)$, $a \neq b$, and $O(a) = O(b)$. We will frequently use the notion of *the first collision of* $A^O(x)$, which is defined as the lexicographically smallest pair (i, j) which is a collision of $A^O(x)$.

If we have more than one oracles $O_1, O_2, \dots$, then we define $A^{O_1,O_2,\cdots}(x)$ *finds a collision for* O_1 and *the first collision of* $A^{O_1,O_2,\cdots}(x)$ *for* O_1 analogously.

We need the following result stating that a random oracle is collision-resistant (in the plain model). This result also serves as a warm-up of the compression method.

Theorem 5.7 (Random Functions are Collision Resistant)**.** *There is a polynomial p such that the following holds. Let $O : \{0,1\}^n \to \{0,1\}^m$ be an oracle. Let A be an oracle circuit of size $S \leq o(2^{m/2})$ such that $A^O(1^n, 1^m)$ finds a collision of O.*

Let $\ell := 2^n m$ and $\mathsf{info} := (n, m, S)$. Then

$$\mathrm{K}^{p(\ell)}(O \mid A, \mathsf{info}) \leq \ell - (m - 2\log S) + O(1).$$

Proof. We describe our compressed encoding of O. Let (i, j) be the first collision of $A^O(1^n, 1^m)$. We write down the indices $i, j \in [S]$ using $2\log S$ bits. Then we simulate $A^O(1^n, 1^m)$. Whenever A makes a query $O(x)$:

- If we have seen this query before, then we have also written down its answer. We do not need to write down anything.
- If this query is made by the j-th gate, then $x = b$, we have already recorded $O(a)$ (the answer of the i-th gate), but we have not recorded $O(b)$ yet. As $O(A) = O(b)$, we do not need to write down anything.
- Otherwise, we write down the answer to this query.

Suppose A made Q distinct queries to O. Then we wrote down $Q - 1$ answers in the above process, costing $(Q - 1)m$ bits. Finally, we write down the rest $2^n - Q$ entries of O, costing $(2^n - Q)m$ bits. It is easy to see that given A, info, and this encoding of O, we can recover O in $\mathrm{poly}(\ell)$ time.

The length of this encoding is

$$2\log S + (Q - 1)m + (2^n - Q)m + O(1) \leq 2^n m - (m - 2\log S) + O(1). \qquad \square$$

Theorem 5.8 (Security of Kilian's Protocol)**.** *Let $x \notin L$. Let $O : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ be an oracle. Given oracle circuits P, Q of size at most S such that*

$$\Pr_{\mathsf{seed} \leftarrow \{0,1\}^{r_{\mathrm{PCP}}}}\left[\mathsf{Verify}^O(\mathsf{Root}, \mathsf{seed}, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{Root} \leftarrow P^O(1^\lambda) \\ \pi \leftarrow Q^O(1^\lambda, \mathsf{seed}) \end{array}\right] \geq \epsilon, \tag{8}$$

we can compute the description of a randomised oracle circuit $\mathsf{FindCol'}^O(r)$ of size $S \cdot \mathrm{poly}(q_{\mathrm{PCP}}, \lambda)$ in $2^{O(\lambda)}$ time, such that

$$\Pr_r\left[\mathsf{FindCol'}^O(r) \textit{ find a collision of } O\right] \geq 2(\epsilon - s_{\mathrm{PCP}})^2/\ell_{\mathrm{PCP}}. \tag{9}$$

Proof. The first step is to construct a candidate PCP proof Π for x. Since $x \notin L$, the PCP verifier accepts Π with probability at most s_{PCP}; the probability of finding a collision will be related to the probability that Kilian's protocol accepts but the PCP verifier does not (which is $\geq \epsilon - s_{\mathrm{PCP}}$).

Recall that for each $j \in [q]$, $i[j]$ is the index of the PCP proof corresponding to auth_j, and $\Pi[i[j]]$ is the $i[j]$-th bit of the PCP proof claimed by auth_j. For every bit i, we say that $Q^O(1^\lambda, \mathsf{seed})$ *claims* $\Pi[i] = 0$ (or 1) if there is some $j \in [q]$ such that $i[j] = i$ and $\Pi[i[j]] = 0$ (or 1). For $i \in [\ell_{\mathrm{PCP}}]$ and $b \in \{0,1\}$, let $P_{i,b}$ be the probability (over seed) that $Q^O(1^\lambda, \mathsf{seed})$ claims $\Pi[i] = b$ and the verifier accepts. That is,

$$P_{i,b} := \Pr_{\mathsf{seed} \leftarrow \{0,1\}^{r_{\mathrm{PCP}}}} \left[\mathsf{Verify}^O(\mathsf{Root}, \mathsf{seed}, \pi) = 1 \wedge Q^O(1^\lambda, \mathsf{seed}) \text{ claims } \Pi[i] = b \right].$$

We construct a PCP proof Π. For each $i \in [\ell_{\mathrm{PCP}}]$, if $P_{i,0} > P_{i,1}$, then we let $\Pi[i] = 0$; otherwise we let $\Pi[i] = 1$. If $\mathsf{Verify}(\mathsf{Root}, \mathsf{seed}, \pi) = 1$, then there are only two possibilities:

- either the PCP verifier accepts Π when given seed as randomness,
- or there is $i \in [\ell_{\mathrm{PCP}}]$ and $b = 1 - \Pi[i]$ such that $Q^O(1^\lambda, \mathsf{seed})$ claims $\Pi[i] = b$.

The first bullet happens w.p. at most s_{PCP}, therefore the second bullet happens w.p. at least $\epsilon - s_{\mathrm{PCP}}$. Thus, the LHS of the following equation, which upper bounds the probability that the second bullet happens, is also at least $\epsilon - s_{\mathrm{PCP}}$:

$$\sum_{i \in [\ell_{\mathrm{PCP}}]} \min\{P_{i,0}, P_{i,1}\} \geq \epsilon - s_{\mathrm{PCP}}. \tag{10}$$

Now we design a randomised oracle circuit $\mathsf{FindCol}'$ that attempts to find a collision of O. Consider the following experiment: randomly sample $\mathsf{seed}_1, \mathsf{seed}_2 \leftarrow \{0,1\}^{r_{\mathrm{PCP}}}$, and invoke Q^O on both seeds to obtain the authentication paths

$$\begin{aligned} \pi_1 &= (\mathsf{auth}_{1,1}, \mathsf{auth}_{1,2}, \ldots, \mathsf{auth}_{1,q}) := Q^O(1^\lambda, \mathsf{seed}_1), \\ \pi_2 &= (\mathsf{auth}_{2,1}, \mathsf{auth}_{2,2}, \ldots, \mathsf{auth}_{2,q}) := Q^O(1^\lambda, \mathsf{seed}_2). \end{aligned}$$

Let $\mathcal{C}$ (for "collision") denote the event that there is some $i \in [\ell_{\mathrm{PCP}}]$ and $b \in \{0,1\}$ such that both the following hold:

- $Q^O(1^\lambda, \mathsf{seed}_1)$ claims $\Pi[i] = b$ and $\mathsf{Verify}^O(\mathsf{Root}, \mathsf{seed}_1, \pi_1) = 1$;
- $Q^O(1^\lambda, \mathsf{seed}_2)$ claims $\Pi[i] = 1 - b$ and $\mathsf{Verify}^O(\mathsf{Root}, \mathsf{seed}_2, \pi_2) = 1$.

Let $\mathsf{FindCol}$ be the algorithm described in Proposition 5.5, under the oracle $\mathsf{Hash} = O$. Consider the following circuit $\mathsf{FindCol}'$:

1. The circuit $\mathsf{FindCol}'$ uses seed_1 and seed_2 as randomness.
2. Then, it invokes Q^O twice to obtain π_1 and π_2.
3. Then, it invokes $\mathsf{FindCol}$ on each pair of $(\mathsf{auth}_{1,i}, \mathsf{auth}_{2,j})$, no matter whether $\mathsf{auth}_{1,i}$ and $\mathsf{auth}_{2,j}$ corresponds to Merkle tree leaves at the same location.

Whenever $\mathcal{C}$ happens, $\mathsf{FindCol}'$ finds a collision of O. On the other hand,

$$\begin{aligned} \Pr_{\mathsf{seed}_1, \mathsf{seed}_2}[\mathcal{C}] &\geq \sum_{i \in [\ell_{\mathrm{PCP}}]} 2 P_{i,0} P_{i,1} \\ &\geq 2 \cdot \sum_{i \in [\ell_{\mathrm{PCP}}]} \min\{P_{i,0}, P_{i,1}\}^2 \\ &\geq (2/\ell_{\mathrm{PCP}}) \left(\sum_{i \in [\ell_{\mathrm{PCP}}]} \min\{P_{i,0}, P_{i,1}\} \right)^2 && \text{Cauchy-Schwarz Inequality} \\ &\geq 2(\epsilon - s_{\mathrm{PCP}})^2 / \ell_{\mathrm{PCP}}. && \text{Eq. (10)} \end{aligned}$$

Note that $\mathsf{FindCol}'$ is a circuit of size $S \cdot \mathrm{poly}(q_{\mathrm{PCP}}, \lambda)$. Thus the lemma is proved. □

Finally, we prove the soundness of Micali's CS proofs in the plain model:

Theorem 5.9 (Security of Micali's CS Proofs)**.** *Let* $O^{\mathsf{MT}} : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$ *and* $O^{\mathsf{FS}} : \{0,1\}^{\lambda} \to \{0,1\}^{r_{\mathrm{PCP}}}$ *be two oracles. Let* $\delta > s_{\mathrm{PCP}}$ *be any parameter and* $\rho := 2\delta^2/\ell_{\mathrm{PCP}}$*. Let* A *be an oracle circuit of size* S *such that*

$$\mathsf{Verify}^{O^{\mathsf{MT}}}(\mathsf{Root}, \mathsf{seed}, \pi) = 1 \text{ where } \begin{cases} (\mathsf{Root}, \pi) \leftarrow A^{O^{\mathsf{MT}}, O^{\mathsf{FS}}}(1^{\lambda}), \\ \mathsf{seed} \leftarrow O^{\mathsf{FS}}(\mathsf{Root}). \end{cases} \tag{11}$$

Let $\ell_{\mathsf{MT}} := 2^{2\lambda} \cdot \lambda$*,* $\ell_{\mathsf{FS}} := 2^{\lambda} \cdot r_{\mathrm{PCP}}$*, and* $\mathsf{info} := (r_{\mathrm{PCP}}, \ell_{\mathrm{PCP}}, \lambda, \delta, S)$*. Then*

$$\text{either} \qquad \mathrm{K}^{p(\ell_{\mathsf{MT}}+\ell_{\mathsf{FS}})}\left(O^{\mathsf{MT}}, O^{\mathsf{FS}} \mid A, \mathsf{info}\right) \le \ell_{\mathsf{MT}} + \ell_{\mathsf{FS}} - \log(1/\delta) + O(\log S),$$

$$\text{or} \qquad \mathrm{pK}_{\rho}^{p(\ell_{\mathsf{MT}}+\ell_{\mathsf{FS}})}\left(O^{\mathsf{MT}}, O^{\mathsf{FS}} \mid A, \mathsf{info}\right) \le \ell_{\mathsf{MT}} + \ell_{\mathsf{FS}} - \lambda + O(\log S).$$

Proof. Without loss of generality, we assume that A always queries $O^{\mathsf{FS}}(\mathsf{Root})$ at the end. We say an oracle gate in A is the *Fiat-Shamir gate* if it is the first O^{FS} oracle gate that queries $O^{\mathsf{FS}}(\mathsf{Root})$; the Fiat-Shamir gate always exists. Let $i_\star$ be the index of the Fiat-Shamir gate. We say the *Fiat-Shamir* query is the query $O^{\mathsf{FS}}(\mathsf{Root})$ that the $i_\star$-th gate queries.

The next step is to define the following process $\tilde{A}^{O^{\mathsf{MT}}, \mathsf{tape}}(\mathsf{seed})$, where $\mathsf{tape} \in (\{0,1\}^{r_{\mathrm{PCP}}})^S$ is a long enough tape consisting of random bits. Roughly speaking, in this process, we use tape and seed to simulate O^{FS}: the Fiat-Shamir query returns seed, while we use tape to answer the other queries. The intuition behind this process is that among all queries to the O^{FS} oracle, *only the Fiat-Shamir query is useful, and we can replace the answers to the other queries with truly random bits.* More precisely, in the process $\tilde{A}^{O^{\mathsf{MT}}, \mathsf{tape}}(\mathsf{seed})$, we simulate $A^{O^{\mathsf{MT}}, O^{\mathsf{FS}}}(1^{\lambda})$, and whenever it queries $O^{\mathsf{FS}}(x)$:

- If we have seen this query before during the simulation, we return the same answer.
- If this query is the $i_\star$-th query, then we return seed.
- Otherwise, we return the next value in tape. That is, suppose this query is the i-th distinct non-Fiat-Shamir query made by A, we return tape_i. (Recall that each element in tape is a string of length r_{PCP}.)

We say $\tilde{A}^{O^{\mathsf{MT}}, \mathsf{tape}}(\mathsf{seed}) = 1$ if it produces (Root, π) such that $\mathsf{Verify}^{O^{\mathsf{MT}}}(\mathsf{Root}, \mathsf{seed}, \pi) = 1$.

Let $\mathsf{tape}_\star \in (\{0,1\}^{r_{\mathrm{PCP}}})^S$ denote the "real" list of answers to the O^{FS} queries made by $A^{O^{\mathsf{MT}}, O^{\mathsf{FS}}}(1^{\lambda})$. That is, we simulate $A^{O^{\mathsf{MT}}, O^{\mathsf{FS}}}(1^{\lambda})$, and whenever it queries $O^{\mathsf{FS}}(x)$, if we have not seen this query before and this query is not the Fiat-Shamir query, then we append it to the end of $\mathsf{tape}_\star$. After this process, suppose that $\mathsf{tape}_\star$ contains S' elements where $S' \le S$, then we add $S - S'$ dummy elements (say they are the all-zero string of length r_{PCP}) to the end of $\mathsf{tape}_\star$. We also let $\mathsf{seed}_\star := O^{\mathsf{FS}}(\mathsf{Root})$ be the "real" answer of the Fiat-Shamir query. By the above definition, we have that $\tilde{A}^{O^{\mathsf{MT}}, \mathsf{tape}_\star}(\mathsf{seed}_\star) = 1$.

Now we fix $\mathsf{tape} := \mathsf{tape}_\star$ and treat seed as an input. During the process $\tilde{A}^{O^{\mathsf{MT}}, \mathsf{tape}}(\mathsf{seed})$, we will output some Root that is independent of seed. (This is because Root is the input of the Fiat-Shamir gate, and we only use seed as the answer of this gate.) Therefore, $\tilde{A}^{O^{\mathsf{MT}}, \mathsf{tape}}(\mathsf{seed})$ can be treated as a prover for Kilian's protocol: after it "commits" to some Root, it receives a random seed from the verifier, and then sends the proof π to the verifier.

We divide our argument into two cases depending on the following quantity

$$p := \Pr_{\mathsf{seed} \leftarrow \{0,1\}^{r_{\mathrm{PCP}}}}\left[\tilde{A}^{O^{\mathsf{MT}}, \mathsf{tape}}(\mathsf{seed}) = 1\right].$$

Case I: Suppose $p < 2\delta$. Then a random seed makes the verifier reject w.p. $1 - p$, but $\mathsf{seed}_\star$ makes the verifier accept. Therefore, we can save $\log(1/p)$ bits when we write down $\mathsf{seed}_\star$.

In this case, our description of O^{MT} and O^{FS} is now as follows. We first write down the oracle O^{MT} in verbatim, costing ℓ_{MT} bits. We also write down $i_\star$ in $\log S$ bits. Then we simulate $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(1^\lambda)$ to obtain $\mathsf{tape}_\star \in (\{0,1\}^{r_{\mathrm{PCP}}})^{S'}$ and $\mathsf{seed}_\star \in \{0,1\}^{r_{\mathrm{PCP}}}$. We write down S' and $\mathsf{tape}_\star$ using $\log S + S' \cdot r_{\mathrm{PCP}}$ bits. To record $\mathsf{seed}_\star$, we use $(r_{\mathrm{PCP}} - \log \frac{1}{2\delta})$ bits to write down the integer K where $\mathsf{seed}_\star$ is the lexicographically K-th smallest string seed such that $\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape}}(\mathsf{seed}) = 1$. Finally, we write down the rest portion of O^{FS} using $(2^\lambda - S' - 1)r_{\mathrm{PCP}}$ bits. The total number of bits is thus

$$\begin{aligned}
&\ell_{\mathsf{MT}} + 2\log S + S' \cdot r_{\mathrm{PCP}} + (r_{\mathrm{PCP}} - \log \frac{1}{2\delta}) + (2^\lambda - S' - 1)r_{\mathrm{PCP}} + O(1) \\
\le &\ell_{\mathsf{MT}} + \ell_{\mathsf{FS}} - (\log \frac{1}{2\delta} - 2\log S) + O(1).
\end{aligned}$$

Case II: Suppose $p \ge 2\delta$. Since the size of $\tilde{A}^{(-),\mathsf{tape}}$ is at most $\mathrm{poly}(S)$, by Theorem 5.8, there is a randomised oracle circuit $\mathsf{FindCol}'^{O^{\mathsf{MT}}}(r)$ of size $\mathrm{poly}(S)$ such that

$$\Pr_r[\mathsf{FindCol}'^{O^{\mathsf{MT}}}(r) \text{ finds a collision of } O^{\mathsf{MT}}] \ge 2(2\delta - s_{\mathrm{PCP}})^2/\ell_{\mathrm{PCP}} \ge 2\delta^2/\ell_{\mathrm{PCP}} = \rho.$$

By Theorem 5.7, for some polynomial p_1, we have

$$\Pr_r\left[\mathrm{K}^{p_1(\ell_{\mathsf{MT}})}\left(O^{\mathsf{MT}} \mid A, \mathsf{tape}, r, \mathsf{info}\right) \le \ell_{\mathsf{MT}} - (\lambda - O(\log S))\right] \ge \rho,$$

which means

$$\mathrm{pK}_\rho^{p_1(\ell_{\mathsf{MT}})}\left(O^{\mathsf{MT}} \mid A, \mathsf{tape}, \mathsf{info}\right) \le \ell_{\mathsf{MT}} - (\lambda - O(\log S)).$$

In this case, our compression of O^{MT} and O^{FS} is as follows. We first write down $i_\star$ using $\log S$ bits. Then we simulate $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(1^\lambda)$ to obtain $\mathsf{tape}_\star \in (\{0,1\}^{r_{\mathrm{PCP}}})^{S'}$. We write down S' and $\mathsf{tape}_\star$ using $\log S + S' \cdot r_{\mathrm{PCP}}$ bits. We then invoke Theorem 5.7 to compress O^{MT} into $\ell_{\mathsf{MT}} - (\lambda - O(\log S))$ bits. Finally, we write down the rest portion of O^{FS} using $(2^\lambda - S') \cdot r_{\mathrm{PCP}}$ bits. The total number of bits is thus

$$\begin{aligned}
&2\log S + S' \cdot r_{\mathrm{PCP}} + \ell_{\mathsf{MT}} - (\lambda - O(\log S)) + (2^\lambda - S') \cdot r_{\mathrm{PCP}} + O(1) \\
\le &\ell_{\mathsf{MT}} + \ell_{\mathsf{FS}} - (\lambda - O(\log S)).
\end{aligned}$$

□

5.3 Security of CS Proofs in the CRS Model

Theorem 5.10 (Security of CS Proofs in the CRS Model)**.** *Suppose that $\lambda \ge 7\log S + 3k + \log \ell_{\mathrm{PCP}}$ and $2^{-k-4} > s_{\mathrm{PCP}}$. Then there is a polynomial p such that the following holds.*

Let $O^{\mathsf{MT}} : \{0,1\}^k \times \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ and $O^{\mathsf{FS}} : \{0,1\}^k \times \{0,1\}^\lambda \to \{0,1\}^{r_{\mathrm{PCP}}}$ be two oracles. Let A be an oracle circuit of size S such that

$$\Pr_{\mathsf{crs} \leftarrow \{0,1\}^k}\left[\mathsf{Verify}^{O^{\mathsf{MT}}(\mathsf{crs},-)}(\mathsf{Root}, \mathsf{seed}, \pi) = 1 \;\middle|\; \begin{array}{c} (\mathsf{Root}, \pi) \leftarrow A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs}) \\ \mathsf{seed} \leftarrow O^{\mathsf{FS}}(\mathsf{crs}, \mathsf{Root}) \end{array}\right] \ge \epsilon.$$

Let $\ell_{\mathsf{MT}} := 2^{k+2\lambda} \cdot \lambda$, $\ell_{\mathsf{FS}} := 2^{k+\lambda} \cdot r_{\mathrm{PCP}}$, and $\mathsf{info} = (k, \lambda, r_{\mathrm{PCP}}, S)$. Then

$$\mathrm{pK}_{2/3}^{p(\ell_{\mathsf{MT}}+\ell_{\mathsf{FS}})}\left(O^{\mathsf{MT}}, O^{\mathsf{FS}} \mid \mathsf{info}\right) \le \ell_{\mathsf{MT}} + \ell_{\mathsf{FS}} + 3.01 S\log S - \epsilon 2^k/S^{4.01}.$$

Proof. First, we choose an index $i_\star$ so that the $i_\star$-th gate of A is likely to be the Fiat-Shamir gate. For a fixed crs, the *Fiat-Shamir gate* of $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$ is the first O^{FS} gate in A that queries $O^{\mathsf{FS}}(\mathsf{crs}, \mathsf{Root})$, where Root is the first output of $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$. Without loss of generality, we may assume that the Fiat-Shamir gate always exists. By averaging, there is an index $i_\star$ such that

$$\Pr_{\mathsf{crs}\leftarrow\{0,1\}^k}\left[\begin{array}{c}\mathsf{Verify}^{O^{\mathsf{MT}}(\mathsf{crs},-)}(\mathsf{Root},\mathsf{seed},\pi)=1\wedge\\ \text{the } i_\star\text{-th gate is the Fiat-Shamir gate}\end{array}\middle|\begin{array}{c}(\mathsf{Root},\pi)\leftarrow A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})\\ \mathsf{seed}\leftarrow O^{\mathsf{FS}}(\mathsf{crs},\mathsf{Root})\end{array}\right]\geq\epsilon/S. \quad (12)$$

Fix this $i_\star$, and we say $\mathsf{crs}\in\{0,1\}^k$ is *good* if Eq. (12) holds for crs. Then there are at least $(\epsilon/S)2^k$ good strings crs.

MT- and FS-compressibility. We classify each good crs into two categories: MT-compressible and FS-compressible. Intuitively, a string crs is MT-compressible if when we run the proof of Theorem 5.9 on this crs, we run into case II; crs is FS-compressible if we run into case I. (Recall that in case I of the proof of Theorem 5.9, we write down O^{MT} in verbatim and compress O^{FS} non-trivially; in case II, we write down O^{FS} in verbatim and compress O^{MT} non-trivially.) The precise definition is a bit involved, as it depends on the proof of Theorem 5.9.

We simulate $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$. Let tape be the list of answers of the O^{FS} oracles, except for the Fiat-Shamir query. That is, whenever $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$ makes a query to O^{FS} that we have not seen before and that is not the Fiat-Shamir query, we append its answer to the end of tape. (Note that we append the answer of *every* query $O^{\mathsf{FS}}(\mathsf{crs}',x')$, regardless of whether $\mathsf{crs}'=\mathsf{crs}$.) We also pad tape with dummy entries so that it contains exactly S elements in $\{0,1\}^{r_{\mathrm{PCP}}}$.

Next, we define the following process denoted as $\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape},\mathsf{crs}}(\mathsf{seed})$. First, we hardwire tape and crs. On input seed, we simulate $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$, but answers the O^{FS} queries in the same manner as the process $\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape}}(\mathsf{seed})$ in the proof of Theorem 5.9. That is, whenever $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$ makes a query to O^{FS}:

1. If we have seen this query before, we return the same answer.
2. If this query is the Fiat-Shamir query, then we return seed (which is our input).
3. Otherwise, we return the next unused element in tape.

Suppose the input of the Fiat-Shamir query is $(\mathsf{crs},\mathsf{Root})$ and the output of the simulated $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$ is (Root',π). We discard Root' and output (Root,π). Note that Root does not depend on seed, therefore this process can be treated as a prover for (unkeyed) Kilian's protocol. We say $\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape},\mathsf{crs}}(\mathsf{seed})=1$ if $\mathsf{Verify}^{O^{\mathsf{MT}}(\mathsf{crs},-)}(\mathsf{Root},\mathsf{seed},\pi)=1$, i.e., the verifier in Kilian's protocol accepts.

Let $p:=\Pr_{\mathsf{seed}\leftarrow\{0,1\}^{r_{\mathrm{PCP}}}}\left[\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape},\mathsf{crs}}(\mathsf{seed})=1\right]$ and $\delta:=2^{-k-4}$. If $p<2\delta$, we say crs is FS-*compressible*. Otherwise, we say crs is MT-*compressible*.

The FS-compressible case. Let $T:=(\epsilon/S)2^{k-1}$, suppose that there are at least T good strings crs that are FS-compressible. Our goal is to compress the *critical entry* of each FS-compressible crs by at least one bit. Here, for every FS-compressible crs, we define $\mathsf{cri}(\mathsf{crs})$, the *critical entry* of crs, to be $O^{\mathsf{FS}}(\mathsf{crs},\mathsf{Root})$, where Root is the Root input of the Fiat-Shamir gate of $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$.

Naïvely, one may think that for every FS-compressible crs, one could save $\log(1/p)\geq\log(1/(2\delta))$ bits when we write down $\mathsf{cri}(\mathsf{crs})$. However, this does not work for the following reason. Suppose we want to compress $\mathsf{cri}(\mathsf{crs})$. Then we need to simulate $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$ and write down every query to O^{FS} that A makes. The critical query only needs $r_{\mathrm{PCP}}-\log(1/p)$ bits to write down, but all other queries need exactly r_{PCP} bits. Suppose that there is a query $O^{\mathsf{FS}}(\mathsf{crs}',\mathsf{Root}')$ that we wrote down verbatim. It might be the case that crs' is also FS-compressible and $\mathsf{cri}(\mathsf{crs}')$ is exactly $O^{\mathsf{FS}}(\mathsf{crs}',\mathsf{Root}')$! In this case, we have already recorded $O^{\mathsf{FS}}(\mathsf{crs}',\mathsf{Root}')$ in verbatim, so there is nothing to save here.

For each crs that is FS-compressible, let $\mathsf{Tape}(\mathsf{crs})$ denote the set of O^{FS} queries that $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$ makes. Now, we pick a large enough subset of FS-compressible strings crs that are "*independent*". We define a *dependency graph* $G = (V, E)$, where:

- The vertices in G are exactly the set of crs's that are FS-compressible.
- For every $\mathsf{crs}, \mathsf{crs}'$, if $\mathsf{cri}(\mathsf{crs}') \in \mathsf{Tape}(\mathsf{crs})$, then we add a *directed* edge $(\mathsf{crs} \to \mathsf{crs}')$ in G.

Let $\succ$ be a linear order over V. (Equivalently, choose a permutation over V and let $\mathsf{crs} \succ \mathsf{crs}'$ if and only if crs appears later than crs' in this permutation.) We say a $\mathsf{crs} \in V$ is *successor-free* if for every outgoing edge $(\mathsf{crs} \to \mathsf{crs}')$, we have $\mathsf{crs} \succ \mathsf{crs}'$. Let K be the number of successor-free crs in V, and consider the sequence $\mathcal{K} = (\mathsf{crs}_1, \mathsf{crs}_2, \dots, \mathsf{crs}_K)$ of successor-free crs, where $\mathsf{crs}_1 \prec \mathsf{crs}_2 \prec \cdots \prec \mathsf{crs}_K$. Let $out(\mathsf{crs})$ be the out-degree of crs, then for every $\mathsf{crs} \in V$, we have $out(\mathsf{crs}) \le S$. Therefore, over a random permutation $\succ$, we have

$$\mathop{\mathbb{E}}_{\succ}[K] = \sum_{\mathsf{crs}\in V} \Pr_{\succ}[\mathsf{crs} \text{ is successor-free}] = \sum_{\mathsf{crs}\in V} \frac{1}{out(\mathsf{crs})} \ge T/S.$$

Now, we fix a $\succ$ such that $K \ge T/S$, which also fixes the sequence $\mathcal{K}$ of successor-free crs. We will succinctly write down the answers of $\mathsf{cri}(\mathsf{crs})$ for every $\mathsf{crs} \in \mathcal{K}$.

The compression. We start by writing down the oracle O^{MT} in verbatim, which costs ℓ_{MT} bits. Then we record the circuit A using $3S \log S$ bits. We also record the integers $(i_\star, K)$ which costs $\log S + k$ bits. After that, we write down the sequence $\mathcal{K}$ using $K \cdot k$ bits.

Now, for i from 1 to K, we write down all answers of queries in $\mathsf{Tape}(\mathsf{crs}_i)$. In particular, we simulate $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$, and whenever A makes a query $O^{\mathsf{FS}}(\mathsf{crs}', x')$:

- If we have already recorded the answer of $O^{\mathsf{FS}}(\mathsf{crs}', x')$, then we do not need to do anything.
- If this query is the Fiat-Shamir query, we also do nothing (for now).
- Otherwise, we spend r_{PCP} bits to write down the answer of $O^{\mathsf{FS}}(\mathsf{crs}', x')$.

After writing down these entries, we can recover tape and thus simulate $\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape},\mathsf{crs}}(\mathsf{seed})$. Note that for any $j < i$, we have $\mathsf{crs}_j \prec \mathsf{crs}_i$, and since crs_j is successor-free, the edge $(\mathsf{crs}_j \to \mathsf{crs}_i)$ is not in E, and thus $\mathsf{cri}(\mathsf{crs}_i) \notin \mathsf{Tape}(\mathsf{crs}_j)$. This means that we have not recorded the answer of $\mathsf{cri}(\mathsf{crs}_i)$ in the first $i-1$ rounds, thus we have the chance to record it (using $r_{\mathrm{PCP}} - \log(1/p)$ bits). Let $\mathsf{seed}_\star$ be the answer of $\mathsf{cri}(\mathsf{crs}_i)$, then we record the integer K such that $\mathsf{seed}_\star$ is the lexicographically K-th smallest string seed such that $\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape},\mathsf{crs}}(\mathsf{seed}) = 1$.

After we have processed crs_i for every $i \in [K]$, we write down the rest of O^{FS} in verbatim.

The total number of bits needed to compress O^{MT} and O^{FS} is

$$\begin{aligned}&\ell_{\mathsf{MT}} + 3S\log S + \log S + k + K \cdot k + \ell_{\mathsf{FS}} - K \cdot \log(1/p) + O(1)\\ \le\, &\ell_{\mathsf{MT}} + \ell_{\mathsf{FS}} + 3.01 S \log S - \epsilon 2^k / S^{4.01},\end{aligned}$$

where the last inequality is because $K \ge 2^{k-1}(\epsilon/S^2) \ge \epsilon 2^k/S^{4.01}$.

The MT-compressible case. Suppose, on the other hand, that there are at less than T good strings crs that are FS-compressible. As there are at least $2T$ good strings crs, there are at least T strings crs that are MT-compressible.

We need to compress the *critical entry* of each MT-compressible crs by at least one bit. Here, the critical entry of crs, denoted as $\mathsf{cri}(\mathsf{crs})$, is the collision of $O^{\mathsf{MT}}(\mathsf{crs}, -)$ found by some circuit related to $\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape},\mathsf{crs}}$.

More precisely, we construct a randomised circuit $\mathsf{FindCol}'_{\mathsf{crs}}$ that finds a collision of $O^{\mathsf{MT}}(\mathsf{crs}, -)$ with good probability. As in the proof of Theorem 5.8, this circuit takes $\mathsf{seed}_1, \mathsf{seed}_2 \leftarrow \{0,1\}^{r_{\mathrm{PCP}}}$ as randomness, invokes $\tilde{A}^{O^{\mathsf{MT}},\mathsf{tape},\mathsf{crs}}$ on both seed_1 and seed_2 to obtain $2q$ authentication paths, and finds collisions of $O^{\mathsf{MT}}(\mathsf{crs}, -)$ among them. The size of $\mathsf{FindCol}'_{\mathsf{crs}}$ is $O(S^3)$ and the success probability is at least $\rho := 2\delta^2/\ell_{\mathrm{PCP}}$.

The next step is to "derandomise" $\mathsf{FindCol}'_{\mathsf{crs}}$. Recall that we prove compression upper bounds w.r.t. the pK^t complexity, so we could assume there is a sufficiently long random string π that helps our compression. We treat π as a random function $\pi : \{0,1\}^k \times [1/\rho] \times [2] \to \{0,1\}^{r_{\mathrm{PCP}}}$. For every $j \in [1/\rho]$, define the circuit $\mathsf{FindCol}'_{\mathsf{crs},j}$ to be the circuit $\mathsf{FindCol}'_{\mathsf{crs}}$ with randomness $\mathsf{seed}_1 = \pi(\mathsf{crs}, j, 1)$ and $\mathsf{seed}_2 = \pi(\mathsf{crs}, j, 2)$. Consider the first j such that $\mathsf{FindCol}'_{\mathsf{crs},j}$ indeed finds a collision of $O^{\mathsf{MT}}(\mathsf{crs}, -)$, and let (i', j') be the first collision found by $\mathsf{FindCol}'_{\mathsf{crs},j}$. Suppose that $i' < j'$, the i'-th gate of $\mathsf{FindCol}'_{\mathsf{crs},j}$ is $O^{\mathsf{MT}}(\mathsf{crs}, a)$, and the j'-th gate is $O^{\mathsf{MT}}(\mathsf{crs}, b)$. Then we define $\mathsf{cri}(\mathsf{crs}) := O^{\mathsf{MT}}(\mathsf{crs}, b)$.

For every MT-compressible crs, let X_{crs} denote the event (over the random variable π) that there is some $j \in [1/\rho]$ such that $\mathsf{FindCol}'_{\mathsf{crs},j}$ finds a collision of $O^{\mathsf{MT}}(\mathsf{crs}, -)$. (If X_{crs} happens, then $\mathsf{cri}(\mathsf{crs})$ is well-defined.) The events X_{crs} are independent, and each one happens w.p. at least $1 - (1-\rho)^{1/\rho} \geq 0.2$. Since there are T such events, by Chernoff bound, w.p. at least $2/3$, X_{crs} is true for at least $0.1T$ choices of crs. We say crs is *good*$^\pi$ if X_{crs} happens; from now on we assume there are at least $0.1T$ good$^\pi$ strings crs.

We use the same argument as in the FS-compressible case to select a sequence of "independent" strings crs. For each good$^\pi$ crs, let j be the smallest number such that $\mathsf{FindCol}'_{\mathsf{crs},j}$ finds a collision of $O^{\mathsf{MT}}(\mathsf{crs}, -)$. Let $\mathsf{Tape}(\mathsf{crs})$ denote the following set of queries:

- every query $O^{\mathsf{MT}}(\mathsf{crs}', a')$ made by $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$;[24]
- every query $O^{\mathsf{MT}}(\mathsf{crs}', a')$ made by $\mathsf{FindCol}'_{\mathsf{crs},j}$.

By building a dependency graph (as in the FS-compressible case) over the good$^\pi$ strings crs, we can select a sequence $\mathcal{K}$ of K good$^\pi$ strings $(\mathsf{crs}_1, \mathsf{crs}_2, \dots, \mathsf{crs}_K)$ such that for every $i > i'$, we have $\mathsf{cri}(\mathsf{crs}_i) \notin \mathsf{Tape}(\mathsf{crs}_{i'})$. Moreover, since each $\mathsf{Tape}(\mathsf{crs})$ contains at most $S_1 \leq O(S^3)$ elements, we have

$$K \geq 0.1T/S_1 \geq 0.05 \cdot \frac{\epsilon 2^k}{S \cdot S_1}.$$

The compression. We start by spending ℓ_{FS} bits to write down the oracle O^{FS} in verbatim. Then we record the circuit A using $3S \log S$ bits. We also record the integers $(i_\star, K)$ which costs $\log S + k$ bits. After that, for each $i \in [K]$, we write down the string crs_i and the smallest integer j_i such that $\mathsf{FindCol}'_{\mathsf{crs}_i,j_i}$ finds a collision for $O^{\mathsf{MT}}(\mathsf{crs}_i, -)$. This takes $K \cdot (k + \log(1/\rho))$ bits.

Now, for i from 1 to K, we write down all answers of queries in $\mathsf{Tape}(\mathsf{crs}_i)$. In particular, we first simulate $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs}_i)$, and whenever A makes a query $O^{\mathsf{MT}}(\mathsf{crs}', a')$ that was not recorded, we record this query. Suppose that the first collision of $O^{\mathsf{MT}}(\mathsf{crs}_i, -)$ found by $\mathsf{FindCol}'_{\mathsf{crs}_i,j_i}$ is the i'-th gate and the j'-th gate of $\mathsf{FindCol}'_{\mathsf{crs}_i,j_i}$. Then we write down these two numbers (i', j'). After that, we simulate $\mathsf{FindCol}'_{\mathsf{crs}_i,j_i}$. Whenever it queries $O^{\mathsf{MT}}(\mathsf{crs}', a')$, if we have not recorded the answer of this query yet, then we record it. The only exception is that we do not need to record the answer of the j'-th gate.

After we have processed crs_i for every $i \in [K]$, we write down the rest of O^{MT} in verbatim.

The total number of bits needed to compress O^{MT} and O^{FS} is

$$\ell_{\mathsf{FS}} + 3S\log S + \log S + k + K \cdot (k + \log(1/\rho)) + \ell_{\mathsf{MT}} - K \cdot (\lambda - 2\log S_1) + O(1)$$

[24] At first look, it may seem that we only need to include the queries made by $\mathsf{FindCol}'_{\mathsf{crs},j}$ in $\mathsf{Tape}(\mathsf{crs})$. However, $\mathsf{FindCol}'_{\mathsf{crs},j}$ depends on tape, and we need to simulate $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$ in order to recover tape, thus it is important that we also record the queries needed by $A^{O^{\mathsf{MT}},O^{\mathsf{FS}}}(\mathsf{crs})$.

$$\leq \ell_{\mathsf{FS}} + \ell_{\mathsf{MT}} + 3.01 S \log S - K(\lambda - 6.5 \log S - k - \log(1/\rho))$$
$$\leq \ell_{\mathsf{FS}} + \ell_{\mathsf{MT}} + 3.01 S \log S - \epsilon 2^k / S^{4.01},$$

where the last equation is because $\lambda - 6.5 \log S - k - \log(1/\rho) \geq \lambda - 6.9 \log S - 3k - \log \ell_{\mathrm{PCP}} \geq 1$ and $K \geq \Omega(\epsilon 2^k / S S_1) \geq \epsilon 2^k / S^{4.01}$. □

5.4 NP-hardness of Gap-mvMCSP^O

Theorem 5.1. *With probability* 1 *over a random oracle* O*,* Gap-mvMCSP^O *is* **NP***-hard under deterministic* $\mathbf{TIME}[2^{\mathrm{polylog}(n)}]^O$ *reductions.*

Moreover, for any language $L \in \mathbf{NP}$ *and any constant* $t \geq 1$*, there is a constant* c_1 *and a mapping computable in deterministic* $2^{\mathrm{polylog}(n)}$ *time with an* O *oracle, that maps an instance* $x \in \{0,1\}^n$ *to a table* T *satisfying the following:*

$$x \in L \implies \mathsf{CC}^O(T) \leq s := O(n^{c_1});$$
$$x \notin L \implies \mathsf{CC}^O_{2^{-\log^t s}}(T) \geq 2^{\log^t s}.$$

Proof. Let $O_N \in \{0,1\}^{2^N}$ denote the truth table of the N-th slice of O, and $O_{\leq N} \in \{0,1\}^{2^{N+1}-2}$ denote the concatenation of $O_1, O_2, \ldots, O_N$. With probability 1 over the random oracle O, for all but finitely many $N \in \mathbb{N}$,

$$\mathrm{K}(O_{\leq N}) > 2^{N+1} - O(N).$$

This fact is easy to see; for completeness, we include a proof in Appendix B.

Let x be an instance of L, $n := |x|$. Assuming $k, r_{\mathrm{PCP}}, \lambda < n^{o(1)}$. By Theorem 5.6, for some constant $c_2 \geq 1$ that only depends on L, if $x \in L$ then there is an honest prover for x of size $S_{\mathsf{honest}} := O(n^{c_2})$. We instantiate the CS proof protocol using the following parameters: $S := 2^{\log^t S_{\mathsf{honest}}}$, $\epsilon := 1/S$, $k := 7 \log S$, $c := k + 5$, $\lambda := 30 \log S + \log \ell_{\mathrm{PCP}}$. It is easy to verify that the technical conditions of Theorem 5.10 are satisfied.

Let $N := 100 \log S + 2 \log \ell_{\mathrm{PCP}}$. We define $O^{\mathsf{MT}} : \{0,1\}^k \times \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ and $O^{\mathsf{FS}} : \{0,1\}^k \times \{0,1\}^\lambda \to \{0,1\}^{r_{\mathrm{PCP}}}$ according to the N-th slice of O. That is:

- On input $\mathsf{crs} \in \{0,1\}^k$ and $a \in \{0,1\}^{2\lambda}$, let $\mathsf{pad} := 0^{N-1-k-2\lambda-\lceil \log \lambda \rceil}$, $O^{\mathsf{MT}}(\mathsf{crs}, a)$ returns the concatenation of $O(0, \mathsf{crs}, a, i, \mathsf{pad})$ for every $i \in [\lambda]$.
- On input $\mathsf{crs} \in \{0,1\}^k$ and $\mathsf{Root} \in \{0,1\}^\lambda$, let $\mathsf{pad} := 0^{N-1-k-\lambda-\lceil \log r_{\mathrm{PCP}} \rceil}$, $O^{\mathsf{FS}}(\mathsf{crs}, \mathsf{Root})$ returns the concatenation of $O(1, \mathsf{crs}, \mathsf{Root}, i, \mathsf{pad})$ for every $i \in [r_{\mathrm{PCP}}]$.

We define the following table T. The rows will be indexed by $\mathsf{crs} \in \{0,1\}^k$, and the columns will be indexed by $(\mathsf{Root}, \pi) \in \{0,1\}^{\ell_{\mathsf{proof}}}$ where $\ell_{\mathsf{proof}} := O(q_{\mathrm{PCP}} \cdot \log \ell_{\mathrm{PCP}} \cdot \lambda) \leq \log^{O(t)} n$ is the length of the CS proof. Let

$$T(\mathsf{crs}, (\mathsf{Root}, \pi)) = 1 \iff \mathsf{Verify}^{O^{\mathsf{MT}}(\mathsf{crs}, -)}(\mathsf{Root}, \mathsf{seed}, \pi) = 1 \text{ where } \mathsf{seed} = O^{\mathsf{FS}}(\mathsf{crs}, \mathsf{Root}).$$

Now we prove the correctness of our reduction. Suppose $x \in L$, then by Theorem 5.6, there is a circuit C of size S_{honest} such that for every $\mathsf{crs} \in \{0,1\}^k$, $T(\mathsf{crs}, C^O(\mathsf{crs})) = 1$. This implies that $\mathsf{CC}^O(T) \leq S_{\mathsf{honest}}$. On the other hand, suppose $x \notin L$. By Theorem 5.10, if there is a size-S oracle circuit C such that

$$\Pr_{\mathsf{crs} \leftarrow \{0,1\}^k}[T(\mathsf{crs}, C^O(\mathsf{crs})) = 1] \geq \epsilon,$$

then

$$\mathrm{pK}^{\mathrm{poly}(2^N)}_{2/3}(O_N \mid O_1, O_2, \ldots, O_{N-1}, O_{N+1}, \ldots) \leq 2^N + 3.01 S \log S - \epsilon 2^k / S^{4.01} \leq 2^N - \Omega(S^{1.5}).$$

Moreover, it is implicit in the proof of Theorem 5.10 that the decompression only needs access to the oracle O up to input length S. Thus we have

$$\mathrm{pK}_{2/3}^{\mathrm{poly}(2^S)}(O_{\leq S}) \leq 2^{S+1} - \Omega(S^{1.5}).$$

And by Fact 2.12,

$$\mathrm{K}(O_{\leq S}) \leq 2^{S+1} - \Omega(S^{1.5}).$$

This could only happen for finitely many input lengths n.

Finally, it is easy to see that our reduction can be computed in deterministic $2^{\mathrm{polylog}(n)}$ time with an O oracle. □

We showed that for some variant of MCSP (namely Gap-mvMCSP), over a random oracle O, the O-relativised version of this variant is **NP**-hard (even to approximate) under $\mathbf{QuasiP}^O$ reductions. We conjecture that the same is true for the original MCSP (and that this is a feasible research question). We believe that this conjecture, if true, would give strong evidence that MCSP is indeed **NP**-complete.

Conjecture 5.11. *With probability* 1 *over a random oracle* O*,* MCSP^O *is* **NP***-hard under* $\mathbf{QuasiP}^O$ *reductions.*

6 Applications

6.1 Pseudorandom Self-Reductions

In this sub-section, one-way functions are assumed to be secure against *non-uniform* adversaries. For simplicity, we only define pseudorandom *mapping* reduction (as opposed to pseudorandom non-adaptive reduction in [HS17, ERSY22]).

Definition 6.1 (Pseudorandom Self-Reducibility, [HS17, ERSY22])**.** Let $\mathscr{C}$ be a complexity class. Let $Q = (\Pi.\textsc{Yes}, \Pi.\textsc{No})$ be a promise problem, where $\Pi.\textsc{Yes}, \Pi.\textsc{No} \subseteq \{0,1\}^\star$, and let $L \subseteq \{0,1\}^\star$ be a language. We say Q is *pseudorandomly (mapping-)reducible* to L with respect to $\mathscr{C}$ if there is a function $g : \{0,1\}^\star \times \{0,1\}^\star \to \{0,1\}^\star$ computable in polynomial time such that the following holds:

(Pseudorandomness) For every input $x \in \{0,1\}^n$, the distributions $g(x, \mathcal{U}_{\mathrm{poly}(n)})$ is indistinguishable from the uniform distribution by $\mathscr{C}$.

(Correctness) For every $x \in \{0,1\}^\star$ and every r: if $x \in \Pi.\textsc{Yes}$ then $g(x,r) \in L$, while if $x \in \Pi.\textsc{No}$ then $g(x,r) \notin L$.

Let $0 < \epsilon < 1 < c$ be two parameters, $\mathrm{Gap}_{\epsilon,c}\mathrm{MOCSP}$ denote the following promise problem:

Input: a truth table $f \in \{0,1\}^N$ and an oracle truth table $O \in \{0,1\}^{N^c}$.

YES instances: $\mathsf{CC}^O(f) \leq N^\epsilon$.

NO instances: $\mathsf{CC}^O(f) > N^{1-\epsilon}$.

By Corollary 4.8, for every constant $\epsilon > 0$, there is a constant $c > 1$ such that $\mathrm{Gap}_{\epsilon,c}\mathrm{MOCSP}$ is **NP**-hard. We show that the same problem admits a pseudorandom self-reduction.

Theorem 6.2. *If one-way functions (resp. subexponentially-secure one-way functions) exist, then for every* $0 < \epsilon < 1/2$ *and* $c > 1$*,* $\mathrm{Gap}_{\epsilon,c}\mathrm{MOCSP}$ *admits a pseudorandom self-reduction against polynomial-size (resp. subexponential-size) adversaries.*

Proof. Suppose that one-way functions exist. Let $\delta := (1-2\epsilon)/10$. By [HILL99, GGM86], there exists a pseudorandom function family $F : \{0,1\}^{N^\delta} \to \{0,1\}^{N^c}$ such that:

- For every $\mathsf{seed} \in \{0,1\}^{N^\delta}$, the circuit complexity of $F(\mathsf{seed})$ is at most $N^{2\delta}$.
- The distribution $F(\mathcal{U}_{N^\delta})$ is $1/N^{\omega(1)}$-indistinguishable from the uniform distribution $\mathcal{U}_{N^c}$.

(If the underlying one-way function is subexponentially secure, then any adversary of size $2^{N^{o(1)}}$ cannot distinguish $F(\mathcal{U}_{N^\delta})$ from $\mathcal{U}_{N^c}$ with advantage $2^{-N^{o(1)}}$.)

Let (f, O) be an instance of $\mathrm{Gap}_{\epsilon,c}\mathrm{MOCSP}$, where $f \in \{0,1\}^N$ and $O \in \{0,1\}^{N^c}$. We randomly sample $\mathsf{seed}_1, \mathsf{seed}_2 \leftarrow \{0,1\}^{N^\delta}$, and let $F(\mathsf{seed}_1)_N$ denote the first N bits of $F(\mathsf{seed}_1)$. Then, we reduce (f, O) to the following distribution:

$$(f \oplus F(\mathsf{seed}_1)_N, O \oplus F(\mathsf{seed}_2)). \tag{13}$$

Given any adversary that distinguishes Eq. (13) from $\mathcal{U}_{N+N^c}$, by hardwiring seed_1 and seed_2, we can construct an adversary that distinguishes $F(\mathsf{seed})$ from $\mathcal{U}_{N^c}$. It follows that Eq. (13) is pseudorandom.

Next, we prove the correctness of the reduction. Let $f' := f \oplus F(\mathsf{seed}_1)_N$, $O' := O \oplus F(\mathsf{seed}_2)$. Since the circuit complexity of $F(\mathsf{seed}_2)$ is at most $N^{2\delta}$, we can simulate the oracle O' by an O-oracle circuit of size $N^{3\delta}$. Similarly, given a size-s circuit computing f, we can construct a size-$(s + N^{3\delta})$ circuit that computes f'. It follows that

$$\mathsf{CC}^{O'}(f') \le \mathsf{CC}^{O'}(f) + N^{3\delta} \le N^{4\delta} \cdot \mathsf{CC}^{O}(f).$$

The same argument also shows that

$$\mathsf{CC}^{O'}(f') \ge \mathsf{CC}^{O}(f)/N^{4\delta}.$$

Let L denote the set of strings (f, O) such that $\mathsf{CC}^O(f) \le N^{\epsilon+5\delta} = N^{1/2}$. We can see that $L \in \mathbf{NP}$ and that Eq. (13) is indeed a reduction from $\mathrm{Gap}_{\epsilon,c}\mathrm{MOCSP}$ to L. □

6.2 Heuristics for Complexity

We present an unconditional deterministic heuristic for Complexity. We will be able to solve the problem in the regime where O is much longer than f. For concreteness, assume that we are given the truth table of $O : \{0,1\}^{2n} \to \{0,1\}$, and want to find $f : \{0,1\}^n \to \{0,1\}$ such that $\mathsf{CC}^O(f) > 2^n/10n$.

Definition 6.3 (The Complexity Problem)**.** Let $m = m(n)$ be a function, the problem $\textsc{Complexity}_{m(n)}$ is the following (total) search problem:

(Input) An oracle truth table $O : \{0,1\}^{m(n)} \to \{0,1\}$.

(Output) A truth table $f : \{0,1\}^n \to \{0,1\}$ such that $\mathsf{CC}^O(f) \ge 2^n/10n$.

Theorem 6.4. *There is,* unconditionally*, a deterministic polynomial-time heuristic for* $\textsc{Complexity}_{2n}$ *under the uniform distribution.*

Proof. Let $\mathcal{O} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a uniformly random function. Let $f : \{0,1\}^n \to \{0,1\}$ be the function given by $f(x) = \bigoplus_{y \in \{0,1\}^n} \mathcal{O}(x, y)$.

We will show that f has high oracle circuit complexity relative to $\mathcal{O}$ with high probability. We do this by a union bound argument. Fix an arbitrary oracle circuit C of size $s = 2^n/10n$. We will show that the probability that $C^{\mathcal{O}}$ computes f is at most $2^{-2^n/3}$. Since there are $2^{O(s \log s)} = O(2^{2^n/10})$ circuits of size at most s, it will follow that with probability $1 - 2^{-\Omega(2^n)}$ that f does not have an $\mathcal{O}$-oracle circuit of size at most s, as desired.

It remains to bound the probability that $C^{\mathcal{O}}$ computes f. To do this, we consider the following equivalent way of sampling $\mathcal{O}$ uniformly at random.

1. To begin, let $\mathcal{O} : \{0,1\}^n \times \{0,1\}^n \to \{0,1,\star\}$ be completely unset (i.e. $\mathcal{O}(x,y) = \star$ for all x and y).
2. While there exists an $\tilde{x} \in \{0,1\}^n$ such that $|\{y : \mathcal{O}(\tilde{x},y) = \star\}| \geq s+1$:
   (a) Simulate running $C^{\mathcal{O}}(\tilde{x})$. Whenever an oracle query (x',y') is made:
      i. if $\mathcal{O}(x',y') = \star$, then set $\mathcal{O}(x',y')$ to be a uniformly random value.
      ii. Respond with the value of $\mathcal{O}(x',y')$.
   (b) For all y' with $\mathcal{O}(\tilde{x},y') = \star$, set the value of $\mathcal{O}(\tilde{x},y')$ to be an (independently) uniformly random bit.
   (c) See if $C^{\mathcal{O}}(\tilde{x}) = f(\tilde{x})$. (This step is not a part of the algorithm. It does not do anything. It is just useful to reference this step in the analysis.)
3. For all x', y' with $\mathcal{O}(x',y') = \star$, set $\mathcal{O}(x',y')$ to be an (independently chosen) uniformly random bit.

We begin by observing some facts about this way of sampling $\mathcal{O}$.

Claim 6.5. *The "while loop" runs at least $(2^n - s)/2$ times.*

Proof. In each loop iteration, the size of $\mathcal{O}^{-1}(\{0,1\})$ increases by at most $s + 2^n$ (because C makes at most s oracle queries and at most 2^n values are set in step Item 2b). Thus, after t iterations we have that

$$\mathbf{E}_{x \leftarrow \{0,1\}^n}[|\{y : \mathcal{O}(x,y) = \star\}|] \geq \frac{2^{2n} - t(2^n + s)}{2^n} = 2^n - (1 + s2^{-n})t \geq 2^n - 2t.$$

For this value to be less than $s+1$ (as it must be for the loop not to run), we must have that $t \geq (2^n - s)/2$. $\diamond$

Next, we show each time the loop iteration runs there is a decent probability that $C^{\mathcal{O}}$ fails to compute f on $\tilde{x}$.

Claim 6.6. *In each iteration of the while loop, the probability (over the randomness in step Item 2b) that at step Item 2c $C^{\mathcal{O}}(\tilde{x}) = f(\tilde{x})$ is at most $1/2$.*

Proof. After step Item 2a finishes, the value of $b = C^{\mathcal{O}}(\tilde{x})$ is determined. But also immediately after step Item 2a finishes we know that there is some y' such that $\mathcal{O}(\tilde{x},y') = \star$. This is because in step Item 2a at most s inputs to $\mathcal{O}$ are set to Boolean values (since C has size at most s) and when the loop iteration begins $|\{y : \mathcal{O}(\tilde{x},y) = \star\}| \geq s+1$.

Thus, since $f(\tilde{x})$ is the parity of $\mathcal{O}(\tilde{x},y)$ over all y, it follows that the probability (over the uniform randomness used in step Item 2b) that $b = f(\tilde{x})$ is exactly half. $\diamond$

Putting the two claims together, we have that the probability that $C^{\mathcal{O}}$ computes f is at most $2^{-(2^n-s)/2} \leq 2^{-2^n/3}$, as desired. $\square$

Acknowledgements

We thank Rahul Santhanam for helpful discussions during the initial stage of this research, Lijie Chen for proving Claim A.1, and anonymous STOC reviewers for their helpful comments. We also thank Yilei Chen, Aayush Jain, Huijia Lin, Amit Sahai, Neekon Vafa, and Vinod Vaikuntanathan for answering questions about our cryptographic assumptions.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. (cit. on p. 11)

[ABK$^+$06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal of Computing*, 35(6):1467–1493, 2006. doi:10.1137/050628994. (cit. on p. 1, 9, 10)

[ABMP01] Michael Alekhnovich, Samuel R. Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is **NP**-hard to linearly approximate. *J. Symb. Log.*, 66(1):171–191, 2001. doi:10.2307/2694916. (cit. on p. 7)

[ACM$^+$21] Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. In *FSTTCS*, volume 213 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.7. (cit. on p. 1, 4, 9)

[AD17] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Information and Computation*, 256:2–8, 2017. doi:10.1016/j.ic.2017.04.004. (cit. on p. 1, 10)

[AFvMV06] Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. Computational depth: Concept and applications. *Theor. Comput. Sci.*, 354(3):391–404, 2006. doi:10.1016/j.tcs.2005.11.033. (cit. on p. 16)

[AGvM$^+$18] Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. *SIAM J. Comput.*, 47(4):1339–1372, 2018. doi:10.1137/17M1157970. (cit. on p. 10)

[AH19] Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. *ACM Transactions on Computation Theory*, 11(4):27:1–27:27, 2019. doi:10.1145/3349616. (cit. on p. 1, 10)

[AHK17] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *Comput. Complex.*, 26(2):469–496, 2017. doi:10.1007/s00037-016-0124-0. (cit. on p. 1)

[AHM$^+$08] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and $\mathbf{AC}^0$ circuits given a truth table. *SIAM J. Comput.*, 38(1):63–84, 2008. doi:10.1137/060664537. (cit. on p. 1)

[AIV19] Eric Allender, Rahul Ilango, and Neekon Vafa. The non-hardness of approximating circuit size. In *Proc. 14th International Computer Science Symposium in Russia (CSR)*, volume 11532 of *Lecture Notes in Computer Science*, pages 13–24, 2019. doi:10.1007/978-3-030-19955-5_2. (cit. on p. 1)

[Ajt83] Miklós Ajtai. Σ_1^1-formulae on finite structures. *Ann. Pure. Appl. Log.*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6. (cit. on p. 3)

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *Proc. 21st Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 2245 of *Lecture Notes in Computer Science*, pages 1–15, 2001. doi:10.1007/3-540-45294-X_1. (cit. on p. 9)

[All17] Eric Allender. The complexity of complexity. In *Computability and Complexity*, volume 10010 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2017. doi:10.1007/978-3-319-50062-1_6. (cit. on p. 1, 9)

[All21] Eric Allender. Vaughan Jones, Kolmogorov complexity, and the new complexity landscape around circuit minimization. *New Zealand Journal of Mathematics*, 52:585–604, 2021. doi:10.53733/148. (cit. on p. 9)

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306. (cit. on p. 7, 42)

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of **NP**. *Journal of the ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901. (cit. on p. 7, 42)

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in $\mathbf{NC}^1$. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989. doi:10.1016/0022-0000(89)90037-8. (cit. on p. 1)

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In *IWCC*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011. doi:10.1007/978-3-642-20901-7_2. (cit. on p. 5)

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. doi:10.1137/S0097539701398521. (cit. on p. 5)

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. **BPP** has subexponential time simulations unless **EXPTIME** has publishable proofs. *Computatioanl Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486. (cit. on p. 1)

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012. doi:10.1145/2160158.2160159. (cit. on p. 5)

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. In *Contemporary Mathematics*, volume 324, pages 71–90. American Mathematical Society, 2003. doi:10.1090/conm/324/05731. (cit. on p. 7, 31)

[BT06] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for **NP** problems. *SIAM Journal of Computing*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974. (cit. on p. 1, 8)

[CDGS18] Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In *EUROCRYPT (1)*, volume 10820 of *Lecture Notes in Computer Science*, pages 227–258. Springer, 2018. doi:10.1007/978-3-319-78381-9_9. (cit. on p. 27, 28, 42)

[CHI$^+$21] Marco Carmosino, Kenneth Hoover, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Lifting for constant-depth circuits and applications to MCSP. In *ICALP*, volume 198 of *LIPIcs*, pages 44:1–44:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.44. (cit. on p. 3)

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638. (cit. on p. 5)

[DS04] Irit Dinur and Shmuel Safra. On the hardness of approximating label-cover. *Inf. Process. Lett.*, 89(5):247–254, 2004. doi:10.1016/j.ipl.2003.11.007. (cit. on p. 7)

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC*, pages 624–633. ACM, 2014. doi:10.1145/2591796.2591884. (cit. on p. 7, 9)

[ERSY22] Reyad Abed Elrazik, Robert Robere, Assaf Schuster, and Gal Yehuda. Pseudorandom self-reductions for **NP**-complete problems. In *ITCS*, volume 215 of *LIPIcs*, pages 65:1–65:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.65. (cit. on p. 8, 9, 56)

[Fei98] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi:10.1145/285055.285059. (cit. on p. 9)

[Fel09] Vitaly Feldman. Hardness of approximate two-level logic minimization and PAC learning with membership queries. *J. Comput. Syst. Sci.*, 75(1):13–26, 2009. doi:10.1016/j.jcss.2008.07.007. (cit. on p. 1)

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:10.1137/0222061. (cit. on p. 1)

[FM05] Gudmund Skovbjerg Frandsen and Peter Bro Miltersen. Reviewing bounds on the circuit size of the hardest functions. *Information Processing Letters*, 95(2):354–357, 2005. doi:10.1016/j.ipl.2005.03.009. (cit. on p. 65)

[FNV17] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In *Public Key Cryptography (1)*, volume 10174 of *Lecture Notes in Computer Science*, pages 121–150. Springer, 2017. doi:10.1007/978-3-662-54365-8_6. (cit. on p. 14)

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. doi:10.1007/3-540-47721-7_12. (cit. on p. 46)

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431. (cit. on p. 3)

[GGH12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. *IACR Cryptol. ePrint Arch.*, page 610, 2012. URL: http://eprint.iacr.org/2012/610. (cit. on p. 7)

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016. doi:10.1137/14095772X. (cit. on p. 5, 12)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503. (cit. on p. 1, 57)

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476. ACM, 2013. doi:10.1145/2488608.2488667. (cit. on p. 3, 5, 7, 11, 21, 22, 30)

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor Carboni Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *CCC*, volume 234 of *LIPIcs*, pages 16:1–16:60. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.16. (cit. on p. 14, 15)

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989. doi:10.1145/73007.73010. (cit. on p. 13, 14)

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. doi:10.1016/0022-0000(84)90070-9. (cit. on p. 5)

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. doi:10.1017/CBO9780511546891. (cit. on p. 11, 13)

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20. ACM, 1986. doi:10.1145/12130.12132. (cit. on p. 2, 3)

[Hås96] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *FOCS*, pages 627–636. IEEE Computer Society, 1996. doi:10.1109/SFCS.1996.548522. (cit. on p. 9)

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708. (cit. on p. 1, 3, 57)

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within **NP**. In *FOCS*, pages 247–258, 2018. doi:10.1109/FOCS.2018.00032. (cit. on p. 1, 2)

[Hir20] Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2020. doi:10.1145/3357713.3384251. (cit. on p. 1)

[Hir22a] Shuichi Hirahara. **NP**-hardness of learning programs and partial MCSP. In *FOCS*, pages 968–979. IEEE, 2022. doi:10.1109/FOCS54457.2022.00095. (cit. on p. 1, 3, 7, 10, 11)

[Hir22b] Shuichi Hirahara. Symmetry of information from meta-complexity. In *CCC*, volume 234 of *LIPIcs*, pages 26:1–26:41. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.26. (cit. on p. 1, 2, 4, 5, 9, 10, 16, 18)

[Hir23] Shuichi Hirahara. Capturing one-way functions via **NP**-hardness of meta-complexity. In *STOC*, 2023. to appear. (cit. on p. 10)

[HOS18] Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. **NP**-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *CCC*, volume 102 of *LIPIcs*, pages 5:1–5:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.5. (cit. on p. 1, 3)

[HP15] John M. Hitchcock and Aduri Pavan. On the **NP**-completeness of the minimum circuit size problem. In *Proc. 35th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 45 of *LIPIcs*, pages 236–245, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.236. (cit. on p. 1, 3)

[HS17] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *CCC*, volume 79 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.7. (cit. on p. 8, 56)

[HW16] Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Proc. 31st Computational Complexity Conference (CCC)*, volume 50 of *LIPIcs*, pages 18:1–18:20, 2016. doi:10.4230/LIPIcs.CCC.2016.18. (cit. on p. 1, 10, 11)

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In *Proc. 33rd Computational Complexity Conference (CCC)*, volume 102 of *LIPIcs*, pages 7:1–7:20, 2018. doi:10.4230/LIPIcs.CCC.2018.7. (cit. on p. 1, 2, 10)

[Ila20a] Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and $\mathbf{AC}^0[p]$. In *Proc. 11th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 151 of *LIPIcs*, pages 34:1–34:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.34. (cit. on p. 1, 4, 6, 7, 9)

[Ila20b] Rahul Ilango. Constant depth formula and partial function versions of MCSP are hard. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2020. doi:10.1109/FOCS46700.2020.00047. (cit. on p. 1, 2, 3)

[ILO20] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. **NP**-hardness of circuit minimization for multi-output functions. In *CCC*, volume 169 of *LIPIcs*, pages 22:1–22:36, 2020. doi:10.4230/LIPIcs.CCC.2020.22. (cit. on p. 1, 10)

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proc. 10th Annual Structure in Complexity Theory Conference*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853. (cit. on p. 1)

[IRS22] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. Robustness of average-case meta-complexity via pseudorandomness. In *STOC*, pages 1575–1583. ACM, 2022. doi:10.1145/3519935.3520051. (cit. on p. 1)

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, pages 60–73. ACM, 2021. doi:10.1145/3406325.3451093. (cit. on p. 3, 5, 12)

[JLS22a] Aayush Jain, Huijia Lin, and Amit Sahai. Personal Communication, 2022. (cit. on p. 5, 13)

[JLS22b] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in $\mathbf{NC}^0$. In *EUROCRYPT (1)*, volume 13275 of *Lecture Notes in Computer Science*, pages 670–699. Springer, 2022. doi:10.1007/978-3-031-06944-4_23. (cit. on p. 5, 12, 13)

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9. (cit. on p. 7, 30)

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314. (cit. on p. 1, 2, 3, 10)

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732. ACM, 1992. doi:10.1145/129712.129782. (cit. on p. 45)

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In *ITCS*, volume 185 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.44. (cit. on p. 9)

[KMN$^+$14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *FOCS*, pages 374–383. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.47. (cit. on p. 3)

[KNY17] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for **NP**. *J. Cryptol.*, 30(2):444–469, 2017. doi:10.1007/s00145-015-9226-0. (cit. on p. 5, 10)

[Ko86] Ker-I Ko. On the notion of infinite pseudorandom sequences. *Theor. Comput. Sci.*, 48(3):9–33, 1986. doi:10.1016/0304-3975(86)90081-2. (cit. on p. 4)

[Ko91] Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM Journal of Computing*, 20(5):962–986, 1991. doi:10.1137/0220059. (cit. on p. 10)

[Kol65] Andrei N Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1965. doi:10.1080/00207166808803030. (cit. on p. 4)

[KS08] Subhash Khot and Rishi Saket. Hardness of minimizing and learning DNF expressions. In *FOCS*, pages 231–240. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.37. (cit. on p. 1, 2)

[Lev] Leonid Levin. Hardness of search problems. URL: https://www.cs.bu.edu/fac/lnd/research/hard.htm. (cit. on p. 1)

[Lev73] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973. (cit. on p. 1)

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254, 2020. doi:10.1109/FOCS46700.2020.00118. (cit. on p. 1, 2)

[LP21a] Yanyi Liu and Rafael Pass. Cryptography from sublinear-time average-case hardness of time-bounded Kolmogorov complexity. In *STOC*, pages 722–735. ACM, 2021. doi:10.1145/3406325.3451121. (cit. on p. 1)

[LP21b] Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on $\mathbf{EXP} \neq \mathbf{BPP}$. In *Proc. 41st Annual International Cryptology Conference (CRYPTO)*, volume 12825 of *Lecture Notes in Computer Science*, pages 11–40. Springer, 2021. doi:10.1007/978-3-030-84242-0_2. (cit. on p. 1)

[LP22] Yanyi Liu and Rafael Pass. On one-way functions from **NP**-complete problems. In *CCC*, volume 234 of *LIPIcs*, pages 36:1–36:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.36. (cit. on p. 1, 4, 9)

[Lup58] Oleg B Lupanov. On the synthesis of switching circuits. *Doklady Akademii Nauk SSSR*, 119(1):23–26, 1958. (cit. on p. 65)

[LV08] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, Third Edition*. Texts in Computer Science. Springer, 2008. doi:10.1007/978-0-387-49820-1. (cit. on p. 14)

[Mer89] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, 1989. doi:10.1007/0-387-34805-0_21. (cit. on p. 43)

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000. doi:10.1137/S0097539795284959. (cit. on p. 3, 7, 8, 42, 46, 47)

[MW17] Cody D. Murray and R. Ryan Williams. On the (non) **NP**-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017. doi:10.4086/toc.2017.v013a004. (cit. on p. 1, 3, 6)

[Raz87] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987. (cit. on p. 3)

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494. (cit. on p. 1, 3)

[RS21] Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. In *Proc. 36th Computational Complexity Conference (CCC)*, volume 200 of *LIPIcs*, pages 35:1–35:58, 2021. doi:10.4230/LIPIcs.CCC.2021.35. (cit. on p. 1)

[Rud17] Michael Rudow. Discrete logarithm and minimum circuit size. *Inf. Process. Lett.*, 128:1–4, 2017. doi:10.1016/j.ipl.2017.07.005. (cit. on p. 10)

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984. doi:10.1007/3-540-39568-7_5. (cit. on p. 5)

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997. doi:10.1007/3-540-69053-0_18. (cit. on p. 7, 30)

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *STOC*, pages 330–335. ACM, 1983. doi:10.1145/800061.808762. (cit. on p. 4)

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404. (cit. on p. 3)

[Smo93] Roman Smolensky. On representations by low-degree polynomials. In *FOCS*, pages 130–138. IEEE Computer Society, 1993. doi:10.1109/SFCS.1993.366874. (cit. on p. 3)

[SS20] Michael Saks and Rahul Santhanam. Circuit lower bounds from **NP**-hardness of MCSP under Turing reductions. In *Proc. 35th Computational Complexity Conference (CCC)*, volume 169 of *LIPIcs*, pages 26:1–26:13, 2020. doi:10.4230/LIPIcs.CCC.2020.26. (cit. on p. 1, 3)

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005. doi:10.1007/11426639_27. (cit. on p. 5)

[Tra84] Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036. (cit. on p. 2)

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Comput. Complex.*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x. (cit. on p. 1)

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and Null-IO from evasive LWE. In *ASIACRYPT (1)*, volume 13791 of *Lecture Notes in Computer Science*, pages 195–221. Springer, 2022. doi:10.1007/978-3-031-22963-3_7. (cit. on p. 6)

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *FOCS*, pages 1–10. IEEE Computer Society, 1985. doi:10.1109/SFCS.1985.49. (cit. on p. 3)

[Zuc07] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.*, 3(1):103–128, 2007. doi:10.4086/toc.2007.v003a006. (cit. on p. 9)

A Optimality of Corollary 4.8

Claim A.1. *Let $0 < \delta_1, \delta_2 < 1$ be two constants such that $\delta_1 + 2\delta_2 > 1$, and $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function where n is large enough. Then there is a circuit C of size $O(2^{\delta_1 n})$ that*

$$\Pr_{x \leftarrow \{0,1\}^n}[C(x) = f(x)] \geq 1/2 + 2^{-\delta_2 n}.$$

The following proof was suggested by Lijie Chen (personal communication).

Proof of Claim A.1. We use the fact that any Boolean function f over n' inputs has correlation $\Omega(2^{-n'/2})$ with some PARITY function. That is, there is a string $y \in \{0,1\}^{n'}$ and a bit b such that

$$\Pr_{x \leftarrow \{0,1\}^{n'}}[\mathsf{IP}(x, y) = f(x) \oplus b] \geq 1/2 + \Omega(2^{-n'/2}),$$

where IP denotes the inner product function.

For every input x, we split it into the concatenation of x_1 and x_2, where $|x_1| = \delta_1 n$ and $|x_2| = (1-\delta_1)n < 2\delta_2 n$. (Without loss of generality we assume $\delta_1 n$ is an integer.) For $x_1 \in \{0,1\}^{\delta_1 n}$, let $g : \{0,1\}^{\delta_1 n} \to \{0,1\}^{(1-\delta_1)n}$ and $h : \{0,1\}^{\delta_1 n} \to \{0,1\}$ encode the PARITY function that has non-trivial correlation with the sub-function $f(x_1, -)$, i.e.,

$$\Pr_{x_2 \leftarrow \{0,1\}^{(1-\delta_1)n}}[\mathsf{IP}(g(x_1), x_2)) = f(x_1, x_2) \oplus h(x_1)] \geq 1/2 + \Omega(2^{-(1-\delta_1)n/2}) \geq 1/2 + 2^{-\delta_2 n}.$$

Using Lupanov's construction [Lup58, FM05], the functions g and h can be computed in $O(n \cdot (2^{\delta_1 n}/n)) \leq O(2^{\delta_1 n})$ size. Let C be the circuit such that $C(x_1, x_2) = \mathsf{IP}(g(x_1), x_2) \oplus h(x_1)$, then it is easy to see that the size of C is $O(2^{\delta_1 n})$ and that

$$\Pr_{x \leftarrow \{0,1\}^n}[C(x) = f(x)] \geq 1/2 + 2^{-\delta_2 n}. \qquad \square$$

B Random Oracles Are Incompressible

Theorem B.1. *Let $O \subseteq \{0,1\}^\star$ be a random oracle, then w.p. 1 there is a constant $c \geq 1$ such that for all but finitely many $N \in \mathbb{N}$,*

$$\mathrm{K}(O_{\leq N}) > 2^{N+1} - cN.$$

Proof. First, we show that for all but finitely many $N \in \mathbb{N}$,

$$\mathrm{K}(O_N \mid O_{\leq(N-1)}) > 2^N - 2N. \tag{14}$$

Suppose that each O_N is randomly generated. Then the probability that Eq. (14) holds is at least $1 - 2^{-2N}$. Fix $N_0 \geq 2$, the probability that for every $N \geq N_0$, Eq. (14) holds is at least

$$\prod_{N \geq N_0}(1 - 2^{-2N}) \geq \prod_{N \geq N_0} 0.1^{2^{-2N}} \geq 0.1^{2^{-2N_0+1}} \geq 1 - 2^{-2N_0+3}.$$

(We used that for $x \in (0, 1/2)$, $1 - x \geq 0.1^x \geq 1 - 4x$.) It follows that for every $\epsilon > 0$, there is $N_0 := \Theta(\log(1/\epsilon))$ such that w.p. $1 - \epsilon$ over a random oracle O, for every $N \geq N_0$, Eq. (14) holds. Thus, w.p. 1 over a random oracle O, Eq. (14) holds for all but finitely many N.

By symmetry of information in Kolmogorov complexity, there is a universal constant $c_1 \geq 1$ such that $\mathrm{K}(O_{\leq N}) \geq \mathrm{K}(O_{\leq(N-1)}) + \mathrm{K}(O_N \mid O_{\leq(N-1)}) - c_1 N$. It is thus easy to prove, by induction on N, that there exists a constant $c \geq 1$ such that for all but finitely many N,

$$\mathrm{K}(O_{\leq N}) > 2^{N+1} - cN. \qquad \square$$