Electronic Colloquium on Computational Complexity, Report No. 213 (2023)

Hard Languages in $\mathsf{NP} \cap \mathsf{coNP}$ and NIZK Proofs from Unstructured Hardness*

Riddhi Ghosal† Yuval Ishai‡ Alexis Korb§ Eyal Kushilevitz¶ Paul Lou‖
Amit Sahai**

June 2023

Abstract

The existence of "unstructured" hard languages in $\mathsf{NP} \cap \mathsf{coNP}$ is an intriguing open question. Bennett and Gill (SICOMP, 1981) asked whether P is separated from $\mathsf{NP} \cap \mathsf{coNP}$ relative to a random oracle, a question that remained open ever since. While a hard language in $\mathsf{NP} \cap \mathsf{coNP}$ can be constructed in a black-box way from a *one-way permutation*, for which only few (structured) candidates exist, Bitansky et al. (SICOMP, 2021) ruled out such a construction based on an *injective one-way function*, an unstructured primitive that is easy to instantiate heuristically. In fact, the latter holds even with a black-box use of indistinguishability obfuscation.

We give the first evidence for the existence of unstructured hard languages in $\mathsf{NP} \cap \mathsf{coNP}$ by showing that if $\mathsf{UP} \not\subseteq \mathsf{RP}$, which follows from the existence of injective one-way functions, the answer to Bennett and Gill's question is affirmative: with probability 1 over a random oracle $\mathcal{O}$, we have that $\mathsf{P}^{\mathcal{O}} \neq \mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}$. Our proof gives a constructive *non-black-box* approach for obtaining candidate hard languages in $\mathsf{NP} \cap \mathsf{coNP}$ from cryptographic hash functions.

The above conditional separation builds on a new construction of *non-interactive zero-knowledge* (NIZK) proofs, with a computationally unbounded prover, to convert a hard promise problem into a hard language. We obtain such NIZK proofs for NP, with a *uniformly random* reference string, from a special kind of hash function which is implied by (an unstructured) random oracle. This should be contrasted with previous constructions of such NIZK proofs that are based on one-way permutations or other *structured* primitives, as well as with (computationally sound) NIZK *arguments* in the random oracle model.

*This is the full version of [GIK+23].

†UCLA. Email: riddhi@cs.ucla.edu.
‡Technion. Email: yuvali@cs.technion.ac.il
§UCLA. Email: alexiskorb@cs.ucla.edu.
¶Technion. Email: eyalk@cs.technion.ac.il
‖UCLA. Email: pslou@cs.ucla.edu.
**UCLA. Email: sahai@cs.ucla.edu.

ISSN 1433-8092

Contents

1 Introduction

What makes us believe that a complexity class contains *hard problems* that cannot be solved in polynomial time?

One source of confidence is the existence of *concrete* computational problems in the class for which no efficient algorithms were found, despite years of intensive efforts. For instance, integer factorization is an example of one such problem. However, the same kind of structure and mathematical elegance that attracts scientists to study a problem also makes this problem more susceptible to algorithmic shortcuts. Indeed, the best known (classical) factoring algorithms are super-polynomially faster than the naive algorithm, and some experts believe that a polynomial-time algorithm is likely to exist. In cryptography, such structured problems from number theory or linear algebra serve as the basis of existing candidates for *public-key encryption*.

A better source of confidence is the existence of *unstructured* candidates for hard problems in the class. These can be obtained by first constructing a provably hard problem based on an unstructured (but computationally inefficient) *random* function, and then heuristically replacing the random function by a "random-looking" efficiently computable function. For example, this methodology can yield candidates for hard-on-average NP languages from every practical[1] cryptographic hash function or *private-key* encryption scheme. This gives us more confidence in the $\mathsf{P} \neq \mathsf{NP}$ conjecture. In the same vein, the ability to use unstructured hardness for *derandomizing* BPP [NW94, IW97] gives us more confidence in the $\mathsf{P} = \mathsf{BPP}$ conjecture.

The curious case of $\mathsf{NP} \cap \mathsf{coNP}$. Unlike the case of NP, candidate hard languages in $\mathsf{NP} \cap \mathsf{coNP}$ are highly structured and scarce. It is known that the existence of a cryptographic one-way permutation (OWP) implies such a language [Bra79, BG81], but the only candidate constructions for OWP rely on the hardness of factoring or discrete logarithm. To make things worse, the class $\mathsf{NP} \cap \mathsf{coNP}$ is not known to contain complete languages [Sip82, HI85], and most current candidates for hard languages in the class are known to have polynomial-time quantum algorithms.[2] Finally, the equality $\mathsf{P} = \mathsf{NP} \cap \mathsf{coNP}$ holds for simple computational models such as decision trees [IN88]. The above state of affairs has led to doubts about whether $\mathsf{P} \neq \mathsf{NP} \cap \mathsf{coNP}$ is the "right" conjecture to make [Sta].

Random oracle separation? A natural approach for using unstructured hardness to separate between classes $\mathcal{C}_1$ and $\mathcal{C}_2$ is to show a separation *relative to a random oracle*. That is, with probability 1 over a random oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}$, we have that $\mathcal{C}_1^{\mathcal{O}} \neq \mathcal{C}_2^{\mathcal{O}}$. Using cryptographic hash functions as a heuristic substitute for $\mathcal{O}$, this may give an explicit candidate separation based on unstructured hardness.[3] Originating from the work of Bennett and Gill [BG81], many such separations were obtained. In particular, a breakthrough work by Rossman et al. [RST15] showed that the polynomial hierarchy is infinite relative to a random oracle, and a very recent breakthrough

[1] Practical hash functions and block ciphers are typically defined only for a fixed level of hardness. However, it is relatively easy to come up with explicit candidates that are meaningful in an asymptotic setting. See [MV15, AGR+16, AGP+19] for some concrete examples.

[2] The only exceptions we are aware of are variants of the stochastic games problem [Con92]. In contrast, there are many natural candidate hard problems in the *promise* version of $\mathsf{NP} \cap \mathsf{coNP}$, and in fact such hard promise problems can be based on the assumption that $\mathsf{P} \neq \mathsf{NP}$ [ESY84]. Our main result is based on such hard promise problems in which the promise set is in NP, whose existence follows from $\mathsf{P} \neq \mathsf{UP}$ or an injective one-way function.

[3] The so-called "random oracle hypothesis," asserting that natural separations relative to a random oracle apply also in the unrelativized world, was refuted by Chang et al. [CCG+94], who showed that relative to a random oracle $\mathsf{IP} \neq \mathsf{PSPACE}$. However, the hypothesis is still plausible for "low-end" separations between P and other natural complexity classes, in the spirit of the random oracle methodology in cryptography.

work by Yamakawa and Zhandry [YZ22] separated BQP machines from BPP machines with respect to NP *search* problems. However, the question of separating $\mathsf{NP} \cap \mathsf{coNP}$ from P relative to a random oracle remains open since it was posed in 1981 by Bennett and Gill. Tardos [Tar89] offered a partial explanation for the difficulty of obtaining such a separation. Furthermore, techniques from more recent works on random oracle separations, such as those developed in [RST15], do not seem helpful. See [Bar] for discussion.

Black-box separation? Another natural approach for obtaining hard languages in $\mathsf{NP} \cap \mathsf{coNP}$ is to construct them from standard cryptographic primitives. Constructions of this kind are almost always (fully) *black-box* [RTV04], in the sense that both the construction and the hardness reduction make an oracle use of the primitive. Indeed, the aforementioned construction of a hard language in $\mathsf{NP} \cap \mathsf{coNP}$ based on a OWP is black-box in this sense, where in particular the NP and coNP verifiers make a black-box use of the OWP. However, as discussed above, OWP is a highly structured primitive: a random function is a permutation with only negligible probability. In contrast, a one-way function (OWF) or even an *injective* OWF, are unstructured primitives that can be realized from random functions. Indeed, a random length-tripling function $f : \{0,1\}^n \to \{0,1\}^{3n}$ is both one-way and injective except with negligible probability. Blum and Impagliazzo [BI87] and Rudich [Rud88] ruled out a black-box construction of a hard language in $\mathsf{NP} \cap \mathsf{coNP}$ from a OWF. Bitansky et al. [BDV21] strengthened this impossibility to rule out (perfectly correct) constructions making a black-box use of an *injective* OWF. A similar result for $\mathsf{UP} \cap \mathsf{coUP}$ is implied by the work of Rosen et al. [RSS21]. The result of [BDV21] was shown to hold even when additionally allowing a black-box use of *indistinguishability obfuscation*. This implies a similar black-box impossibility based on a host of other cryptographic primitives, including public-key encryption. Altogether, the prospects of using cryptography to get unstructured hardness in $\mathsf{NP} \cap \mathsf{coNP}$ seemed low.

1.1 Our Results

We give the first evidence for the existence of unstructured hard languages in $\mathsf{NP} \cap \mathsf{coNP}$. Concretely, we show that if $\mathsf{UP} \not\subseteq \mathsf{RP}$, which follows from the existence of injective one-way functions, the answer to Bennett and Gill's question is affirmative.

Theorem 1.1. *If* $\mathsf{UP} \not\subseteq \mathsf{RP}$*, then with probability* 1 *over the choice of a random oracle* $\mathcal{O}$*,* $\mathsf{P}^{\mathcal{O}} \neq \mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}$*.*

Similarly to the previous separation between P and $\mathsf{NP} \cap \mathsf{coNP}$ based on one-way permutations, the conclusion is in fact stronger: $\mathsf{P}^{\mathcal{O}} \neq \mathsf{UP}^{\mathcal{O}} \cap \mathsf{coUP}^{\mathcal{O}}$. Moreover, if we make the stronger assumption that an injective one-way function exists, the conclusion holds with average-case hardness. What makes our conditional separation interesting is the *unstructured assumption*. Indeed, as discussed above, an injective one-way function (which implies $\mathsf{UP} \not\subseteq \mathsf{RP}$) is unstructured in the sense that it can be obtained with overwhelming probability from a random function.

The proof of Theorem 1.1 departs from the black-box impossibility framework of [BDV21] in two ways. First, it only provides a probabilistic rather than perfect correctness guarantee. Second, it inherently makes a *non-black-box* use of a hard UP language. In fact, the proof gives us a *constructive* separation in the random oracle model that makes a non-black-box use of any injective OWF. If we further instantiate the injective OWF and the random oracle using a cryptographic hash function, we get an explicit algorithm for converting the code of the hash function into the code of an NP-verifier and a coNP-verifier which are *heuristically* conjectured to define a hard language in $\mathsf{NP} \cap \mathsf{coNP}$.

The curious case of NIZK proofs. The proof of Theorem 1.1 relies on a novel connection with another intriguing open question: the possibility of constructing non-interactive zero-knowledge (NIZK) *proofs* for NP from unstructured hardness assumptions. We consider here NIZK proofs in the "uniform random string" (URS) model [BFM88], where the prover and verifier share a *uniformly random* reference string,[4] and allow both an honest and a malicious prover to be computationally unbounded. Even in this rather liberal setting, all previous NIZK proofs for NP relied on *structured* assumptions (see Section 1.2). The Fiat-Shamir heuristic [FS87] can be used to obtain NIZK for NP in the *random oracle model*, without requiring any structure. However, the protocols obtained via this approach are currently limited to *arguments* that only guarantee soundness against computationally bounded provers, except for very special NP languages [IV19]. Our second main result gives the first construction of NIZK proofs for NP from unstructured hardness.

Theorem 1.2. *There exists an (unbounded-prover, URS-based)* NIZK *proof system for* NP *in the random oracle model.*

In fact, here we can instantiate the random oracle by an explicit hash function, or family of hash functions, that has simple to state (but nonstandard) properties of a random function (see Definition 4.6 for more details). Roughly speaking, we need $H : \{0,1\}^n \to \{0,1\}^n$ to satisfy three properties: (1) The image size $H(\{0,1\}^n)$ covers a δ-fraction of the co-domain, where $0 < \delta < 1$ can be approximated to a high level of precision in time $\mathsf{poly}(n)$; (2) The output $H(x)$ on a random input x is pseudorandom; (3) It is hard to distinguish between $(x, H(x))$ for a random x and $(H^{-1}(y), y)$, where y is a uniformly random image of H and $H^{-1}(y)$ is a random preimage of y. While this new primitive is implied by the existence of a one-way permutation, what makes it useful for our purposes is the fact that it can be constructed from a random oracle.

From NIZK proofs to a conditional separation. Finally, our main conditional separation (Theorem 1.1) is obtained by applying the NIZK proof to a language defined by the hard UP-relation. This inherently makes non-black-box use of the relation. The high-level idea is to use the NIZK proof to convert a hard *promise problem* in $\mathsf{NP} \cap \mathsf{coNP}$, naturally defined by the UP-relation, into a hard *language* in $\mathsf{NP} \cap \mathsf{coNP}$. Here we rely on the fact that the language defining the promise is in NP. The statistical soundness of the NIZK is used to eliminate the promise, whereas the zero-knowledge property of the NIZK ensures that hardness is maintained. See the technical overview in Section 2 for more details.

Open problems. Our work gives the first evidence that unstructured hardness implies the existence of hard languages in $\mathsf{NP} \cap \mathsf{coNP}$. However, because we make essential non-black-box use of unstructured hardness, our techniques do not yield an unconditional separation in the random oracle model. Resolving this 42-year-old question remains open. On the NIZK side, we leave open the questions of instantiating our construction from standard cryptographic assumptions and obtaining efficient-prover variants.

1.2 Related Work

Hard languages in $\mathsf{NP} \cap \mathsf{coNP}$. A language in $(\mathsf{NP} \cap \mathsf{coNP}) \setminus P$ is easy to construct directly based on the conjectured hardness of factoring integers or computing discrete logarithms. For

[4] Allowing a structured reference string would make (unbounded-prover) NIZK proofs realizable from any one-way function [Ps05]. However, in the context of our main application, this relaxation would only lead to a hard *promise problem* in $\mathsf{NP} \cap \mathsf{coNP}$, which is easy to obtain directly from any hard UP language.

instance, a factoring-based language can include the pairs (x, i) such that the i-th bit of (a canonical representation of) the factorization of x is 1.

A similar approach can be used to construct a hard language in $\mathsf{NP} \cap \mathsf{coNP}$ from any *one-way permutation* (OWP): a length-preserving, injective one-way function [Bra79, BG81]. While a deterministic OWP can be constructed assuming the one-wayness of factoring and variants of the discrete logarithm problem [GLN11], it is much easier to construct randomized *families* of OWPs, where each permutation is specified by a key. However, in order to directly obtain a hard language in $\mathsf{NP} \cap \mathsf{coNP}$ from such a family, it is crucial that the family be *certifiable* in the sense that one can efficiently recognize valid keys. All known constructions of such OWP families rely on the hardness of factoring or discrete logarithm. While a OWP family can also be constructed from indistinguishability obfuscation and a one-way function [BPW16], this OWP family is not certifiable.

A very different candidate for a hard language in $\mathsf{NP} \cap \mathsf{coNP}$ was given by Condon [Con92], who showed that the problem of deciding which player has the greatest chance of winning a stochastic two-player game is in $\mathsf{NP} \cap \mathsf{coNP}$. The best known (randomized) algorithms for this language run in time $2^{O(\sqrt{n})}$ [Lud95].

Complexity theoretic applications of NIZK. Our work uses NIZK proofs for NP to convert a hard promise problem in $\mathsf{NP} \cap \mathsf{coNP}$ into a hard language. There are several previous works that use different flavors of NIZK proofs in complexity theoretic contexts. In particular, variants of NIZK were used by Naor [Nao96] to separate two notions of learning distributions, by Hsiao et al. [HLR07] to separate two notions of computational entropy, by Ishai et al. [IKOS10] to rule out invertible sampling algorithms for general distributions, and by Hubácek et al. [HNY17] to construct a hard search problem in TFNP with at most two solutions. These works are similar to ours in that NIZK is used to enforce some promise while respecting hardness. However, whereas these works relied on prior constructions of NIZK that were based on structured hardness assumptions, our current application requires a new flavor of NIZK proofs based on unstructured hardness.

Assumptions for NIZK proofs. Unlike most cryptographic applications of NIZK proofs, in this work we allow the honest prover to be computationally unbounded. Pass and shelat [Ps05] construct such NIZK proofs for NP (in fact, AM) from any one-way function. Unlike our construction, which only requires a *uniform* reference string (URS), this construction inherently relies on a *structured* reference string which is picked from a special distribution. This makes it unsuitable for our complexity theoretic application. A technical similarity between the construction from [Ps05] and ours is that both use each segment of the reference string to define a hidden bit that depends on whether the segment belongs to the image of some function. Finally, Ball et al. [BDK20] show how to convert any unbounded-prover NIZK proof in the URS model into a ZAP (2-message witness-indistinguishable proof), extending the bounded-prover construction of Dwork and Naor [DN07].

Allowing structured assumptions, NIZK proofs for NP in the URS model can be based on OWP families [FLS90, BY96, CL18] (hence on the hardness of factoring and discrete logarithm), the decision linear assumption on bilinear groups [GOS06], or indistinguishability obfuscation and one-way functions [BP15]. Known lattice-based constructions of NIZK proofs [CCH+19, PS19] require a structured reference string. This requirement can be eliminated by settling for (computationally sound) *arguments*, which do not suffice for our purposes. Similarly, we cannot rely on NIZK arguments in the random oracle model [FS86]. Instead, we present a new construction of NIZK *proofs* in the random oracle model.

Finally, we note that all of the above NIZK constructions except those based on one-way func-

tions and discrete logarithm satisfy the stronger property of *efficient-prover* NIZK, where the prover can be implemented efficiently given an NP-witness. Such NIZK *arguments* are also known under the (subexponential) decisional Diffie-Hellman (DDH) assumption [JJ21]. The existence of efficient-prover NIZK *proofs* for NP in the random oracle model remains open.

Randomness vs. structure. While a random function has many useful properties that are hard to (provably) realize by explicit constructions, in this work we view a uniformly random function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ as an "unstructured" object. This view is common both in theoretical computer science and mathematics (see, e.g., [Bar, Tao]). Following this view, is useful to draw a dividing line between "unstructured" and "structured" cryptographic primitives or hardness assumptions depending on whether they can be based on a random oracle. Beyond the well-known separation between OWF and public-key encryption [IR89], this dichotomy is also useful when considering different flavors of OWF. To illustrate this, consider the following three flavors of OWF:

1. An injective OWF $f : \{0,1\}^n \to \{0,1\}^n$, namely a OWP;
2. An injective OWF $f : \{0,1\}^n \to \{0,1\}^{1.5n}$;
3. An injective OWF $f : \{0,1\}^n \to \{0,1\}^{3n}$.

It is clear that $1 \Rightarrow 2 \Rightarrow 3$. But is there a qualitative separation? The above dichotomy puts (3) in the "unstructured" category, since a random $f : \{0,1\}^n \to \{0,1\}^{3n}$ is injective with overwhelming probability, whereas (1) and (2) seem to be in the "structured" category, since it is not clear how to efficiently build such functions from a random oracle. In particular, a random $f : \{0,1\}^n \to \{0,1\}^{1.5n}$ is *not* injective, except with negligible probability. This taxonomy also suggests a qualitative difference in the abundance and concrete efficiency of explicit candidates. Indeed, candidates for (3) can be based on any of the many practical proposals for cryptographic hash functions and block ciphers, whereas known candidates for (1) and (2) are based on specific "public-key" assumptions, leading to significantly worse concrete efficiency and polynomial-time quantum attacks.

2 Technical Overview

In this section we give a high-level technical overview of our results. We start, in Section 2.1, by describing the general strategy of using NIZK to construct a hard language in $\mathsf{NP} \cap \mathsf{coNP}$. Then, in Section 2.2, we describe our construction of NIZK proofs in the random oracle model, which is our main technical contribution.

2.1 Separating P from NP ∩ coNP Using NIZK

Our primary contribution is achieving a separation between $\mathsf{P}^{\mathcal{O}}$ and $\mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}$, with probability 1 over the choice of the random oracle $\mathcal{O}$, assuming an *unstructured* source of hardness. An example of an unstructured source of hardness that suffices for our separation result is the existence of an injective one-way function.[5] In fact, it will suffice to merely assume $\mathsf{UP} \not\subseteq \mathsf{RP}$, which is implied by the existence of injective one-way functions. Formally, our main theorem is the following:

[5] Recall that we view an injective one-way function as an "unstructured" primitive because a random oracle $\mathcal{O} : \{0,1\}^n \to \{0,1\}^{3n}$ is both injective and one-way with overwhelming probability. "Unstructured hardness" is typically associated with random functions: see, e.g., [Bar, BDV21] for discussion.

Theorem 2.1. *Assuming* $\mathsf{UP} \not\subseteq \mathsf{RP}$*, we have:*

$$\Pr_{\mathcal{O}}\left[\mathsf{P}^{\mathcal{O}} \neq \mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}\right] = 1.$$

Towards a hard language in $\mathsf{NP} \cap \mathsf{coNP}$. We begin by considering any language $\mathcal{L}_0$ in $\mathsf{UP} \setminus \mathsf{RP}$. Thus, all statements in $\mathcal{L}_0$ have a *unique* witness, and all statements not in $\mathcal{L}_0$ have *no* witnesses. The first task is to construct a new language whose statements have both NP and coNP certificates. With the hope of making progress in this aspect, let us define

$$\mathcal{L}_1 = \{(x, i) : \exists w, (x, w) \in R_{\mathcal{L}_0} \wedge w_i = 1\},$$

where $R_{\mathcal{L}_0}$ is the efficiently recognizable Boolean relation for $\mathcal{L}_0$. The corresponding complement language is

$$\overline{\mathcal{L}}_1 = \{(x, i) : (\exists w, (x, w) \in R_{\mathcal{L}_0} \wedge w_i = 0) \vee (\nexists w, (x, w) \in R_{\mathcal{L}_0})\}.$$

Observe that for all statements (x, i) where $x \in \mathcal{L}_0$, there is valid certificate for the membership of (x, i) in both $\mathcal{L}_1$ and $\overline{\mathcal{L}}_1$. This certificate is the *unique* witness w for x's membership in $\mathcal{L}_0$: Given w, simply check that $(x, w) \in R_{\mathcal{L}_0}$, then the ith bit of this witness is the evidence that either $(x, i) \in \mathcal{L}_1$ or $(x, i) \notin \mathcal{L}_1$. Unfortunately, for statements of the form (x, i) where $x \notin \mathcal{L}_0$, there is no clear coNP certificate (for membership in $\overline{\mathcal{L}}_1$).

Can proofs be useful? The issue above can however be resolved if we could attach an additional component to the instance which would act as a *proof* for the fact that $x \in \mathcal{L}_0$. Specifically, consider the existence of a non-interactive proof system defined by a pair of machines (P, V), consisting of an (unbounded) prover P and an efficient deterministic verifier V, for the language $\mathcal{L}_0$. We require perfect completeness and perfect soundness (for all $x \notin \mathcal{L}_0$, there is no string π such that $V(x, \pi) = 1$). This gives us a new language

$$\mathcal{L}_2 = \{(x, i, \pi) : (\exists w, (x, w) \in R_{\mathcal{L}_0} \wedge w_i = 1) \wedge V(x, \pi) = 1\}.$$

We now observe that this language is in both NP and coNP. Whenever $x \in \mathcal{L}_0$, both the NP and coNP certificate are the unique witness w such that $(x, w) \in R_{\mathcal{L}_0}$. When $x \notin \mathcal{L}_0$, there does not exist any w, such that $(x, w) \in R_{\mathcal{L}_0}$, but $V(x, \pi)$ must always output 0 due to the perfect soundness of our proof system. Thus, our coNP verifier runs this polynomial time proof verification algorithm and outputs 1 if $V(x, \pi) = 0$. In this case, any string can serve as a coNP certificate because it is never used by the coNP verifier.

What about hardness? With what we have written so far, there is no reason for $\mathcal{L}_2$ to be hard, since the proof π above could simply consist of the (unique) witness for $x \in \mathcal{L}_0$, in which case of course no hardness would exist for $\mathcal{L}_2$. Clearly, we need to ask more from our proof system to prevent the proof π from destroying hardness.

Indeed, we want to actually be able to *prove* hardness for $\mathcal{L}_2$. Namely, we would like to show that if $\mathcal{L}_2 \in \mathsf{P}$ then it must be the case that $\mathcal{L}_0 \in \mathsf{RP}$, resulting in a contradiction. Here's a natural reduction idea: Suppose D is a polynomial time Turing machine that decides $\mathcal{L}_2$. Then we might try to construct an efficient Turing machine M, which given input x, attempts to decide if $x \in \mathcal{L}_0$, in the following manner,

1. Somehow, create a valid "fake" proof π for proving $x \in \mathcal{L}_0$, even if in fact x might not actually be in $\mathcal{L}_0$.

2. Attempt to recover a witness w for x by iterating through all indices i up to some polynomial bound on the witness length, $p(|x|)$, and setting $w_i \leftarrow D(x, i, \pi)$.

3. Assuming M was able to complete Step 1, M checks if $(x, w) \in R_{\mathcal{L}_0}$ and if so, outputs 1.

Obviously, Step 1 above looks highly suspicious. If the proof system is sound, how can it allow for such "fake" proofs?

Zero knowledge to the rescue? This seeming contradiction is exactly what the notion of (Non-Interactive) Zero Knowledge (NIZK) protocols [BFM88] seem to have been created to solve. In a NIZK protocol, there is indeed a simulator that can create the kinds of valid "fake" proofs that we seek. Since we are seeking a separation in the random oracle model, we need to construct such NIZK proofs for UP, with (near)-perfect soundness[6].

In fact, it turns out that in the random oracle model, no such NIZK proofs were known prior to our work. The main technical tool that we contribute in this work is indeed the construction of such NIZK proofs for NP with near-perfect soundness in the random oracle model. We also show how to obtain them from a concrete unstructured hardness conjecture we call a δ-$\mathsf{Dense\text{-}Pseudorandom\text{-}Hash\text{-}Function}$ (δ-$\mathsf{Dense\text{-}PRHF}$). We will return to this below in Section 2.2, but for now, let us assume that we have built such a near-perfectly sound NIZK proof system in the random oracle model.

A hard language in $\mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}$. We are now ready to put it all together. Let $\Pi_{\mathsf{NIZK}} = (\mathcal{P}^{(\cdot)}, \mathcal{V}^{(\cdot)})$ be a NIZK proof system for membership in the language $\mathcal{L}_0 \in \mathsf{UP} \setminus \mathsf{RP}$ in the random oracle model such that $\mathcal{V}^{(\cdot)}$ is deterministic and Π_{NIZK} satisfies perfect completeness, near-perfect soundness, and zero-knowledge against computationally unbounded oracle Turing machines restricted to polynomially many oracle queries. For syntax purposes below, let $\mathsf{Sim} \coloneqq (\mathsf{SimProof}, \mathsf{SimRO})$ be the zero knowledge simulator for this proof system. Here, $\mathsf{SimProof}$ upon input an instance x, outputs a proof π and some state st. SimRO gets this state st from $\mathsf{SimProof}$ and simulates random oracle queries (see Def. 3.10 for the security definition). We now construct our hard language that depends on the random oracle $\mathcal{O}$:

$$\mathcal{L}^{\mathcal{O}} \coloneqq \left\{ (x, i, \pi) : (\exists w, (x, w) \in R_{\mathcal{L}_0} \text{ and } w_i = 1) \wedge (\Pi_{\mathsf{NIZK}}.\mathcal{V}^{\mathcal{O}}(x, \pi) = 1) \right\}$$

The intuition for why $\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}} \in \mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}\right] = 1$ is identical to the reasoning for why $\mathcal{L}_2 \in \mathsf{NP} \cap \mathsf{coNP}$. We now explain why $\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}} \notin \mathsf{P}^{\mathcal{O}}\right] = 1$ by following the reduction template laid out above. For the sake of contradiction assume, $\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}} \in \mathsf{P}^{\mathcal{O}}\right] > \varepsilon$ for some constant $\varepsilon > 0$. The classic Bennett-Gill paper [BG81] shows that this assumption implies the existence of some polynomial-time oracle Turing machine $D^{(\cdot)}$ such that $D^{\mathcal{O}}(x, i, \pi)$ correctly decides the membership of (x, i, π) with probability 1 over the choice of $\mathcal{O}$. Now we construct a probabilistic polynomial-time Turing machine M that, upon given an input x,

1. Use $\mathsf{SimProof}(x)$ to generate a proof π and some state st.

2. For each index $i \in [p(|x|)]$, where $p(|x|)$ is the polynomial bound on the witness length guaranteed by the definition of UP, set $w_i \leftarrow D^{\mathsf{SimRO}(\mathsf{st}, \cdot)}(x, i, \pi)$ (when D queries q on its oracle tape, M simulates the oracle responses with $\mathsf{SimRO}(\mathsf{st}, q)$).

[6] More formally, what we need is that with probability 1 over the choice of random oracle, only a finite number of statements not in $\mathcal{L}_0$ admit valid proofs.

3. Finally, if $(x, w) \in R_{\mathcal{L}_0}$ (or if a prefix $w_1 \dots w_t$ is a witness, for some $t \in [p(|x|)]$), then M outputs 1. Otherwise M outputs 0.

Observe that if $x \notin \mathcal{L}_0$, then there is no witness w such that $(x, w) \in R_{\mathcal{L}_0}$ so M will always correctly output 0. If $x \in \mathcal{L}_0$, the perfect completeness and zero-knowledge property of Π_{NIZK} guarantees that M will recover a valid witness w and output 1 with all but negligible probability. This implies that M is a RP decider for $\mathcal{L}_0$ contradicting our assumption that $\mathcal{L}_0 \in \mathsf{UP} \setminus \mathsf{RP}$.

The only thing that remains is to describe how to construct our main technical tool, which we believe will be of independent interest: a NIZK proof system with near-perfect soundness in the random oracle model.

2.2 NIZK Proofs from Unstructured Hardness

NIZK arguments for NP, satisfying computational soundness, have been known to exist in the random oracle model since the classic work of Fiat and Shamir [FS87]. However, the Fiat-Shamir paradigm inherently results in NIZK protocols that admit proofs for false statements. We instead look elsewhere for inspiration.

Our starting point is [FLS90] which constructs NIZK proofs with near-perfect soundness in the uniform common random string (URS) model from one-way-permutations (OWPs). Unfortunately, we know of no unstructured sources of hardness that yield OWPs. Our initial idea is to try to look into the protocol of [FLS90], and see what happens if we replace the OWP found there with a random oracle[7]. Let us see what happens if we do so.

The construction of [FLS90] proceeds in two steps. First, they show how to construct NIZK proofs in an intermediate model, which they call the $\mathsf{Hidden\text{-}Bits}$ models. In this model, the prover has access to a private uniformly random bit string r, and can choose which bits of r to reveal to or hide from the verifier. The verifier learns only the bits of r selected by the prover. As their second step, they show how to instantiate this hidden bit string functionality using a URS and a OWP. At a high level, the prover interprets the URS as a sequence of output values $y_1 \dots y_t$, where each y_i constitutes a commitment to the i^{th} bit r_i of the hidden bit string. Each r_i is set to be equal to $h(x_i)$ where h is a hardcore bit function of the OWP and x_i is the unique value such that $\mathsf{OWP}(x_i) = y_i$. Thus, each bit of r is uniquely determined by the OWP and the URS. The prover can open any bit r_i by simply sending over the corresponding pre-image x_i. However, by the one-way-ness of the OWP, the verifier is unable to learn any value of r_i unless it is revealed by the prover.

Now, suppose we were to replace the OWP in the above construction with a random oracle. The resulting NIZK proof now fails in several ways. First, a random function will not be *surjective* on a constant fraction of its codomain with high probability. Thus, the above protocol is no longer complete as r_i is undefined whenever y_i does not have a pre-image. Secondly, as a random function need not be *injective*, then the proof is no longer sound since whenever there exist multiple pre-images $x_i^{(1)}$ and $x_i^{(2)}$ of y_i such that $h(x_i^{(1)}) \neq h(x_i^{(2)})$, then the prover can choose to open r_i to either value. Even if the random oracle were to only be 2-to-1, this would still lead to exponentially many choices of possible hidden bit strings r, breaking soundness. Thus, we must use a different manner to determine r from the URS and the random oracle.

A new way of determining the hidden bit string. Our key idea is to determine r_i by whether or not there exists any pre-image for y_i under the random oracle. In particular, we set $r_i = 1$ if there exist no pre-images for y_i, and set $r_i = 0$ if there exists at least one pre-image for y_i. Observe

[7]Note that replacing the URS with a random oracle is trivial.

that the resulting string r may not be uniformly random as its distribution depends on the density of the image in the co-domain. However, if the random oracle is a function from n bits to n bits, then we can show that each bit of r is 1 with probability roughly equal to e^{-1}. Now, the prover can reveal that $r_i = 0$ by sending over a pre-image x_i for y_i (which the verifier can check), and can claim that $r_i = 1$ by simply stating that "there is no pre-image". Observe that the prover cannot lie about the value of r_i whenever $r_i = 1$ because he cannot invent a pre-image when no such pre-image exists. However, the prover can easily lie about the value of r_i whenever $r_i = 0$ by simply claiming that "there is no pre-image" even if one exists. Thus, a cheating prover can lie in a one-sided manner. This idea leads us to invent a new model for NIZK proofs. As the one-sided nature of this tampering resembles a Z-channel in information theory, we call it the Z-Tamperable-Hidden-Bits model.

The Z-Tamperable-Hidden-Bits Model. The Z-Tamperable-Hidden-Bits model (Definition 4.2) is identical to the Hidden-Bits model except that a cheating prover can lie about the hidden bit string r at every index where $r_i = 0$, and r can be sampled from a fixed biased distribution. In particular, the prover is given access to a bit string r where each bit of r is independently and identically sampled from a Bernoulli distribution. The (honest) prover can choose which bits of r to reveal or hide from the verifier, and the verifier learns only the bits of r selected by the prover. However, a cheating prover can first modify the hidden bit string r by changing any '0's of his choice in r to '1's. This results in a new string $\tilde{r}$. The prover then chooses which bits of the resulting string $\tilde{r}$ to reveal or hide from the verifier, and the verifier receives the corresponding bits of $\tilde{r}$.

This model gives a lot of power to the cheating prover. Nevertheless, as we will show below, we can modify the protocol and enforce certain statistical tests to limit the power of the cheating prover, and actually achieve near-perfect soundness, while preserving zero knowledge.

Instantiating the Z-Tamperable-Hidden-Bits Model. Observe that using the method detailed above, we can use a random oracle to instantiate the functionality of the Z-Tamperable-Hidden-Bits model. In fact, we only need a hash function that satisfies several simple properties, which are satisfied (with overwhelming probability) by a random function. We capture these properties by a new primitive which we call a δ-Dense-Pseudorandom-Hash-Function (Definition 4.6). Roughly speaking, this is a hash function $H_{\mathsf{PRHF}} : \{0,1\}^n \to \{0,1\}^n$ which is (1) pseudorandom, (2) has an image which is approximately δ-Dense in the co-domain, and (3) satisfies a property we call pre-image pseudorandomness, which guarantees indistinguishability between two ways of sampling (x, y) such that $H_{\mathsf{PRHF}}(x) = y$.

2.2.1 NIZK Proofs in the Z-Tamperable-Hidden-Bits Model

It now remains to show how to construct NIZK proofs in the Z-Tamperable-Hidden-Bits model. Our construction modifies the proof system of [FLS90] in that we alter key parameter sizes, sample r from a biased distribution, and add specific statistical checks to the verifier's algorithm.

Warmup: A Proof for Graph Hamiltonicity from Structured Hidden Bit Strings. As a warmup, we first build a NIZK proof for the NP-complete language Graph-Hamiltonicity in an artificial variant of the Z-Tamperable-Hidden-Bits model where the hidden bit string is chosen from a structured distribution. In particular, if the instance G is graph on n-nodes, then r is chosen uniformly from the set of all adjacency matrices of n-cycle graphs on n-nodes.[8]

[8] This is identical to the setup in [FLS90].

The idea is to have the prover prove in zero-knowledge that the graph H represented by r is a subgraph of some permutation of G. To show that H is a subgraph of G, the prover first finds a Hamiltonian cycle C_G in G and a permutation π such that $\pi(C_G) = H$. Then, the prover shows that every edge of H is contained in $\pi(G)$, by showing that all non-edges of $\pi(G)$ are not edges of H: this is done by revealing that the matrix elements corresponding to non-edges of $\pi(G)$ in the adjacency matrix r of H are '0'.

For completeness, observe that a Hamiltonian graph always contains an n-cycle subgraph (i.e. the Hamiltonian cycle). For soundness, first observe that, by definition, no non-Hamiltonian graph has an n-cycle subgraph. Additionally, although a cheating prover can cheat by changing '0's in r to '1's (which adds edges to H), this does not help the cheating prover, since if H is not a subgraph of some permutation π of G, then neither is any graph H' formed by adding edges to H. So, in fact the Z-Tamperability of the cheating prover does not help in this case. For zero knowledge, observe that the verifier only learns a permutation π and that the elements of r corresponding to $\pi(G)$ are '0's. If r is chosen uniformly from the set of adjacency matrices of all n-cycle graphs, then π is uniform over the set of all permutations. Thus, a simulator need only pick a random permutation π and set the corresponding edges of r to be '0'.

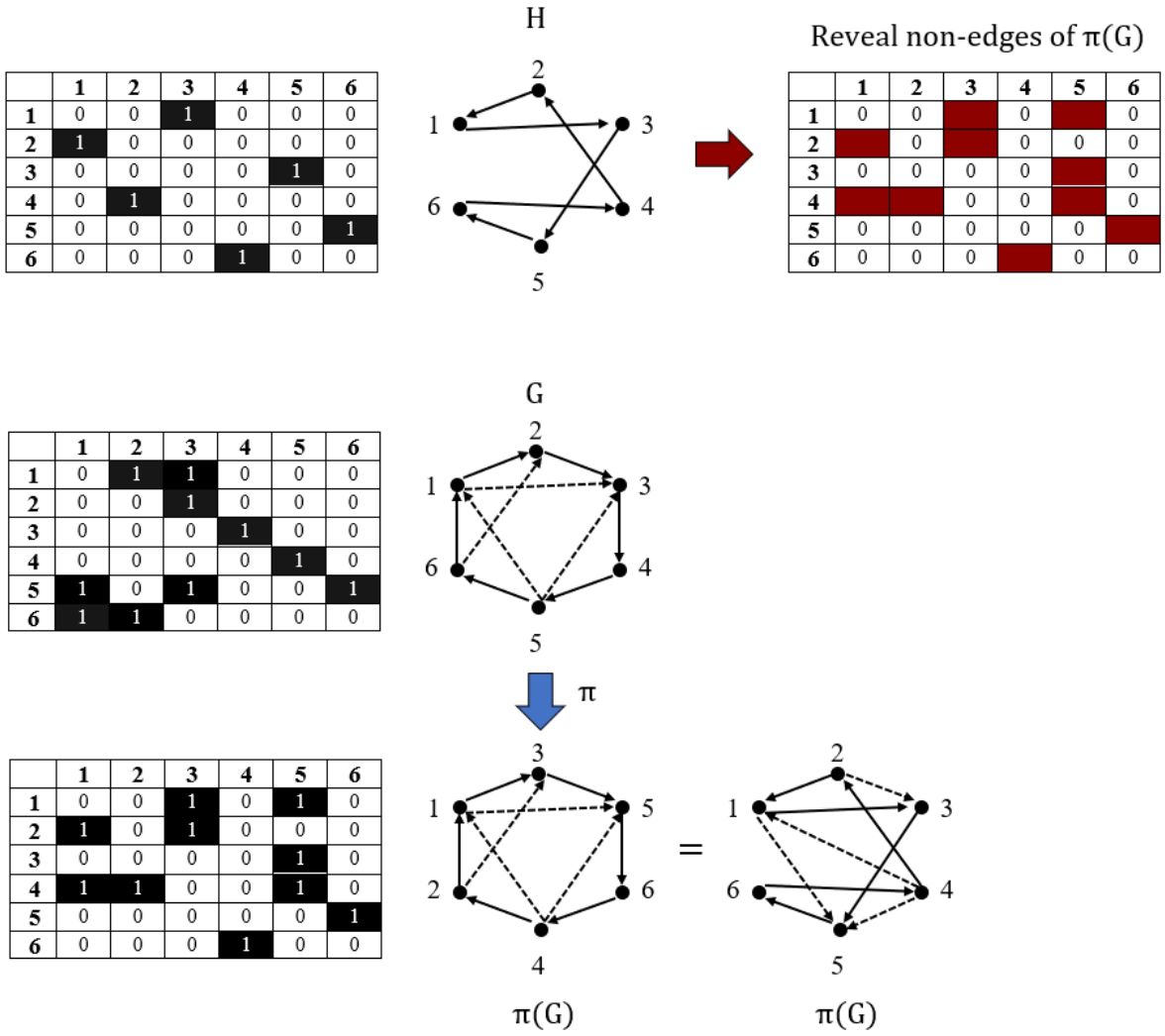

Figure 1: Proving that H is a subgraph of G

How to actually sample n-cycle graphs. Now, in the Z-Tamperable-Hidden-Bits model, instead of having r sampled from the set of all n-cycle graphs, each bit of r is sampled from i.i.d. Bernoulli distributions. Thus, if we simply interpret r as an $n \times n$ adjacency matrix, then we have only a negligible probability of getting an n-cycle. The work of [FLS90] solved this issue by introducing a clever matrix sampling technique, which allows us to generate n-cycle graphs with inverse polynomial probability. Let $c > 1$ be some constant parameter, which was presented as the fixed value $c = 2$ in the work of [FLS90]. We will modify this constant because of difficulties that arise in the Z-Tamperable-Hidden-Bits model, as we will explain shortly.

We use r to sample a sequence of $n^c \times n^c$ matrices $M^{(i)}$, where each element of $M^{(i)}$ is set to 1 with probability $\frac{1}{n^{2c-1}}$ so that the expected number of '1's in $M^{(i)}$ is n. We will say that $M^{(i)} \in$ Type-H if $M^{(i)}$ contains an $n \times n$ submatrix $S^{(i)}$ which is the adjacency matrix of an n-cycle graph $H^{(i)}$, and contains '0's everywhere else. The prover will add all matrices of Type-H to a set CycleSet which will be dealt with in a special way.

For all indices $i \notin$ CycleSet, the prover will simply discard the corresponding matrices by fully

revealing them to the verifier. The verifier will check that prover does not discard any valid matrices $M^{(i)} \in$ Type-H.

For all indices $i \in$ CycleSet, the prover will specify and reveal all rows and columns that are not part of the submatrix $S^{(i)}$. The prover will then prove that the graph $H^{(i)}$ represented by $S^{(i)}$ is a subgraph of some permutation of G using the protocol described above. The verifier will check that the rows and columns not in the specified submatrix $S^{(i)}$ contain only '0's, and will check that the subgraph protocol verifies on $S^{(i)}$.

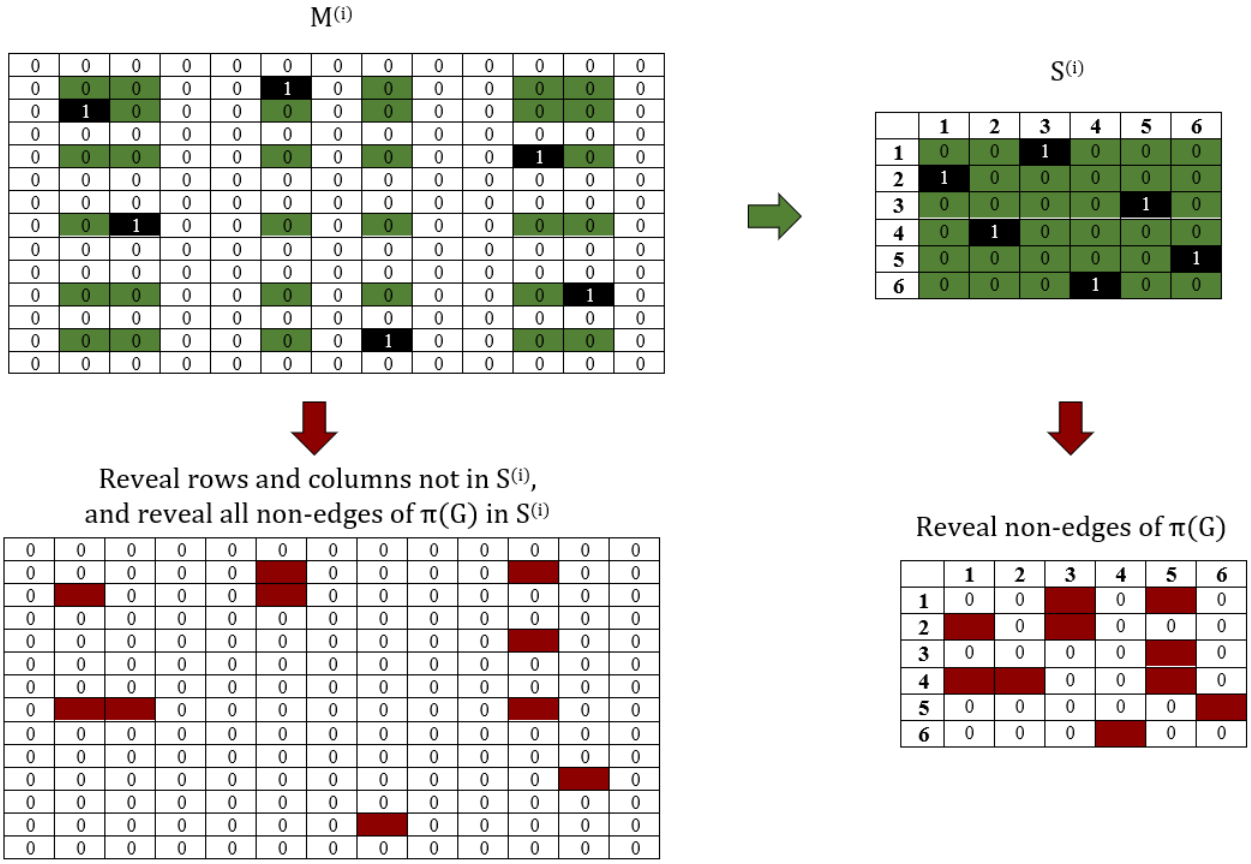

Figure 2: Sampling submatrices that contain n-cycles.

Now, for $c = 2$, the probability a matrix $M^{(i)}$ is in type Type-H is $\Omega(n^{-1.5})$. Thus, if we sample enough such matrices, we will get sufficiently many matrices of type Type-H. This is sufficient for constructing a NIZK proof in the regular Hidden-Bits model. For soundness, observe that the prover must perform the subgraph protocol on every matrix $M^{(i)} \in$ Type-H since the verifier will check that the prover does not discard any valid matrices. Therefore, as long as there is at least one matrix $M^{(i)} \in$ Type-H, then we have soundness in the Hidden-Bits model. Zero knowledge follows as the simulator can easily simulate random matrices $M^{(i)} \notin$ Type-H and can simulate the revealed indices of $M^{(i)} \in$ Type-H by picking a random submatrix $S^{(i)}$, a random permutation π and setting all rows and columns not in $S^{(i)}$ and all non-edges of $\pi(G)$ in $S^{(i)}$ to be '0'.

Unfortunately, this is insufficient for proving soundness in the Z-Tamperable-Hidden-Bits model. Recall that the one-sided tamperability means that the cheating prover can add '1's to any matrix

$M^{(i)}$, but cannot remove[9] any '1's. Thus, a cheating prover can simply invalidate all matrices $M^{(i)} \in$ Type-H by adding additional '1's to them. This allows the prover to discard any troublesome Type-H matrices.

Constraining the cheating prover. In order to prevent the cheating prover from simply invalidating all Type-H matrices, we will add a statistical check to the verifier.

As a first attempt, we try having the verifier check that CycleSet contains at least a $\Omega(n^{-1.5})$ fraction of the matrices. This way, if the prover simply invalidates all matrices of Type-H, then CycleSet will become empty, and the verifier will reject. However, observe that for every matrix in CycleSet, the prover only has to pass the subgraph check (by showing that the '1's of the matrix are contained in an $n \times n$ submatrix which defines a subgraph of some permutation of G). Thus, the cheating prover can fill the empty spots in CycleSet with any matrices not in Type-H which can pass the subgraph check. As there are enough such matrices to make up for the difference, we will need a different type of statistical check.

Now, since a cheating prover cannot pass the subgraph check with any matrix $M^{(i)} \in$ Type-H (as G is not Hamiltonian), then the prover must invalidate each of these matrices. But the only way the prover can invalidate these matrices is by adding '1's to them. Thus, this will increase the number of discarded matrices (i.e. matrices $M^{(i)}$ where $i \notin$ CycleSet) with at least $n+1$ '1's. This leads us to the idea for our statistical check: We will have the verifier count the number of discarded matrices with at least $n+1$ '1's, and reject if the number if much greater than expected.

Now, since the cheating prover cannot remove '1's from matrices, then he cannot decrease the count of discarded matrices with at least $n+1$ '1's by simply removing '1's. However, the cheating prover can instead decrease this count by adding into CycleSet the indices of all matrices $M^{(i)}$ that can pass the subgraph check and have at least $n+1$ '1's . Now, in order to pass the subgraph check, a matrix $M^{(i)}$ must have all of its '1's contained within an $n \times n$ submatrix.

Our key insight is that if we increase the dimensions of our matrices (by increasing the parameter c specified above), then the matrices become very sparse. Therefore, the vast majority of matrices will not have any '1's fall into the same row or column, i.e., the '1's of the matrix will form a permutation submatrix. But if a matrix has at least $n+1$ '1's, and these '1's form a permutation submatrix, then these '1's cannot be contained within an $n \times n$ submatrix! Therefore, such a matrix cannot pass the subgraph check (even if the prover adds additional '1's to the matrix). To complete the proof of soundness, it suffices to show that the expected number of matrices $M^{(i)}$ with at least $n+1$ '1's whose '1's do *not* form a permutation submatrix, and thus could pass the subgraph check, is much smaller than the expected number of matrices $M^{(i)} \in$ Type-H. Thus, with overwhelming probability over the choice of the hidden bit string, a cheating prover must either inflate the count of discarded matrices $M^{(i)}$ with at least $n+1$ '1's or get caught by the subgraph check. In either case, the verifier will reject.

These statistical checks therefore preclude cheating by the prover that would enable the prover to avoid dealing with the matrices in Type-H. As we observed above, if the cheating prover is forced to reveal the permutation π and all the non-edges of $\pi(G)$ in the cycle-subgraph in even one matrix in Type-H, we have soundness. Then, with a standard application of the Borel–Cantelli lemma, we obtain near-perfect soundness, together with zero knowledge.

[9] We will set matrix elements to '1' iff the corresponding bits of r are all '1's. Since the cheating prover can only change '0's in r to '1's, this means that the cheating prover can also only change '0's in $M^{(i)}$ to '1's.

3 Preliminaries

Notation.

- We say that a function $f(n)$ is negligible in n if $f(n) = n^{-\omega(1)}$, and we denote it by $f(n) = \mathsf{negl}(n)$.
- We say that a function $g(n)$ is polynomial in n if $g(n) = p(n)$ for some fixed polynomial p, and we denote it by $g(n) = \mathsf{poly}(n)$.
- For $t \in \mathbb{N}$, we use $[t]$ to denote the set $\{1, \ldots, t\}$.
- If R is a random variable, then $r \leftarrow R$ denotes sampling r from R. If T is a set, then $i \leftarrow T$ denotes sampling i uniformly at random from T.

Definition 3.1 (Polynomial-Query-Bounded Oracle Turing Machine)**.** *We say that an oracle Turing machine $M^{(\cdot)}$ is polynomial-query-bounded if there exists a polynomial $p(\cdot) : \mathbb{N} \to \mathbb{N}$ such that for any input $x \in \{0,1\}^*$ and for any oracle $\mathcal{O}$, the execution of $M^{\mathcal{O}}(x)$ makes at most $p(|x|)$ many queries to $\mathcal{O}$.*

Lemma 3.2 (Chebyshev's Inequality)**.** *Let X be a random variables with finite expected value μ and finite non-zero variance σ^2. Then, for any $k > 0$,*

$$\Pr\left[|X - \mu| \geq k\sigma\right] \leq \frac{1}{k^2}$$

Lemma 3.3 (Chernoff Bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables taking values in $\{0,1\}$, and let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. Then a two-sided Chernoff bound for $0 \leq \delta \leq 1$ is*

$$\Pr\left[|X - \mu| \geq \delta\mu\right] \leq 2 \cdot \exp\left(-\frac{\delta^2\mu}{3}\right)$$

And a one sided Chernoff bound for $0 \leq \delta \leq 1$ is

$$\Pr[X \geq (1+\delta)\mu] \leq \exp\left(\frac{-\delta^2\mu}{3}\right)$$

Definition 3.4 (Statistical Distance)**.** *Let D_1 and D_2 be two random variables over some probability space with support in X. The statistical distance between D_1 and D_2 is*

$$\Delta(D_1, D_2) = \frac{1}{2} \sum_{x \in X} |\Pr[D_1 = x] - \Pr[D_2 = x]|$$

Lemma 3.5. *If $A = (A_1, \ldots, A_n)$ and $B = (B_1, \ldots, B_n)$ are sequences of independent random variables such that*

$$\forall i \in [n],\ \Delta(A_i, B_i) \leq \varepsilon$$

then

$$\Delta((A_1, \ldots, A_n), (B_1, \ldots, B_n)) \leq n\epsilon$$

The lemma above follows from [DH12]. Refer to Lemma A.1 in Appendix A.

Lemma 3.6 (First Borel-Cantelli Lemma (BC1))**.** *Suppose that $\{E_n\}_{n \in \mathbb{N}}$ is a sequence of events in a probability space. If*

$$\sum_{n \in \mathbb{N}} \Pr[E_n] < \infty$$

then the probability that infinitely many of them occur is 0. In other words, with probability 1, only a finite number of the events occur.

3.1 Random Oracles

A random oracle is a function $\mathcal{O} : \{0,1\}^* \to \{0,1\}$. When using random oracles, we use $\Pr_{\mathcal{O}}$ to denote the probability when the random oracle $\mathcal{O}$ is chosen uniformly from the set of all functions from $\{0,1\}^*$ to $\{0,1\}$.

Lemma 3.7. *For any function $m : \mathbb{N} \to \mathbb{N}$, a random oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}$ can be equivalently interpreted as a family of countably-infinite independent random functions $\{\mathcal{O}_n : \{0,1\}^n \to \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$.*

Proof. For $n \in \mathbb{N}$, we define

$$\mathcal{O}_n(x) = F_n(x,1)\| \cdots \|F_n(x,m).$$

where each $F_n : \{0,1\}^{n+\lceil \log m(n)\rceil} \to \{0,1\}$ is a function such that $F_n(x,i) = \mathcal{O}(1^n\|0\|x\|i)$, where $|x| = n$ and $i \in [m]$ is given by its binary representation. Note that over the probability of $\mathcal{O}$, for all $k \neq \ell \in \mathbb{N}$, $\mathcal{O}_k$ and $\mathcal{O}_\ell$ are independent random functions because they always query the oracle $\mathcal{O}$ at disjoint points. □

3.2 Non-Interactive Zero-Knowledge Proofs

We define the standard notion of a non-interactive zero-knowledge (NIZK) proof with a computationally unbounded prover in the uniform reference string (URS) model and in the random oracle (RO) model.

Remark 3.8. Our definitions of soundness are *adaptive*, meaning that a cheating prover's choice of $x \notin \mathcal{L}$ can depend on the urs and/or the random oracle $\mathcal{O}$.

Notation. If for sufficiently large n, a NIZK proof satisfies soundness with respect to some explicit negligible function $\varepsilon_s(n)$, we say that it achieves $\varepsilon_s(n)$-soundness.

Definition 3.9 ((Unbounded-Prover) NIZK Proof in the URS Model)**.** *An (unbounded-prover) non-interactive zero-knowledge* (NIZK) *proof system in the uniform reference string* (URS) *model for a language $\mathcal{L}$ is a tuple of algorithms $\Pi = (\mathsf{Gen}, P, V)$:*

- $\mathsf{Gen}(1^n)$ *is a* PPT *algorithm that given an input length n outputs a uniformly random bit-string* urs *of length $p_{\mathsf{urs}}(n)$, for some (fixed) polynomial p_{urs}.*
- $P(\mathsf{urs}, x)$ *is an (unbounded, randomized) honest prover algorithm that takes as input a reference string* $\mathsf{urs} \in \{0,1\}^{p_{\mathsf{urs}}(|x|)}$ *and $x \in \mathcal{L}$ and outputs a proof π.*
- $V(\mathsf{urs}, x, \pi)$ *is a polynomial-time deterministic verifier that takes as input a reference string* $\mathsf{urs} \in \{0,1\}^{p_{\mathsf{urs}}(|x|)}$*, x, and a proof π, and outputs either 1 (accept) or 0 (reject).*

We require Π to satisfy the following completeness, soundness, and zero knowledge requirements.

- ***Perfect Completeness:*** *For all $x \in \mathcal{L}$,*

$$\Pr\left[V(\mathsf{urs}, x, \pi) = 1 : \mathsf{urs} \leftarrow \mathsf{Gen}(1^{|x|}), \pi \leftarrow P(\mathsf{urs}, x)\right] = 1$$

  where the probability is over the choice of urs *and the randomness of P.*

- ***(Adaptive) Statistical Soundness:*** *There exists a negligible function $\varepsilon_s(\cdot)$ such that for all $n \in \mathbb{N}$,*

$$\Pr_{\mathsf{urs}\leftarrow\mathsf{Gen}(1^n)}[\exists(x,\pi) \text{ such that } x \notin \mathcal{L}, |x| = n, \text{ and } V(\mathsf{urs}, x, \pi) = 1] \leq \varepsilon_s(n)$$

- ***Computational Zero Knowledge:*** *There exists a* PPT *simulator* Sim *and a negligible function* $\varepsilon_{zk}(\cdot)$ *such that for all* $x \in \mathcal{L}$ *and all non-uniform polynomial-sized adversaries* $\mathcal{A}$*,*

$$\Big| \Pr\left[\mathcal{A}(\mathsf{urs}, x, \pi) = 1 : \mathsf{urs} \leftarrow \mathsf{Gen}(1^{|x|}), \pi \leftarrow P(\mathsf{urs}, x)\right] - \Pr\left[\mathcal{A}(\mathsf{urs}^*, x, \pi^*) = 1 : (\mathsf{urs}^*, \pi^*) \leftarrow \mathsf{Sim}(x)\right] \Big| \leq \varepsilon_{zk}(|x|)$$

where the probability is over the choice of urs *and the randomness of* P*,* Sim*, and* $\mathcal{A}$*.*

When defining NIZK proofs in the random oracle (RO) model, we allow all algorithms access to the random oracle. Additionally, the zero knowledge simulator is allowed to program the random oracle. We will actually prove zero knowledge against all polynomial-query-bounded oracle Turing machines (Def. 3.1), rather than against computationally bounded adversaries.

Definition 3.10 ((Unbounded-Prover) NIZK Proof in the RO Model)**.** *An (unbounded-prover) non-interactive zero-knowledge* (NIZK) *proof system in the random oracle* (RO) *model for a language* $\mathcal{L}$ *is a tuple of algorithms* $\Pi = (P^{(\cdot)}, V^{(\cdot)})$ *with access to a random oracle* $\mathcal{O}$*.*

- $P^{\mathcal{O}}(x)$ *is an (unbounded, randomized) honest prover algorithm that is given oracle access to a random oracle* $\mathcal{O}$*, takes as input* $x \in \mathcal{L}$*, and outputs a proof* π*.*
- $V^{\mathcal{O}}(x, \pi)$ *is a polynomial-time deterministic verifier that is given oracle access to a random oracle* $\mathcal{O}$*, takes as input* x *and a proof* π*, and outputs either 1 (accept) or 0 (reject).*

We require Π *to satisfy the following completeness, soundness, and zero knowledge requirements.*

- ***Perfect Completeness:*** *For all* $x \in \mathcal{L}$*,*

$$\Pr\left[V^{\mathcal{O}}(x, \pi) = 1 : \pi \leftarrow P^{\mathcal{O}}(x)\right] = 1$$

where the probability is over the choice of $\mathcal{O}$ *and the randomness of* P*.*

- ***(Adaptive) Statistical Soundness:*** *There exists a negligible function* $\varepsilon_s(\cdot)$ *such that for all* $n \in \mathbb{N}$*,*

$$\Pr_{\mathcal{O}}[\exists(x, \pi) \text{ such that } x \notin \mathcal{L}, |x| = n, \text{ and } V^{\mathcal{O}}(x, \pi) = 1] \leq \varepsilon_s(n)$$

- ***Zero Knowledge:*** *There exists a stateful* PPT *simulator* $\mathsf{Sim} = (\mathsf{SimProof}, \mathsf{SimRO})$ *and a negligible function* $\varepsilon_{zk}(\cdot)$ *such that for all* $x \in \mathcal{L}$ *and all polynomial-query-bounded oracle Turing machines* $\mathcal{A}^{(\cdot)}$ *,*

$$\Big| \Pr\left[\mathcal{A}^{\mathcal{O}}(x, \pi) = 1 : \pi \leftarrow P^{\mathcal{O}}(x)\right] - \Pr\left[\mathcal{A}^{\mathsf{SimRO}(\mathsf{st}, \cdot)}(x, \pi^*) = 1 : (\mathsf{st}, \pi^*) \leftarrow \mathsf{SimProof}(x)\right] \Big| \leq \varepsilon_{zk}(|x|)$$

where the probability is over the choice of $\mathcal{O}$ *and the randomness of* P*,* Sim*, and* $\mathcal{A}$*.*

4 Definitions

We first define a notion of Efficiently-Approximable constants.

Definition 4.1 (Efficiently-Approximable Constant)**.** *We say that a real number $\gamma \in (0,1)$ is* Efficiently-Approximable *if there exists a polynomial-time deterministic algorithm* Approx_γ *such that for all $n \in \mathbb{N}$,*

- $\mathsf{Approx}_\gamma(1^n)$ *outputs a rational number $\hat{\gamma} = a/b$, where a, b are integers in binary representation.*
- *For all $n \in \mathbb{N}$, $\left|\gamma - \mathsf{Approx}_\gamma(1^n)\right| \leq \frac{1}{2^n}$.*

4.1 Z-Tamperable-Hidden-Bits (ZHB) Model

We now define NIZK proofs in the Z-Tamperable-Hidden-Bits model. This is the same as the Hidden-Bits model of [FLS90] except that we (1) allow a dishonest prover to tamper with the hidden bits by flipping any '0's of their choice to '1's, and (2) allow the hidden bits to be sampled from a fixed biased distribution. As the one-sided nature of this tampering resembles a Z-channel in information theory, we call it the Z-Tamperable-Hidden-Bits model.

Definition 4.2 (NIZK Proof in the Z-Tamperable-Hidden-Bits Model)**.** *An (unbounded-prover) non-interactive zero-knowledge* (NIZK) *proof system in the* Z-Tamperable-Hidden-Bits (ZHB) *model for a language $\mathcal{L}$ is a family of algorithms $\{\Pi_\gamma = (\mathsf{Gen}_\gamma, P_\gamma, V_\gamma)\}$ parameterized by* Efficiently-Approximable *constants $\gamma \in (0,1)$, where*

- $\mathsf{Gen}_\gamma(1^n)$ *is a* PPT *hidden bit string generator that given an input length n outputs a hidden bit string $r \in \{0,1\}^{p_{\gamma,\mathsf{hbs}}(n)}$ for some (fixed) polynomial $p_{\gamma,\mathsf{hbs}}$ where each bit r_i of r is sampled independently from a Bernoulli distribution with $\Pr[r_i = 1] = \hat{\gamma}$ where $\hat{\gamma} = \mathsf{Approx}_\gamma(1^n)$.*
- *$P_\gamma(r, x)$ is an (unbounded, randomized) honest prover algorithm that takes as input a hidden bit string $r \in \{0,1\}^{p_{\gamma,\mathsf{hbs}}(|x|)}$ and $x \in L$, and outputs a set of indices $\mathcal{I} \subseteq [p_{\gamma,\mathsf{hbs}}(|x|)]$ and a proof π.*
- *$V_\gamma(x, \mathcal{I}, \{r_i\}_{i\in\mathcal{I}}, \pi)$ is a polynomial-time deterministic verifier that takes as input x, an index set $\mathcal{I} \subseteq [p_{\gamma,\mathsf{hbs}}(|x|)]$, the bits $\{r_i\}_{i\in\mathcal{I}}$ of the hidden bit string r corresponding to $\mathcal{I}$, and a proof π, and outputs either 1 (accept) or 0 (reject).*

We require Π to satisfy the following completeness, soundness, and zero knowledge requirements.

- ***Perfect Completeness:*** *For all* Efficiently-Approximable *constants $\gamma \in (0,1)$ and all $x \in \mathcal{L}$,*

$$\Pr\left[V_\gamma(x, \mathcal{I}, \{r_i\}_{i\in\mathcal{I}}, \pi) = 1 : r \leftarrow \mathsf{Gen}_\gamma(1^{|x|}), (\mathcal{I}, \pi) \leftarrow P_\gamma(r, x)\right] = 1$$

  where the probability is over the randomness of Gen_γ and P_γ.

- ***(Adaptive) Statistical Soundness:*** *For all* Efficiently-Approximable *constants $\gamma \in (0,1)$, there exists a negligible function $\varepsilon_{\gamma,s}(\cdot)$ such that for all $n \in \mathbb{N}$,*

$$\Pr_{r\leftarrow\mathsf{Gen}_\gamma(1^n)}\Big[\exists(x, \mathcal{I}, f, \pi) \text{ such that } x \notin L, |x| = n, f \in \{0,1\}^{|r|} \\ \text{and } \ V_\gamma(x, \mathcal{I}, \{\tilde{r}_i = \mathsf{ZeroFlip}(r_i, f_i)\}_{i\in\mathcal{I}}, \pi) = 1\Big] \leq \varepsilon_{\gamma,s}(n)$$

where

$$\mathsf{ZeroFlip}(r_i, f_i) = \begin{cases} r_i \oplus f_i & \text{if } r_i = 0 \\ r_i & \text{if } r_i = 1 \end{cases}$$

- ***Statistical Zero Knowledge:*** *For all* Efficiently-Approximable *constants* $\gamma \in (0,1)$*, there exists a* PPT *simulator* Sim_γ *and a negligible function* $\varepsilon_{\gamma,zk}(\cdot)$ *such that for all* $x \in \mathcal{L}$ *and all adversaries* $\mathcal{A}$*,*

$$\Big| \Pr\left[\mathcal{A}(x, \mathcal{I}, \{r_i\}_{i\in\mathcal{I}}, \pi) = 1 : r \leftarrow \mathsf{Gen}_\gamma(1^{|x|}), (\mathcal{I}, \pi) \leftarrow P_\gamma(r, x)\right] - \Pr\left[\mathcal{A}(x, \mathcal{I}^*, \{r_i^*\}_{i\in\mathcal{I}^*}, \pi^*) = 1 : (\mathcal{I}^*, \{r_i^*\}_{i\in\mathcal{I}^*}, \pi^*) \leftarrow \mathsf{Sim}_\gamma(x)\right] \Big| \leq \varepsilon_{\gamma,zk}(|x|)$$

where the probability is over the randomness of $\mathsf{Gen}_\gamma, P_\gamma, \mathsf{Sim}_\gamma$*, and* $\mathcal{A}$*.*

Notation. If $\{\Pi_\gamma = (\mathsf{Gen}_\gamma, P_\gamma, V_\gamma)\}$ is a NIZK proof system in the ZHB model, we may say that Π_γ for some fixed Efficiently-Approximable constant $\gamma \in (0,1)$ satisfies perfect completeness. By this, we mean that the perfect completeness property holds with respect to this specific γ (as opposed to for all Efficiently-Approximable constants $\gamma \in (0,1)$). Similarly, we may say that Π_γ satisfies soundness or zero knowledge, meaning that these properties hold with respect to this specific γ.

We also define two variants of the properties above.

Definition 4.3 (Perfect Zero Knowledge)**.** *We say that a* NIZK *proof system* $\{\Pi_\gamma = (\mathsf{Gen}_\gamma, P_\gamma, V_\gamma)\}$ *in the* Z-Tamperable-Hidden-Bits *model has* perfect zero knowledge *if for all* Efficiently-Approximable *constants* $\gamma \in (0,1)$*, there exists a* PPT *simulator* Sim_γ *such that for all* $x \in \mathcal{L}$*, the following two distribution ensembles are identically distributed:*

$$\{(x, \mathcal{I}, \{r_i\}_{i\in\mathcal{I}}, \pi) : r \leftarrow \mathsf{Gen}_\gamma(1^{|x|}), (\mathcal{I}, \pi) \leftarrow P_\gamma(r, x)\}$$

and

$$\{(x, \mathcal{I}^*, \{r_i^*\}_{i\in\mathcal{I}^*}, \pi^*) : (\mathcal{I}^*, \{r_i^*\}_{i\in\mathcal{I}^*}, \pi^*) \leftarrow \mathsf{Sim}_\gamma(x)\}$$

Definition 4.4 (Statistical Completeness)**.** *We say that a* NIZK *proof system* $\{\Pi_\gamma = (\mathsf{Gen}_\gamma, P_\gamma, V_\gamma)\}$ *in the* Z-Tamperable-Hidden-Bits *model has* statistical completeness *if for all* Efficiently-Approximable *constants* $\gamma \in (0,1)$*, there exists a negligible function* $\varepsilon_{\gamma,c}(\cdot)$ *such that for all* $x \in \mathcal{L}$*,*

$$\Pr\left[V_\gamma(x, \mathcal{I}, \{r_i\}_{i\in\mathcal{I}}, \pi) = 1 : r \leftarrow \mathsf{Gen}_\gamma(1^{|x|}), (\mathcal{I}, \pi) \leftarrow P_\gamma(r, x)\right] \geq 1 - \varepsilon_{\gamma,c}(n)$$

Remark 4.5. An *unbounded-prover* NIZK proof system for an NP language $\mathcal{L}$ in the Z-Tamperable-Hidden-Bits model which satisfies *statistical completeness* and *perfect zero knowledge* implies a NIZK proof system for $\mathcal{L}$ in the Z-Tamperable-Hidden-Bits model which satisfies *perfect completeness* and *statistical zero knowledge*.

We modify the prover of the former so that before outputting a proof π and index set $\mathcal{I}$, the prover first checks whether the deterministic verifier would accept. If the verifier would reject, the (unbounded) prover instead computes and outputs a witness w for its input x. We modify the verifier so that it also accepts if it receives a witness w for x. Observe that we now get perfect completeness as the verifier always accepts an honest prover's proof. Soundness is unchanged since if $x \notin \mathcal{L}$, then there is no witness w for x. For zero knowledge, observe that whenever the prover does not output a witness w, the same zero knowledge simulator perfectly simulates the prover. As the prover only outputs a witness w with negligible probability, the simulator now achieves statistical zero knowledge.

4.2 δ-Dense-PRHFs

We define a variant of a hash function, which we call a δ-Dense-Pseudorandom-Hash-Function (δ-Dense-PRHF). This is a hash function which has two pseudorandomness properties and whose image is roughly δ-dense in the co-domain [10].

Definition 4.6 (δ-Dense-PRHF)**.** *Let $\delta \in (0,1)$. A deterministic polynomial-time algorithm H_{PRHF} is a δ-Dense-PRHF with stretch function $\ell : \mathbb{N} \to \mathbb{N}$ if it satisfies the following properties:*

- ***Pseudorandom:*** *There exists a negligible function $\varepsilon_{\mathsf{PR}}(\cdot)$ such that for all non-uniform polynomial-sized adversaries $\mathcal{A}$ and all $n \in \mathbb{N}$,*

$$\left| \Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(1^n, H_{\mathsf{PRHF}}(x)) = 1] - \Pr_{y \leftarrow \{0,1\}^{\ell(n)}}[\mathcal{A}(1^n, y) = 1] \right| \leq \varepsilon_{\mathsf{PR}}(n)$$

- ***Pre-Image Pseudorandom:*** *There exists a negligible function $\varepsilon_{\mathsf{PIPR}}(\cdot)$ such that for all non-uniform polynomial-sized adversaries $\mathcal{A}$ and all $n \in \mathbb{N}$,*

$$\left| \Pr_{(x^*,y^*) \leftarrow \mathcal{D}_0(1^n)}[\mathcal{A}(x^*, y^*) = 1] - \Pr_{(x^*,y^*) \leftarrow \mathcal{D}_1(1^n)}[\mathcal{A}(x^*, y^*) = 1] \right| \leq \varepsilon_{\mathsf{PIPR}}(n)$$

where we define

$\mathcal{D}_0(1^n)$*:*

1. $x^* \leftarrow \{0,1\}^n$
2. $y^* = H_{\mathsf{PRHF}}(x^*)$
3. *Output* (x^*, y^*)

$\mathcal{D}_1(1^n)$*:*

1. $y^* \leftarrow \mathsf{Image}_n$ *where* $\mathsf{Image}_n = \{y \in \{0,1\}^{\ell(n)} : \exists x \in \{0,1\}^n,\ y = H_{\mathsf{PRHF}}(x)\}$
2. $x^* \leftarrow \mathsf{Pre\text{-}Image}_n(y^*)$ *where* $\mathsf{Pre\text{-}Image}_n(y^*) = \{x \in \{0,1\}^n : y^* = H_{\mathsf{PRHF}}(x)\}$
3. *Output* (x^*, y^*)

- δ-Dense*: There exists a negligible function $\varepsilon_{\mathsf{dense}}(\cdot)$ such that for all $n \in \mathbb{N}$,*

$$\Pr_{y \leftarrow \{0,1\}^{\ell(n)}}[\exists x \in \{0,1\}^n \text{ s.t. } H_{\mathsf{PRHF}}(x) = y] \in [\delta - \varepsilon_{\mathsf{dense}}(n), \delta + \varepsilon_{\mathsf{dense}}(n)]$$

Remark 4.7. A random oracle, when interpreted as a random function from n bits to n bits, satisfies all of the properties of a $\left(1 - \frac{1}{e}\right)$-Dense-PRHF in the random oracle model. See Appendix B for details.

Remark 4.8 ($\frac{1}{2}$-Dense-PRHF from structured assumptions)**.** We can construct a $\frac{1}{2}$-Dense-PRHF from any OWP as follows. Define $H_{\mathsf{PRHF}}(x) = P(x)||h(x)$ where P is a OWP and h is a hardcore predicate for P. It is easy to verify that this construction satisfies all the necessary properties.

[10] Note that our hash function does is not compressing. In fact any function mapping n to $n + c$ bits for some constant $c \geq 0$ satisfying the properties below suffice.

5 NIZK Proofs for NP in the Z-Tamperable-Hidden-Bits Model

We now show how to build NIZK proofs in our new Z-Tamperable-Hidden-Bits (ZHB) model. We prove the following:

Theorem 5.1. *There exists a* NIZK *proof system for* NP *in the* Z-Tamperable-Hidden-Bits *model.*

To prove this, we build a NIZK proof system in the Z-Tamperable-Hidden-Bits model for the language

$$\mathsf{Graph\text{-}Hamiltonicity} = \{\text{graphs } G \mid G \text{ contains a Hamiltonian cycle}\}$$

Since Graph-Hamiltonicity is NP-complete, then this implies a NIZK proof system in the Z-Tamperable-Hidden-Bits model for every language in NP.

Additional Definitions.

- **Matrix Classifications:** Throughout this section, we will use the following matrix classifications for matrices $M \in \{0,1\}^{n^4 \times n^4}$:

Type	Description
Type-H ("H" for "Hamiltonian")	$M \in$ Type-H iff M contains a submatrix $S \in \{0,1\}^{n \times n}$ that is the adjacency matrix of some Hamiltonian cycle H on n nodes, and every element of M that is not in S is '0'.
Type-M1 ("M1" for "many '1's")	$M \in$ Type-M1 iff M contains $\geq n+1$ '1's
Type-PB1 ("PB1" for "permutation with bounded '1's")	$M \in$ Type-PB1 iff M contains a permutation submatrix $S \in \{0,1\}^{(n+k)\times(n+k)}$ for some $1 \leq k \leq n$, and every element of M that is not in S is '0'.

Table 1: Matrix Classifications

Note that there are some matrices $M \in \{0,1\}^{n^4 \times n^4}$ which do not fall into any of these three categories and that Type-PB1 $\subset$ Type-M1.

- MatGen**:** We define a PPT matrix generation algorithm MatGen:

MatGen(1^n):

1. Output a matrix $M \in \{0,1\}^{n^4 \times n^4}$ where for $j,k \in [n^4]$, we independently sample each element $M_{j,k}$ of M by setting

$$M_{j,k} = \begin{cases} 1 & \text{with probablility } \frac{1}{n^7} \\ 0 & \text{with probability } 1 - \frac{1}{n^7} \end{cases}$$

5.1 Construction

We build a NIZK proof system ZHB-NIZK $= \{(\mathsf{Gen}_\gamma, P_\gamma, V_\gamma)\}$ (parameterized by Efficiently-Approximable constants $\gamma \in (0,1)$) for Graph-Hamiltonicity in the Z-Tamperable-Hidden-Bits model which satisfies *statistical completeness* and *perfect zero knowledge*. However, as shown in Remark 4.5, this implies a scheme which satisfies *perfect completeness* and *statistical zero knowledge*. Please refer to the technical overview (Section 2) for a high level overview of our construction.

Remark 5.2. This is the same construction as the one used in [FLS90] for building NIZK proofs for NP in the Hidden-Bits model except that we

- have increased the dimensions of our matrices,
- sample the hidden bit string r according to our new parameter γ,
- adjust accordingly the method for sampling matrices from r to ensure that the expected number of '1's in a sampled matrix is still n,
- and have added a statistical check (Step 6) to the verifier's algorithm.

Remark 5.3. For all Efficiently-Approximable constants $\gamma \in (0,1)$, if P_γ is additionally given a witness for G (i.e. a Hamiltonian cycle C_G of G) as input, then P_γ can be made PPT.

$\mathsf{Gen}_\gamma(1^{|G|})$:

1. Let n be the number of nodes in a graph of size $|G|$.
2. $\hat{\gamma} \leftarrow \mathsf{Approx}_\gamma(1^n)$.
3. For $i \in [n^{14} \log_{1/\hat{\gamma}}(n^7)]$, sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = \hat{\gamma}$ and $\Pr[r_i = 0] = 1-\hat{\gamma}$.
4. Output $r = r_1 \dots r_{n^{14} \log_{1/\hat{\gamma}}(n^7)}$

Notation. For r such that $|r| = n^{14} \log_{1/\hat{\gamma}}(n^7)$, we will interpret r as n^6 blocks, each containing $n^8 \cdot \log_{1/\hat{\gamma}}(n^7)$ bits. Every block can be interpreted as $n^4 \times n^4$ sub-blocks, each containing $\log_{1/\hat{\gamma}}(n^7)$ bits. Let

- $r[i]$ denote the i^{th} block (this will represent a matrix $M^{(i)}$)
- and $r[i][j,k]$ denote the $(j \times k)^{th}$ sub-block of $r[i]$ (this will represent an element $M^{(i)}_{j,k}$).

$P_\gamma(r, G)$:

1. **Setup:** Set $\mathcal{I} = \emptyset$. Set $\mathsf{CycleSet} = \emptyset$. Let n be the number of nodes in G. Let $\hat{\gamma} = \mathsf{Approx}_\gamma(1^n)$.
2. **Compute Witness:** Compute a Hamiltonian cycle C_G of G.
3. For $i \in [n^6]$:
   (a) **Compute matrix $M^{(i)}$ from $r[i]$:** Determine a matrix $M^{(i)} \in \{0,1\}^{n^4 \times n^4}$ where for $j,k \in [n^4]$, we set
   $$M^{(i)}_{j,k} = \begin{cases} 1 & \text{if } r[i][j,k] = 1^{\log_{1/\hat{\gamma}}(n^7)} \\ 0 & \text{else} \end{cases}$$
   (Observe that if r is sampled from $\mathsf{Gen}_\gamma(1^{|G|})$, then $\Pr[M^{(i)}_{j,k} = 1] = \frac{1}{n^7}$. This means that over the probability of r, $M^{(i)} \leftarrow \mathsf{MatGen}(1^n)$.)
   (b) **Check whether $M^{(i)} \in$ Type-H:** If $M^{(i)}$ does not contain exactly n '1's or there exists a row or column with two or more '1's, then $M^{(i)} \notin$ Type-H. Otherwise, $M^{(i)}$

has a permutation submatrix $S^{(i)} \in \{0,1\}^{n \times n}$. In this case, interpret $S^{(i)}$ as the adjacency matrix of some graph $H^{(i)}$ on n nodes. If $H^{(i)}$ is a Hamiltonian cycle, then $M^{(i)} \in \mathsf{Type\text{-}H}$. Otherwise, $M^{(i)} \notin \mathsf{Type\text{-}H}$.

(c) **If $M^{(i)} \notin \mathsf{Type\text{-}H}$:**

i. **Reveal $M^{(i)}$:** Add the indices corresponding to $r[i]$ to $\mathcal{I}$.

(d) **If $M^{(i)} \in \mathsf{Type\text{-}H}$:**

i. Let $S^{(i)}$ and $H^{(i)}$ be the submatrix and Hamiltonian cycle respectively that are determined from $M^{(i)}$ in Step 3b

ii. **Add i to $\mathsf{CycleSet}$.**

iii. **Reveal that all rows and columns not in $S^{(i)}$ contain only '0's:** Let $\mathsf{SRows}^{(i)}$ and $\mathsf{SCols}^{(i)}$ be the rows and columns respectively of $M^{(i)}$ that correspond to submatrix $S^{(i)}$. "Reveal" all values not in $\mathsf{SRows}^{(i)}$ and $\mathsf{SCols}^{(i)}$ by adding the indices corresponding to $\{r[i][j,k] \mid (j,k) \notin (\mathsf{SRows}^{(i)} \times \mathsf{SCols}^{(i)})\}$ to $\mathcal{I}$.

iv. **Prove that $H^{(i)}$ is a subgraph of $\pi^{(i)}(G)$ for some permutation $\pi^{(i)}$ by showing that all non-edges of $\pi^{(i)}(G)$ are not edges of $H^{(i)}$ (by revealing that the elements in $S^{(i)}$ corresponding to non-edges of $\pi^{(i)}(G)$ are '0'):** Compute a permutation $\pi^{(i)}$ such that $\pi^{(i)}(C_G) = H^{(i)}$. "Reveal" all values in the set

$$\{S^{(i)}_{t,s} \mid (t,s) \text{ is not an edge of } \pi^{(i)}(G)\}$$

by adding the indices of $r[i]$ corresponding to this set to $\mathcal{I}$.

4. Output $(\mathcal{I}, \pi = (\mathsf{CycleSet}, \{\pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}\}_{i \in \mathsf{CycleSet}}))$.

$V_\gamma(G, \mathcal{I}, \{\tilde{r}\}_{i \in \mathcal{I}}, \pi)$:

1. Parse $\pi = (\mathsf{CycleSet}, \{\pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}\}_{i \in \mathsf{CycleSet}})$.

2. **Setup:** Let n be the number of nodes in G. Let $\hat{\gamma} = \mathsf{Approx}_\gamma(1^n)$.

3. For $i \in [n^6]$:

(a) **Compute matrix $\tilde{M}^{(i)}$ from $\tilde{r}[i]$:** Determine a matrix $\tilde{M}^{(i)} \in \{0,1,\perp\}^{n^4 \times n^4}$ where for $j,k \in [n^4]$, we set

$$\tilde{M}^{(i)}_{j,k} = \begin{cases} \perp & \text{if any of the indices corresponding to } \tilde{r}[i][j,k] \text{ are not in } \mathcal{I} \\ 1 & \text{else if } \tilde{r}[i][j,k] = 1^{\log_{1/\hat{\gamma}}(n^7)} \\ 0 & \text{else} \end{cases}$$

4. **Check that for every $i \notin \mathsf{CycleSet}$, $\tilde{M}^{(i)}$ is valid**:

(a) For $i \in [n^6] \backslash \mathsf{CycleSet}$,

i. **Check that $\tilde{M}^{(i)}$ is fully revealed:** If $\tilde{M}^{(i)}$ contains a $\perp$, then reject.

ii. **Check that $\tilde{M}^{(i)} \notin \mathsf{Type\text{-}H}$:** If $\tilde{M}^{(i)}$ contains exactly n '1's and there does not exist a row or column with two or more '1's, then $\tilde{M}^{(i)}$ has a permutation

submatrix $\tilde{S}^{(i)} \in \{0,1\}^{n \times n}$. In this case, interpret $\tilde{S}^{(i)}$ as the adjacency matrix of some graph $\tilde{H}^{(i)}$ on n nodes. If $\tilde{H}^{(i)}$ is a Hamiltonian cycle, then reject.

5. **Check that for every $i \in \mathsf{CycleSet}$, $\tilde{M}^{(i)}$ is valid**:

   (a) For $i \in \mathsf{CycleSet}$:

      i. **Compute submatrix $\tilde{S}^{(i)}$:** Reject if $|\mathsf{SRows}^{(i)}| \neq n$ or $|\mathsf{SCols}^{(i)}| \neq n$. Otherwise, let $\tilde{S}^{(i)} \in \{0,1,\perp\}^{n \times n}$ be the submatrix of $\tilde{M}^{(i)}$ corresponding to the rows and columns specified in $\mathsf{SRows}^{(i)}$ and $\mathsf{SCols}^{(i)}$ respectively.

      ii. **Check that all rows and columns not in $\tilde{S}^{(i)}$ contain only '0's:** If $\tilde{M}^{(i)}_{j,k} \neq 0$ for any $(j,k) \notin (\mathsf{SRows}^{(i)} \times \mathsf{SCols}^{(i)})$, then reject.

      iii. **Check that each element in $\tilde{S}^{(i)}$ which corresponds to a non-edge of $\pi^{(i)}(G)$ is '0':** If $\tilde{S}^{(i)}_{t,s} \neq 0$ for any (t,s) that is not an edge of $\pi^{(i)}(G)$, then reject.

6. **Check that there are not too many matrices which contain $\geq n+1$ '1's:**

   (a) Compute

$$p_n := \Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}M1}]$$
$$= 1 - \sum_{i=0}^{n} \binom{n^8}{i} \left(\frac{1}{n^7}\right)^i \left(1 - \frac{1}{n^7}\right)^{n^8 - i}$$

   (b) Compute $w = |\{i \in [n^6] \mid \tilde{M}^{(i)} \text{ contains } \geq n+1 \text{ '1's}\}|$.

   (c) Reject if $w > n^6 \cdot p_n + n^4$.

7. **Accept:** Output 1 (accept).

5.2 Completeness

Lemma 5.4. ZHB-NIZK *satisfies statistical completeness in the* Z-Tamperable-Hidden-Bits *model.*

Proof. Let $\gamma \in (0,1)$ be any Efficiently-Approximable constant. If G is Hamiltonian, then an honest prover P_γ can always perform its part of the protocol. Additionally, in an honest execution, each $\tilde{M}^{(i)}$ agrees with $M^{(i)}$ on all positions where $\tilde{M}^{(i)} \neq \perp$. The verifier V_γ will accept if the proof passes the following conditions:

- **For every $i \notin \mathsf{CycleSet}$, $\tilde{M}^{(i)}$ is valid (Step 4):** This always holds for an honest prover's proof since an honest prover sets $i \notin \mathsf{CycleSet}$ if and only if $M^{(i)} \notin \mathsf{Type\text{-}H}$ and fully reveals all such matrices $M^{(i)}$ by including the indices corresponding to $r[i]$ in $\mathcal{I}$.

- **For every $i \in \mathsf{CycleSet}$, $\tilde{M}^{(i)}$ is valid (Step 5):** This always holds for an honest prover's proof. An honest prover sets $i \in \mathsf{CycleSet}$ if and only if matrix $M^{(i)} \in \mathsf{Type\text{-}H}$. Thus, $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of a Hamiltonian cycle $H^{(i)}$ on n-nodes. Furthermore, the only '1's that occur in $M^{(i)}$ are in $S^{(i)}$, so the prover can indeed reveal that all other rows and columns contain only '0's. Now observe that since G is Hamiltonian, there always exists a permutation $\pi^{(i)}$ such that the Hamiltonian cycle $H^{(i)}$ is a subgraph of $\pi^{(i)}(G)$. Therefore, for such a $\pi^{(i)}$, P can reveal that all the elements of $S^{(i)}$

which correspond to non-edges of $\pi^{(i)}(G)$ are '0' since every '1' in $S^{(i)}$ forms an edge of $H^{(i)}$ and thus must also be an edge of $\pi^{(i)}(G)$.

- **There are not too many matrices which contain $\geq n+1$ '1's (Step 6)**: Let $r \leftarrow \mathsf{Gen}_\gamma(1^{|G|})$, and for $i \in [n^6]$, let $M^{(i)}$ be computed from $r[i]$ as in Step 3a of the prover's algorithm. Let W be a random variable representing the number of matrices $M^{(i)}$ which contain $\geq n+1$ '1's. Then,

$$\mathbb{E}[W] = n^6 \cdot p_n$$

where

$$\begin{aligned} p_n &= \Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}M1}] \\ &= \Pr_{r \leftarrow \mathsf{Gen}_\gamma(1^{|G|})}\begin{bmatrix}\text{A matrix } M^{(i)} \text{ computed from } r[i] \text{ as in Step 3a} \\ \text{of the prover's algorithm contains} \geq n+1 \text{ '1's}\end{bmatrix} \end{aligned}$$

Therefore, by a Chernoff bound,

$$\Pr[W > n^6 \cdot p_n + n^4] \leq e^{-n^6 \cdot p_n/(3 \cdot p_n^2 \cdot n^4)} = e^{-n^2/(3 \cdot p_n)} = \mathsf{negl}(n)$$

so the prover passes this check with overwhelming probability over the choice of $r \leftarrow \mathsf{Gen}_\gamma(1^{|G|})$.

Thus, we have statistical completeness since a verifier accepts an honest proof with overwhelming probability over the choice of $r \leftarrow \mathsf{Gen}_\gamma(1^{|G|})$. □

5.3 Zero Knowledge

Lemma 5.5. ZHB-NIZK *satisfies perfect zero knowledge in the* Z-Tamperable-Hidden-Bits *model.*

Proof. This proof is essentially identical to the proof in [FLS90] as the prover's algorithm is identical except for some minor differences in parameters.

Let $\gamma \in (0,1)$ be any Efficiently-Approximable constant. We define our PPT simulator:

$\mathsf{Sim}_\gamma(G)$:

1. **Setup:** Set $\widehat{\mathcal{I}} = \emptyset$. Set $\widehat{\mathsf{CycleSet}} = \emptyset$. Let n be the number of nodes in G. Let $\hat{\gamma} \leftarrow \mathsf{Approx}_\gamma(1^n)$.
2. **Sample $\widehat{r}$:** Sample $\widehat{r} \leftarrow \mathsf{Gen}_\gamma(1^{|G|})$.
3. For $i \in [n^6]$:
   (a) **Sample matrix $M^{(i)}$ from $\widehat{r}[i]$:** Determine a matrix $M^{(i)} \in \{0,1\}^{n^4 \times n^4}$ where for $j,k \in [n^4]$, we set

$$M^{(i)}_{j,k} = \begin{cases} 1 & \text{if } \widehat{r}[i][j,k] = 1^{\log_{1/\hat{\gamma}}(n^7)} \\ 0 & \text{else} \end{cases}$$

   (b) **Check whether $M^{(i)} \in$ Type-H:** If $M^{(i)}$ does not contain exactly n '1's or there exists a row or column with two or more '1's, then $M^{(i)} \notin \mathsf{Type\text{-}H}$. Otherwise, $M^{(i)}$ has a permutation submatrix $S^{(i)} \in \{0,1\}^{n \times n}$. In this case, interpret $S^{(i)}$ as the

adjacency matrix of some graph $H^{(i)}$ on n nodes. If $H^{(i)}$ is a Hamiltonian cycle, then $M^{(i)} \in \mathsf{Type\text{-}H}$. Otherwise, $M^{(i)} \notin \mathsf{Type\text{-}H}$.

(c) **If $M^{(i)} \notin \mathsf{Type\text{-}H}$:**

i. **Reveal $M^{(i)}$:** Add the indices corresponding to $r[i]$ to $\mathcal{I}$.

(d) **If $M^{(i)} \in \mathsf{Type\text{-}H}$:**

i. **Add i to $\widehat{\mathsf{CycleSet}}$.**

ii. **Zero out $\widehat{r}[i]$:** Set $\widehat{r}[i] = 0^{n^8 \cdot \log_{1/\hat{\gamma}}(n^7)}$.

iii. **Pick a random submatrix $\widehat{S}^{(i)}$ and reveal all rows and columns not in $\widehat{S}^{(i)}$:** Sample $\widehat{\mathsf{SRows}}^{(i)}$ and $\widehat{\mathsf{SCols}}^{(i)}$ as random subsets of size n from $[n^4]$. "Reveal" all values not in $\widehat{S}^{(i)}$ by adding the indices corresponding to $\{r[i][j,k] \mid (j,k) \notin (\widehat{\mathsf{SRows}}^{(i)} \times \widehat{\mathsf{SCols}}^{(i)})\}$ to $\mathcal{I}$.

iv. **Pick a random permutation $\widehat{\pi}$ and reveal all elements in $\widehat{S}^{(i)}$ corresponding to non-edges of $\widehat{\pi}^{(i)}(G)$:** Pick a random permutation $\widehat{\pi}^{(i)}$ on n nodes. "Reveal" all values in the set

$$\{\widehat{S}^{(i)}_{t,s} \mid (t,s) \text{ is not an edge of } \widehat{\pi}^{(i)}(G)\}$$

by adding the indices of $r[i]$ corresponding to this set to $\mathcal{I}$.

4. Output $(\widehat{\mathcal{I}}, \{\widehat{r}_i\}_{i\in\widehat{\mathcal{I}}}, \widehat{\pi} = (\widehat{\mathsf{CycleSet}}, \{\widehat{\pi}^{(i)}, \widehat{\mathsf{SRows}}^{(i)}, \widehat{\mathsf{SCols}}^{(i)}\}_{i\in\widehat{\mathsf{CycleSet}}}))$.

Let G be any Hamiltonian graph on n nodes. To prove perfect zero knowledge, we need to to show that the following two distributions are equal:

$$D_0 = (I, \{R_i\}_{i\in I}, (\mathsf{CYCLESET}, \{\Pi^{(i)}, \mathsf{SROWS}^{(i)}, \mathsf{SCOLS}^{(i)}\}_{i\in\mathsf{CYCLESET}}))$$

$$D_1 = (\widehat{I}, \{\widehat{R}_i\}_{i\in\widehat{I}}, (\widehat{\mathsf{CYCLESET}}, \{\widehat{\Pi}^{(i)}, \widehat{\mathsf{SROWS}}^{(i)}, \widehat{\mathsf{SCOLS}}^{(i)}\}_{i\in\widehat{\mathsf{CYCLESET}}}))$$

where

- $R \coloneqq (R_1, \ldots, R_{n^{14}\log_{1/\hat{\gamma}}(n^7)}), I, \mathsf{CYCLESET}, \Pi^{(i)}, \mathsf{SROWS}^{(i)}, \mathsf{SCOLS}^{(i)}$ are random variables for $r, \mathcal{I}, \mathsf{CycleSet}, \pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}$ respectively where

$$r \leftarrow \mathsf{Gen}_\gamma(1^{|G|}) \text{ and } (\mathcal{I}, (\mathsf{CycleSet}, \{\pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}\}_{i\in\mathsf{CycleSet}})) \leftarrow P_\gamma(r, G)$$

- $\widehat{R} \coloneqq (\widehat{R}_1, \ldots, \widehat{R}_{n^{14}\log_{1/\hat{\gamma}}(n^7)}), \widehat{I}, \widehat{\mathsf{CYCLESET}}, \widehat{\Pi}^{(i)}, \widehat{\mathsf{SROWS}}^{(i)}, \widehat{\mathsf{SCOLS}}^{(i)}$ are random variables for $\widehat{r}, \widehat{\mathcal{I}}, \widehat{\mathsf{CycleSet}}, \widehat{\pi}^{(i)}, \widehat{\mathsf{SRows}}^{(i)}, \widehat{\mathsf{SCols}}^{(i)}$ respectively where

$$(\widehat{\mathcal{I}}, \{\widehat{r}_i\}_{i\in\widehat{\mathcal{I}}}, (\widehat{\mathsf{CycleSet}}, \{\widehat{\pi}^{(i)}, \widehat{\mathsf{SRows}}^{(i)}, \widehat{\mathsf{SCols}}^{(i)}\}_{i\in\widehat{\mathsf{CycleSet}}})) \leftarrow \mathsf{Sim}_\gamma(G)$$

We proceed by splitting the joint distribution into a chain of conditional distributions. The equality of D_0 and D_1 then follows from the fact that the individual conditional distributions are indeed identical:

1. For any fixed values $(\mathsf{CycleSet}, \{r[i]\}_{i\notin\mathsf{CycleSet}}, \{\pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}\}_{i\in\mathsf{CycleSet}})$, the conditional distribution of

$$(I, \{R_i\}_{i\in I} \mid \mathsf{CYCLESET} = \mathsf{CycleSet}, \{R[i]\}_{i\notin\mathsf{CYCLESET}} = \{r[i]\}_{i\notin\mathsf{CycleSet}}$$
$$\{\Pi^{(i)}, \mathsf{SROWS}^{(i)}, \mathsf{SCOLS}^{(i)}\}_{i\in\mathsf{CYCLESET}} = \{\pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}\}_{i\in\mathsf{CycleSet}})$$

is identical to the conditional distribution of

$$(\widehat{I}, \{\widehat{R}_i\}_{i\in\widehat{I}} \mid \widehat{\mathsf{CYCLESET}} = \mathsf{CycleSet}, \{\widehat{R}[i]\}_{i\notin\mathsf{CYCLESET}} = \{r[i]\}_{i\notin\mathsf{CycleSet}}$$
$$\{\widehat{\Pi}^{(i)}, \widehat{\mathsf{SROWS}}^{(i)}, \widehat{\mathsf{SCOLS}}^{(i)}\}_{i\in\widehat{\mathsf{CYCLESET}}} = \{\pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}\}_{i\in\mathsf{CycleSet}})$$

This is because the values of $(I, \{R_i\}_{i\in I})$ can be deterministically determined from the values of $(G, \mathsf{CYCLESET}, \{R[i]\}_{i\notin\mathsf{CYCLESET}}, \{\Pi^{(i)}, \mathsf{SROWS}^{(i)}, \mathsf{SCOLS}^{(i)}\}_{i\in\mathsf{CYCLESET}})$ in the same way that the values of $(\widehat{I}, \{\widehat{R}_i\}_{i\in\widehat{I}})$ can be deterministically determined from $(G, \widehat{\mathsf{CYCLESET}}, \{\widehat{R}[i]\}_{i\notin\widehat{\mathsf{CycleSet}}}, \{\widehat{\Pi}^{(i)}, \widehat{\mathsf{SROWS}}^{(i)}, \widehat{\mathsf{SCOLS}}^{(i)}\}_{i\in\widehat{\mathsf{CYCLESET}}})$. In both cases,

- For every $i \notin \mathsf{CycleSet}$, I (and $\widehat{I}$) will contain the indices corresponding to $\{r[i]\}_{i\notin\mathsf{CycleSet}}$. For each of these indices, the corresponding value of R (and $\widehat{R}$) is determined by $\{r[i]\}_{i\notin\mathsf{CycleSet}}$.
- For every $i \in \mathsf{CycleSet}$, I (and $\widehat{I}$) will contain the indices corresponding to the rows and columns of the i^{th} matrix that are not part of the submatrix $S^{(i)}$ formed by $\mathsf{SRows}^{(i)}$ and $\mathsf{SCols}^{(i)}$ along with the indices of the elements of $S^{(i)}$ corresponding to non-edges of $\pi^{(i)}(G)$. For each of these indices, the corresponding value of R (and $\widehat{R}$) is '0'.

2. For any fixed $\mathsf{CycleSet}$, the conditional distribution of

$$(\{\Pi^{(i)}, \mathsf{SROWS}^{(i)}, \mathsf{SCOLS}^{(i)}\}_{i\in\mathsf{CYCLESET}} \mid \mathsf{CYCLESET} = \mathsf{CycleSet})$$

is identical to the conditional distribution of

$$(\{\widehat{\Pi}^{(i)}, \widehat{\mathsf{SROWS}}^{(i)}, \widehat{\mathsf{SCOLS}}^{(i)}\}_{i\in\mathsf{CYCLESET}} \mid \widehat{\mathsf{CYCLESET}} = \mathsf{CycleSet})$$

and both conditional distributions are independent of G, $\{R[i]\}_{i\notin\mathsf{CYCLESET}}$, and $\{\widehat{R}[i]\}_{i\notin\widehat{\mathsf{CYCLESET}}}$. This is because in the real world game (distribution D_0), for each i, conditioned on $i \in \mathsf{CycleSet}$, $M^{(i)}$ is independently and uniformly distributed over all matrices of type $\mathsf{Type\text{-}H}$. Thus the permutation submatrix $S^{(i)}$ of $M^{(i)}$ is uniformly distributed over all possible $n \times n$ submatrices of $M^{(i)}$, meaning that $\mathsf{SROWS}^{(i)}$ and $\mathsf{SCOLS}^{(i)}$ are uniformly distributed over all subsets of size n of $[n^4]$. Additionally, as the Hamiltonian cycle $H^{(i)}$ given by $S^{(i)}$ is uniformly distributed over all Hamiltonian cycles on n nodes, then for any Hamiltonian cycle C_G of G, the permutation $\pi^{(i)}$ such that $\pi^{(i)}(C_G) = H^{(i)}$ is uniformly distributed over all permutations on n nodes. Thus the real world distribution (distribution D_0) matches the simulated distribution (distribution D_1).

3. The distributions $(\mathsf{CYCLESET}, \{R[i]\}_{i\notin\mathsf{CYCLESET}})$ and $(\widehat{\mathsf{CYCLESET}}, \{\widehat{R}[i]\}_{i\notin\widehat{\mathsf{CYCLESET}}})$ are identical as we sample these variables in the same manner.

4. Applying chain rule to the above distributions, we get the desired result.

□

5.4 Soundness

Lemma 5.6. $\mathsf{ZHB\text{-}NIZK}$ *satisfies (adaptive) statistical soundness in the* $\mathsf{Z\text{-}Tamperable\text{-}Hidden\text{-}Bits}$ *model. In particular, for sufficiently large* n, $\mathsf{ZHB\text{-}NIZK}$ *achieves* $\frac{1}{e^n}$*-adaptive-statistical-soundness.*

Notation. For $n \in \mathbb{N}$, we will use $|G_n|$ to denote the size of graphs with n nodes.

5.4.1 Proof Overview

Let $\gamma \in (0,1)$ be any Efficiently-Approximable constant and let $n \in \mathbb{N}$ be sufficiently large. We want to show that with high probability over $r \leftarrow \mathsf{Gen}_\gamma(1^{|G_n|})$, the verifier will not accept any cheating proof for any non-Hamiltonian graph G.

To show this, we define the following variables: Let P^*_γ be any (unbounded) cheating prover. Let $r \leftarrow \mathsf{Gen}_\gamma(1^{|G_n|})$, and consider any non-Hamiltonian graph G on n nodes and any cheating proof $(\mathcal{I}, f, \pi = (\mathsf{CycleSet}, \{\pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}\})_{i \in \mathsf{CycleSet}})$ output by $P^*_\gamma(r)$. Let $\{\tilde{r}_i\}_{i \in \mathcal{I}} = \{\mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathcal{I}}$. Let $M^{(i)}$ be the matrix that an honest prover would produce from $r[i]$, and let $\tilde{M}^{(i)}$ be the matrix that a honest verifier would produces from $\tilde{r}[i]$.

Soundness in the Original Hidden Bits Model. In the original soundness proof of [FLS90] in the regular Hidden-Bits model, it was shown that if any $M^{(i)} \in$ Type-H, then the verifier will reject.

- If $i \in \mathsf{CycleSet}$, P^* must prove that the '1's in $M^{(i)}$ form a graph $H^{(i)}$ which is a subgraph of $\pi^{(i)}(G)$ for some permutation $\pi^{(i)}$ and some non-Hamiltonian graph G. But, the '1's of any $M^{(i)} \in$ Type-H form a Hamiltonian cycle. Since no Hamiltonian cycle is a subgraph of any permutation of a non-Hamiltonian graph, then P^* cannot include $i \in \mathsf{CycleSet}$ without the verifier rejecting.
- If $i \notin \mathsf{CycleSet}$, then P^* must fully reveal $M^{(i)}$. However, the verifier will reject if it sees any fully revealed matrix of type Type-H.

To finish the proof, [FLS90] show that the probability that there exists at least one $M^{(i)} \in$ Type-H is overwhelming.

Soundness in the Z-Tamperable-Hidden-Bits Model. In contrast, in our Z-Tamperable-Hidden-Bits model, the cheating prover has much more power. In particular, the cheating prover can change any '0's in the hidden bit string to '1's. Thus, for any $M^{(i)} \in$ Type-H, the cheating prover can invalidate this matrix by adding '1's to $M^{(i)}$ so that the verifier's $\tilde{M}^{(i)} \notin$ Type-H. Therefore, the cheating prover can safely set $i \notin \mathsf{CycleSet}$ and can fully reveal this matrix without the verifier rejecting.

In order to prevent this behavior, the verifier will count the number of fully revealed matrices which contain at least $n+1$ '1's (i.e. those of type Type-M1) and reject if the number is much greater than expected. The hope is that since the cheating prover must invalidate all matrices $M^{(i)} \in$ Type-H by adding '1's to these matrices (which ensures that $\tilde{M}^{(i)} \in$ Type-M1), then this should push the total number of matrices in Type-M1 beyond the verifier's bound, causing the verifier to reject.

Unfortunately, the prover can also cheat in a second manner. The prover can choose not to fully reveal some matrices $M^{(i)} \in$ Type-M1 by setting $i \in \mathsf{CycleSet}$ and pretending that these were matrices of type Type-H . This will lower the apparent number of matrices of type Type-M1 that count toward the verifier's bound. However, for every $i \in \mathsf{CycleSet}$, the prover must show that the '1's in $M^{(i)}$ form a graph $H^{(i)}$ which is a subgraph of $\pi^{(i)}(G)$ for some permutation $\pi^{(i)}$. Therefore, the matrices $M^{(i)}$ for which the prover can cheat in this second manner are limited by this constraint.

Indeed, we will show that the expected number of matrices $M^{(i)} \in$ Type-M1 for which the prover can cheat in this second manner is far less than the expected number of matrices $M^{(i)} \in$ Type-H. Thus, with high probability, the cheating prover must either inflate the count of matrices

$\tilde{M}^{(i)} \in \mathsf{Type\text{-}M1}$ beyond the verifier's bound, or put some $i \in \mathsf{CycleSet}$ for some matrix $\tilde{M}^{(i)}$ which will not pass the subgraph check. In either case, the verifier will reject. In particular, we show the last step by proving that there is a large class of matrices $\mathsf{Type\text{-}PB1} \subseteq \mathsf{Type\text{-}M1}$ for which the prover cannot cheat in this second manner. $\mathsf{Type\text{-}PB1}$ is the class of matrices whose '1's form a permutation submatrix on $n+k$ nodes for $1 \leq k \leq 2n$. The prover cannot cheat in the second manner on matrices $M^{(i)} \in \mathsf{Type\text{-}PB1}$ since the prover cannot fit the '1's of these matrices into an $n \times n$ submatrix of $M^{(i)}$.

Thus, our proof will proceed in two parts:

1. First, we will show that if $M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})$, then the verifier will only accept a proof for a non-Hamiltonian graph G if $i \notin \mathsf{CycleSet}$ and $\tilde{M}^{(i)} \in \mathsf{Type\text{-}M1}$.

2. We will then show that with overwhelming probability over r, there are enough matrices $M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})$ to cause the corresponding number of matrices $\tilde{M}^{(i)} \in \mathsf{Type\text{-}M1}$ to exceed the bound that the verifier checks in Step 6. In particular, we will prove

$$\Pr_{r \leftarrow \mathsf{Gen}_\gamma(1^{|G_n|})} \left[\left| \{i \mid M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})\} \right| > n^6 \cdot p_n + n^4 \right] \geq 1 - \mathsf{negl}(|G_n|)$$

   where $p_n = \Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}M1}]$.

Together, these two steps imply adaptive statistical soundness.

5.4.2 Formal Proof

Let $\gamma \in (0,1)$ be any $\mathsf{Efficiently\text{-}Approximable}$ constant and let $n \in \mathbb{N}$. Let P^*_γ be any (unbounded) cheating prover. Let $r \leftarrow \mathsf{Gen}_\gamma(1^{|G_n|})$, and consider any non-Hamiltonian graph G on n nodes and any cheating proof $(\mathcal{I}, f, \pi = (\mathsf{CycleSet}, \{\pi^{(i)}, \mathsf{SRows}^{(i)}, \mathsf{SCols}^{(i)}\})_{i \in \mathsf{CycleSet}})$ output by $P^*_\gamma(r)$. Let $\{\tilde{r}_i\}_{i \in \mathcal{I}} = \{\mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathcal{I}}$. Let $M^{(i)}$ be the matrix that an honest prover would produce from $r[i]$, and let $\tilde{M}^{(i)}$ be the matrix that a honest verifier would produces from $\tilde{r}[i]$.

Using ZeroFlip. Observe that P^*_γ can change any '0' in r to a '1' in $\tilde{r}$ since P^* can specify f. However, P^*_γ cannot change any '1' in r to a '0' in $\tilde{r}$. Now, the verifier only uses $\tilde{r}$ to compute matrices $\tilde{M}^{(i)}$. Since $\tilde{M}^{(i)}_{j,k} = 1$ if and only if $\tilde{r}[i][j,k] = 1^{\log_{1/\hat{\gamma}}(n^7)}$, then P^*_γ can change any '0' in $M^{(i)}$ to a '1' in $\tilde{M}^{(i)}$, but cannot change any '1' in $M^{(i)}$ to a '0' in $\tilde{M}^{(i)}$.

Lemma 5.7. *If the verifier accepts, then for every $i \in \mathsf{CycleSet}$, all the '1's in $M^{(i)}$ must be contained in some submatrix $S^{(i)} \in \{0,1\}^{n \times n}$ which is the adjacency matrix of some graph $H^{(i)}$ which is a subgraph of $\pi^{(i)}(G)$.*

Proof. Suppose that the verifier accepts. Let $i \in \mathsf{CycleSet}$, and let $S^{(i)}$ and $\tilde{S}^{(i)}$ be the respective submatrices of $M^{(i)}$ and $\tilde{M}^{(i)}$ specified by $\mathsf{SRows}^{(i)}$ and $\mathsf{SCols}^{(i)}$. Let $H^{(i)}$ be the graph whose adjacency matrix is $S^{(i)}$.

In Step 5, the verifier first checks that everything in $\tilde{M}^{(i)}$ which is not part of $\tilde{S}^{(i)}$ is '0'. Thus, since P^*_γ cannot change any '1's in $M^{(i)}$ to '0's in $\tilde{M}^{(i)}$, then the '1's of the original $M^{(i)}$ must fit within the $n \times n$ submatrix $S^{(i)}$.

Now, suppose that $H^{(i)}$ were not a subgraph of $\pi^{(i)}(G)$. Then, $H^{(i)}$ contains an edge that is a non-edge of $\pi^{(i)}(G)$, which means that the corresponding position in $S^{(i)}$ is a '1'. Since P^*_γ cannot change any '1's in $S^{(i)}$ to '0's in $\tilde{S}^{(i)}$, then this means that the corresponding position in

$\tilde{S}(i)$ is not '0'. But then the verifier will reject in Step 5, since there is a non-edge of $\pi^{(i)}(G)$ whose corresponding position in $\tilde{S}(i)$ is not '0'. Therefore, $H^{(i)}$ must be a subgraph of $\pi^{(i)}(G)$.

□

Lemma 5.8. *If the verifier accepts a proof for a non-Hamiltonian graph G and $M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})$, then $i \notin \mathsf{CycleSet}$ and $\tilde{M}^{(i)} \in \mathsf{Type\text{-}M1}$.*

Proof. Let $M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})$ and suppose that the verifier accepts. We have two cases.

- If $M^{(i)} \in \mathsf{Type\text{-}H}$:

  Then, $M^{(i)}$ contains a submatrix $S^{(i)}$ which is the adjacency matrix of some Hamiltonian cycle $H^{(i)}$. Furthermore, this is the only $n \times n$ submatrix of $M^{(i)}$ that contains all the '1's of $M^{(i)}$. Since G is non-Hamiltonian, then $H^{(i)}$ is not a subgraph of $\pi^{(i)}(G)$ for any permutation $\pi^{(i)}$. Thus, by Lemma 5.7, we must have that $i \notin \mathsf{CycleSet}$.

  Now, since $i \notin \mathsf{CycleSet}$ and because the verifier did not reject in Step 4, it must be the case that P^* revealed all of $\tilde{M}^{(i)}$ and that $\tilde{M}^{(i)} \notin \mathsf{Type\text{-}H}$. Since $M^{(i)} \in \mathsf{Type\text{-}H}$, this means that P^* made use of its $\mathsf{ZeroFlip}$ string f to ensure that $\tilde{M}^{(i)} \notin \mathsf{Type\text{-}H}$. The only way that P^* can ensure this is by changing at least one '0' in $M^{(i)}$ to a '1' in $\tilde{M}^{(i)}$. But since $M^{(i)}$ had exactly n '1's, this means that $\tilde{M}^{(i)}$ will have at least $n+1$ '1's, meaning that $\tilde{M}^{(i)} \in \mathsf{Type\text{-}M1}$.

- If $M^{(i)} \in \mathsf{Type\text{-}PB1}$:

  Then, $M^{(i)}$ contains a permutation submatrix in $\{0,1\}^{(n+k)\times(n+k)}$ for some $1 \leq k \leq n$. But this means that the '1's of $M^{(i)}$ cannot fit within an $n \times n$ submatrix of $M^{(i)}$, so by Lemma 5.7, we must have that $i \notin \mathsf{CycleSet}$.

  Now, since $i \notin \mathsf{CycleSet}$ and because the verifier did not reject in Step 4, it must be the case that P^*_γ revealed all of $\tilde{M}^{(i)}$. Since P^*_γ cannot change any '1's in $M^{(i)}$ to '0's in $\tilde{M}^{(i)}$ and since $M^{(i)}$ already contains at least $n+1$ '1's, then $\tilde{M}^{(i)}$ must also contain at least $n+1$ '1's.

□

Lemma 5.9. *For all* $\mathsf{Efficiently\text{-}Approximable}$ *constants $\gamma \in (0,1)$,*

$$\Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})] = p_n + \Omega\left(\frac{1}{n^{1.5}}\right)$$

where $p_n = \Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}M1}]$

Proof. Since $\mathsf{Type\text{-}H}$ and $\mathsf{Type\text{-}PB1}$ are disjoint and $\mathsf{Type\text{-}PB1} \subset \mathsf{Type\text{-}M1}$, we get that

$$\begin{aligned}&\Pr[M \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})]\\ &= \Pr[M \in \mathsf{Type\text{-}H}] + \Pr[M \in \mathsf{Type\text{-}PB1} \mid M \in \mathsf{Type\text{-}M1}] \cdot \Pr[M \in \mathsf{Type\text{-}M1}]\end{aligned}$$

To continue the proof, we will bound each of these probabilities using the following two lemmas.

Lemma 5.10. *For all* $\mathsf{Efficiently\text{-}Approximable}$ *constants $\gamma \in (0,1)$ and for sufficiently large n,*

$$\Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}PB1} \mid M \in \mathsf{Type\text{-}M1}] \cdot \Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}M1}] \geq p_n\left(1 - O\left(\frac{1}{n^2}\right)\right) - \mathsf{negl}(n)$$

where $p_n = \Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}M1}]$.

Proof. Recall that $M \in \mathsf{Type\text{-}PB1}$ if M contains a permutation submatrix $S \in \{0,1\}^{(n+k)\times(n+k)}$ for some $1 \le k \le n$, and every element of M that is not in S is '0'. Equivalently, $M \in \mathsf{Type\text{-}PB1}$ if M contains exactly $n+k$ '1's for some $1 \le k \le n$ and no row or column of M has 2 or more '1's. Thus,

$$\begin{aligned}
&\Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}PB1} \mid M \in \mathsf{Type\text{-}M1}] \\
&= \sum_{k=1}^{n} \Big(\Pr[M \text{ has no row or column with 2 or more '1's} \mid M \text{ has exactly } n+k \text{ '1's}] \\
&\qquad \cdot \Pr[M \text{ has exactly } n+k \text{ '1's} \mid M \text{ has} > n \text{ '1's}] \Big) \\
&\ge \Pr[M \text{ has no row or column with 2 or more '1's} \mid M \text{ has exactly 2n '1's}] \\
&\quad \cdot \sum_{k=1}^{n} \Big(\Pr[M \text{ has exactly } n+k \text{ '1's} \mid M \text{ has} > n \text{ '1's}] \Big) \\
&= \Pr[M \text{ has no row or column with 2 or more '1's} \mid M \text{ has exactly 2n '1's}] \\
&\quad \cdot \Pr[M \text{ has} \le 2n \text{ '1's} \mid M \text{ has} > n \text{ '1's}]
\end{aligned}$$

where the first inequality follows from the fact that the probability of getting a collision decreases with fewer '1's. Now, since the expected number of '1's in M is n, by a Chernoff bound, we get

$$\begin{aligned}
&\Pr[M \text{ has} \le 2n \text{ '1's} \mid M \text{ has} > n \text{ '1's}] \\
&= 1 - \Pr[M \text{ has} > 2n \text{ '1's} \mid M \text{ has} > n \text{ '1's}] \\
&= 1 - \Pr[M \text{ has} > 2n \text{ '1's}] \,/\, \Pr[M \text{ has} > n \text{ '1's}] \\
&\ge 1 - e^{-n/3}/p_n
\end{aligned}$$

Then, by a union bound,

$$\begin{aligned}
&\Pr[M \text{ has no row or column with 2 or more '1's} \mid M \text{ has exactly 2n '1's}] \\
&= 1 - \Pr[M \text{ has at least one row or column with 2 or more '1's} \mid M \text{ has exactly 2n '1's}] \\
&\ge 1 - (\Pr[M \text{ has at least one row with 2 or more '1's} \mid M \text{ has exactly 2n '1's}] \\
&\qquad + \Pr[M \text{ has at least one column with 2 or more '1's} \mid M \text{ has exactly 2n '1's}]) \\
&= 1 - 2 \cdot \Pr[M \text{ has at least one row with 2 or more '1's} \mid M \text{ has exactly 2n '1's}] \\
&\ge 1 - 2 \sum_{i,j \in [n], i<j} \Pr\Big[\text{the } i^{th} \text{ and } j^{th} \text{ '1' in } M \text{ occur in the same row}\Big] \\
&\ge 1 - 2\binom{2n}{2} \frac{n^4}{(n^8 - 2n)} \\
&= 1 - O\left(\frac{1}{n^2}\right)
\end{aligned}$$

where the last inequality follows since conditioned on M having exactly 2n '1's, the set of '1's in M are uniformly distributed over all sets of size $2n$ of the n^8 possible positions in M. Thus, there are at most n^4 positions for the j^{th} '1' that are in the same row as the i^{th} '1', and there are at least

$n^8 - 2n$ positions remaining in the matrix. Thus,

$$\begin{aligned}
&\Pr[M \in \mathsf{Type\text{-}PB1} \mid M \in \mathsf{Type\text{-}M1}] \cdot \Pr[M \in \mathsf{Type\text{-}M1}] \\
&\geq \left(1 - O\left(\frac{1}{n^2}\right)\right) \cdot (1 - e^{-n/3}/p_n) \cdot p_n \\
&= p_n \left(1 - O\left(\frac{1}{n^2}\right)\right) - \mathsf{negl}(n)
\end{aligned}$$

□

Lemma 5.11. *For all* Efficiently-Approximable *constants* $\gamma \in (0, 1)$*,*

$$\Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}H}] = \Omega\left(\frac{1}{n^{1.5}}\right)$$

Proof. $M \in$ Type-H if M contains a submatrix $S \in \{0,1\}^{n \times n}$ which is the adjacency matrix of a Hamiltonian cycle on n nodes, and and every element of M that is not in S is '0'. Equivalently, $M \in$ Type-H if M contains exactly n '1's which form a permutation matrix $S \in \{0,1\}^{n \times n}$ which is an n-cycle. Thus,

$$\begin{aligned}
&\Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}H}] \\
&= \Pr\left[S \text{ is an } n\text{-cycle} \mid M \text{ has exactly n '1's which form a permutation submatrix } S \in \{0,1\}^{n \times n}\right] \\
&\quad \cdot \Pr\left[M \text{ has no row or column with 2 or more '1's} \mid M \text{ has exactly n '1's}\right] \\
&\quad \cdot \Pr\left[M \text{ has exactly n '1's}\right]
\end{aligned}$$

Now,

$$\begin{aligned}
&\Pr\left[S \text{ is an } n\text{-cycle} \mid M \text{ has exactly n '1's which form a permutation submatrix } S \in \{0,1\}^{n \times n}\right] \\
&= \frac{(n-1)!}{n!} = \frac{1}{n}
\end{aligned}$$

Furthermore, using essentially the same calculation as in Lemma 5.10, we get

$$\Pr\left[M \text{ has no row or column with 2 or more '1's} \mid M \text{ has exactly n '1's}\right] = 1 - O\left(\frac{1}{n^2}\right)$$

By Chebyshev's inequality,

$$\begin{aligned}
\Pr\left[\left|(\text{number of '1's in } M) - n\right| > \sqrt{2n}\right] &\leq \frac{\mathsf{Var}(\text{number of '1's in } M)}{n^2} \\
&= \frac{n^8 \cdot \left(\frac{1}{n^7}\right)\left(1 - \frac{1}{n^7}\right)}{2n} \\
&= \frac{1}{2} - \frac{1}{2n^7} \\
&\leq \frac{1}{2}
\end{aligned}$$

Thus, since the expected number of '1's in M is n, then

$$\begin{aligned}\Pr\left[M \text{ has exactly n '1's}\right] &\geq \frac{1}{2\sqrt{2n}} \sum_{i=n-\sqrt{2n}}^{n+\sqrt{2n}} \Pr\left[M \text{ has exactly } i \text{ '1's}\right] \\ &= \frac{1}{2\sqrt{2n}} \left(1 - \Pr\left[\left|(\text{number of '1's in } M) - n\right| > \sqrt{2n}\right]\right) \\ &\geq \frac{1}{4\sqrt{2n}}\end{aligned}$$

Therefore,

$$\Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}H}] \geq \frac{1}{n} \cdot \left(1 - O\left(\frac{1}{n^2}\right)\right) \cdot \frac{1}{4\sqrt{2n}} = \Omega\left(\frac{1}{n^{1.5}}\right)$$

□

Thus, by Lemma 5.10 and Lemma 5.11,

$$\begin{aligned}&\Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})] \\ &= \Pr[M \in \mathsf{Type\text{-}H}] + \Pr[M \in \mathsf{Type\text{-}PB1} \mid M \in \mathsf{Type\text{-}M1}] \cdot \Pr[M \in \mathsf{Type\text{-}M1}] \\ &= \Omega\left(\frac{1}{n^{1.5}}\right) + \left(1 - O\left(\frac{1}{n^2}\right)\right) \cdot p_n - \mathsf{negl}(n) \\ &= p_n + \Omega\left(\frac{1}{n^{1.5}}\right)\end{aligned}$$

which completes the proof. □

Corollary 5.12. *For all* Efficiently-Approximable *constants* $\gamma \in (0,1)$ *and for all sufficiently large* $n \in \mathbb{N}$*,*

$$\Pr_{r \leftarrow \mathsf{Gen}_\gamma(1^{|G_n|})}\left[\left|\left\{i \mid M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})\right\}\right| > n^6 \cdot p_n + n^4\right] \geq 1 - \frac{1}{e^n}$$

where $p_n = \Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in \mathsf{Type\text{-}M1}]$.

Proof. Since by taking the probability over r, each $M[i]$ is sampled independently according to $\mathsf{MatGen}(1^n)$, then by Lemma 5.9,

$$\begin{aligned}\mu :&= \mathop{\mathbb{E}}_{r \leftarrow \mathsf{Gen}_\gamma(1^{|G|})}\left[\left|\left\{i \mid M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})\right\}\right|\right] \\ &= n^6 \cdot \Pr_{M \leftarrow \mathsf{MatGen}(1^n)}[M \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})] \\ &= n^6 \cdot p_n + n^6 \cdot \Omega\left(\frac{1}{n^{1.5}}\right) \\ &= n^6 \cdot p_n + \Omega(n^{4.5})\end{aligned}$$

Thus, by a Chernoff bound, for sufficiently large $n \in \mathbb{N}$,

$$\begin{aligned}
&\Pr_{r \leftarrow \mathsf{Gen}_\gamma(1^{|G|})}\left[\left|\{i \mid M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})\}\right| > n^6 \cdot p_n + n^4\right] \\
&= 1 - \Pr_{r \leftarrow \mathsf{Gen}_\gamma(1^{|G|})}\left[\left|\{i \mid M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})\}\right| \leq n^6 \cdot p_n + n^4\right] \\
&\geq 1 - \Pr_{r \leftarrow \mathsf{Gen}_\gamma(1^{|G|})}\left[\left|\{i \mid M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})\}\right| < \left(1 - \frac{1}{n^2}\right)\mu\right] \\
&\geq 1 - e^{-\frac{\mu}{3n^4}} = 1 - e^{-\Omega(n^2)} \\
&\geq 1 - e^{-n}
\end{aligned}$$

□

Finally, we can prove soundness.

Lemma 5.13. *For all* Efficiently-Approximable *constants* $\gamma \in (0,1)$ *and for all sufficiently large* $n \in \mathbb{N}$*,*

$$\Pr_{r \leftarrow \mathsf{Gen}_\gamma(1^{|G_n|})}\Big[\exists (G, \mathcal{I}, f, \pi) \text{ such that } G \text{ is a non-Hamiltonian graph on } n \text{ nodes}, \\ \text{and } V_\gamma(G, \mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathcal{I}}, \pi) = 1\Big] \leq \frac{1}{e^n}.$$

where we define

$$\mathsf{ZeroFlip}(r_i, f_i) = \begin{cases} r_i \oplus f_i & \text{if } r_i = 0 \\ r_i & \text{if } r_i = 1 \end{cases}$$

Proof. Let $\gamma \in (0,1)$ be any Efficiently-Approximable constant. By Corollary 5.12, for sufficiently large $n \in \mathbb{N}$, except with probability at most e^{-n} over the choice of $r \leftarrow \mathsf{Gen}_\gamma(1^{|G_n|})$, we have

$$\left|\{i \mid M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})\}\right| > n^6 \cdot p_n + n^4$$

By Lemma 5.8, if the verifier accepts a proof for a non-Hamiltonian graph G, then for each i such that $M^{(i)} \in (\mathsf{Type\text{-}H} \cup \mathsf{Type\text{-}PB1})$, it must be the case that $i \notin \mathsf{CycleSet}$ and $\tilde{M}^{(i)} \in \mathsf{Type\text{-}M1}$. Thus, for sufficiently large $n \in \mathbb{N}$, except with probability at most e^{-n} over the choice of $r \leftarrow \mathsf{Gen}_\gamma(1^{|G_n|})$, any accepting proof for a non-Hamiltonian graph G must have

$$\left|\{i \mid \tilde{M}^{(i)} \in \mathsf{Type\text{-}M1}\}\right| > n^6 \cdot p_n + n^4$$

However, this means that the verifier will reject in Step 6. □

6 NIZK Proofs for NP in the Random Oracle Model

In this section, we show how to construct NIZK proofs in the Random Oracle (RO) model. We prove the following:

Theorem 6.1. *There exists an (unbounded-prover)* NIZK *proof system for* NP *in the* Random Oracle (RO) *model.*

Proof. We first define some additional notation.

Notation.

- We use $\mathsf{pad}_n(i)$ to denote the binary representation of i padded to n bits.
- We use ROS_n to denote the set of all functions from n bits to n bits.
- Extending Lemma 3.7 slightly, we will interpret our random oracle $\mathcal{O}$ as two families of functions:
  - $\{\mathcal{O}_n^{\mathsf{urs}} : \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$ where $\mathcal{O}_n^{\mathsf{urs}}(x) = \mathcal{O}(0\|1^n\|0\|x\|\mathsf{pad}_n(1)) \dots \mathcal{O}(0\|1^n\|0\|x\|\mathsf{pad}_n(n))$ and
  - $\{\mathcal{O}_n^{p} : \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$ where $\mathcal{O}_n^{p}(x) = \mathcal{O}(1\|1^n\|0\|x\|\mathsf{pad}_n(1)) \dots \mathcal{O}(1\|1^n\|0\|x\|\mathsf{pad}_n(n))$.

  Note that if we are given a query to $\mathcal{O}$, we can determine whether or not the query belongs to $\mathcal{O}_n^{\mathsf{urs}}$ or $\mathcal{O}_n^{p}$ by checking the format of the query.

Proof Overview. In the previous section, we showed how to build a NIZK proof in the Z-Tamperable-Hidden-Bits model. To build a NIZK proof in the RO model, we will instantiate the hidden bit string functionality of the Z-Tamperable-Hidden-Bits model by determining a hidden bit string r from the random oracle $\mathcal{O}$ in the following way: We first create a $\mathsf{urs} = y_1 \dots y_{p_{\mathsf{hbs}(n)}}$ using multiple invocations of $\mathcal{O}_n^{\mathsf{urs}}$ on fixed inputs, and then set each bit r_i of the hidden bit string equal to 1 if and only if y_i does not contain a pre-image under $\mathcal{O}_n^{p}$. Since $\mathcal{O}_n^{p}$ is $(1-e^{-1})$-Dense with overwhelming probability, then r_i will be sampled similarly to our ZHB hidden bit string generator $\mathsf{ZHB.Gen}_{e^{-1}}(1^n)$ with overwhelming probability. Observe that the prover can reveal that $r_i = 0$ by sending over a pre-image v_i for y_i (which the verifier can check), and can claim that $r_i = 1$ by simply stating that "there is no pre-image".

For soundness, observe that just like in the Z-Tamperable-Hidden-Bits model, the prover can lie about r_i whenever $r_i = 0$, but cannot lie about r_i whenever $r_i = 1$ because the prover cannot send over a pre-image when no such pre-image exists. Thus, the soundness reduces to the soundness of the Z-Tamperable-Hidden-Bits model construction.

For zero knowledge, the simulator will first run the ZHB simulator on x to get a simulated ZHB proof and simulated hidden bits. The simulator will then simulate the urs and random oracle values using a lazy sampling method that is consistent with these simulated hidden bits. We then use properties of the random oracle and the security of the ZHB simulator to show that sampling our values in this manner results in a similar distribution as in the real world experiment.

6.1 Construction

Let $\mathcal{L}$ be an NP language. By Theorem 5.1, there exists a NIZK proof system $\{\mathsf{ZHB\text{-}NIZK}_\gamma = (\mathsf{ZHB.Gen}_\gamma, \mathsf{ZHB}.P_\gamma, \mathsf{ZHB}.V_\gamma)\}$ for $\mathcal{L}$ in the Z-Tamperable-Hidden-Bits model. Let $p_{\gamma,\mathsf{hbs}}(n)$ be the length of the hidden bit string of $\mathsf{ZHB\text{-}NIZK}_\gamma$ on inputs of length n.

For the remainder of this proof, we will set $\gamma = \frac{1}{e}$, which is an Efficiently-Approximable constant. This will give us $\mathsf{ZHB\text{-}NIZK}_{e^{-1}} = (\mathsf{ZHB.Gen}_{e^{-1}}, \mathsf{ZHB}.P_{e^{-1}}, \mathsf{ZHB}.V_{e^{-1}})$. For notational convenience, we will refer to $p_{e^{-1},\mathsf{hbs}}(n)$ simply by $p_{\mathsf{hbs}}(n)$.

We build a NIZK proof system $\mathsf{RO\text{-}NIZK} = (P^{(\cdot)}, V^{(\cdot)})$ for $\mathcal{L}$ in the RO model which satisfies *statistical completeness*. However, as shown in Remark 6.3, we can easily use this to build a NIZK proof system with *perfect completeness*.

$P^{\mathcal{O}}(x)$:

1. Let $n = |x|$.
2. For $i \in [p_{\mathsf{hbs}}(n)]$, set $y_i = \mathcal{O}_n^{\mathsf{urs}}(\mathsf{pad}_n(i))$
3. For $i \in [p_{\mathsf{hbs}}(n)]$,
   (a) If $\exists z \in \{0,1\}^n$ such that $\mathcal{O}_n^p(z) = y_i$,
   sample $v_i \leftarrow \{z \in \{0,1\}^n : \mathcal{O}_n^p(z) = y_i\}$, and set $r_i = 0$.
   (b) If $\nexists z \in \{0,1\}^n$ such that $\mathcal{O}_n^p(z) = y_i$, set $v_i = \perp$ and $r_i = 1$.
4. $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.
5. Return $\pi \coloneqq (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$.

$V^{\mathcal{O}}(x, \pi)$:

1. Let $n = |x|$, and parse $\pi = (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$.
2. For $i \in [p_{\mathsf{hbs}}(n)]$, set $y_i = \mathcal{O}_n^{\mathsf{urs}}(\mathsf{pad}_n(i))$.
3. For $i \in \mathsf{ZHB}.\mathcal{I}$,
   (a) If $v_i = \perp$, set $\tilde{r}_i = 1$.
   (b) If $v_i \neq \perp$, and $\mathcal{O}_n^p(v_i) = y_i$, set $\tilde{r}_i = 0$.
   (c) If $v_i \neq \perp$, and $\mathcal{O}_n^p(v_i) \neq y_i$, reject.
4. Return $\mathsf{ZHB}.V_{e^{-1}}(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r}_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$

□

6.2 Completeness

Lemma 6.2. *If* $\mathsf{ZHB\text{-}NIZK}_{e^{-1}}$ *satisfies perfect completeness in the* Z-Tamperable-Hidden-Bits *model, then* RO-NIZK *satisfies statistical completeness in the* RO *model. In particular, the verifier will only reject an honest prover's proof on at most a constant number of inputs.*

Proof. For sufficiently large[11] n, every possible string $r \in \{0,1\}^{p_{\mathsf{hbs}}(n)}$ can be output by $\mathsf{ZHB}.\mathsf{Gen}_{e^{-1}}(1^n)$. Thus, the perfect completeness of $\mathsf{ZHB\text{-}NIZK}_{e^{-1}}$ implies that for all sufficiently large n, all $x \in \mathcal{L}$ such that $|x| = n$, and all $r \in \{0,1\}^{p_{\mathsf{hbs}}(n)}$,

$$\Pr\left[\mathsf{ZHB}.V_{e^{-1}}(x, \mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathcal{I}}, \mathsf{ZHB}.\pi) = 1 : (\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)\right] = 1$$

Therefore, since for any random oracle $\mathcal{O}$, an honest verifier $V^{\mathcal{O}}$ always generates bits $\{\tilde{r}_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}$ that correctly correspond to the string $r \in \{0,1\}^{p_{\mathsf{hbs}}(n)}$ produced by an honest $P^{\mathcal{O}}$, then for sufficiently large n, $V^{\mathcal{O}}$ will always accept when the prover is honest. Thus, $V^{\mathcal{O}}$ will only reject an honest prover's proof on at most a constant number of inputs (those corresponding to small n). □

[11] That is, n such that $\mathsf{Approx}_{e^{-1}}(1^n) \in (0,1)$

Remark 6.3. Our NIZK proof system RO-NIZK $= (P^{(\cdot)}, V^{(\cdot)})$ only satisfies *statistical completeness*. However, as shown in the above proof, the verifier will only fail to accept an honest prover's proof on at most a constant number of inputs. Thus, we can easily construct a new NIZK proof system with *perfect completeness* without incurring any loss in soundness or zero knowledge by simply hardcoding into the verifier whether each of these constantly many values are in the language or not, and having the verifier accept or reject accordingly.

6.3 Soundness

Lemma 6.4. *If* ZHB-NIZK$_{e^{-1}}$ *satisfies adaptive statistical soundness in the* Z-Tamperable-Hidden-Bits *model then* RO-NIZK *satisfies adaptive statistical soundness in the* RO *model.*

Proof. Let $X = (X_1, \ldots, X_{p_{\mathsf{hbs}}(n)})$ where each X_i is an independent Bernoulli random variable with $\Pr[X_i = 1] = \mathsf{Approx}_{e^{-1}}(1^n)$. Then X is a random variable whose distribution is identical to the distribution of hidden bit strings output by $\mathsf{ZHB.Gen}_{e^{-1}}(1^n)$.

Similarly, let $Y = (Y_1, \ldots, Y_{p_{\mathsf{hbs}}(n)})$ where each Y_i is an independent Bernoulli random variable with $\Pr[Y_i = 1] = 1 - \delta_{\mathcal{O}}$ where $\delta_{\mathcal{O}} = \Pr_{y \leftarrow \{0,1\}^n}[\exists x \in \{0,1\}^n \text{ s.t. } \mathcal{O}^p_n(x) = y]$ and $\mathcal{O}$ is a uniformly sampled random oracle. Then, Y is a random variable whose distribution is identical to the distribution of the strings $r = r_1 \ldots r_{p_{\mathsf{hbs}}(n)}$ generated by the honest prover $P^{(\cdot)}$ when given oracle access to a uniformly sampled random oracle.

Let $D_{0,n}$ and $D_{1,n}$ denote the distributions followed by X and Y respectively. We will show that the statistical distance between $D_{0,n}$ and $D_{1,n}$ is negligible. Observe that

1. With overwhelming probability over the choice of the random oracle, $(1 - \delta_{\mathcal{O}})$ is negligibly close to e^{-1} by the $(1 - e^{-1})$-Dense property of $\mathcal{O}^p_n$.
2. $|\mathsf{Approx}_{e^{-1}}(1^n) - e^{-1}| \leq \frac{1}{2^n}$.

This implies that with overwhelming probability over the choice of the random oracle, $(1-\delta_{\mathcal{O}})$ is negligibly close to $\mathsf{Approx}_{e^{-1}}(1^n)$. Thus, the statistical distance between each X_i and Y_i is negligible. Since all the X_i's and Y_i's are mutually independent, it must be that X and Y are independent. Now, applying Lemma 3.5 yields $\Delta(D_{0,n}, D_{1,n}) \leq p_{\mathsf{hbs}(n)} \cdot \mathsf{negl}(n) = \mathsf{negl}(n)$.

Now, observe that the verifier $V^{\mathcal{O}}(x, \pi)$ either

- outputs $\mathsf{ZHB}.V_{e^{-1}}(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ for some string f and where r is the string that would be generated by an honest prover given oracle $\mathcal{O}$,
- or rejects if $v_i \neq \perp$ and $\mathcal{O}^p_n(v_i) \neq y_i$.

Assume that $V^{\mathcal{O}}$ accepts a proof $\pi = (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$. Hence, for each $i \in [p_{\mathsf{hbs}}(n)]$, the cheating prover can set $v_i = \perp$ even if $\exists z \in \{0,1\}^n$ such that $\mathcal{O}^p_n(z) = y_i$. In this way, the prover can forcefully change a '0' in r_i to a '1' in $\tilde{r}_i$. However, the prover cannot change any '1' in r_i to a '0' in $\tilde{r}_i$ since the prover cannot convince the verifier that there actually is a pre-image for y_i if there is no such pre-image. Note that this scenario is identical to the ZeroFlip function used by the cheating Z-Tamperable-Hidden-Bits prover. Thus, it suffices to show that

$$\Pr_{r \leftarrow D_{1,n}}\Big[\exists(x, \mathsf{ZHB}.\mathcal{I}, f, \mathsf{ZHB}.\pi) \text{ such that } x \notin L, |x| = n, f \in \{0,1\}^{|r|} \\ \text{and } \mathsf{ZHB}.V_{e^{-1}}(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) = 1\Big] \leq \mathsf{negl}(n).$$

Now, by the adaptive statistical soundness of $\mathsf{ZHB\text{-}NIZK}_{e^{-1}}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $n \in \mathbb{N}$,

$$\Pr_{r \leftarrow D_{0,n}} \Big[\exists (x, \mathsf{ZHB}.\mathcal{I}, f, \mathsf{ZHB}.\pi) \text{ such that } x \notin L, |x| = n, f \in \{0,1\}^{|r|}, \\ \text{and } \mathsf{ZHB}.V_{e^{-1}}(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) = 1 \Big] \leq \mathsf{negl}(n).$$

Since $D_{0,n}$ and $D_{1,n}$ are statistically close, then for all $n \in \mathbb{N}$, we also have

$$\Pr_{r \leftarrow D_{1,n}} \Big[\exists (x, \mathsf{ZHB}.\mathcal{I}, f, \mathsf{ZHB}.\pi) \text{ such that } x \notin L, |x| = n, f \in \{0,1\}^{|r|} \\ \text{and } \mathsf{ZHB}.V_{e^{-1}}(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) = 1 \Big] \leq \mathsf{negl}(n)$$

This concludes the proof.

□

6.4 Zero Knowledge

Lemma 6.5. *If* $\mathsf{ZHB\text{-}NIZK}_{e^{-1}}$ *satisfies statistical zero knowledge in the* Z-Tamperable-Hidden-Bits *model, then* RO-NIZK *satisfies zero knowledge against polynomial-query-bounded oracle Turing machines.*

Proof. Let $\mathsf{ZHB.Sim}_{e^{-1}}$ be the PPT zero knowledge simulator for $\mathsf{ZHB\text{-}NIZK}_{e^{-1}}$. We construct a PPT zero knowledge simulator $\mathsf{Sim} = (\mathsf{SimProof}, \mathsf{SimRO})$ for RO-NIZK:

- $\mathsf{SimProof}(x)$:
  1. Let $n = |x|$.
  2. **Create Registers:** Set $\mathsf{Reg}_L = \{\}$, $\mathsf{Reg}_R = \{\}$, and $\mathsf{Reg}_O = \{\}$.
  3. **Simulate the Prover:**
     (a) For $i \in [p_{\mathsf{hbs}}(n)]$, sample $y_i \leftarrow \{0,1\}^n$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.
     (b) **Abort if output value repeated:** If $\exists j \neq k$, such that $y_i = y_j$, abort.
     (c) $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB.Sim}_{e^{-1}}(x)$
     (d) For $i \in [\mathsf{ZHB}.\mathcal{I}]$,
        i. If $r_i = 0$, sample $v_i \leftarrow \{0,1\}^n \backslash \{v_k\}_{k<i}$, and add (v_i, y_i) to Reg_R.
        ii. If $r_i = 1$, set $v_i = \perp$, and add $(\perp, y_i)$ to Reg_R
     (e) Set $\mathsf{st} = (\mathsf{Reg}_L, \mathsf{Reg}_R, \mathsf{Reg}_O)$ and $\pi = (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$
     (f) Output (st, π)
- $\mathsf{SimRO}(\mathsf{st}, w)$:
  1. If w encodes an input z to oracle $\mathcal{O}_n^{\mathsf{urs}}$ and an index i (i.e. $w = (0||1^n||0||z||\mathsf{pad}_n(i))$), sample $u \leftarrow \mathsf{SimRO}_L(\mathsf{st}, z)$, and output u_i.
  2. If w encodes an input z to oracle $\mathcal{O}_n^{p}$ and an index i (i.e. $w = (1||1^n||0||z||\mathsf{pad}_n(i))$), sample $u \leftarrow \mathsf{SimRO}_R(\mathsf{st}, z)$, and output u_i.
  3. For all other oracle queries, output $\mathsf{SimO}(\mathsf{st}, w)$.
- $\mathsf{SimRO}_L(\mathsf{st}, z)$

1. Parse $\mathsf{st} = (\mathsf{Reg}_L, \mathsf{Reg}_R, \mathsf{Reg}_O)$
2. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u', return u'.
3. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$, add (z, u) to Reg_L, and return u.

- $\mathsf{SimRO}_R(\mathsf{st}, z)$
  1. Parse $\mathsf{st} = (\mathsf{Reg}_L, \mathsf{Reg}_R, \mathsf{Reg}_O)$
  2. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u', return u'.
  3. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$.
     (a) **Abort if output value repeated:** If $(z', u) \in \mathsf{Reg}_R$ for some z', abort.
     (b) Add (z, u) to Reg_R, and return u.
- $\mathsf{SimO}(\mathsf{st}, w)$:
  1. Parse $\mathsf{st} = (\mathsf{Reg}_L, \mathsf{Reg}_R, \mathsf{Reg}_O)$
  2. **Respond Consistently:** If $(w, u') \in \mathsf{Reg}_O$ for some u, return u'.
  3. **Lazy Sample:** Sample $u \leftarrow \{0,1\}$, add (w, u) to Reg_O, and return u.

Interpreting Oracle Queries. Observe that $\mathcal{P}^{\mathcal{O}}$ on inputs of length n only queries the random oracle at points corresponding to $\mathcal{O}_n^{\mathsf{urs}}$ and $\mathcal{O}_n^p$. Thus, since $\mathcal{O}$ is a random oracle, then oracle queries for inputs not to $\mathcal{O}_n^{\mathsf{urs}}$ or $\mathcal{O}_n^p$ are independent of both the honest prover's proof and the oracle's behavior with respect to $\mathcal{O}_n^{\mathsf{urs}}$ and $\mathcal{O}_n^p$. Hence, $\mathcal{O}_n^{\mathsf{urs}}$ and $\mathcal{O}_n^p$ can be assumed to be independently and uniformly distributed over all functions from n bits to n bits. For this reason, in the following hybrids, for notational simplicity, we will ignore oracle queries that are not to $\mathcal{O}_n^{\mathsf{urs}}$ or $\mathcal{O}_n^p$. Equivalently, we can view the adversary as having oracle access to both a left oracle and a right oracle, which are each functions from n bits to n bits. In the real world, the left oracle is $\mathcal{O}_n^{\mathsf{urs}}$ and the right oracle is $\mathcal{O}_n^p$. In the simulated world, the left oracle is $\mathsf{SimRO}_L(\mathsf{st}, \cdot)$ and the right oracle is $\mathsf{SimRO}_R(\mathsf{st}, \cdot)$.

We show that Sim is a zero knowledge simulator using a hybrid argument. Observe that our first hybrid represents the real world game.

$\mathbf{Hybrid}_0^{\mathcal{A}}(x)$: Real world game

1. Let $n = |x|$.
2. **Sample $\mathcal{O}_n^{\mathsf{urs}}$:** Sample random oracle $\mathcal{O}_n^{\mathsf{urs}} \leftarrow \mathsf{ROS}_n$.
3. **Sample $\mathcal{O}_n^p$:** Sample random oracle $\mathcal{O}_n^p \leftarrow \mathsf{ROS}_n$.
4. **Simulate the Prover:**
   (a) For $i \in [p_{\mathsf{hbs}}(n)]$, set $y_i = \mathcal{O}_n^{\mathsf{urs}}(\mathsf{pad}_n(i))$
   (b) For $i \in [p_{\mathsf{hbs}}(n)]$,
      i. If $\exists z \in \{0,1\}^n$ such that $\mathcal{O}_n^p(z) = y_i$,
      sample $v_i \leftarrow \{z \in \{0,1\}^n : \mathcal{O}_n^p(z) = y_i\}$, and set $r_i = 0$.
      ii. If $\nexists z \in \{0,1\}^n$ such that $\mathcal{O}_n^p(z) = y_i$, set $v_i = \perp$ and $r_i = 1$.

(c) $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.

(d) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.

5. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,
  1. Return $\mathcal{O}_n^{\mathsf{urs}}(z)$
- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,
  1. Return $\mathcal{O}_n^{p}(z)$.

Next, we make some notational changes which do not affect the output of the hybrid. In particular, we keep track of certain oracle queries and responses in registers. If the oracle is queried at a point already in a register, we simply respond accordingly. We also explicitly define the image set of $\mathcal{O}_n^p$. Observe that this hybrid is identical to the previous hybrid.

$\mathbf{Hybrid}_1^{\mathcal{A}}(x)$:

1. Let $n = |x|$.
2. **Change: Create Oracle Registers:** Set $\mathsf{Reg}_L = \{\}$ and $\mathsf{Reg}_R = \{\}$.
3. **Sample $\mathcal{O}_n^{\mathsf{urs}}$:** Sample random oracle $\mathcal{O}_n^{\mathsf{urs}} \leftarrow \mathsf{ROS}_n$.
4. **Sample $\mathcal{O}_n^{p}$:** Sample random oracle $\mathcal{O}_n^{p} \leftarrow \mathsf{ROS}_n$.
5. **Change: Compute Image:** Let $\mathsf{Image}_n = \{u \in \{0,1\}^n \mid \exists z \in \{0,1\}^n \text{ s.t. } \mathcal{O}_n^p(z) = u\}$.
6. **Simulate the Prover:**
   (a) **Change:** For $i \in [p_{\mathsf{hbs}}(n)]$, set $y_i = \mathcal{O}_n^{\mathsf{urs}}(\mathsf{pad}_n(i))$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.
   (b) For $i \in [p_{\mathsf{hbs}}(n)]$,
      i. **Change:** If $y_i \in \mathsf{Image}_n$, sample $v_i \leftarrow \{z \in \{0,1\}^n : \mathcal{O}_n^p(z) = y_i\}$, set $r_i = 0$, and add (v_i, y_i) to Reg_R.
      ii. **Change:** If $y_i \notin \mathsf{Image}_n$, set $v_i = \perp$, set $r_i = 1$, and add $(\perp, y_i)$ to Reg_R.
   (c) $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.
   (d) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.
7. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,
  1. **Change: Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u', return u'.
  2. **Change:** Set $u = \mathcal{O}_n^{\mathsf{urs}}(z)$, add (z, u) to Reg_L, and return u.
- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,

1. **Change: Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u', return u'.
2. **Change:** Set $u = \mathcal{O}_n^p(z)$, add (z, u) to Reg_R, and return u.

In this hybrid, rather than sampling $\mathcal{O}_n^{\mathsf{urs}}$ all at once, we sample it using lazy sampling.

$\mathbf{Hybrid}_2^{\mathcal{A}}(x)$:

1. Let $n = |x|$.
2. **Create Oracle Registers:** Set $\mathsf{Reg}_L = \{\}$ and $\mathsf{Reg}_R = \{\}$.
3. ~~**Sample $\mathcal{O}_n^{\mathsf{urs}}$: Sample random oracle $\mathcal{O}_n^{\mathsf{urs}} \leftarrow \mathsf{ROS}_n$.**~~
4. **Sample $\mathcal{O}_n^p$:** Sample random oracle $\mathcal{O}_n^p \leftarrow \mathsf{ROS}_n$.
5. **Compute Image:** Let $\mathsf{Image}_n = \{u \in \{0,1\}^n \mid \exists z \in \{0,1\}^n \text{ s.t. } \mathcal{O}_n^p(z) = u\}$.
6. **Simulate the Prover:**
   (a) **Change:** For $i \in [p_{\mathsf{hbs}}(n)]$, sample $y_i \leftarrow \{0,1\}^n$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.
   (b) For $i \in [p_{\mathsf{hbs}}(n)]$,
      i. If $y_i \in \mathsf{Image}_n$, sample $v_i \leftarrow \{z \in \{0,1\}^n : \mathcal{O}_n^p(z) = y_i\}$, set $r_i = 0$, and add (v_i, y_i) to Reg_R.
      ii. If $y_i \notin \mathsf{Image}_n$, set $v_i = \perp$, set $r_i = 1$, and add $(\perp, y_i)$ to Reg_R.
   (c) $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.
   (d) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.
7. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u', return u'.
  2. **Change: Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$, add (z, u) to Reg_L, and return u.
- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u', return u'.
  2. Set $u = \mathcal{O}_n^p(z)$, add (z, u) to Reg_R, and return u.

Lemma 6.6. *For all adversaries $\mathcal{A}^{(\cdot)}$ and all $x \in \mathcal{L}$,*

$$\left|\Pr\left[\mathbf{Hybrid}_1^{\mathcal{A}}(x) = 1\right] - \Pr\left[\mathbf{Hybrid}_2^{\mathcal{A}}(x) = 1\right]\right| = 0$$

Proof. The hybrids are identically distributed. The only difference between $\mathbf{Hybrid}_1^{\mathcal{A}}$ and $\mathbf{Hybrid}_2^{\mathcal{A}}$ is the way in which we compute responses for queries to the left oracle (which include the strings $y_1, \dots, y_{p_{\mathsf{hbs}}(n)}$). However, note that the distributions of these query responses are identical in both hybrids. In both cases, each distinct query z to the left oracle is mapped to an independent random output value. The register Reg_L is used to ensure that the queries are answered consistently. □

We add a simple abort condition if any of the y_i's are repeated or if there is a collision in the output values of the right oracle.

$\mathbf{Hybrid}_3^{\mathcal{A}}(x)$:

1. Let $n = |x|$.
2. **Create Oracle Registers:** Set $\mathsf{Reg}_L = \{\}$ and $\mathsf{Reg}_R = \{\}$.
3. **Sample $\mathcal{O}_n^p$:** Sample random oracle $\mathcal{O}_n^p \leftarrow \mathsf{ROS}_n$.
4. **Compute Image:** Let $\mathsf{Image}_n = \{u \in \{0,1\}^n \mid \exists z \in \{0,1\}^n \text{ s.t. } \mathcal{O}_n^p(z) = u\}$.
5. **Simulate the Prover:**
   (a) For $i \in [p_{\mathsf{hbs}}(n)]$, sample $y_i \leftarrow \{0,1\}^n$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.
   (b) **Change: Abort if output value repeated:** If $\exists j \neq k$, such that $y_i = y_j$, abort.
   (c) For $i \in [p_{\mathsf{hbs}}(n)]$,
      i. If $y_i \in \mathsf{Image}_n$, sample $v_i \leftarrow \{z \in \{0,1\}^n : \mathcal{O}_n^p(z) = y_i\}$, set $r_i = 0$, and add (v_i, y_i) to Reg_R.
      ii. If $y_i \notin \mathsf{Image}_n$, set $v_i = \perp$, set $r_i = 1$, and add $(\perp, y_i)$ to Reg_R.
   (d) $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.
   (e) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.
6. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u', return u'.
  2. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$, add (z, u) to Reg_L, and return u.
- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u, return u'.
  2. **Change:** Set $u = \mathcal{O}_n^p(z)$.
     (a) **Change: Abort if output value repeated:** If $(z', u) \in \mathsf{Reg}_R$ for some z', abort.
     (b) **Change:** Add (z, u) to Reg_R, and return u.

Lemma 6.7. *For all polynomial-query-bounded oracle Turing machines adversaries $\mathcal{A}^{(\cdot)}$ and all $x \in \mathcal{L}$,*

$$\left|\Pr\left[\mathbf{Hybrid}_2^{\mathcal{A}}(x) = 1\right] - \Pr\left[\mathbf{Hybrid}_3^{\mathcal{A}}(x) = 1\right]\right| \leq \mathsf{negl}(|x|)$$

Proof. $\mathbf{Hybrid}_2^{\mathcal{A}}$ is identical to $\mathbf{Hybrid}_3^{\mathcal{A}}$ given the latter does not abort.

Note that if an adversary makes at most $q(n)$ queries to the right oracle, then at any instance, Reg_R can have at-most $(q(n) + p_{\mathsf{hbs}}(n))$ entries. Thus, by union bound, the probability of abort in step 2a is upper bounded by $\frac{q(n)\cdot(q(n)+p_{\mathsf{hbs}}(n))}{2^n}$. After including the probability of aborting if any

$y_i = y_j$ for $i \neq j$, then we get that the probability of abort is at most

$$1 - \frac{(2^n - p_{\mathsf{hbs}}(n))^{(p_{\mathsf{hbs}}(n))}}{(2^n)^{(p_{\mathsf{hbs}}(n))}} + \frac{q(n) \cdot (q(n) + p_{\mathsf{hbs}}(n))}{2^n}$$
$$= 1 - \left(1 - \frac{p_{\mathsf{hbs}}(n)}{2^n}\right)^{(p_{\mathsf{hbs}}(n))} + \frac{q(n) \cdot (q(n) + p_{\mathsf{hbs}}(n))}{2^n} \leq \varepsilon(n),$$

for some negligible function $\varepsilon(n)$. Hence, the lemma follows. □

Rather than computing Image_n from $\mathcal{O}^p_n$, we now compute the density of the image of a randomly sampled function from ROS_n and sample an independent random Image_n with the same density. We then lazy sample our right oracle in a manner consistent with our new Image_n.

Hybrid$^{\mathcal{A}}_4(x)$:

1. Let $n = |x|$.
2. **Create Oracle Registers:** Set $\mathsf{Reg}_L = \{\}$ and $\mathsf{Reg}_R = \{\}$.
3. ~~**Sample $\mathcal{O}^p_n$:** Sample random oracle $\mathcal{O}^p_n \leftarrow \mathsf{ROS}_n$.~~
4. **Change: Compute Density:** Sample random oracle $\mathcal{O}'_n \leftarrow \mathsf{ROS}_n$.
   Let $\delta' = 2^{-n} \cdot |\{u \in \{0,1\}^n \mid \exists z \in \{0,1\}^n \text{ s.t. } \mathcal{O}'_n(z) = u\}|$.
5. **Change: Sample Image:** Sample Image_n as a random set of size $2^n \cdot \delta'$ from $\{0,1\}^n$.
6. **Simulate the Prover:**
   (a) For $i \in [p_{\mathsf{hbs}}(n)]$, sample $y_i \leftarrow \{0,1\}^n$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.
   (b) **Abort if output value repeated:** If $\exists j \neq k$, such that $y_i = y_j$, abort.
   (c) For $i \in [p_{\mathsf{hbs}}(n)]$,
      i. **Change:** If $y_i \in \mathsf{Image}_n$, sample $v_i \leftarrow \{0,1\}^n \backslash \{v_k\}_{k<i}$, set $r_i = 0$, and add (v_i, y_i) to Reg_R.
      ii. If $y_i \notin \mathsf{Image}_n$, set $v_i = \perp$, set $r_i = 1$, and add $(\perp, y_i)$ to Reg_R.
   (d) $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \ldots r_{p_{\mathsf{hbs}}(n)}$.
   (e) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.
7. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u, return u'.
  2. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$, add (z, u) to Reg_L, and return u.
- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u', return u'.

2. **Change:** Sample $u \leftarrow \mathsf{Image}_n$.
   (a) **Abort if output value repeated:** If $(z', u) \in \mathsf{Reg}_R$ for some z', abort.
   (b) Add (z, u) to Reg_R, and return u.

Lemma 6.8. *For all polynomial-query-bounded oracle Turing machines $\mathcal{A}^{(\cdot)}$ and all $x \in \mathcal{L}$,*

$$\left|\Pr\left[\mathbf{Hybrid}_3^{\mathcal{A}}(x) = 1\right] - \Pr\left[\mathbf{Hybrid}_4^{\mathcal{A}}(x) = 1\right]\right| = 0$$

Proof. Conditioned on neither hybrid aborting in step 2a, then the output distribution of $\mathbf{Hybrid}_4^{\mathcal{A}}$ is identical to that of $\mathbf{Hybrid}_3^{\mathcal{A}}$. Observe that

- The generated string $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$ is identically distributed in both the hybrids because δ' is the exact probability that a randomly chosen y_i has a pre-image under some function from n bits to n bits. As a consequence, since $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi)$ depend only on (r, x) then these values are also identically distributed in both hybrids.
- In both hybrids, whether or not each y_i has a pre-image or not is consistent with r. This is the only correlation between $y_1 \dots y_{p_{\mathsf{hbs}}(n)}$ and r.
- Conditioned on y_i having a pre-image under a random function $\mathcal{O}_n^p$, then over the choice of the random function, any value v_i is equally likely to be in the pre-image of y_i as long as v_i is not already known to be in the pre-image of some other output value and as long as no other pre-image of y_i is already known. Thus, the distributions of the v_i's in both hybrids are identical.
- If $\mathbf{Hybrid}_3^{\mathcal{A}}$ does not abort, then responses to distinct queries to the right oracle $\mathcal{O}_n^p$ are distinct elements from the image. Over the choice of the random function $\mathcal{O}_n^p$ of $\mathbf{Hybrid}_3^{\mathcal{A}}$, every element is equally likely to be in the image of any particular input. Thus, it is equivalent to sample the image first as in $\mathbf{Hybrid}_4$ and respond to distinct queries with uniformly sampled distinct elements from the image.

Finally, we observe that the probability of either hybrid aborting are identical, hence we get that the output distributions of $\mathbf{Hybrid}_3^{\mathcal{A}}$ and $\mathbf{Hybrid}_4^{\mathcal{A}}$ are identical.

□

Rather than first sampling Image_n and $\mathsf{urs} = y_1 \dots y_n$, and then setting r accordingly, we sample r and urs first and then set Image_n to be a random set which is consistent with r and urs.

$\mathbf{Hybrid}_5^{\mathcal{A}}(x)$:

1. Let $n = |x|$.
2. **Create Oracle Registers:** Set $\mathsf{Reg}_L = \{\}$ and $\mathsf{Reg}_R = \{\}$.
3. **Compute Density:** Sample random oracle $\mathcal{O}'_n \leftarrow \mathsf{ROS}_n$.
   Let $\delta' = 2^{-n} \cdot |\{u \in \{0,1\}^n \mid \exists z \in \{0,1\}^n \text{ s.t. } \mathcal{O}'_n(z) = u\}|$.
4. ~~**Sample Image:** Sample Image_n as a random set of size $2^n \cdot \delta'$ from $\{0,1\}^n$.~~
5. **Change: Create Image Set:** Set $\mathsf{Image}_n = \{\}$.
6. **Simulate the Prover:**

(a) For $i \in [p_{\mathsf{hbs}}(n)]$, sample $y_i \leftarrow \{0,1\}^n$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.

(b) **Abort if output value repeated:** If $\exists j \neq k$, such that $y_i = y_j$, abort.

(c) For $i \in [p_{\mathsf{hbs}}(n)]$,

i. **Change:** Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = (1 - \delta')$.

ii. **Change:** If $r_i = 0$, sample $v_i \leftarrow \{0,1\}^n \backslash \{v_k\}_{k<i}$, add (v_i, y_i) to Reg_R, and add y_i to Image_n.

iii. **Change:** If $r_i = 1$, set $v_i = \perp$, and add $(\perp, y_i)$ to Reg_R.

iv. **Change: Extend Image Set:** Sample a random set of size $(2^{-n} \cdot \delta' - |\mathsf{Image}_n|)$ from $\{0,1\}^n \backslash \{y_i\}_{i \in [p_{\mathsf{hbs}}(n)]}$ and add it to Image_n.

(d) $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.

(e) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.

7. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,

1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u', return u'.
2. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$, add (z, u) to Reg_L, and return u.

- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,

1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u', return u'.
2. Sample $u \leftarrow \mathsf{Image}_n$.
   (a) **Abort if output value repeated:** If $(z', u) \in \mathsf{Reg}_R$ for some z', abort.
   (b) Add (z, u) to Reg_R, and return u.

Lemma 6.9. *For all adversaries $\mathcal{A}^{(\cdot)}$ and all $x \in \mathcal{L}$,*

$$\left|\Pr\left[\mathbf{Hybrid}_4^{\mathcal{A}}(x) = 1\right] - \Pr\left[\mathbf{Hybrid}_5^{\mathcal{A}}(x) = 1\right]\right| = 0$$

Proof. The output distribution of $\mathbf{Hybrid}_5^{\mathcal{A}}$ is identical to that of $\mathbf{Hybrid}_4^{\mathcal{A}}$. Here we just exchange the order in which we sample r, $\mathsf{urs} = y_1 \dots y_{p_{\mathsf{hbs}}(n)}$, and Image_n. Observe that

- The distributions of the r_i's remain unchanged since δ' is the exact probability that a randomly chosen y_i is in Image_n.
- Conditioned on any fixed r, the image set Image_n of $\mathbf{Hybrid}_4^{\mathcal{A}}$ is distributed uniformly over all image sets which are consistent with r. Thus, it is equivalent to sample r first and then sample Image_n as in $\mathbf{Hybrid}_5^{\mathcal{A}}$.
- In both hybrids, whether or not each y_i has a pre-image or not is consistent with r and Image_n.

Thus, since all other output values are generated from r, $y_1 \dots y_{p_{\mathsf{hbs}}(n)}$, and Image_n in the same way in both hybrids, then the hybrids have identical output distributions. □

We observe that we no longer need to keep track of Image_n since it is equivalent to simply lazy sample in a manner consistent with our y_i's and the previous queries.

$\mathbf{Hybrid}_6^{\mathcal{A}}(x)$:

1. Let $n = |x|$.
2. **Create Oracle Registers:** Set $\mathsf{Reg}_L = \{\}$ and $\mathsf{Reg}_R = \{\}$.
3. **Compute Density:** Sample random oracle $\mathcal{O}'_n \leftarrow \mathsf{ROS}_n$.
   Let $\delta' = 2^{-n} \cdot |\{u \in \{0,1\}^n \mid \exists z \in \{0,1\}^n \text{ s.t. } \mathcal{O}'_n(z) = u\}|$.
4. ~~**Create Image Set:** Set $\mathsf{Image}_n = \{\}$.~~
5. **Simulate the Prover:**
   (a) For $i \in [p_{\mathsf{hbs}}(n)]$, sample $y_i \leftarrow \{0,1\}^n$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.
   (b) **Abort if output value repeated:** If $\exists j \neq k$, such that $y_i = y_j$, abort.
   (c) For $i \in [p_{\mathsf{hbs}}(n)]$,
      i. Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = \delta'$.
      ii. If $r_i = 0$, sample $v_i \leftarrow \{0,1\}^n \backslash \{v_k\}_{k<i}$,
      add (v_i, y_i) to Reg_R, ~~and add y_i to Image_n.~~
      iii. If $r_i = 1$, set $v_i = \perp$, and add $(\perp, y_i)$ to Reg_R
      iv. ~~**Extend Image Set:** Sample a random set of size $(2^{-n} \cdot \delta' - |\mathsf{Image}_n|)$ from $\{0,1\}^n \backslash \{y_i\}_{i \in [p_{\mathsf{hbs}}(n)]}$ and add it to Image_n.~~
   (d) $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.
   (e) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.
6. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u, return u'.
  2. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$, add (z, u) to Reg_L, and return u.
- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u', return u'.
  2. **Change: Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$.
     (a) **Abort if output value repeated:** If $(z', u) \in \mathsf{Reg}_R$ for some z', abort.
     (b) Add (z, u) to Reg_R, and return u.

Lemma 6.10. *For all adversaries $\mathcal{A}^{(\cdot)}$ and all $x \in \mathcal{L}$,*

$$\left|\Pr\left[\mathbf{Hybrid}_5^{\mathcal{A}}(x) = 1\right] - \Pr\left[\mathbf{Hybrid}_6^{\mathcal{A}}(x) = 1\right]\right| = 0$$

Proof. The output distribution of $\mathbf{Hybrid}_6^{\mathcal{A}}$ is identical to that of $\mathbf{Hybrid}_5^{\mathcal{A}}$. The only difference between the two hybrids is that in $\mathbf{Hybrid}_6^{\mathcal{A}}$ we no longer extend and completely define the image set Image_n as in Step 6(c)iv in $\mathbf{Hybrid}_5^{\mathcal{A}}$. This impacts the simulation in only one way: upon oracle queries to the right oracle, u is now sampled randomly from $\{0,1\}^n$ rather than Image_n.

Observe that the joint distribution of $(y_i\}_{i \in [p_{\mathsf{hbs}}(n)]})$ and the oracle responses have identical distribution in both hybrids. This gives us the desired result.

□

We now use the $(1-e^{-1})$-Dense property of the random oracle to exchange the density $(1-\delta')$ with $(1-\mathsf{Approx}_{e^{-1}}(1^n))$.

$\mathbf{Hybrid}_7^{\mathcal{A}}(x)$:

1. Let $n = |x|$.
2. **Create Oracle Registers:** Set $\mathsf{Reg}_L = \{\}$ and $\mathsf{Reg}_R = \{\}$.
3. ~~**Compute Density:** Sample random oracle $\mathcal{O}'_n \leftarrow \mathsf{ROS}_n$.~~
   ~~Let $\delta' = 2^{-n} \cdot |\{u \in \{0,1\}^n \mid \exists z \in \{0,1\}^n \text{ s.t. } \mathcal{O}'_n(z) = u\}|$.~~
4. **Simulate the Prover:**
   (a) For $i \in [p_{\mathsf{hbs}}(n)]$, sample $y_i \leftarrow \{0,1\}^n$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.
   (b) **Abort if output value repeated:** If $\exists j \neq k$, such that $y_i = y_j$, abort.
   (c) For $i \in [p_{\mathsf{hbs}}(n)]$,
      i. **Change:** Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = (1 - \mathsf{Approx}_{e^{-1}}(1^n))$.
      ii. If $r_i = 0$, sample $v_i \leftarrow \{0,1\}^n \backslash \{v_k\}_{k<i}$, and add (v_i, y_i) to Reg_R.
      iii. If $r_i = 1$, set $v_i = \perp$, and add $(\perp, y_i)$ to Reg_R.
   (d) $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.
   (e) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.
5. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u, return u'.
  2. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$, add (z, u) to Reg_L, and return u.
- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u', return u'.
  2. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$.
     (a) **Abort if output value repeated:** If $(z', u) \in \mathsf{Reg}_R$ for some z', abort.
     (b) Add (z, u) to Reg_R, and return u.

Lemma 6.11. *For all adversaries $\mathcal{A}^{(\cdot)}$ and all $x \in \mathcal{L}$,*

$$\left|\Pr\left[\mathbf{Hybrid}_6^{\mathcal{A}}(x) = 1\right] - \Pr\left[\mathbf{Hybrid}_7^{\mathcal{A}}(x) = 1\right]\right| \leq \mathsf{negl}(|x|)$$

Proof. The only difference between the two hybrids is the way in which r is sampled. By Lemma B.3, a random function from n bits to n bits is $(1-e^{-1})$-Dense with overwhelming probability. Thus, with overwhelming probability, the computed density δ' is negligibly close to $(1-e^{-1})$ which is in turn negligibly close to $(1 - \mathsf{Approx}_{e^{-1}}(1^n))$. Therefore, the distribution of each r_i in $\mathbf{Hybrid}_6^{\mathcal{A}}$ is negligibly close to the distribution of each r_i in $\mathbf{Hybrid}_7^{\mathcal{A}}$. Thus, by Lemma A.1, the statistical

distance between the two distributions of r in the two hybrids is at most $p_{\mathsf{hbs}}(n) \cdot \mathsf{negl}(n)$ which is negligible. Therefore, the output distributions of the two hybrids are statistically close.

□

Finally, we replace the generation of r and $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi)$ with the ZHB simulator.

$\mathbf{Hybrid}_8^{\mathcal{A}}(x)$:

1. Let $n = |x|$.
2. **Create Oracle Registers:** Set $\mathsf{Reg}_L = \{\}$ and $\mathsf{Reg}_R = \{\}$.
3. **Simulate the Prover:**
   (a) For $i \in [p_{\mathsf{hbs}}(n)]$, sample $y_i \leftarrow \{0,1\}^n$, and add $(\mathsf{pad}_n(i), y_i)$ to Reg_L.
   (b) **Abort if output value repeated:** If $\exists j \neq k$, such that $y_i = y_j$, abort.
   (c) **Change:** $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB.Sim}_{e^{-1}}(x)$
   (d) **Change:** For $i \in [\mathsf{ZHB}.\mathcal{I}]$,
      i. ~~Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = e^{-1}$.~~
      ii. If $r_i = 0$, sample $v_i \leftarrow \{0,1\}^n \backslash \{v_k\}_{k<i}$, and add (v_i, y_i) to Reg_R.
      iii. If $r_i = 1$, set $v_i = \perp$, and add $(\perp, y_i)$ to Reg_R
   (e) ~~$(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_{e^{-1}}(r, x)$ where $r = r_1 \ldots r_{p_{\mathsf{hbs}}(n)}$.~~
   (f) Send $\pi := (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}^{(\cdot)}$.
4. Answer $\mathcal{A}^{(\cdot)}$'s oracle queries as described below, and output whatever $\mathcal{A}^{(\cdot)}$ outputs.

- **Left Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the left oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_L$ for some u, return u'.
  2. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$, add (z, u) to Reg_L, and return u.
- **Right Oracle:** When $\mathcal{A}^{(\cdot)}$ makes a query z to the right oracle,
  1. **Respond Consistently:** If $(z, u') \in \mathsf{Reg}_R$ for some u', return u'.
  2. **Lazy Sample:** Sample $u \leftarrow \{0,1\}^n$.
     (a) **Abort if output value repeated:** If $(z', u) \in \mathsf{Reg}_R$ for some z', abort.
     (b) Add (z, u) to Reg_R, and return u.

Lemma 6.12. *If* $\mathsf{ZHB\text{-}NIZK}_{e^{-1}}$ *satisfies statistical zero knowledge in the* Z-Tamperable-Hidden-Bits *model, then for all polynomial-query-bounded oracle Turing machines* $\mathcal{A}^{(\cdot)}$ *and all* $x \in \mathcal{L}$,

$$\left|\Pr\left[\mathbf{Hybrid}_7^{\mathcal{A}}(x) = 1\right] - \Pr\left[\mathbf{Hybrid}_8^{\mathcal{A}}(x) = 1\right]\right| \leq \mathsf{negl}(|x|)$$

Proof. This follows by a straightforward reduction to the statistical zero knowledge of $\mathsf{ZHB\text{-}NIZK}_{e^{-1}}$. Suppose there exists a polynomial-query-bounded oracle Turing machines $\mathcal{A}$ that could distinguish between the two hybrids with non-negligible advantage on some input $x \in \mathcal{L}$. We build an adversary $\mathcal{B}$ that breaks the statistical zero knowledge of $\mathsf{ZHB\text{-}NIZK}_{e^{-1}}$. $\mathcal{B}$ is given $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ which are either generated by the honest prover and hidden bit string generator or generated by the

ZHB simulator. $\mathcal{B}$ sets $n = |x|$, creates oracle registers Reg_L and Reg_R, samples $\mathsf{urs} = y_1 \dots y_{\mathsf{hbs}(n)}$ as in both hybrids, and aborts if any $y_j = y_k$ for any $j \neq k$. $\mathcal{B}$ then samples $\{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}$ as in both hybrids and adds values (v_i, y_i) for $i \in \mathsf{ZHB}.\mathcal{I}$ to Reg_R (as in step 3(d)ii and step 3(d)iii of $\mathbf{Hybrid}_8$). $\mathcal{B}$ then sends $\pi = (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ to $\mathcal{A}$. $\mathcal{B}$ responds to $\mathcal{A}$'s queries to the left and right oracles as in both hybrids. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. Observe that if $\mathcal{B}$ received an input generated by $\mathsf{ZHB}.\mathsf{Sim}_{e^{-1}}(x)$, then $\mathcal{B}$ exactly emulates $\mathbf{Hybrid}_8$. If $\mathcal{B}$ received an input generated by $\mathsf{ZHB}.\mathsf{Gen}_{e^{-1}}(1^{|x|})$ and $\mathsf{ZHB}.P_{e^{-1}}(x)$, then $\mathcal{B}$ emulates a hybrid which we call $\mathbf{Hybrid}^{\mathcal{A}}_{7,2}$.

Observe that $\mathbf{Hybrid}^{\mathcal{A}}_{7,2}$ is identical to $\mathbf{Hybrid}^{\mathcal{A}}_{7}$ except that $\mathbf{Hybrid}^{\mathcal{A}}_{7}$ may add up to $p_{\mathsf{hbs}}(n)$ additional values of the form (v_i, y_i) for either randomly picked v_i or $v_i = \perp$ to Reg_R. However, as long as the following conditions hold, then the output distributions of $\mathbf{Hybrid}^{\mathcal{A}}_{7}$ and $\mathbf{Hybrid}^{\mathcal{A}}_{7_2}$ are identical:

- The adversary in $\mathbf{Hybrid}^{\mathcal{A}}_{7,2}$ does not query its right oracle at the additional values v_i that would have been sampled in $\mathbf{Hybrid}^{\mathcal{A}}_{7}$. Observe that as these additional v_i values are independent, random distinct elements, then the probability that $\mathcal{A}$ queries one of these additional v_i's after making $q(n)$ queries is bounded by $\frac{q(n)p_{\mathsf{hbs}}(n)}{2^n - p_{\mathsf{hbs}}(n)}$ which is negligible.

- The challenger in $\mathbf{Hybrid}^{\mathcal{A}}_{7,2}$ never samples a response u that corresponds to some y_i when responding to queries to the right oracle. Observe that if the adversary makes at most $q(n)$ queries, then the probability that this event occurs is bounded by $\frac{q(n)p_{\mathsf{hbs}}(n)}{2^n}$ which is negligible.

Thus, the distingiushing advantage is negligible. □

We observe that our final hybrid matches the simulator exactly where the left oracle is SimRO_L and the right oracle is SimRO_R. Since all the intermediate hybrids are indistinguishable to *polynomial-query-bounded* oracle Turing machines, we get our desired result.

□

7 NIZK Proofs for NP in the URS Model from δ-Dense-PRHFs

We now build NIZK proofs in the uniform reference string (URS) model using a δ-$\mathsf{Dense\text{-}PRHF}$. We prove the following:

Theorem 7.1. *If there exists a* δ-$\mathsf{Dense\text{-}PRHF}$ *for some* $\mathsf{Efficiently\text{-}Approximable}$ *constant* $\delta \in (0, 1)$*, then there exists an (unbounded-prover)* NIZK *proof system for* NP *in the* URS *model.*

Remark 7.2. Recall that a random oracle, when interpreted as a function from n bits to n bits, satisfies all of the properties of a $\left(1 - \frac{1}{e}\right)$-$\mathsf{Dense\text{-}PRHF}$ in the random oracle model (see Remark 4.7 and Appendix B). Indeed, our construction of NIZK proofs in the RO model is the same as in this construction except that we instantiate the δ-$\mathsf{Dense\text{-}PRHF}$ H_{PRHF} for $\delta = \left(1 - \frac{1}{e}\right)$ with a random oracle and generate the urs using the random oracle. However, for technical reasons, which we explain in Remark 7.6, the proof of zero knowledge in this construction does not necessarily imply zero knowledge in the RO model. For this reason, in the previous section, we separately proved how to construct NIZK proofs in the RO model. The zero knowledge proof in that construction bypasses the issue by allowing the simulator to program the random oracle.

Proof of Theorem 7.1. We first provide a brief proof overview.

Proof Overview. In a previous section, we showed how to build NIZK proofs in the Z-Tamperable-Hidden-Bits model. To build a NIZK proof in the URS model, we will instantiate the hidden bit string functionality of the Z-Tamperable-Hidden-Bits model by determining a hidden bit string r from the $\mathsf{urs} = y_1 \dots y_{p_{\mathsf{hbs}}(n)}$ and the δ-Dense-PRHF H_{PRHF} in the following way: We set each bit r_i of the hidden bit string equal to 1 if and only if y_i does not contain a pre-image under H_{PRHF}. Since H_{PRHF} is δ-Dense, then r_i will be sampled similarly to our ZHB hidden bit string generator $\mathsf{ZHB.Gen}_\delta(1^n)$. Observe that the prover can reveal that $r_i = 0$ by sending over a pre-image v_i for y_i (which the verifier can check), and can claim that $r_i = 1$ by simply stating that "there is no pre-image".

For soundness, observe that just like in the Z-Tamperable-Hidden-Bits model, the prover can lie about r_i whenever $r_i = 0$, but cannot lie about r_i whenever $r_i = 1$ because the prover cannot send over a pre-image when no such pre-image exists. Thus, the soundness reduces to the soundness of the Z-Tamperable-Hidden-Bits model construction.

For zero knowledge, the simulator will first generate a $\mathsf{urs} = y_1 \dots y_{p_{\mathsf{hbs}}(n)}$ in such a way that the simulator knows a pre-image v_i under H_{PRHF} for every y_i. We use the pseudorandomness and pre-image pseudorandomness properties of H_{PRHF} to argue that the urs sampled by the simulator is indistinguishable from a random urs. Then, the simulator will run the ZHB simulator on x to get a simulated ZHB proof and simulated hidden bits. The simulator can open the corresponding bits of r to the correct values since the simulator knows a pre-image for every y_i in the urs. Zero knowledge then reduces to the security of the ZHB simulator.

7.1 Construction

Let $\mathcal{L}$ be an NP language and let H_{PRHF} be a δ-Dense-PRHF for some Efficiently-Approximable constant $\delta \in (0, 1)$ with stretch function $\ell : \mathbb{N} \to \mathbb{N}$. By Theorem 5.1, there exists a NIZK proof system $\{\mathsf{ZHB\text{-}NIZK}_\gamma = (\mathsf{ZHB.Gen}_\gamma, \mathsf{ZHB}.P_\gamma, \mathsf{ZHB}.V_\gamma)\}$ for $\mathcal{L}$ in the Z-Tamperable-Hidden-Bits. Let $p_{\gamma,\mathsf{hbs}}(n)$ be the length of the hidden bit string of $\mathsf{ZHB\text{-}NIZK}_\gamma$ on inputs of length n.

For the remainder of this proof, we will set $\gamma = 1 - \delta$, which is an Efficiently-Approximable constant. For notational convenience, we will refer to $p_{\gamma,\mathsf{hbs}}(n)$ simply by $p_{\mathsf{hbs}}(n)$.

We build a NIZK proof system $\mathsf{URS\text{-}NIZK} = (\mathsf{URS.Gen}, \mathsf{URS}.P, \mathsf{URS}.V)$ for $\mathcal{L}$ in the URS model which satisfies *statistical completeness*. However, similarly to Remark 6.3, we can easily use this to build a NIZK proof system with *perfect completeness* without incurring any loss in soundness or zero knowledge.

$\mathsf{URS.Gen}(1^n)$:

1. For $i \in [p_{\mathsf{hbs}}(n)]$, sample a uniformly random $y_i \in \{0,1\}^{\ell(n)}$.
2. Output $\mathsf{urs} = y_1 \dots y_{p_{\mathsf{hbs}}(n)}$.

$\mathsf{URS}.P(\mathsf{urs}, x)$:

1. Parse urs as $y_1, \dots, y_{p_{\mathsf{hbs}}(n)}$ where $n = |x|$.
2. For $i \in [p_{\mathsf{hbs}}(n)]$,
   (a) If $\exists z \in \{0,1\}^n$ such that $H_{\mathsf{PRHF}}(z) = y_i$,
   sample $v_i \leftarrow \{z \in \{0,1\}^n : H_{\mathsf{PRHF}}(z) = y_i\}$, and set $r_i = 0$.

(b) If $\nexists z \in \{0,1\}^n$ such that $H_{\mathsf{PRHF}}(z) = y_i$, set $v_i = \perp$ and $r_i = 1$.

3. $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.

4. Return $\pi \coloneqq (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$.

$\mathsf{URS}.V(\mathsf{urs}, x, \pi)$:

1. Parse urs as $y_1, \dots, y_{p_{\mathsf{hbs}}(n)}$ where $n = |x|$.

2. Parse π as $(\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$.

3. For $i \in \mathsf{ZHB}.\mathcal{I}$,

   - If $v_i = \perp$, set $\tilde{r}_i = 0$.
   - If $v_i \neq \perp$, and $H_{\mathsf{PRHF}}(v_i) = y_i$, set $\tilde{r}_i = 1$.
   - If $v_i \neq \perp$, and $H_{\mathsf{PRHF}}(v_i) \neq y_i$, reject.

4. Return $\mathsf{ZHB}.V_\gamma(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r}_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$

□

7.2 Completeness

Lemma 7.3. *If H_{PRHF} is a δ-Dense-PRHF for some Efficiently-Approximable constant $\delta \in (0,1)$ and ZHB-NIZK$_\gamma$ with $\gamma = 1 - \delta$ satisfies perfect completeness in the Z-Tamperable-Hidden-Bits model, then URS-NIZK satisfies statistical completeness in the URS model. In particular, the verifier will only reject an honest prover's proof on at most a constant number of inputs.*

Proof. This proof is nearly identical to the proof of completeness in the RO model (Lemma 6.2). Thus, we defer the proof to Appendix C.2 □

7.3 Soundness

Lemma 7.4. *If H_{PRHF} is a δ-Dense-PRHF for some Efficiently-Approximable constant $\delta \in (0,1)$ and ZHB-NIZK$_\gamma$ with $\gamma = 1 - \delta$ satisfies adaptive statistical soundness in the Z-Tamperable-Hidden-Bits model, then URS-NIZK satisfies adaptive statistical soundness in the URS model.*

Proof. This proof is nearly identical to the proof of soundness in the RO model (Lemma 6.4) . Thus, we defer the proof to Appendix C.2 □

7.4 Zero Knowledge

Lemma 7.5. *If H_{PRHF} is a δ-Dense-PRHF for some Efficiently-Approximable constant $\delta \in (0,1)$ and ZHB-NIZK$_\gamma$ with $\gamma = 1 - \delta$ satisfies statistical zero knowledge in the Z-Tamperable-Hidden-Bits model, then URS-NIZK satisfies computational zero knowledge in the URS model.*

Remark 7.6. For technical reasons, the proof of zero knowledge in this construction does not necessarily imply zero knowledge in the RO model if we simply instantiate the δ-Dense-PRHF H_{PRHF} with a random oracle and generate the urs using the random oracle. This is because the proof of zero knowledge here uses non-uniform reductions which contain advice dependent

on H_{PRHF}. Thus, if we try to proceed with the proof using a random oracle instead of H_{PRHF}, then we require our random oracle to satisfy both *pseudorandomness* and *pre-image pseudorandomness* (as per Def 4.6) against *oracle-dependent* adversaries. While a random oracle is $\left(1-\frac{1}{e}\right)$-$\mathsf{Dense}$ (see Remark 4.7 and Appendix B) and also satisfies standard pseudorandomness against oracle-dependent adversaries [DGK17, GGKT05], it is unclear how to prove *pre-image pseudorandomness* against such adversaries (though we conjecture it to be true and leave the proof for future work).

Proof of Lemma 7.5. Let $\mathsf{ZHB.Sim}_\gamma$ be the PPT zero knowledge simulator for $\mathsf{ZHB\text{-}NIZK}_\gamma$. We construct a PPT zero knowledge simulator $\mathsf{URS.Sim}$ for $\mathsf{URS\text{-}NIZK}$:

$\mathsf{URS.Sim}(x)$:

1. Let $n = |x|$.
2. $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i\in\mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB.Sim}_\gamma(x)$.
3. For $i \in [p_{\mathsf{hbs}}(n)]$, sample $v_i \leftarrow \{0,1\}^n$, and set $y_i = H_{\mathsf{PRHF}}(v_i)$.
4. For $i \in \mathsf{ZHB}.\mathcal{I}$,
   (a) If $r_i = 1$, set $v'_i = \perp$
   (b) If $r_i = 0$, set $v'_i = v_i$.
5. Output $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i\in\mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$.

We will now prove that the output of Sim is computationally indistinguishable from the joint distribution of the real urs and an honest prover's proof. We show this via a hybrid argument. Observe that $\mathbf{Hybrid}_0$ represents the real world experiment.

$\mathbf{Hybrid}_0(x)$:

1. Let $n = |x|$.
2. For $i \in [p_{\mathsf{hbs}}(n)]$,
   (a) Sample $y_i \leftarrow \{0,1\}^{\ell(n)}$.
   (b) If $\exists z \in \{0,1\}^n$ such that $H_{\mathsf{PRHF}}(z) = y_i$,
   and sample $v_i \leftarrow \{z \in \{0,1\}^n : H_{\mathsf{PRHF}}(z) = y_i\}$, and set $r_i = 0$.
   (c) If $\nexists z \in \{0,1\}^n$ such that $H_{\mathsf{PRHF}}(z) = y_i$, set $v_i = \perp$ and $r_i = 1$.
3. $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.
4. Output $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i\in\mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$.

For our next hybrid, we sample r_i first and use it to determine v_i and y_i.

$\mathbf{Hybrid}_1(x)$:

1. Let $n = |x|$.
2. For $i \in [p_{\mathsf{hbs}}(n)]$,
   (a) **Change:** Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = \gamma'$

where $\gamma' = 1 - \Pr_{y \leftarrow \{0,1\}^{\ell(n)}}[\exists z \text{ s.t. } H_{\mathsf{PRHF}}(z) = y]$

3. $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.

4. **Change:** Define $\mathsf{Image}_n = \{y \in \{0,1\}^{\ell(n)} : \exists z \in \{0,1\}^n,\ y = H_{\mathsf{PRHF}}(z)\}$ and $\mathsf{Pre\text{-}Image}_n(y) = \{z \in \{0,1\}^n : H_{\mathsf{PRHF}}(z) = y\}$

5. **Change:** For $i \in [p_{\mathsf{hbs}}(n)]$,

   (a) **Change:** If $r_i = 0$, sample $y_i \leftarrow \mathsf{Image}_n$ and $v_i \leftarrow \mathsf{Pre\text{-}Image}_n(y_i)$, and set $v_i' = v_i$.

   (b) **Change:** If $r_i = 1$, sample $y_i \leftarrow \{0,1\}^{\ell(n)} \backslash \mathsf{Image}_n$, and set $v_i' = \bot$.

6. **Change:** Output $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v_i'\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$.

Lemma 7.7. *For all adversaries $\mathcal{A}$ and all $x \in \mathcal{L}$,*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_0(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_1(x)) = 1]| = 0$$

Proof. The hybrids are identically distributed. Observe that in $\mathbf{Hybrid}_0$, the following properties hold:

- The distribution of $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi)$ only depends on r and x.
- For any $i \neq j$, the distribution of (r_i, v_i, y_i) is independently and identically distributed to the distribution of (r_j, v_j, y_j).
- For any i, $\Pr[r_i = 1] = \Pr_{y \leftarrow \{0,1\}^{\ell(n)}}[\nexists z \text{ s.t. } H_{\mathsf{PRHF}}(z) = y]$
- For any i, conditioned on $r_i = 0$, y_i is uniformly distributed over all elements in the image of H_{PRHF}, and v_i is uniformly distributed over all pre-images of y_i.
- For any i, conditioned on $r_i = 1$, y_i is uniformly distributed over all elements that are not in the image of H_{PRHF}, and $v_i = \bot$.

Thus, it is equivalent to first sample each r_i from the correct distribution and then sample (y_i, v_i) according to the induced conditional probabilities. As $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi)$ only depends on r, as long as r has the correct distribution, then the final output distribution $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v_i'\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$ will be distributed the same as before. □

We now use the pre-image pseudorandomness of H_{PRHF} to change our method of sampling when $r_i = 0$.

$\mathbf{Hybrid}_2(x)$:

1. Let $n = |x|$.

2. For $i \in [p_{\mathsf{hbs}}(n)]$,

   (a) Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = \gamma'$ where $\gamma' = 1 - \Pr_{y \leftarrow \{0,1\}^{\ell(n)}}[\exists z \text{ s.t. } H_{\mathsf{PRHF}}(z) = y]$

3. $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.

4. Define $\mathsf{Image}_n = \{y \in \{0,1\}^{\ell(n)} : \exists z \in \{0,1\}^n,\ y = H_{\mathsf{PRHF}}(z)\}$ and $\mathsf{Pre\text{-}Image}_n(y) = \{z \in \{0,1\}^n : H_{\mathsf{PRHF}}(z) = y\}$

5. For $i \in [p_{\mathsf{hbs}}(n)]$

   (a) **Change:** If $r_i = 0$, sample $v_i \leftarrow \{0,1\}^n$, set $y_i = H_{\mathsf{PRHF}}(v_i)$, and set $v'_i = v_i$

   (b) If $r_i = 1$, sample $y_i \leftarrow \{0,1\}^{\ell(n)} \backslash \mathsf{Image}_n$, and set $v'_i = \perp$.

6. Output $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$.

Lemma 7.8. *If H_{PRHF} is a δ-*Dense-PRHF *for some* Efficiently-Approximable *constant $\delta \in (0,1)$, then for all non-uniform polynomial-sized adversaries $\mathcal{A}$ and all $x \in \mathcal{L}$,*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_1(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_2(x)) = 1]| \leq \mathsf{negl}(|x|)$$

Proof. We define a series of intermediate hybrids. Let $\mathbf{Hybrid}_{1,j}$ be the same as $\mathbf{Hybrid}_1$ except that in step 5a, for $i \leq j$, if $r_i = 0$. we sample (y_i, v_i, v'_i) according to $\mathbf{Hybrid}_2$ by sampling $v_i \leftarrow \{0,1\}^n$, setting $y_i = H_{\mathsf{PRHF}}(v_i)$, and setting $v'_i = v_i$. Observe that $\mathbf{Hybrid}_{1,0} = \mathbf{Hybrid}_1$ and that $\mathbf{Hybrid}_{1,p_{\mathsf{hbs}}(n)} = \mathbf{Hybrid}_2$.

Now, suppose for sake of contradiction that there exists an $x \in \mathcal{L}$ and a non-uniform polynomial-sized adversary that can distinguish between the output of $\mathbf{Hybrid}_1(x)$ and $\mathbf{Hybrid}_2(x)$ with greater than $\mathsf{negl}(|x|)$ probability. Then, there exist an $x \in \mathcal{L}$, $j \in [p_{\mathsf{hbs}}(|x|)]$, a non-negligible function $\mu(\cdot)$, and a non-uniform polynomial-sized adversary $\mathcal{A}$ such that

$$\left|\Pr[\mathcal{A}(\mathbf{Hybrid}_{1,j-1}(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_{1,j}(x)) = 1]\right| > \frac{\mu(|x|)}{p_{\mathsf{hbs}}(|x|)}$$

We build a non-uniform polynomial-sized adversary $\mathcal{B}$ that breaks the pre-image pseudorandomness of H_{PRHF}. $\mathcal{B}$ non-uniformly fixes the values of $(r, \mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi)$ computed as in steps 1-3 of $\mathbf{Hybrid}_{1,j-1}$ and $\mathbf{Hybrid}_{1,j}$ and the values of $\{(y_i, v_i, v'_i)\}_{i \neq j}$ computed as in step 5 of $\mathbf{Hybrid}_{1,j-1}$ and $\mathbf{Hybrid}_{1,j}$ which maximize the distinguishing advantage of $\mathcal{A}$. We assume that the non-uniformly fixed $r_j = 0$, as otherwise the hybrids are identical and $\mathcal{A}$'s distinguishing advantage is 0. Then, $\mathcal{B}$ receives (v^*, y^*), where (v^*, y^*) is sampled from either $D_0(1^{|x|})$ or $D_1(1^{|x|})$ as defined according to Definition 4.6. $\mathcal{B}$ sets $v'_j = v^*$ and $y_j = y^*$, and sends $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(|x|)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$ to $\mathcal{A}$. $\mathcal{B}$ then outputs whatever $\mathcal{A}$ outputs. Observe that if (v^*, y^*) is sampled from $D_0(1^{|x|})$, then $\mathcal{B}$ exactly emulates $\mathbf{Hybrid}_{1,j}$, and if (v^*, y^*) is sampled from $D_1(1^{|x|})$, then $\mathcal{B}$ exactly emulates $\mathbf{Hybrid}_{1,j-1}$. Thus, since $\mathcal{B}$ non-uniformly fixed values that maximized $\mathcal{A}$'s distinguishing probability, then $\mathcal{B}$ distinguishes between $D_0(1^{|x|})$ and $D_1(1^{|x|})$ with at least $\frac{\mu(|x|)}{p_{\mathsf{hbs}}(|x|)}$ probability. Thus, $\mathcal{B}$ breaks the pre-image pseudorandomness of H_{PRHF}. □

We will now change our method of sampling when $r_i = 1$. First, we prove that δ-Dense-PRHFs are psuedorandom with respect to the flat distribution of all elements that are in the co-domain, but not in the image.

Lemma 7.9. *If H_{PRHF} is a δ-*Dense-PRHF *for some* Efficiently-Approximable *constant $\delta \in (0,1)$, then there exists a negligible function $\varepsilon_{\mathsf{NPIPR}}$ such that for all non-uniform polynomial-sized adversaries $\mathcal{A}$ and all $n \in \mathbb{N}$,*

$$\left|\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(1^n, H_{\mathsf{PRHF}}(x)) = 1] - \Pr_{y \leftarrow \{0,1\}^{\ell(n)} \backslash \mathsf{Image}_n}[\mathcal{A}(1^n, y) = 1]\right| \leq \varepsilon_{\mathsf{NPIPR}}(n)$$

where we define $\mathsf{Image}_n = \{y \in \{0,1\}^{\ell(n)} : \exists x \in \{0,1\}^n,\ y = H_{\mathsf{PRHF}}(x)\}$

Proof. Let $n \in \mathbb{N}$, and let $\mathcal{A}$ be any non-uniform polynomial-sized adversary. Let $\mathsf{Image}_n = \{y \in \{0,1\}^{\ell(n)} : \exists x \in \{0,1\}^n,\ y = H_{\mathsf{PRHF}}(x)\}$. Now,

$$\begin{aligned}\Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\mathcal{A}(1^n, y) = 1] = &\Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\mathcal{A}(1^n, y) = 1 \mid y \in \mathsf{Image}_n] \Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[y \in \mathsf{Image}_n] \\ &+ \Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\mathcal{A}(1^n, y) = 1 \mid y \notin \mathsf{Image}_n] \Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[y \notin \mathsf{Image}_n]\end{aligned}$$

Now,

- By the δ-Dense property of H_{PRHF},

$$\Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[y \in \mathsf{Image}_n] = \delta \pm \mathsf{negl}(n)$$

- By the pseudorandom property of H_{PRHF},

$$\Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\mathcal{A}(1^n, y) = 1] = \Pr_{x\leftarrow\{0,1\}^{n}}[\mathcal{A}(1^n, H_{\mathsf{PRHF}}(x)) = 1] \pm \mathsf{negl}(n)$$

- By the pre-image psuedorandom property of H_{PRHF},

$$\Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\mathcal{A}(1^n, y) = 1 \mid y \in \mathsf{Image}_n] = \Pr_{x\leftarrow\{0,1\}^{n}}[\mathcal{A}(1^n, H_{\mathsf{PRHF}}(x)) = 1] \pm \mathsf{negl}(n)$$

Plugging these into our first equation, we get that

$$\begin{aligned}\Pr_{x\leftarrow\{0,1\}^{n}}[\mathcal{A}(1^n, H_{\mathsf{PRHF}}(x)) = 1] = \delta \cdot &\Pr_{x\leftarrow\{0,1\}^{n}}[\mathcal{A}(1^n, H_{\mathsf{PRHF}}(x)) = 1] \\ &+ (1-\delta) \cdot \Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\mathcal{A}(1^n, y) = 1 \mid y \notin \mathsf{Image}_n] \\ &\pm \mathsf{negl}(n)\end{aligned}$$

Therefore,

$$(1-\delta) \cdot \Pr_{x\leftarrow\{0,1\}^{n}}[\mathcal{A}(1^n, H_{\mathsf{PRHF}}(x)) = 1] = (1-\delta) \cdot \Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\mathcal{A}(1^n, y) = 1 \mid y \notin \mathsf{Image}_n] \pm \mathsf{negl}(n)$$

which implies that

$$\Pr_{x\leftarrow\{0,1\}^{n}}[\mathcal{A}(1^n, H_{\mathsf{PRHF}}(x)) = 1] = \Pr_{y\leftarrow\{0,1\}^{\ell(n)}\setminus\mathsf{Image}_n}[\mathcal{A}(1^n, y) = 1] \pm \mathsf{negl}(n).$$

□

Hybrid$_3(x)$:

1. Let $n = |x|$.
2. For $i \in [p_{\mathsf{hbs}}(n)]$,
   (a) Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = \gamma'$
   where $\gamma' = 1 - \Pr_{y\leftarrow\{0,1\}^{\ell(n)}}[\exists z \text{ s.t. } H_{\mathsf{PRHF}}(z) = y]$
3. $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)$ where $r = r_1 \ldots r_{p_{\mathsf{hbs}}(n)}$.

4. ~~Define $\mathsf{Image}_n = \{y \in \{0,1\}^{\ell(n)} : \exists z \in \{0,1\}^n,\ y = H_{\mathsf{PRHF}}(z)\}$ and $\mathsf{Pre\text{-}Image}_n(y) = \{z \in \{0,1\}^n : H_{\mathsf{PRHF}}(z) = y\}$~~

5. For $i \in [p_{\mathsf{hbs}}(n)]$

   (a) If $r_i = 0$, sample $v_i \leftarrow \{0,1\}^n$, set $y_i = H_{\mathsf{PRHF}}(v_i)$, and set $v'_i = v_i$.

   (b) **Change:** If $r_i = 1$, sample $v_i \leftarrow \{0,1\}^n$, set $y_i = H_{\mathsf{PRHF}}(v_i)$, and set $v'_i = \bot$.

6. Output $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$.

Lemma 7.10. *If H_{PRHF} is a δ-*Dense-PRHF *for some* Efficiently-Approximable *constant $\delta \in (0,1)$, then for all non-uniform polynomial-sized adversaries $\mathcal{A}$ and all $x \in \mathcal{L}$,*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_2(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_3(x)) = 1]| \leq \mathsf{negl}(|x|)$$

Proof. We define a series of intermediate hybrids. Let $\mathbf{Hybrid}_{2,j}$ be the same as $\mathbf{Hybrid}_2$ except that in step 5b, for $i \leq j$, if $r_i = 1$, we sample (y_i, v'_i) according to $\mathbf{Hybrid}_3$ by sampling $v_i \leftarrow \{0,1\}^n$, setting $y_i = H_{\mathsf{PRHF}}(v_i)$, and setting $v'_i = \bot$. Observe that $\mathbf{Hybrid}_{2,0} = \mathbf{Hybrid}_2$ and that $\mathbf{Hybrid}_{2,p_{\mathsf{hbs}}(n)} = \mathbf{Hybrid}_3$.

Now, suppose for sake of contradiction that there exists an $x \in \mathcal{L}$ and a non-uniform polynomial-sized adversary and that can distinguish between the output of $\mathbf{Hybrid}_2(x)$ and $\mathbf{Hybrid}_3(x)$ with greater than $\mathsf{negl}(|x|)$ probability. Then, there exist an $x \in \mathcal{L}$, $j \in [p_{\mathsf{hbs}}(|x|)]$, a non-negligible function $\mu(\cdot)$, and a non-uniform polynomial-sized adversary $\mathcal{A}$ such that

$$\left|\Pr[\mathcal{A}(\mathbf{Hybrid}_{2,j-1}(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_{2,j}(x)) = 1]\right| > \frac{\mu(|x|)}{p_{\mathsf{hbs}}(|x|)}$$

We build a non-uniform polynomial-sized adversary $\mathcal{B}$ that breaks the property of H_{PRHF} defined in Lemma 7.9. $\mathcal{B}$ non-uniformly fixes the values of $(r, \mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi)$ computed as in steps 1-3 of $\mathbf{Hybrid}_{2,j-1}$ and $\mathbf{Hybrid}_{2,j}$ and the values of $\{(y_i, v_i, v'_i)\}_{i \neq j}$ computed as in step 5 of $\mathbf{Hybrid}_{2,j-1}$ and $\mathbf{Hybrid}_{2,j}$ which maximize the distinguishing advantage of $\mathcal{A}$. We assume that the non-uniformly fixed $r_j = 1$, as otherwise the hybrids are identical and $\mathcal{A}$'s distinguishing advantage is 0. Then, $\mathcal{B}$ receives y^*, where y^* is either sampled as $H_{\mathsf{PRHF}}(x^*)$ for a uniform $x^* \in \{0,1\}^n$, or y^* is sampled from $\{0,1\}^{\ell(n)} \backslash \mathsf{Image}_n$. $\mathcal{B}$ sets $v'_j = \bot$ and $y_j = y^*$, and sends $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(|x|)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$ to $\mathcal{A}$. $\mathcal{B}$ then outputs whatever $\mathcal{A}$ outputs. Observe that if y^* is sampled as $H_{\mathsf{PRHF}}(x^*)$ for a uniform $x^* \in \{0,1\}^n$, then $\mathcal{B}$ exactly emulates $\mathbf{Hybrid}_{2,j}$, and if y^* is sampled from $\{0,1\}^{\ell(n)} \backslash \mathsf{Image}_n$, then $\mathcal{B}$ exactly emulates $\mathbf{Hybrid}_{2,j-1}$. Thus, since $\mathcal{B}$ non-uniformly fixed values that maximized $\mathcal{A}$'s distinguishing probability, then $\mathcal{B}$ distinguishes between these two distributions of y^* with at least $\frac{\mu(|x|)}{p_{\mathsf{hbs}}(|x|)}$ probability. Thus, $\mathcal{B}$ contradicts Lemma 7.9. □

As we only need the values of v_i, when $i \in [\mathsf{ZHB}.\mathcal{I}]$, we rearrange our hybrid:

$\mathbf{Hybrid}_4(x)$:

1. Let $n = |x|$.

2. For $i \in [p_{\mathsf{hbs}}(n)]$,

   (a) Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = \gamma'$
   where $\gamma' = 1 - \Pr_{y \leftarrow \{0,1\}^{\ell(n)}}[\exists z \text{ s.t. } H_{\mathsf{PRHF}}(z) = y]$

3. $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.

4. **Change:** For $i \in [p_{\mathsf{hbs}}(n)]$, sample $v_i \leftarrow \{0,1\}^n$ and set $y_i = H_{\mathsf{PRHF}}(v_i)$.
5. **Change:** For $i \in \mathsf{ZHB}.\mathcal{I}$,
   (a) **Change:** If $r_i = 0$, set $v'_i = v_i$.
   (b) **Change:** If $r_i = 1$, set $v'_i = \perp$.
6. Output $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$.

Lemma 7.11. *For all adversaries $\mathcal{A}$ and all $x \in \mathcal{L}$,*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_3(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_4(x)) = 1]| = 0$$

Proof. The hybrids are identically distributed. □

We now exchange the exact density of H_{PRHF} with its approximated density.

$\mathbf{Hybrid}_5(x)$:

1. Let $n = |x|$.
2. For $i \in [p_{\mathsf{hbs}}(n)]$,
   (a) **Change:** Sample $r_i \in \{0,1\}$ such that $\Pr[r_i = 1] = 1 - \mathsf{Approx}_\delta(1^n)$
3. $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)$ where $r = r_1 \dots r_{p_{\mathsf{hbs}}(n)}$.
4. For $i \in [p_{\mathsf{hbs}}(n)]$, sample $v_i \leftarrow \{0,1\}^n$, and set $y_i = H_{\mathsf{PRHF}}(v_i)$.
5. For $i \in \mathsf{ZHB}.\mathcal{I}$,
   (a) If $r_i = 1$, set $v'_i = \perp$
   (b) If $r_i = 0$, set $v'_i = v_i$.
6. Output $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$.

Lemma 7.12. *If H_{PRHF} is a δ-Dense-PRHF for some* Efficiently-Approximable *constant $\delta \in (0,1)$, then for all adversaries $\mathcal{A}$ and all $x \in \mathcal{L}$,*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_4(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_5(x)) = 1]| \leq \mathsf{negl}(|x|)$$

Proof. Let $D_{0,x}$ be the distribution of r when r is sampled according to $\mathbf{Hybrid}_4(x)$, and let $D_{1,x}$ be the distribution of r when r is sampled according to $\mathbf{Hybrid}_5(x)$. Since H_{PRHF} is δ-Dense and δ is Efficiently-Approximable, then for all $n \in \mathbb{N}$

$$\gamma' = 1 - \Pr_{y \leftarrow \{0,1\}^{\ell(n)}}[\exists z \text{ s.t. } H_{\mathsf{PRHF}}(z) = y] = (1 - \delta) \pm \mathsf{negl}(n) = (1 - \mathsf{Approx}_\delta(1^n)) \pm \mathsf{negl}(n)$$

Thus, for each i, $\Pr[r_i = 1]$ when sampled according to $\mathbf{Hybrid}_4$ is negligibly close to $\Pr[r_i = 1]$ when sampled according to $\mathbf{Hybrid}_5$. Therefore, since the r_i's are sampled independently, by Lemma 3.5,

$$\Delta(D_{0,x}, D_{1,x}) \leq p_{\mathsf{hbs}}(n) \cdot \mathsf{negl}(|x|)$$

As the only difference between these two hybrids is that r is sampled either according to $D_{0,x}$ or $D_{1,x}$, then for all $x \in \mathcal{L}$,

$$\Delta(\mathbf{Hybrid}_4(x), \mathbf{Hybrid}_5(x)) \leq p_{\mathsf{hbs}}(n) \cdot \mathsf{negl}(|x|)$$

□

Finally, we simulate the values of $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathcal{I}}, \mathsf{ZHB}.\pi)$ using $\mathsf{Sim}_\gamma(x)$.

$\mathbf{Hybrid}_6(x)$:

1. Let $n = |x|$.
2. **Change:** $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.\mathsf{Sim}_\gamma(x)$.
3. For $i \in [p_{\mathsf{hbs}}(n)]$, sample $v_i \leftarrow \{0,1\}^n$, and set $y_i = H_{\mathsf{PRHF}}(v_i)$.
4. For $i \in \mathsf{ZHB}.\mathcal{I}$,
   (a) If $r_i = 1$, set $v'_i = \perp$
   (b) If $r_i = 0$, set $v'_i = v_i$.
5. Output $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$.

Lemma 7.13. *If $\mathsf{ZHB}.\Pi_\gamma$ satisfies statistical zero knowledge, then for all adversaries $\mathcal{A}$ and all $x \in \mathcal{L}$,*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_5(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_6(x)) = 1]| \leq \mathsf{negl}(|x|)$$

Proof. Suppose for sake of contradiction that there exists an adversary $\mathcal{A}$ and an $x \in \mathcal{L}$ such that

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_5(x)) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_6(x)) = 1]| > \mathsf{negl}(|x|)$$

We build an adversary $\mathcal{B}$ that breaks the statistical zero knowledge of $\mathsf{ZHB}.\Pi_\gamma$. $\mathcal{B}$ first receives $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ from its zero knowledge challenger. $\mathcal{B}$ samples $(\{y_i\}_{i \in [p_{\mathsf{hbs}}(n)]}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}})$ as in $\mathbf{Hybrid}_5$ and $\mathbf{Hybrid}_6$, sends $(\mathsf{urs} = (y_1 \dots y_{p_{\mathsf{hbs}}(n)}), \pi = (\mathsf{ZHB}.\mathcal{I}, \{v'_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi))$ to $\mathcal{A}$, and outputs whatever $\mathcal{A}$ outputs. Observe that if $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{Sim}(x)$, then $\mathcal{B}$ exactly emulates $\mathbf{Hybrid}_6(x)$, and if $(\mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ is sampled such that $r \leftarrow \mathsf{Gen}_\gamma(1^n)$ (with $\mathsf{Approx}_\gamma(1^n) = 1 - \mathsf{Approx}_\delta(1^n)$) and $(\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)$, then $\mathcal{B}$ exactly emulates $\mathbf{Hybrid}_5(x)$. Thus, $\mathcal{B}$'s distinguishing advanatage is non-negligible, which breaks the statistical zero knowledge of $\mathsf{ZHB}.\Pi_\gamma$. □

We observe that our final hybrid is identical to our simulator. Thus, by combining our hybrid indistinguishability lemmas, we get computational zero knowledge.

□

8 Separating P from NP ∩ coNP

As an application of our NIZK proof in the random oracle model, we show a separation of NP∩coNP and P relative to a random oracle, assuming that UP ⊈ RP. Starting with a language $\mathcal{L}_0 \in \mathsf{UP} \setminus \mathsf{RP}$, we will construct a language $\mathcal{L}$ that relies on a *NIZK proof* with desired properties given in Lemma 8.5. Such a language $\mathcal{L}$ has an efficient NP verifier and an efficient coNP verifier which crucially use the NIZK verifier. However, $\mathcal{L}$ remains hard to solve in our modified construction due to the non-adaptive zero-knowledge property of the NIZK proof.

Definition 8.1 (The class UP)**.** *A language $\mathcal{L} \subseteq \{0,1\}^*$ is in the complexity class UP if there is a polynomial-time decidable relation $R_{\mathcal{L}} \subseteq \{0,1\}^* \times \{0,1\}^*$ and a polynomial $p(\cdot)$, such that for every $x \in \{0,1\}^*$:*

1. *If $x \in \mathcal{L}$, then there is a* unique *$w \in \{0,1\}^*$ such that $|w| \leq p(|x|)$ and $(x,w) \in R_{\mathcal{L}}$.*
2. *If $x \notin \mathcal{L}$, then there is no $w \in \{0,1\}^*$ such that $|w| \leq p(|x|)$ and $(x,w) \in R_{\mathcal{L}}$.*

Lemma 8.2. *If there exists an injective one-way function $f : \{0,1\}^* \to \{0,1\}^*$, then $\mathsf{UP} \not\subseteq \mathsf{RP}$.*

Proof. Let $\mathcal{L} \coloneqq \{(y,i) : \exists x, f(x) = y \wedge x_i = 1\}$. Then observe that $\mathcal{L} \in \mathsf{UP}$ where the relation verification is done by a polynomial time Turing machine that takes as input (y,i) and w and outputs 1 if $f(w) = y$ and $w_i = 1$ and otherwise outputs 0. For every $(y,i) \in \mathcal{L}$, injectivity guarantees a unique x such that $f(x) = y$ and $x_i = 1$ and for every $(y,i) \notin \mathcal{L}$, there is no string x satisfying the two conditions. Since f is one-way, there is no RP-decider for $\mathcal{L}$ or else this RP-decider can be used to recover a preimage bit-by-bit. □

Theorem 8.3. *If $\mathsf{UP} \not\subseteq \mathsf{RP}$, then with probability 1 over the choice of a random oracle $\mathcal{O}$, $\mathsf{P}^{\mathcal{O}} \neq \mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}$.*

Proof of Theorem 8.3. The proof will follow directly from Lemma 8.4 and Lemma 8.6 which are stated below. □

To prove this theorem, we start with a language $\mathcal{L}_0 \in \mathsf{UP} \setminus \mathsf{RP}$. Let $R'_{\mathcal{L}_0}$ be a UP-relation for $\mathcal{L}_0$, and $p(\cdot)$ be a corresponding polynomial bound on the witness length, as required by Definition 8.1. Let $R_{\mathcal{L}_0} \subseteq \{0,1\}^* \times \{0,1\}^*$ be a polynomial-time decidable relation such that $(x,w) \in R_{\mathcal{L}_0}$ if and only if $(x,w) \in R'_{\mathcal{L}_0}$ and $|w| \leq p(|x|)$.

To obtain a language that has an NP and a coNP-verifier, we will construct a new language whose statements also include a NIZK proof for membership in the language $\mathcal{L}_0$. Let $\Pi'_{\mathsf{NIZK}} = (\mathcal{P}'^{(\cdot)}, \mathcal{V}'^{(\cdot)})$ be a NIZK proof system for membership in the language $\mathcal{L}_0$ in the random oracle model, according to Lemma 8.5. Crucially, $\mathcal{V}'^{(\cdot)}$ is deterministic (by Definition 3.10). We now construct our primary language of interest that depends on the random oracle $\mathcal{O}$:

$$\mathcal{L}^{\mathcal{O}} \coloneqq \left\{ (x,i,\pi) : (\exists w, (x,w) \in R_{\mathcal{L}_0} \text{ and } w_i = 1) \wedge (\Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x,\pi) = 1) \right\}$$

where $i \in \mathbb{N}$ is given by its standard binary representation.

Let $R_{\mathcal{L}^{\mathcal{O}}}$ be a NP relation defined by the language $\mathcal{L}^{\mathcal{O}}$. We first give a polynomial-time oracle aided Turing machine $\mathcal{V}^{(\cdot)}_{\mathsf{NP}}$ that, when given access to a random oracle $\mathcal{O}$, recognizes $R_{\mathcal{L}^{\mathcal{O}}}$ with probability 1 over the choice of $\mathcal{O}$. We then give a polynomial-time oracle-aided Turing machine $\mathcal{V}^{(\cdot)}_{\mathsf{coNP}}$ such that, when given access to a random oracle $\mathcal{O}$, recognizes the complement language

$$\overline{\mathcal{L}}^{\mathcal{O}} \coloneqq \left\{ (x,i,\pi) : (\nexists w, (x,w) \in R_{\mathcal{L}_0} \text{ and } w_i = 1) \vee (\Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x,\pi) = 0) \right\}$$

Lemma 8.4. $\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}} \in \mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}\right] = 1$.

We will use the following intermediate lemma to prove this result.

Lemma 8.5 (Soundness Amplification of NIZK proofs in the Random Oracle Model)**.** *If for any language $L \in \mathsf{NP}$ there exists a NIZK proof system $\Pi_{\mathsf{NIZK}} = (\mathcal{P}^{(\cdot)}, \mathcal{V}^{(\cdot)})$ in the random oracle model with perfect completeness, statistical soundness, and computational zero-knowledge, then for any*

language $L \in \mathsf{NP}$, there exists a NIZK proof system $\Pi'_{\mathsf{NIZK}} = (\mathcal{P}'^{(\cdot)}, \mathcal{V}'^{(\cdot)})$ with perfect completeness, computational zero-knowledge, and the following soundness condition :

$$\Pr_{\mathcal{O}}\left[\forall x \notin L, \forall \pi\{0,1\}^*, \mathcal{V}'^{\mathcal{O}}(x,\pi) = 0\right] = 1.$$

We call such a property as almost-sure *soundness.*

Proof. For $i \in \mathbb{N}$, let $X_i(\mathcal{O})$ be an indicator random variable over a choice of random oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2, \ldots)$ such that $X_i = 1$ if there exists a string $x \notin L$ of length i and a proof $\pi \in \{0,1\}^*$ such that $\mathcal{V}^{\mathcal{O}}(x,\pi) = 1$, and $X_i = 0$ otherwise. Let us refer to this as a *bad* proof. By the $\left(\frac{1}{e^i} + \frac{2p_{\mathsf{hbs}}(i)}{2^{i/4}}\right)$-statistical soundness of Π_{NIZK} from Lemma 6.4 we know that $\Pr_{\mathcal{O}}[X_i = 1] \leq \frac{1}{e^i} + \frac{2p_{\mathsf{hbs}}(i)}{2^{i/4}}$. As a result, the following series converges

$$\sum_{n\in\mathbb{N}} \Pr[X_n = 1] = \sum_{i\in\mathbb{N}} \left(\frac{1}{e^i} + \frac{2p_{\mathsf{hbs}}(i)}{2^{i/4}}\right) \leq \sum_{i\in\mathbb{N}} \frac{1}{i^2} < \infty.$$

Now, by applying BC1 (Lemma 3.6), we know that the number of input lengths for which *bad* proofs exists are only finitely many. In other words. there exists some constant $n_0 \in \mathbb{N}$ such that for all $i > n_0$, $X_i = 0$.

For Π'_{NIZK}, the prover algorithm $\mathcal{P}'^{(\cdot)}$ remains identical as $\mathcal{P}^{(\cdot)}$. Thus, the completeness and zero knowledge remains unaffected. We give the following modified polynomial-time verification algorithm as follows:

$\mathcal{V}'(x,\pi)$:

1. If $|x| \leq n_0$, brute force check if $x \in L$. If yes, then accept, otherwise reject.
2. Else return $\mathcal{V}(x,\pi)$.

Note that $\mathcal{V}'$ is an efficient algorithm because n_0 is a fixed constant. Thus, $\mathcal{V}'$ never accepts a wrong proof when the length of the statement is at-most n_0. When $|x| > n_0$, the analysis above shows that $\Pr_{\mathcal{O}}[X_i = 1] = 0$, hence we get the desired result. □

Proof of Lemma 8.4. Let $R_{\mathcal{L}^{\mathcal{O}}}$ be a NP relation defined by the language $\mathcal{L}^{\mathcal{O}}$. In other words, if $(x,i,\pi) \in \mathcal{L}^{\mathcal{O}}$, then there exists a witness $w \in \{0,1\}^*$ such that $((x,i,\pi),w) \in R_{\mathcal{L}^{\mathcal{O}}}$. From here on wards, for any string $w \in \{0,1\}^*$, let w_i denote the ith bit of w and if $i > |w|$, we use the convention that $w_i = 0$. We now give a polynomial-time oracle aided Turing machine $\mathcal{V}^{(\cdot)}_{\mathsf{NP}}$ that, when given access to a random oracle $\mathcal{O}$, recognizes $R_{\mathcal{L}^{\mathcal{O}}}$ with probability 1 over the choice of $\mathcal{O}$.

$\mathcal{V}^{\mathcal{O}}_{\mathsf{NP}}((x,i,\pi),w)$:

1. If $(x,w) \in R_{\mathcal{L}_0}$ and $w_i = 1$ and $\Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x,\pi) = 1$, then output 1.
2. Else, output 0.

Observe that for any random oracle $\mathcal{O}$, $\Pi'_{\mathsf{NIZK}}.\mathcal{V}^{\mathcal{O}}$ has polynomial runtime in the input length, so $\mathcal{V}^{\mathcal{O}}_{\mathsf{NP}}$ has polynomial runtime as well. There are two cases to consider:

1. For any random oracle $\mathcal{O}$, for any $(x,i,\pi) \in \mathcal{L}^{\mathcal{O}}$, by definition of $\mathcal{L}^{\mathcal{O}}$, there exists a witness w^* such that $(x,w^*) \in R_{\mathcal{L}_0}$ and $w^*_i = 1$, and $\Pi_{\mathsf{NIZK}'}.\mathcal{V}'^{\mathcal{O}}(x,\pi) = 1$. Therefore, this witness w^* for

x is also a witness for the statement (x, i, π). Given this witness w^*, $\mathcal{V}^{\mathcal{O}}_{\mathsf{NP}}((x, i, \pi), w^*) = 1$. Since this holds for any random oracle $\mathcal{O}$, we have, for the same polynomial $p(\cdot)$ defined above, the following probability statement:

$$\Pr_{\mathcal{O}}\left[\forall (x, i, \pi) \in \mathcal{L}^{\mathcal{O}}, \exists w \in \{0,1\}^*, \mathcal{V}^{\mathcal{O}}_{\mathsf{NP}}((x, i, \pi), w) = 1\right] = 1$$

2. For any random oracle $\mathcal{O}$, for $(x, i, \pi) \notin \mathcal{L}^{\mathcal{O}}$, we have $(x, i, \pi) \in \overline{\mathcal{L}}^{\mathcal{O}}$.

   - If $x \notin \mathcal{L}_0$, then $(x, w) \notin R_{\mathcal{L}_0}$ for all $w \in \{0,1\}^*$, so $\mathcal{V}^{\mathcal{O}}_{\mathsf{NP}}((x, i, \pi), w^*) = 0$ for all $w \in \{0,1\}^*$.
   - If $x \in \mathcal{L}_0$, then there exists a unique witness w^* for x. This w^* acts as the witness for the statement (x, i, π)
     - If $w^*_i = 0$, then $\mathcal{V}^{\mathcal{O}}_{\mathsf{NP}}((x, i, \pi), w^*) = 0$.
     - Otherwise, if $w^*_i = 1$, the definition of the complement language $\overline{\mathcal{L}}^{\mathcal{O}}$ implies that $\Pi_{\mathsf{NIZK}'}.\mathcal{V}'^{\mathcal{O}}(x, \pi) = 0$, so $\mathcal{V}^{\mathcal{O}}_{\mathsf{NP}}((x, i, \pi), w^*) = 0$.

   Since the above analysis holds for any random oracle $\mathcal{O}$, we have, for the same polynomial $p(\cdot)$ defined above, the following probability statement:

$$\Pr_{\mathcal{O}}\left[\forall (x, i, \pi) \notin \mathcal{L}^{\mathcal{O}}, \forall w \in \{0,1\}^*, \mathcal{V}^{\mathcal{O}}_{\mathsf{NP}}((x, i, \pi), w) = 0\right] = 1$$

$\mathcal{V}^{\mathcal{O}}_{\mathsf{coNP}}((x, i, \pi), w)$:

1. If either $\Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x, \pi) = 0$ or $(\Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x, \pi) = 1 \wedge (x, w) \in R_{\mathcal{L}_0} \wedge w_i = 0)$, then output 1.
2. Else, output 0.

We now show the correctness of the coNP verifier. First, we recall the almost-sure soundness property of our NIZK protocol Π'_{NIZK} for the language $\mathcal{L}_0$ from Lemma 8.5:

$$\Pr_{\mathcal{O}}\left[\forall x \notin \mathcal{L}_0, \forall \pi \in \{0,1\}^*, \Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x, \pi) = 0\right] = 1.$$

1. For any random oracle $\mathcal{O}$, consider any $(x, i, \pi) \in \overline{\mathcal{L}}^{\mathcal{O}}$.

   (a) If $x \notin \mathcal{L}_0$, then for all strings $w \in \{0,1\}^*$, $(x, w) \notin R_{\mathcal{L}_0}$.
   Combining this observation with the almost-sure property of Π'_{NIZK} for language $\mathcal{L}_0$ gives the following probability statement:

$$\Pr_{\mathcal{O}}\left[\forall x \notin \mathcal{L}_0, \forall i \in \{0,1\}^*, \forall \pi \in \{0,1\}^*, \mathcal{V}^{\mathcal{O}}_{\mathsf{coNP}}((x, i, \pi), w) = 1\right] = 1.$$

   (b) Now consider any $x \in \mathcal{L}_0$. Then there exists a unique witness w^* such that $(x, w^*) \in R_{\mathcal{L}_0}$ which is also the witness for the statement (x, i, π).

      i. If $\Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x, \pi) = 0$, then $\mathcal{V}^{\mathcal{O}}_{\mathsf{coNP}}((x, i, \pi), w^*) = 1$.

ii. If $\Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x,\pi)=1$, then it must be that $w_i^*=0$ by the definition of $\overline{\mathcal{L}}^{\mathcal{O}}$. Then w^* serves as witness for (x,i,π) since

$$\Pi'_{\mathsf{NIZK}}.\mathcal{V}'^{\mathcal{O}}(x,\pi)=1 \wedge (x,w^*)\in R_{\mathcal{L}_0} \wedge w_i^*=0,$$

so $\mathcal{V}^{\mathcal{O}}_{\mathsf{coNP}}((x,i,\pi),w^*)=1$.

Therefore, we have the following probability statement:

$$\Pr_{\mathcal{O}}\left[\forall (x,i,\pi)\in\overline{\mathcal{L}}^{\mathcal{O}}, \exists w\in\{0,1\}^*, \mathcal{V}^{\mathcal{O}}_{\mathsf{coNP}}((x,i,\pi),w)=1\right]=1.$$

2. For any random oracle $\mathcal{O}$, consider any $(x,i,\pi)\notin\overline{\mathcal{L}}^{\mathcal{O}}$ which is a condition equivalent to $(x,i,\pi)\in\mathcal{L}^{\mathcal{O}}$. Then there exists a unique witness w^* such that $(x,w^*)\in R_{\mathcal{L}_0}$ and $w_i^*=1$ and $\Pi_{\mathsf{NIZK}'}.\mathcal{V}'^{\mathcal{O}}(x,\pi)=1$ by definition of $\mathcal{L}^{\mathcal{O}}$. Since there is no other witness w for x, by the construction of $\mathcal{V}^{(\cdot)}_{\mathsf{coNP}}$, we have $\mathcal{V}^{\mathcal{O}}_{\mathsf{coNP}}((x,i,\pi),w^*)=0$. Since this analysis holds for any random oracle $\mathcal{O}$, we have the following probability statement:

$$\Pr_{\mathcal{O}}\left[\forall (x,i,\pi)\notin\overline{\mathcal{L}}^{\mathcal{O}}, \forall w\in\{0,1\}^*, \mathcal{V}^{\mathcal{O}}_{\mathsf{coNP}}((x,i,\pi),w)=0\right]=1.$$

□

Lemma 8.6. *Assuming* $\mathsf{UP}\not\subseteq\mathsf{RP}$, $\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}}\notin\mathsf{P}^{\mathcal{O}}\right]=1$.

To show the desired statement, we will use the following Lemma from [BG81].

Lemma 8.7 (Lemma 1 in [BG81])**.** *Let $\mathcal{L}^{(\cdot)}$ be an oracle-dependent language such that any bit of the oracle (an infinite binary sequence) only affects the membership of finitely many statements in $\mathcal{L}^{\mathcal{O}}$. Let $\mathcal{D}^{(\cdot)}=\{D_1^{(\cdot)},D_2^{(\cdot)},\ldots\}$ be the set of all possible polynomial-time oracle aided Turing Machines. If there does not exist any $k\in\mathbb{N}$ such that*

$$\Pr_{\mathcal{O}}\left[\left(\forall (x,i,\pi)\in\mathcal{L}^{\mathcal{O}}, D_k^{\mathcal{O}}(x,i,\pi)=1\right)\wedge\left(\forall (x,i,\pi)\notin\mathcal{L}^{\mathcal{O}}, D_k^{\mathcal{O}}(x,i,\pi)=0\right)\right]=1$$

then

$$\Pr_{\mathcal{O}}\left[\exists k\in\mathbb{N} \text{ such that } \left(\forall (x,i,\pi)\in\mathcal{L}^{\mathcal{O}}, D_k^{\mathcal{O}}(x,i,\pi)=1\right)\wedge\left(\forall (x,i,\pi)\notin\mathcal{L}^{\mathcal{O}}, D_k^{\mathcal{O}}(x,i,\pi)=0\right)\right]=0.$$

We now provide an overview on how we will use Lemma 8.7 to prove Lemma 8.6. Recall our primary language of interest:

$$\mathcal{L}^{\mathcal{O}}:=\left\{(x,i,\pi):(\exists w,(x,w)\in R_{\mathcal{L}_0}\text{ and }w_i=1)\wedge(\Pi'_{\mathsf{NIZK}}.\mathcal{V}^{\mathcal{O}}(x,\pi)=1)\right\}.$$

Note that the only way $\mathcal{L}^{\mathcal{O}}$ is dependent on the random oracle is via the NIZK verifier $\Pi'_{\mathsf{NIZK}}.\mathcal{V}^{\mathcal{O}}(x,\pi)$. This means that there always exists an $\exists n\in\mathbb{N}$, such that changing a bit of $\mathcal{O}$ influences the membership of at-most 2^n instance of $\mathcal{L}^{\mathcal{O}}$ which satisfies the condition required to use Lemma 8.7.

Let $\mathcal{D}^{(\cdot)}=\{D_1^{(\cdot)},D_2^{(\cdot)},\ldots\}$ be the set of all possible polynomial-time oracle aided Turing Machines. Now observe the following definitional equivalence: For any oracle $\mathcal{O}$, the definition of $\mathsf{P}^{\mathcal{O}}$ gives that an oracle-dependent language $\mathcal{L}^{\mathcal{O}}\in\mathsf{P}^{\mathcal{O}}$ if and only if there exists $k\in\mathbb{N}$ such that two

conditions hold: (1) for all statements $z \in \mathcal{L}^{\mathcal{O}}$, $D_k^{\mathcal{O}}(z) = 1$ and (2) for all statements $z \notin \mathcal{L}^{\mathcal{O}}$, $D_k^{\mathcal{O}}(z) = 0$. This definitional equivalence implies that, for any constant $\varepsilon > 0$:

$$\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}} \in \mathsf{P}^{\mathcal{O}}\right] > \varepsilon$$

if and only if

$$\Pr_{\mathcal{O}}\left[\exists k \in \mathbb{N} \text{ s.t. } (\forall z \in \mathcal{L}^{\mathcal{O}}, D_k^{\mathcal{O}}(z) = 1) \wedge (\forall z \notin \mathcal{L}^{\mathcal{O}}, D_k^{\mathcal{O}}(z) = 0)\right] > \varepsilon.$$

Using the contrapositive of Lemma 8.7, the latter probability statement above implies that there exists a $k \in \mathbb{N}$ such that

$$\Pr_{\mathcal{O}}\left[\left(\forall (x,i,\pi) \in \mathcal{L}^{\mathcal{O}}, D_k^{\mathcal{O}}(x,i,\pi) = 1\right) \wedge \left(\forall (x,i,\pi) \notin \mathcal{L}^{\mathcal{O}}, D_k^{\mathcal{O}}(x,i,\pi) = 0\right)\right] = 1.$$

In other words, if we assume that $\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}} \in \mathsf{P}^{\mathcal{O}}\right] > \varepsilon$, then in fact we have a particular polynomial-time oracle-aided Turing machine $D_k^{(\cdot)}$ which is a decider for $\mathcal{L}^{(\cdot)}$ with probability 1 over the choice of a random oracle $\mathcal{O}$.

For statements $x \in \mathcal{L}_0$, we observe that the oracle-aided TM $D_k^{(\cdot)}$, when given access to a random oracle $\mathcal{O}$, is also able to decide, using its input (x, i, π), whether the ith bit of the unique witness w to x is 1 or 0. Therefore, polynomially many invocations of the oracle-aided TM $D_k^{(\cdot)}$ will allow us to recover the entire witness w. Using this oracle-aided TM $D_k^{(\cdot)}$, we will construct a RP decider for $\mathcal{L}_0$ to contradict the assumption that $\mathcal{L}_0 \in \mathsf{UP} \setminus \mathsf{RP}$.

We will now build an RP decider that, instead of being given access to a random oracle and a proof string, *simulates* both the proof string and the answers to random oracle queries and recovers the ith bit of the unique witness for any statement $x \in \mathcal{L}_0$. This ability to correctly (with overwhelming probability) simulate the proof string and the random oracle answers follows from the zero-knowledge property of the NIZK in the random oracle model for the language $\mathcal{L}_0$. We now recall this property in verbatim: There exists a stateful PPT simulator $\mathsf{Sim} = (\mathsf{SimProof}, \mathsf{SimRO})$ and a negligible function $\varepsilon_{zk}(\cdot)$ such that for all $x \in \mathcal{L}$ and all *polynomial-query-bounded* oracle-aided Turing machines $\mathcal{A}^{(\cdot)}$ (cf. Def. 3.1),

$$\Bigg| \Pr\left[\mathcal{A}^{\mathcal{O}(\cdot)}(x,\pi) = 1 : \mathcal{O} \leftarrow RO, \pi \leftarrow P^{\mathcal{O}(\cdot)}(x)\right] \\ - \Pr\left[\mathcal{A}^{\mathsf{SimRO}(\mathsf{st},\cdot)}(x,\pi^*) = 1 : (\mathsf{st},\pi^*) \leftarrow \mathsf{SimProof}(x)\right] \Bigg| \leq \varepsilon_{zk}(|x|)$$

where the probability is over the choice of $\mathcal{O}$ and the randomness of P, Sim, and $\mathcal{A}$.

We can therefore build another RP machine that, given an input x, first attempts to recover a witness w for x, and then checks if $(x, w) \in R_{\mathcal{L}_0}$ which is an efficiently verifiable relation. For all $x \in \mathcal{L}_0$, the machine's correctness will hold by the zero-knowledge property and for all $x \notin \mathcal{L}_0$, the machine's correctness will be perfectly guaranteed by the fact that no witness exists.

We now proceed with the formal proof of Lemma 8.6.

Proof of Lemma 8.6. Assume for sake of contradiction that there exists some constant $\varepsilon > 0$ such that $\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}} \in \mathsf{P}^{\mathcal{O}}\right] > \varepsilon$. Then by Lemma 8.7, there exists a polynomial-time oracle-aided Turing machine $D^{(\cdot)}$ such that

$$\Pr_{\mathcal{O}}\left[\left(\forall (x,i,\pi) \in \mathcal{L}^{\mathcal{O}}, D^{\mathcal{O}}(x,i,\pi) = 1\right) \wedge \left(\forall (x,i,\pi) \notin \mathcal{L}^{\mathcal{O}}, D^{\mathcal{O}}(x,i,\pi) = 0\right)\right] = 1.$$

For all $i \in \mathbb{N}$, we define the polynomial-time oracle-aided Turing machine $D_{(i)}^{(\cdot)}$ as follows:

$D^{\mathcal{O}}_{(i)}(x, \pi)$:

1. Output $D^{\mathcal{O}}(x, i, \pi)$.

The above machine is defined for syntactical purposes in the proof. Now we define the probabilistic machine $\tilde{D}$ as follows:

$\tilde{D}(x, i)$:

1. $(\mathsf{st}, \pi) \leftarrow \mathsf{SimProof}(x)$.
2. Output $D^{\mathsf{SimRO}(\mathsf{st},\cdot)}(x, i, \pi)$.

Note that in the above definition, we do not run $D^{\mathcal{O}}_{(i)}$ (we cannot hardcode infinitely many TM descriptions, over all $i \in \mathbb{N}$, into $\tilde{D}$).

Finally, we define another probabilistic Turing machine D^* as follows. Let p be a polynomial that bounds the length of witnesses given by the definition UP and the fact that $\mathcal{L}_0 \in \mathsf{UP}$.

$D^*(x)$:

1. For $i \in [p(|x|)]$.
   (a) If $\tilde{D}(x, i) = 1$, set $w_i \leftarrow 1$.
   (b) Else, set $w_i \leftarrow 0$.
2. If w (or any prefix of w) satisfies $(x, w) \in R_{\mathcal{L}_0}$, then output 1. Else, output 0.

We claim that D^* is an RP decider for $\mathcal{L}_0$.

1. Observe that if $x \notin \mathcal{L}_0$, then there does not exist a witness w for x so $D^*(x)$ will always output 0 (by Step 2 of D^*).

2. Now consider any $x \in \mathcal{L}_0$. As a high-level overview, we will argue that for any fixed value of $i \in [p(|x|)]$, Step 1 of D^* will recover w_i with all but negligible probability in the length of x. Then we will apply a union bound to show that the probability that w_i for all indices $i \in [p(|x|)]$ are all correctly recovered happens with all but negligible probability.

   Formally, for any fixed $x \in \mathcal{L}_0$ and for any fixed index $i \in [p(|x|)]$, let w denote the unique witness for x such that $(x, w) \in R_{\mathcal{L}_0}$. Then over the coins of the $\mathsf{Sim} = (\mathsf{SimProof}, \mathsf{SimRO})$ and the coins of the machine, we have

$$\begin{aligned}
&\Pr\left[\tilde{D}(x, i) = 1 : (x \in \mathcal{L}_0 \wedge w_i = 1)\right] \\
&\quad = \Pr\left[D^{\mathsf{SimRO}(\mathsf{st},\cdot)}(x, i, \pi) = 1 : (x \in \mathcal{L}_0 \wedge w_i = 1), (\mathsf{st}, \pi) \leftarrow \mathsf{SimProof}(x)\right] \\
&\quad = \Pr\left[D^{\mathsf{SimRO}(\mathsf{st},\cdot)}_{(i)}(x, \pi) = 1 : (x \in \mathcal{L}_0 \wedge w_i = 1), (\mathsf{st}, \pi) \leftarrow \mathsf{SimProof}(x)\right] \qquad (1)
\end{aligned}$$

   where all three equivalences are by construction of the machines present in the probability statements.

Then, by the zero-knowledge property of the NIZK in the random oracle model for the language $\mathcal{L}_0$, we have the existence of a negligible function ε such that

$$\left| \Pr_{\text{coins of } \mathsf{Sim}} \left[D_{(i)}^{\mathsf{SimRO}(\mathsf{st},\cdot)}(x,\pi) = 1 : (x \in \mathcal{L}_0 \wedge w_i = 1), (\mathsf{st},\pi) \leftarrow \mathsf{SimProof}(x) \right] \right.$$
$$\left. - \Pr_{\mathcal{O}} \left[D_{(i)}^{\mathcal{O}}(x,\pi) = 1 : (x \in \mathcal{L}_0 \wedge w_i = 1), \pi \leftarrow P^{\mathcal{O}}(x) \right] \right| \leq \varepsilon(|x|). \tag{2}$$

Then observe that the probability statement

$$\Pr_{\mathcal{O}} \left[\forall (x,i,\pi) \in \mathcal{L}^{\mathcal{O}}, D^{\mathcal{O}}(x,i,\pi) = 1 \right] = 1$$

along with the perfect completeness of the NIZK protocol implies:

$$\Pr_{\mathcal{O}} \left[D^{\mathcal{O}}(x,i,\pi) = 1 : (x \in \mathcal{L}_0 \wedge w_i = 1), \pi \leftarrow P^{\mathcal{O}}(x) \right] = 1.$$

Now, by the definition of the polynomial-time oracle-aided machine $D_{(i)}^{(\cdot)}$, we have,

$$\Pr_{\mathcal{O}} \left[D_{(i)}^{\mathcal{O}}(x,\pi) = 1 : (x \in \mathcal{L}_0 \wedge w_i = 1), \pi \leftarrow P^{\mathcal{O}}(x) \right] = 1. \tag{3}$$

Together, Equation 2 and Equation 3 imply

$$\Pr_{\text{coins of } \mathsf{Sim}} \left[D_{(i)}^{\mathsf{SimRO}(\mathsf{st},\cdot)}(x,\pi) = 1 : (x \in \mathcal{L}_0 \wedge w_i = 1), (\mathsf{st},\pi) \leftarrow \mathsf{SimProof}(x) \right] \geq 1 - \varepsilon(|x|). \tag{4}$$

By symmetry, we also have

$$\Pr_{\text{coins of } \mathsf{Sim}} \left[D_{(i)}^{\mathsf{SimRO}(\mathsf{st},\cdot)}(x,\pi) = 0 : (x \in \mathcal{L}_0 \wedge w_i = 0), (\mathsf{st},\pi) \leftarrow \mathsf{SimProof}(x) \right] \geq 1 - \varepsilon(|x|). \tag{5}$$

Therefore, by combining Equation 1, Equation 4, and Equation 5, we have that for any $x \in \mathcal{L}_0$, for any fixed $i \in [p(|x|)]$, for any bit $b \in \{0,1\}$:

$$\Pr_{\text{coins of } \mathsf{Sim}} \left[\tilde{D}(x,i) = b : (x \in \mathcal{L}_0 \wedge w_i = 1 - b), \right] \leq \varepsilon(|x|). \tag{6}$$

Then, by a union bound over all values of $i \in [p(|x|)]$,

$$\Pr_{\text{coins of } \mathsf{Sim}} \left[\exists i \in [p(|x|)], \tilde{D}(x,i) = b : (x \in \mathcal{L}_0 \wedge w_i = 1 - b) \right]$$
$$\leq p(|x|) \cdot \varepsilon(|x|) \tag{7}$$

Note that $p(|x|) \cdot \varepsilon(|x|)$ is negligible in $|x|$. Therefore, we conclude that for all $x \in \mathcal{L}_0$, the first step of D^* correctly recovers the witness w with probability at least $1 - p(|x|)\varepsilon(|x|)$ probability.

The two cases imply that D^* is an RP decider for the language $\mathcal{L}_0$, contradicting the assumption that $\mathcal{L}_0 \in \mathsf{UP} \setminus \mathsf{RP}$. Therefore, it must be that

$$\Pr_{\mathcal{O}} \left[\mathcal{L}^{\mathcal{O}} \in \mathsf{P}^{\mathcal{O}} \right] = 0.$$

This gives us the desired result.

□

8.1 Explicit "Unstructured" Candidates for Hard Languages in $\mathsf{NP} \cap \mathsf{coNP}$

While the conclusion of Theorem 8.3 is existential, the proof is actually *constructive* in the sense that the code of the NP-verifier and the coNP-verifier in the conclusion can be efficiently constructed from the code of the UP-verifier or injective one-way function in the assumption.

This yields an efficient compiler that constructs candidate hard languages in $\mathsf{NP} \cap \mathsf{coNP}$ from cryptographic hash functions. The compiler uses the random oracle heuristic [BR93], where a construction is designed and analyzed by assuming the availability of a random function, and the latter is heuristically instantiated by using a concrete cryptographic hash function. While not sound in general [CGH04], the counterexamples are typically contrived.

More concretely, the proof of Theorem 8.3 implies the following explicit construction of an "unstructured" language $\mathcal{L}$ which is *heuristically* conjectured to be in $\mathsf{NP} \cap \mathsf{coNP} \setminus \mathsf{P}$. Start by taking any language $\mathcal{L}_0$ in $\mathsf{UP} \setminus \mathsf{RP}$. Such a language is implied by the existence of an *injective* one-way function, which in turn can be heuristically constructed by instantiating a length-tripling random oracle $\mathcal{O}_0 : \{0,1\}^n \to \{0,1\}^{3n}$. (Note that such a random $\mathcal{O}_0$ will be both one-way and injective with overwhelming probability.) We now apply the construction of Theorem 8.3 to obtain the separating language $\mathcal{L}^{\mathcal{O}}$, with efficient $\mathsf{NP}^{\mathcal{O}}$ and $\mathsf{coNP}^{\mathcal{O}}$ verifiers, which we make explicit by instantiating the random oracle $\mathcal{O}$ with a cryptographic hash function.

Unlike most applications of random oracles in cryptography and complexity theory, including the recent construction of a *search problem* in $\mathsf{BQP} \setminus \mathsf{BPP}$ [YZ22], here we make a non-black-box use of the oracle implementation. This is also the case for some previous cryptographic applications of the random oracle methodology, e.g., in the context of incrementally verifiable computation [Val08, CCS22].

9 Acknowledgements

We thank the anonymous STOC reviewers, Nir Bitansky, Ran Canetti, Itai Dinur, Russell Impagliazzo, Xiao Liang, Alex Lombardi, Raghu Meka, Moni Naor, Ron Rothblum, Avishay Tal, and Vinod Vaikuntanathan for helpful comments, pointers, and suggestions. This research was supported in part from a Simons Investigator Award, DARPA SIEVE award, NTT Research, NSF Frontier Award 1413955, BSF grant 2018393, a Xerox Faculty Research Award, a Google Faculty Research Award, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024. Y. Ishai was additionally supported by ERC Project NTSC (742754), and ISF grant 2774/20. E. Kushilevitz was additionally supported by ISF grant 2774/20.

10 References

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for MPC, and more. In *ESORICS 2019, Part II*, pages 151–171, 2019.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT 2016, Part I*, pages 191–219, 2016.

[Bar] Boaz Barak. Why do we care about random oracles? `https://www.boazbarak.org/Courses/avg_case_depth.pdf`. Accessed: 2023-12-27.

[BDK20] Marshall Ball, Dana Dachman-Soled, and Mukul Kulkarni. New techniques for zero-knowledge: Leveraging inefficient provers to reduce assumptions, interaction, and trust. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 674–703. Springer, 2020.

[BDV21] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure versus hardness through the obfuscation lens. *SIAM J. Comput.*, 50(1):98–144, 2021.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[BG81] Charles H Bennett and John Gill. Relative to a random oracle A, $\mathsf{P}^{\mathrm{A}} \neq \mathsf{NP}^{\mathrm{A}} \neq \mathsf{co\text{-}NP}^{\mathrm{A}}$ with probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981.

[BI87] Manuel Blum and Russell Impagliazzo. Generic oracles and oracle classes (extended abstract). In *FOCS 1987*, pages 118–126, 1987.

[BP15] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 401–427. Springer, 2015.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 474–502, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS 1993*, pages 62–73, 1993.

[Bra79] Gilles Brassard. Relativized cryptography. In *FOCS 1979*, pages 383–391, 1979.

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *J. Cryptol.*, 9(3):149–166, 1996.

[CCG+94] Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *J. Comput. Syst. Sci.*, 49(1):24–39, 1994.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019.

[CCS22] Megan Chen, Alessandro Chiesa, and Nicholas Spooner. On succinct non-interactive arguments in relativized worlds. In *EUROCRYPT 2022, Part II*, pages 336–366, 2022.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CL18] Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 476–506. Springer, 2018.

[Con92] Anne Condon. The complexity of stochastic games. *Inf. Comput.*, 96(2):203–224, 1992.

[DGK17] Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 473–495, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[DH12] Frank Den Hollander. Probability theory: the coupling method. `https://mathematicaster.org/teaching/lcs22/hollander_coupling.pdf`, 2012.

[DK18] Apoorvaa Deshpande and Yael Kalai. Proofs of ignorance and applications to 2-message witness hiding. Cryptology ePrint Archive, Paper 2018/896, 2018. `https://eprint.iacr.org/2018/896`.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[ESY84] Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Inf. Control.*, 61(2):159–173, 1984.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd Annual ACM Symposium on Theory of Computing*, pages 416–426, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM journal on Computing*, 35(1):217–246, 2005.

[GIK+23] Riddhi Ghosal, Yuval Ishai, Alexis Korb, Eyal Kushilevitz, Paul Lou, and Amit Sahai. Hard languages in NP ∩ conp and NIZK proofs from unstructured hardness. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1243–1256. ACM, 2023.

[GLN11] Oded Goldreich, Leonid A. Levin, and Noam Nisan. On constructing 1-1 one-way functions. In Oded Goldreich, editor, *Studies in Complexity and Cryptography.*, volume 6650 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 2011.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

[HI85] Juris Hartmanis and Neil Immerman. On complete problems for np$\cap$conp. In Wilfried Brauer, editor, *Automata, Languages and Programming, 12th Colloquium, Nafplion, Greece, July 15-19, 1985, Proceedings*, volume 194 of *Lecture Notes in Computer Science*, pages 250–259. Springer, 1985.

[HLR07] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 169–186, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.

[HNY17] Pavel Hubácek, Moni Naor, and Eylon Yogev. The journey from NP to TFNP hardness. In Christos H. Papadimitriou, editor, *ITCS 2017: 8th Innovations in Theoretical Computer Science Conference*, volume 4266, pages 60:1–60:21, Berkeley, CA, USA, January 9–11, 2017. LIPIcs.

[IKOS10] Yuval Ishai, Abishek Kumarasubramanian, Claudio Orlandi, and Amit Sahai. On invertible sampling and adaptive security. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 466–482, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[IN88] Russell Impagliazzo and Moni Naor. Decision trees and downward closures. In *Proceedings: Third Annual Structure in Complexity Theory Conference, Georgetown Uni-*

versity, Washington, D. C., USA, June 14-17, 1988, pages 29–38. IEEE Computer Society, 1988.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 44–61. ACM, 1989.

[IV19] Vincenzo Iovino and Ivan Visconti. Non-interactive zero knowledge proofs in the random oracle model. In Claude Carlet, Sylvain Guilley, Abderrahmane Nitaj, and El Mamoun Souidi, editors, *Codes, Cryptology and Information Security - Third International Conference, C2SI 2019, Rabat, Morocco, April 22-24, 2019, Proceedings - In Honor of Said El Hajji*, volume 11445 of *Lecture Notes in Computer Science*, pages 118–141. Springer, 2019.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2021.

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 158–189, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[KZ20] Benjamin Kuykendall and Mark Zhandry. Towards non-interactive witness hiding. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 627–656, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

[Lud95] Walter Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Inf. Comput.*, 117(1):151–155, 1995.

[MV15] Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. *J. ACM*, 62(6):46:1–46:29, 2015.

[Nao96] Moni Naor. Evaluation may be easier than generation (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 74–83. ACM, 1996.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[Ps05] Rafael Pass and Abhi shelat. Unconditional characterizations of non-interactive zero-knowledge. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 118–134, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[RSS21] Alon Rosen, Gil Segev, and Ido Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? *J. Cryptol.*, 34(1):8, 2021.

[RST15] Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1030–1048. IEEE Computer Society, 2015.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[Rud88] Steven Rudich. *Limits on the provable consequences of one-way functions*. PhD thesis, UC Berkeley, 1988.

[Sip82] Michael Sipser. On relativization and the existence of complete sets. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings*, volume 140 of *Lecture Notes in Computer Science*, pages 523–531. Springer, 1982.

[Sta] Reasons to believe $\mathsf{P} \neq \mathsf{NP} \cap \mathsf{coNP}$ (or not). `https://cstheory.stackexchange.com/questions/20021/reasons-to-believe-p-ne-np-cap-conp-or-not`. Accessed: 2023-03-19.

[Tao] Terry Tao. The dichotomy between structure and randomness. `https://www.math.ucla.edu/~tao/preprints/Slides/icmslides2.pdf`. Accessed: 2023-12-27.

[Tar89] Gábor Tardos. Query complexity, or why is it difficult to seperate $\mathsf{NP}^{\mathsf{A}} \cap \mathsf{coNP}^{\mathsf{A}}$ from P^{A} by random oracles A? *Comb.*, 9(4):385–392, 1989.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.

[YZ22] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 69–74. IEEE, 2022.

A Preliminaries Continued

Lemma A.1 (Maximal Coupling [DH12])**.** *Let $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$ be two sequence of independent random variables on probability spaces $\Omega_1, \ldots, \Omega_n$ respectively. The maximal coupling theorem states that there exists random variables $A' = (A'_1, \ldots, A'_n)$ and $B' = (B'_1, \ldots, B'_n)$ on the product probability space $\Omega_1 \times \ldots \times \Omega_n$ such that,*

- *$A'_1, \ldots, A'_n$ are independent random variables on the probability space $\Omega_1, \ldots \Omega_n$ respectively. The same holds for $B'_1, \ldots, B'_n$.*
- *$A = (A_1, \ldots, A_n)$ and A' (similarly $B = (B_1, \ldots, B_n)$ and B') are identically distributed.*
- *$\forall i \in [n],\ \Delta(A_i, B_i) = \Pr[A'_i \neq B'_i]$ and $\Delta(A, B) = \Pr[A' \neq B']$.*

Lemma A.2. *If $A = (A_1, \ldots, A_n)$ and $B = (B_1, \ldots, B_n)$ are sequences of independent random variables such that*

$$\forall i \in [n],\ \Delta(A_i, B_i) \leq \varepsilon$$

then

$$\Delta((A_1, \ldots, A_n), (B_1, \ldots, B_n)) \leq n\epsilon$$

Proof. Using Lemma A.1, we have,

$$\begin{aligned}
\Delta(A, B) &= \Pr[A' \neq B'] \\
&= \Pr[A'_1 \neq B'_1 \text{ or } \ldots \text{ or } A'_n \neq B'_n] && \text{Follows from the first point of Lemma A.1} \\
&\leq \sum_i \Pr[A'_i \neq B'_i] && \text{Union Bound} \\
&= \sum_i \Delta(A_i, B_i) && \text{Follows from third point of Lemma A.1}
\end{aligned}$$

□

B A Random Oracle is a $\left(1 - \frac{1}{e}\right)$-Dense-PRHF

In this section, we show that a length-preserving random oracle satisfies all of the properties of a $\left(1 - \frac{1}{e}\right)$-Dense-PRHF with probability 1.

First, a random oracle satisfies the pseudorandomness requirement of PRHF.

Lemma B.1 (Theorem 6 of [DGK17, GGKT05])**.** *For all polynomial-query-bounded oracle Turing machines $\mathcal{A}$, and all $n \in \mathbb{N}$*

$$\left| \Pr_{\mathcal{O}_n, x \leftarrow \{0,1\}^n} \left[\mathcal{A}^{\mathcal{O}_n}\left(\mathcal{O}(x)\right)\right] - \Pr_{\mathcal{O}_n, y \leftarrow \{0,1\}^n} \left[\mathcal{A}^{\mathcal{O}_n}\left(y\right)\right] \right| \leq \mathsf{negl}(n)$$

where $\mathcal{O}_n$ is a random length-preserving oracle.

Secondly, a random oracle satisfies the pre-image pseudorandomness property.

Lemma B.2. *For any polynomially-query-bounded adversary $\mathcal{A}$ making at-most $q = \mathsf{poly}(n)$ oracle queries.*

$$\left| \Pr_{(x,y) \leftarrow \mathcal{D}_0(1^n), \mathcal{O}_n} [\mathcal{A}^{\mathcal{O}_n}(x, y) = 1] - \Pr_{(x,y) \leftarrow \mathcal{D}_1(1^n), \mathcal{O}_n} [\mathcal{A}^{\mathcal{O}_n}(x, y) = 1] \right| \leq \mathsf{negl}(n),$$

where $\mathcal{O}_n : \{0,1\}^n \to \{0,1\}^n$, is chosen uniformly at random and

$\mathcal{D}_0(1^n)$:

1. $x \leftarrow \{0,1\}^n$
2. $y = \mathcal{O}_n(x)$
3. *Output* (x, y)

$\mathcal{D}_1(1^n)$:

1. $y \leftarrow \mathsf{Image}_n$ *where* $\mathsf{Image}_n = \{y \in \{0,1\}^n : \exists x \in \{0,1\}^n,\ y = \mathcal{O}_n(x)\}$
2. $x \leftarrow \mathsf{PreImage}_n(y^*)$ *where* $\mathsf{PreImage}_n(y) = \{x \in \{0,1\}^n : \mathcal{O}_n(x) = y\}$
3. *Output* (x, y)

Proof. Our proof uses a sequence of hybrids. Intuitively, we would like to exploit the fact that an adversary only gets polynomially many oracle queries.

Assume wlog that $\mathcal{A}$ makes oracle queries only after it receives (x, y). This works because one can design the experiment to out the pair (x, y) as soon as $\mathcal{A}$ makes the first oracle query.

Hybrid H_0: This is the case when $\mathcal{A}$ interacts with $\mathcal{D}_0$.

1. $x \leftarrow \{0,1\}^n$
2. $y = \mathcal{O}_n(x)$
3. Send (x, y) to $\mathcal{A}$

Upon Oracle query a to $\mathcal{O}_n$:

- Return $\mathcal{O}_n(a)$

Hybrid H_1: Remains same as Hybrid H_0 for sampling (x, y) except we sample y at random from the co-domain.

1. $x \leftarrow \{0,1\}^n$
2. $y \leftarrow \{0,1\}^n$
3. Send (x, y) to $\mathcal{A}$

Upon Oracle query a to $\mathcal{O}_n$:

1. If $a == x$ then return y
2. Else return $\mathcal{O}_n(a)$.

Note that hybrids H_0 and H_1 are identically distributed over the choice of the random oracle. Hence,

$$\left| \Pr_{(x,y)\leftarrow\mathcal{D}_0(1^n),\mathcal{O}_n}[\mathcal{A}^{\mathcal{O}_n}_{H_0}(x, y) = 1] - \Pr_{(x,y)\leftarrow\mathcal{D}_0(1^n),\mathcal{O}_n}[\mathcal{A}^{\mathcal{O}_n}_{H_1}(x, y) = 1] \right| = 0.$$

Hybrid H_2: Here we sample y from the image set of $\mathcal{O}_n$ instead of the entire codomain.

1. $y \leftarrow \mathsf{Image}_n$
2. $x \leftarrow \{0,1\}^n$
3. Send (x, y) to $\mathcal{A}$

Upon Oracle query a to $\mathcal{O}_n$:

1. If $a == x$ then return y
2. Else return $\mathcal{O}_n(a)$.

Let Y be a random variable corresponding to the sampling from the image set. Then the probability that a fixed y is chosen over the randomness of the oracle is,

$$\Pr_{Y,\mathcal{O}_n}[Y = y] = \Pr_{Y}[Y = y | y \in \mathsf{Image}_n] \Pr_{\mathcal{O}_n}[y \in \mathsf{Image}_n].$$

Now, $\Pr_{\mathcal{O}_n}[y \in \mathsf{Image}_n]$=$\Pr_{\mathcal{O}_n}[\exists x s.t. \mathcal{O}_n(x) = y] = 1 - \frac{1}{e}$. Thus
Thus,

$$\Pr_{Y,\mathcal{O}_n}[Y = y] = \frac{1}{\mathsf{Image_n}} \left(1 - \frac{1}{e}\right).$$

Using Lemma B.3 we have that for all but a negligible fraction of the random oracle, $|\mathsf{Image}_n| \in \left((1 - e^{-1})2^n - \mathsf{negl}(n), (1 - e^{-1})2^n + \mathsf{negl}(n)\right)$. Therefore,

$$|\Pr_{Y,\mathcal{O}_n}[Y = y] - \frac{1}{2^n}| \leq \mathsf{negl}(n).$$

Since the sampling of x and the other oracle query responses given a fixed sampled y is identical in both the hybrids, and that the probability of sampling a y in Hybrid H_1 was 2^{-n}, we have that

$$\left| \Pr_{(x,y)\leftarrow\mathcal{D}_0(1^n),\mathcal{O}_n}[\mathcal{A}^{\mathcal{O}_n}_{H_1}(x, y) = 1] - \Pr_{(x,y)\leftarrow\mathcal{D}_0(1^n),\mathcal{O}_n}[\mathcal{A}^{\mathcal{O}_n}_{H_2}(x, y) = 1] \right| \leq \mathsf{negl}(n).$$

Hybrid H_3: Here we we sample x as a random preimage of y and do not program the random oracle.

1. $y \leftarrow \mathsf{Image}_n$
2. $x \leftarrow \mathsf{Preimage}_n(y)$
3. Send (x, y) to $\mathcal{A}$

Upon Oracle query a to $\mathcal{O}_n$:

1. return $\mathcal{O}_n(a)$.

The view of the adversary in H_2 and H_3 is identical. Thus,

$$\left| \Pr_{(x,y)\leftarrow\mathcal{D}_0(1^n),\mathcal{O}_n}[\mathcal{A}^{\mathcal{O}_n}_{H_2}(x, y) = 1] - \Pr_{(x,y)\leftarrow\mathcal{D}_1(1^n),\mathcal{O}_n}[\mathcal{A}^{\mathcal{O}_n}_{H_3}(x, y) = 1] \right| = 0.$$

Combining the above sequence of equations gives us the desired result:

$$\left| \Pr_{(x,y)\leftarrow\mathcal{D}_0(1^n),\mathcal{O}_n}[\mathcal{A}^{\mathcal{O}_n}(x, y) = 1] - \Pr_{(x,y)\leftarrow\mathcal{D}_1(1^n),\mathcal{O}_n}[\mathcal{A}^{\mathcal{O}_n}(x, y) = 1] \right| \leq \mathsf{negl}(n).$$

□

Finally, a random oracle, when interpreted as a random function from n bits to n bits, is $\left(1-\frac{1}{e}\right)$ dense.

Lemma B.3. *There exist negligible functions $\varepsilon_R, \varepsilon_{\mathsf{dense}}$ such that for all $n \in \mathbb{N}$,*

$$\Pr_{\mathcal{O}_n}\left[p_{\mathcal{O}_n} \in [(1-e^{-1}) - \varepsilon_{\mathsf{dense}}(n), (1-e^{-1}) + \varepsilon_{\mathsf{dense}}(n)]\right] \geq 1 - \varepsilon_R(n)$$

where $p_{\mathcal{O}_n} = \Pr_{y \leftarrow \{0,1\}^n}[\exists x \in \{0,1\}^n \text{ s.t. } \mathcal{O}_n(x) = y]$, and $\mathcal{O}_n : \{0,1\}^n \to \{0,1\}^n$.

Proof. Let $n \in \mathbb{N}$, and let y_i be the i^{th} value in $\{0,1\}^n$. For $i \in [2^n]$, we define random variable Y_i over the choice of $\mathcal{O} \leftarrow \mathsf{ROS}_n$ by

$$Y_i = \begin{cases} 1 & \text{if } \nexists x_i \in \{0,1\}^n \text{ such that } \mathcal{O}(x_i) = y_i \\ 0 & \text{else} \end{cases}$$

Let $Y = \sum_{i=1}^{2^n} Y_i$. Now, by linearity of expectation,

$$\mathbb{E}[Y] = \sum_{i=1}^{2^n} \mathbb{E}[Y_i] = \sum_{i=1}^{2^n} \left(1 - \frac{1}{2^n}\right)^{2^n} = 2^n \cdot \left(1 - \frac{1}{2^n}\right)^{2^n}$$

Furthermore,

$$\begin{aligned}
\mathbb{E}[Y^2] &= \sum_{i,j \in [2^n]} \mathbb{E}[Y_i Y_j] \\
&= \sum_{i,j \in [2^n], i \neq j} \mathbb{E}[Y_i Y_j] + \sum_{i \in [2^n]} \mathbb{E}[Y_i^2] \\
&= \sum_{i,j \in [2^n], i \neq j} \left(1 - \frac{2}{2^n}\right)^{2^n} + \sum_{i \in [2^n]} \mathbb{E}[Y_i] \\
&= 2 \cdot \binom{2^n}{2} \left(1 - \frac{2}{2^n}\right)^{2^n} + 2^n \left(1 - \frac{1}{2^n}\right)^{2^n} \\
&= 2^n(2^n - 1) \left(1 - \frac{2}{2^n}\right)^{2^n} + 2^n \left(1 - \frac{1}{2^n}\right)^{2^n}
\end{aligned}$$

Therefore, since $\left(1 - \frac{1}{n}\right)^n \leq \frac{1}{e} + \mathsf{negl}(n)$ for large enough n,

$$\begin{aligned}
\mathsf{Var}[Y] &= \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 \\
&= 2^n(2^n - 1) \left(1 - \frac{2}{2^n}\right)^{2^n} + 2^n \left(1 - \frac{1}{2^n}\right)^{2^n} - \left(2^n \cdot \left(1 - \frac{1}{2^n}\right)^{2^n}\right)^2 \\
&\leq 2^n(2^n - 1)e^{-2} + 2^n e^{-1} - 2^{2n} e^{-2} + \mathsf{negl}(n) \\
&= 2^n(e^{-1} - e^{-2}) + \mathsf{negl}(n)
\end{aligned}$$

By Chebyshev's inequality,

$$\Pr\left[\left|Y - 2^n e^{-1}\right| > 2^{3n/4}\right] \leq \frac{2^n(e^{-1} - e^{-2}) + \mathsf{negl}(n)}{2^{6n/4}} = \mathsf{negl}(n)$$

Thus,

$$\Pr\left[Y \cdot 2^{-n} \in [e^{-1} - 2^{-n/4}, e^{-1} + 2^{-n/4}]\right] \geq 1 - \mathsf{negl}(n).$$

This immediately implies

$$\Pr\left[1 - Y \cdot 2^{-n} \in [(1 - e^{-1}) - 2^{-n/4}, (1 - e^{-1}) + 2^{-n/4}]\right] \geq 1 - \mathsf{negl}(n)$$

which completes the proof. □

Note: The lemmas above have been written in a way which only allows $\mathcal{A}$ to make oracle queries to an n-bit length preserving oracle ($\mathcal{O}_n$). However, in the random oracle model, the adversary can indeed make queries to $\mathcal{O}$ with arbitrary length inputs. The same proofs do work in this setting as well since oracle responses to query length other than n are completely independent from the input length of interest (i.e., n).

Corollary B.4. *A length-preserving random function $\mathcal{O}$ satisfies the properties of a $\left(1 - \frac{1}{e}\right)$-*Dense PRHF *with probability* 1 *relative to $\mathcal{O}$.*

We call a random oracle $\mathcal{O}$ "bad" if there exists an (oracle dependent) uniform adversary $\mathcal{A}$ that breaks any of the properties in the above three lemmata. This corollary claims that the set of such "bad" oracles has a measure 0. This proof follows from the application of the Borel Cantelli Lemma (cf Lemma 3.6) in the following way: Let us fix a uniform adversary $\mathcal{A}$. For this particular $\mathcal{A}$, the set of corresponding "bad" random oracles has a measure 0. This is obtained by applying the Borel Cantelli Lemma independently on Lemmata B.1, B.2, and B.3 (as used in the proof of Lemma 8.5). Finally, note that there are only countably many uniform adversaries and that a countable union of measure 0 sets also has a measure 0. Hence, by taking a union bound over all uniform adversaries, we conclude the probability that there exists an adversary which breaks any of the PRHF properties over a random choice of $\mathcal{O}$ is 0.

C Completeness and Soundness for NIZK Proofs in the URS model.

In this section, we formally prove completeness (Lemma 7.3) and soundness (Lemma 7.4) for our construction of NIZK proofs in the URS model.

C.1 Completeness

Lemma C.1. *If H_{PRHF} is a δ-*Dense-PRHF *for some* Efficiently-Approximable *constant $\delta \in (0, 1)$ and* ZHB-NIZK$_\gamma$ *with $\gamma = 1 - \delta$ satisfies perfect completeness in the* Z-Tamperable-Hidden-Bits *model, then* URS-NIZK *satisfies statistical completeness in the* URS *model.*

Proof. For all Efficiently-Approximable constants $\gamma \in (0, 1)$ and for large enough[12] n, every possible string $r \in \{0,1\}^{p_{\mathsf{hbs}}(n)}$ can be output by $\mathsf{Gen}_\gamma(1^n)$. Thus, for $\gamma = 1 - \delta$, the perfect completeness of ZHB-NIZK$_\gamma$ implies that for all sufficiently large n, all $x \in \mathcal{L}$ such that $|x| = n$, and all $r \in \{0,1\}^{p_{\mathsf{hbs}}(n)}$,

$$\Pr\left[\mathsf{ZHB}.V_\gamma(x, \mathsf{ZHB}.\mathcal{I}, \{r_i\}_{i \in \mathcal{I}}, \mathsf{ZHB}.\pi) = 1 : (\mathsf{ZHB}.\mathcal{I}, \mathsf{ZHB}.\pi) \leftarrow \mathsf{ZHB}.P_\gamma(r, x)\right] = 1$$

Since an honest URS.V always generates bits $\{\tilde{r}_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}$ that correctly correspond to the string $r \in \{0,1\}^{p_{\mathsf{hbs}}(n)}$ produced by an honest URS.P, then for sufficiently large n, URS.V will always accept when the prover is honest. Thus, URS.V will only reject an honest prover's proof on at most some constant number of inputs. □

[12] For n large enough such that $\mathsf{Approx}_\gamma(1^n) \in (0, 1)$.

C.2 Soundness

Lemma C.2. *If* H_{PRHF} *is a* δ-Dense-PRHF *for some* Efficiently-Approximable *constant* $\delta \in (0,1)$ *and* $\mathsf{ZHB\text{-}NIZK}_\gamma$ *with* $\gamma = 1-\delta$ *satisfies adaptive statistical soundness in the* Z-Tamperable-Hidden-Bits *model, then* URS-NIZK *satisfies adaptive statistical soundness in the* URS *model.*

Proof. Let $X = (X_1, \ldots, X_{p_{\mathsf{hbs}}(n)})$ where each X_i is an independent Bernoulli random variable with $\Pr[X_i = 1] = \hat{\gamma}$ where $\hat{\gamma} = 1 - \mathsf{Approx}_\delta(1^n)$. Then X is a random variable whose distribution is identical to the distribution of hidden bit strings output by $\mathsf{ZHB.Gen}_\gamma(1^n)$.

Similarly, let $Y = (Y_1, \ldots, Y_{p_{\mathsf{hbs}}(n)})$ where each Y_i is an independent Bernoulli random variable with $\Pr[Y_i = 1] = \gamma' = 1-\delta'$ where $\delta' = \Pr_{y \leftarrow \{0,1\}^{\ell(n)}}[\exists x \in \{0,1\}^n \text{ s.t. } H_{\mathsf{PRHF}}(x) = y]]$. Then, Y is a random variable whose distribution is identical to the distribution of the strings $r = r_1 \ldots r_{p_{\mathsf{hbs}}(n)} \in \{0,1\}^{p_{\mathsf{hbs}}(n)}$ generated by the honest $\mathsf{URS}.P$ given a uniform reference string $\mathsf{urs} \leftarrow \{0,1\}^{p_{\mathsf{urs}}(n)}$.

Let $D_{0,n}$ and $D_{1,n}$ denote the distributions followed by X and Y respectively. We will show that the statistical distance between $D_{0,n}$ and $D_{1,n}$ is negligible. Observe that

1. δ' is negligibly close to δ by the δ-Dense property of H_{PRHF}
2. $|\mathsf{Approx}_\delta(1^n) - \delta| \leq 2^{-n}$.

This implies that γ' is negligibly close to $\hat{\gamma}$. Since all the X_i's and Y_i's are mutually independent, it must be that X and Y are independent. Now, applying Lemma 3.5 yields $\Delta(D_{0,n}, D_{1,n}) \leq p_{\mathsf{hbs}}(n) \cdot \mathsf{negl}(n)$ which is negligible.

Now, observe that the verifier $\mathsf{URS}.V(\mathsf{urs}, x, \pi)$ either

- outputs $\mathsf{ZHB}.V_\gamma(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$ for some string f where r is the string that would be generated by an honest prover given urs,
- or rejects if $v_i \neq \perp$ and $H_{\mathsf{PRHF}}(v_i) \neq y_i$.

Assume that $\mathsf{URS}.V$ accepts a proof $\pi = (\mathsf{ZHB}.\mathcal{I}, \{v_i\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi)$. Hence, for each $i \in [p_{\mathsf{hbs}}(n)]$, the cheating prover can set $v_i = \perp$ even if $\exists z \in \{0,1\}^n$ such that $H_{\mathsf{PRHF}}(z) = y_i$. In this way, the prover can forcefully change a '0' in r_i to a '1' in $\tilde{r}_i$. However, the prover cannot change any '1' in r_i to a '0' in $\tilde{r}_i$ since the prover cannot convince the verifier that there actually is a pre-image for y_i if there is no such pre-image. Note that this scenario is identical to the ZeroFlip function used by the cheating Z-Tamperable-Hidden-Bits prover. Thus, it suffices to show that

$$\Pr_{r \leftarrow D_{1,n}}\Big[\exists(x, \mathsf{ZHB}.\mathcal{I}, f, \mathsf{ZHB}.\pi) \text{ such that } x \notin L, |x| = n, f \in \{0,1\}^{|r|}, \\ \text{and } \mathsf{ZHB}.V_\gamma(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) = 1\Big] \leq \mathsf{negl}(n).$$

Now, by the statistical soundness of $\mathsf{ZHB\text{-}NIZK}_\gamma$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $n \in \mathbb{N}$,

$$\Pr_{r \leftarrow D_{0,n}}\Big[\exists(x, \mathsf{ZHB}.\mathcal{I}, f, \mathsf{ZHB}.\pi) \text{ such that } x \notin L, |x| = n, f \in \{0,1\}^{|r|}, \\ \text{and } \mathsf{ZHB}.V_\gamma(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) = 1\Big] \leq \mathsf{negl}(n).$$

Since $D_{0,n}$ and $D_{1,n}$ are statistically close, then for all constants $\gamma \in (0,1)$ and all $n \in \mathbb{N}$ it must be that,

$$\Pr_{r \leftarrow D_{1,n}} \Big[\exists (x, \mathsf{ZHB}.\mathcal{I}, f, \mathsf{ZHB}.\pi) \text{ such that } x \notin L, |x| = n, f \in \{0,1\}^{|r|}, \\ \text{and } \mathsf{ZHB}.V_\gamma(x, \mathsf{ZHB}.\mathcal{I}, \{\tilde{r_i} = \mathsf{ZeroFlip}(r_i, f_i)\}_{i \in \mathsf{ZHB}.\mathcal{I}}, \mathsf{ZHB}.\pi) = 1 \Big] \leq \mathsf{negl}(n).$$

This concludes the proof.

□

D Non-interactive Witness Hiding

For any NP language $\mathcal{L}$, we will consider a statement-witness relation denoted $R_{\mathcal{L}}$ for $\mathcal{L}$, and we define for any $x \in \mathcal{L}$ the set $R_{\mathcal{L}}(x) = \{w : (x,w) \in R_{\mathcal{L}}\}$. We use the notation $\mathcal{L}(x) = \begin{cases} 1 & \text{if } x \in \mathcal{L} \\ 0 & \text{if } x \notin \mathcal{L} \end{cases}$.

Definition D.1 (Non-interactive Witness Hiding Proofs w.r.t. a Distribution)**.** *Let $\mathcal{P}$ be a randomized computationally unbounded algorithm and let $\mathcal{V}$ be a PPT algorithm. For a language $\mathcal{L}_0 \in \mathsf{NP}$, let $\mathcal{D}$ be a probability ensemble where distribution $\mathcal{D}(\lambda)$ has support over $\mathcal{L}_0$. We say that a protocol $\Pi = (\mathcal{P}, \mathcal{V})$ is non-interactive witness hiding (NIWH) proof system with respect to a probability ensemble $\mathcal{D}$ if the following three properties hold:*

1. ***(Perfect) Completeness**: For all $x \in \mathcal{L}_0$, for all $\lambda \in \mathbb{N}$*

$$\Pr\left[\begin{array}{c} \mathcal{P}(1^\lambda, x) \to \pi, \\ \mathcal{V}(1^\lambda, x, \pi) \to 1 \end{array} \right] = 1$$

   where the probability is over the coins of $\mathcal{P}$ and $\mathcal{V}$.

2. ***(Perfect) Soundness**: For all $x \notin \mathcal{L}_0$, and for all $\lambda \in \mathbb{N}$, there does not exist $\pi \in \{0,1\}^*$ such that $\mathcal{V}(1^\lambda, x, \pi) = 1$.*

3. ***(Computational) Witness Hiding**: If for all polynomial-sized $\mathcal{A}$ there exists a negligible function $\mu : \mathbb{N} \to [0,1]$ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr_{x \sim \mathcal{D}(\lambda)} \Big[\mathcal{A}(1^\lambda, x) \in R_{\mathcal{L}}(x) \Big] \leq \mu(\lambda),$$

   then for all polynomial-sized $\mathcal{A}'$ there exists a negligible function $\mu' : \mathbb{N} \to [0,1]$ such that for all $\lambda \in \mathbb{N}$

$$\Pr_{x \sim \mathcal{D}(\lambda)} \Big[\mathcal{A}'(1^\lambda, x, \mathcal{P}(x)) \in R_{\mathcal{L}}(x) \Big] \leq \mu'(\lambda).$$

Remark D.2. For similar definitions of witness hiding, see [DK18, JKKR17, KZ20]. As remarked by [KZ20], these definitions are all weaker than the original definition of witness hiding given by [FS90].

Theorem D.3. *If there exists an injective one-way function $f : \{0,1\}^* \to \{0,1\}^*$ and a NIWH proof system $(\mathcal{P}, \mathcal{V})$ for all of NP, then $\mathsf{P} \neq \mathsf{NP} \cap \mathsf{coNP}$.*

Proof. Consider the NP language $\mathcal{L}_0 \coloneqq \{y : \exists x, f(x) = y\}$ and we consider a NIWH proof system $(\mathcal{P}, \mathcal{V})$ for $\mathcal{L}_0$. We now construct a language $\mathcal{L}$ as

$$\mathcal{L} \coloneqq \big\{(y, \pi, i) : \mathcal{V}(y, \pi) = 1 \wedge (\exists x, f(x) = y \wedge x_i = 1)\big\}.$$

The proof follows directly from Lemma D.4 and Lemma D.5 which are stated and proven as follows.

Lemma D.4. $\mathcal{L} \in \mathsf{NP} \cap \mathsf{coNP}$.

Proof. For convenience, observe that $\overline{\mathcal{L}} \coloneqq \{0,1\}^* \setminus \mathcal{L}$ can be expressed as

$$\overline{\mathcal{L}} \coloneqq \big\{(y, \pi, i) : \mathcal{V}(y, \pi) = 0 \vee (\nexists x, f(x) = y \wedge x_i = 1)\big\}.$$

Observe $\mathcal{L} \in \mathsf{NP}$ by constructing the following NP-verifier:

$\mathcal{V}_{\mathsf{NP}}((y, \pi, i), w)$:

1. If $\mathcal{V}(y, \pi) = 1 \wedge f(w) = y \wedge w_i = 1$, then output 1. Else, output 0.

If $(y, \pi, i) \in \mathcal{L}$, then the unique preimage x such that $f(x) = y$ is the witness, on which $\mathcal{V}_{\mathsf{NP}}$ correctly outputs 1. Otherwise, $(y, \pi, i) \notin \mathcal{L}$ and one of three conditions fails by definition of the language, so $\mathcal{V}_{\mathsf{NP}}$ correctly outputs 0.

Observe $\mathcal{L} \in \mathsf{coNP}$ by constructing the following coNP-verifier:

$\mathcal{V}_{\mathsf{coNP}}((y, \pi, i), w)$:

1. If $\mathcal{V}(y, \pi) = 0 \vee (f(w) = y \wedge w_i = 0)$, then output 1. Else, output 0.

For correctness of this coNP verifier, if $(y, \pi, i) \in \overline{\mathcal{L}}$, then we consider the two subcases where either $\mathcal{V}(y, \pi) = 0$ or $\mathcal{V}(y, \pi) = 1$. In the first subcase where $\mathcal{V}(y, \pi) = 0$, by construction $\mathcal{V}_{\mathsf{coNP}}$ correctly outputs 1. In the remaining subcase where $\mathcal{V}(y, \pi) = 1$, by the perfect soundness of $(\mathcal{P}, \mathcal{V})$, $y \in \mathcal{L}_0$ so there must exist a preimage x such that $f(x) = y$. Moreover, $x_i = 0$ since $(y, \pi, i) \in \overline{\mathcal{L}_0}$. Therefore the preimage x serves as the witness and $\mathcal{V}_{\mathsf{coNP}}$ correctly outputs 1 given this witness.

If $(y, \pi, i) \notin \overline{\mathcal{L}}$ then by construction $\mathcal{V}_{\mathsf{coNP}}$ correctly outputs 0. □

Lemma D.5. $\mathcal{L} \notin \mathsf{P}$.

Proof. Consider the probability ensemble $\mathcal{D}$ over $\mathcal{L}_0$ where $\mathcal{D}(\lambda)$ is defined by the following sampling procedure: Sample a uniform random $x \leftarrow \{0,1\}^\lambda$ and output $y \leftarrow f(x)$. The one-wayness of f guarantees that for all polynomial-sized $\mathcal{A}$ there exists a negligible function $\mu : \mathbb{N} \to [0,1]$ such that

$$\Pr_{y \sim \mathcal{D}(\lambda)} [\mathcal{A}(y) \in R_{\mathcal{L}_0}(y)] \leq \mu(\lambda).$$

Then, the witness hiding property of $(\mathcal{P}, \mathcal{V})$ implies that for all polynomial-sized $\mathcal{A}'$ there exists a negligible function $\mu' : \mathbb{N} \to [0,1]$ such that

$$\Pr_{y \sim \mathcal{D}(\lambda)} \big[\mathcal{A}'(y, \mathcal{P}(y)) \in R_{\mathcal{L}_0}(y)\big] \leq \mu'(\lambda).$$

Suppose for sake of contradiction that there exists a polynomial-time decider $\mathcal{M} : \{0,1\}^* \to \{0,1\}$ for $\mathcal{L}$. That is, for all (y, π, i), $\mathcal{M}(y, \pi, i) = \mathcal{L}((y, \pi, i))$. Then we can build a polynomial-sized $\mathcal{B}$ that inverts f when given both y and $\mathcal{P}(y)$ that succeeds with probability 1. For any value $\lambda \in \mathbb{N}$, $\mathcal{B}$ is given an input sample y sampled according to $\mathcal{D}(\lambda)$ and $\pi \leftarrow \mathcal{P}(y)$. $\mathcal{B}$ is defined as follows:

$\mathcal{B}(1^\lambda, y, \pi)$

1. For $i \in [\lambda]$:
   (a) Set $z_i \leftarrow \mathcal{M}(y, \pi, i)$.
2. Output $z = (z_1, \ldots z_\lambda)$.

Since $\mathcal{M}$ is a decider for $\mathcal{L}$, $y \in \mathcal{L}_0$, and $\pi \leftarrow \mathcal{P}(y)$, it follows that $\mathcal{M}(y, \pi, i) = x_i$ where $f(x) = y$. Then $z = x$ and $\mathcal{B}$ recovers x with probability 1 contradicting the witness hiding property of $(\mathcal{P}, \mathcal{V})$. Therefore, no such polynomial-time decider $\mathcal{M}$ can exist and $\mathcal{L} \notin \mathsf{P}$.

□

□

E Average-case Hardness

In Section 8, we considered a language $\mathcal{L}^{\mathcal{O}}$ with worst-case hardness in $\mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}$. We now construct a language $\mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}$ such that with probability 1 over the choice of random oracle $\mathcal{O}$, $\mathcal{L}^{\mathcal{O}}_{\mathsf{avg}} \in \mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}$ and $\mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}$ satisfies average-case hardness with respect to some distribution over inputs.

Let $f' : \{0,1\}^* \to \{0,1\}^*$ be any injective one-way function. Define the injective one-way function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f(x, r) = (f'(x), r)$ so that $\langle x, r \rangle$ is a hardcore predicate for f. Let Π_{NIZK} be a NIZK protocol in the RO model with perfect completeness, almost-sure soundness, and zero-knowledge (see Def. 3.10) for the language $\{y : \exists x, f'(x) = y\}$ and let $q(\cdot)$ be a polynomial bound on all valid proof lengths. Then define the following language:

$$\mathcal{L}^{\mathcal{O}}_{\mathsf{avg}} := \left\{ (y, \pi, r) : (\exists x, f'(x) = y \text{ and } \langle x, r \rangle = 1) \wedge (\Pi_{\mathsf{NIZK}}.\mathcal{V}^{\mathcal{O}}(x, \pi) = 1) \right\}$$

where we consider for any fixed oracle $\mathcal{O}$ the following distribution with the following (inefficient) sampling procedure:

$\mathsf{Samp}^{\mathcal{O}}(1^\lambda)$:

1. Sample uniform random $x \in \{0,1\}^\lambda, r \in \{0,1\}^\lambda$.
2. Set $(y, r) = f(x, r)$.
3. Set $\pi \leftarrow \Pi_{\mathsf{NIZK}}.\mathcal{P}^{\mathcal{O}}(1^\lambda, y)$.
4. Output (y, π, r).

The fact that $\Pr_{\mathcal{O}}\left[\mathcal{L}^{\mathcal{O}}_{\mathsf{avg}} \in \mathsf{NP}^{\mathcal{O}} \cap \mathsf{coNP}^{\mathcal{O}}\right] = 1$ follows identically as before. It remains to show that this language satisfies average-case hardness with respect to $\mathsf{Samp}^{\mathcal{O}}(1^\lambda)$.

Theorem E.1. *Suppose f' is an injective one-way function and let $\mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}$ be as defined above. Then, for all polynomial query bounded oracle-aided* PPT *$\mathcal{A}^{(\cdot)}$, there exists a negligible function $\mu : \mathbb{N} \to [0,1]$ such that for all $\lambda \in \mathbb{N}$*

$$\Pr_{\mathcal{O}, \mathsf{Samp}}\left[\mathcal{A}(y, \pi, r) = \mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}(y, \pi, r) : (y, \pi, r) \leftarrow \mathsf{Samp}^{\mathcal{O}}(1^\lambda)\right] \leq \frac{1}{2} + \mu(\lambda)$$

where $\mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}(y, \pi, r) = 1$ if $(y, \pi, r) \in \mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}$ and $\mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}(y, \pi, r) = 0$ otherwise.

Proof. Suppose for sake of contradiction that there exists a polynomial query bounded oracle-aided PPT $\mathcal{A}^{(\cdot)}$ such that

$$\Pr_{\mathcal{O},\mathsf{Samp}}\left[\mathcal{A}^{\mathcal{O}}(y,\pi,r) = \mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}(y,\pi,r) : (y,\pi,r) \leftarrow \mathsf{Samp}^{\mathcal{O}}(1^\lambda)\right] \geq \frac{1}{2} + \frac{1}{n^c}$$

for some constant $c > 0$. Then we will construct a PPT machine $\mathcal{B}$ that on input $(1^\lambda, (y,r) = f(x,r))$ for a randomly chosen $x \in \{0,1\}^\lambda$ and $r \in \{0,1\}^\lambda$, recovers $\langle x, r\rangle_2$, contradicting the hardcore property of the predicate defined by $r \in \{0,1\}^\lambda$.

$\mathcal{B}(1^\lambda, (y,r))$:

1. Obtain $\pi \leftarrow \mathsf{SimProof}(1^\lambda, (y,r))$.
2. Output $b \leftarrow \mathcal{A}^{\mathsf{SimO}}(1^\lambda, y, r, \pi)$.

To analyze the probability of this reduction outputting $\langle x, r\rangle_2$, where x is the unique preimage of y under f', we consider the following polynomial query bounded oracle-aided (computationally unbounded) machine $\tilde{\mathcal{B}}^{(\cdot)}$:

$\tilde{\mathcal{B}}^{\mathcal{O}}(1^\lambda, (y,r))$:

1. Obtain $\pi \leftarrow \Pi_{\mathsf{NIZK}}.\mathcal{P}^{\mathcal{O}}(1^\lambda, (y,r))$.
2. Output $b \leftarrow \mathcal{A}^{\mathcal{O}}(1^\lambda, y, r, \pi)$.

Then immediately by the zero-knowledge property of Π_{NIZK} (for language $\mathcal{L}_f = \{(y,r) : \exists x, f(x,r) = (y,r)\}$), we have that for all $b' \in \{0,1\}$, for all $(y,r) \in \mathcal{L}_f$ for all sufficiently large $\lambda \in \mathbb{N}$:

$$\left| \Pr_{\text{coins of } \mathcal{B}}[\mathcal{B}(y,r) = \langle x, r\rangle_2 : (y,r) = f(x,r)] - \Pr_{\mathcal{O},\text{ coins of } \tilde{\mathcal{B}}^{(\cdot)}}\left[\tilde{\mathcal{B}}^{\mathcal{O}}(y,r) = \langle x, r\rangle_2 : (y,r) = f(x,r)\right] \right| \leq \mathsf{negl}(\lambda).$$

However, observe that

$$\Pr_{\mathcal{O},\text{ coins of } \tilde{\mathcal{B}}^{(\cdot)},x,r}\left[\tilde{\mathcal{B}}^{\mathcal{O}}(y,r) = \langle x, r\rangle_2 : (y,r) = f(x)\right] = \Pr_{\mathcal{O},\mathsf{Samp}}\left[\mathcal{A}^{\mathcal{O}}(y,\pi,r) = \mathcal{L}^{\mathcal{O}}_{\mathsf{avg}}(y,\pi,r) : (y,\pi,r) \leftarrow \mathsf{Samp}^{\mathcal{O}}(1^\lambda)\right] \geq \frac{1}{2} + \frac{1}{n^c}.$$

Therefore, $\Pr_{\text{coins of } \mathcal{B},x,r}[\mathcal{B}(y,r) = \langle x, r\rangle_2 : (y,r) = f(x)]$ is non-negligibly larger than $1/2$, contradicting the hardcore bit property of f. Therefore, no polynomial query bounded oracle-aided PPT $\mathcal{A}^{(\cdot)}$ with such guessing advantage can exist.

□

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il