On Zero-Knowledge PCPs: Limitations, Simplifications, and Applications*

Yuval Ishai† Mohammad Mahmoody‡ Amit Sahai§ David Xiao¶

Abstract

We revisit the question of *Zero-Knowledge PCPs*, studied by Kilian, Petrank, and Tardos (STOC '97). A ZK-PCP is defined similarly to a standard probabilistically checkable proof (PCP), except that the view of any (possibly malicious) verifier can be efficiently simulated up to a small statistical distance. Kilian *et al.* obtained a ZK-PCP for **NEXP** in which the proof oracle is in $\mathbf{EXP}^{\mathbf{NP}}$. They also obtained a ZK-PCP for **NP** in which the proof oracle is computable in polynomial-time, but this ZK-PCP is only zero-knowledge against *bounded-query* verifiers who make at most an *a priori fixed* polynomial number of queries. The existence of ZK-PCPs for **NP** with efficient oracles and arbitrary polynomial-time malicious verifiers was left open. This question is motivated by the recent line of work on cryptography using tamper-proof hardware tokens: an efficient ZK-PCP (for any language) is *equivalent* to a statistical zero-knowledge proof using only a single stateless token sent to the verifier.

We obtain the following results regarding efficient ZK-PCPs:

Negative Result on Efficient ZK-PCPs. We settle the above question negatively and show that any language or promise problem with efficient ZK-PCPs must be in **SZK** (the class of promise problems with a statistical zero-knowledge *single prover* proof system). Therefore, no **NP**-complete problem can have an efficient ZK-PCP unless $\mathbf{NP} \subseteq \mathbf{SZK}$ (which also implies $\mathbf{NP} \subseteq \mathbf{coAM}$ and the polynomial-time hierarchy collapses).

Simplifying Bounded-Query ZK-PCPs. The bounded-query zero-knowledge PCP of Kilian *et al.* starts from a *weakly-sound* bounded-query ZK-PCP of Dwork *et al.* (CRYPTO '92) and amplifies its soundness by introducing and constructing a new primitive called a *locking scheme* which is an unconditional oracle-based analogue of a commitment scheme. We simplify the ZK-PCP of Kilian *et al.* by presenting an elementary new construction of locking schemes. Our locking scheme is purely combinatorial.

Black-Box Sublinear ZK Arguments via ZK-PCPs. Kilian used PCPs to construct sublinear communication zero-knowledge arguments for **NP** by making *non-black-box* use of collision-resistant hash functions (STOC '92). We show that ZK-PCPs can be used to get black-box variants of this result with improved round complexity, as well as an *unconditional* zero-knowledge variant of Micali's non-interactive CS Proofs (FOCS '94) in the Random Oracle Model.

*This work is based on previous works of Ishai, Mahmoody, and Sahai [IMS12], and Mahmoody and Xiao [MX13].

†Computer Science Department, Technion. yuvali@cs.technion.ac.il. Research done in part while visiting UCLA. Supported by the European Research Council as part of the ERC project CaC (grant 259426), ISF grant 1361/10, and BSF grant 2008411.

‡Computer Science Department, Cornell. mohammad@cs.cornell.edu. Research done in part while visiting UCLA and LIAFA. Supported in part by NSF Awards CNS-1217821 and CCF-0746990, AFOSR Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2- 0211.

§Computer Science Department, UCLA. sahai@cs.ucla.edu.

¶LIAFA, CNRS, Université Paris 7. dxiao@liafa.univ-paris-diderot.fr.

Keywords: Zero-Knowledge, Probabilistically Checkable Proofs, Sublinear Communication.

Contents

1 Introduction

The seminal work of Goldwasser, Micali, and Rackoff [GMR89] changed the classical notion of a mathematical proof by incorporating randomness and interaction. This change was initially motivated by the intriguing possibility of zero knowledge proofs – proofs that carry no extra knowledge other than being convincing. The result of Goldreich, Micali, and Wigderson [GMW91] showed that any **NP** statement can be proved in a zero-knowledge (ZK) manner, making ZK proofs a central tool for cryptographic protocol design; this was later extended by Ben-Or *et al.* [BGG+88] to any language in **PSPACE**. All these fundamental results, however, relied on the assumption that one-way functions exist. Ostrovsky and Wigderson [OW93] showed that computational assumptions (similar to one-way functions) are indeed necessary for zero-knowledge for non-trivial languages.

Motivated by the goal of achieving *unconditionally* secure zero-knowledge proofs for **NP**, Ben-Or, Goldwasser, Kilian and Wigderson [BGKW88] introduced the model of multi-prover interactive proofs (MIP) and presented a perfect ZK protocol for any statement that is provable in the MIP model. Shortly after, Babai, Fortnow, and Lund [BFL90] showed that in fact any language in **NEXP** can be proved in the MIP model. Fortnow, Rompel, and Sipser [FRS94] studied the MIP model further and observed that as a proof system it is equivalent to another model in which an *oracle* encodes a probabilistically checkable proof (PCP) which is queried by an efficient randomized verifier.[1] The difference between a prover and a PCP oracle is that a prover can keep an internal state, and hence its answer to a given question can depend on other questions. Therefore, soundness against a PCP oracle is potentially easier to achieve than soundness against a malicious prover. This line of work culminated in the celebrated PCP theorem [AS98, ALM+98].

Zero-Knowledge PCPs. In this work we study *zero-knowledge proofs* in the PCP model. A zero-knowledge PCP (ZK-PCP) is defined similarly to a standard PCP, except that the view of any (possibly malicious) verifier can be efficiently simulated up to a small statistical distance. It is instructive to note that zero-knowledge PCPs are incomparable to traditional ZK proofs: since the PCP model makes the prover less powerful, achieving soundness may become easier whereas achieving zero-knowledge may become harder.

The original ZK protocol of [GMW91] for **NP** implicitly relies on *honest-verifier* zero-knowledge PCP for the **NP**-complete problem of 3-coloring of graphs. In this PCP the prover takes any 3-coloring of the input graph, randomly permutes the 3 colors, and writes down the colors as the PCP string. The verifier chooses a random edge, reads the colors of the vertices of that edge, and accepts iff the colors are different. This ZK-PCP has two disadvantages: **(1)** it is only zero-knowledge against *honest verifiers* (a malicious verifier can learn whether the colors of two non-adjacent nodes are identical), and **(2)** the soundness error is very large: $1 - 1/m$ where m is the number of edges. Dwork *et al.* [DFK+92],[2] relying on the PCP theorem [ALM+98, AS98], improved the ZK-PCP implicit in [GMW91] in both directions. Their construction implies a ZK-PCP for **NP** of polynomial length and with a constant alphabet size such that: **(1)** the PCP is zero-knowledge against verifiers who ask *any* pair of queries (but not more), and **(2)** the soundness error is constant. However, the soundness error of this ZK-PCP could not be easily reduced further while maintaining ZK

[1] The PCP oracle is often identified with the proof string defined by its truth-table, in which case the output domain of the oracle is referred to as the *PCP alphabet*.

[2] This formulation of the result of [DFK+92] is due to [KPT97].

against malicious verifiers. Furthermore, it could not be made zero-knowledge against arbitrary polynomial-time verifiers, simply because it has polynomial length and a malicious verifier could read the entire proof string.

Kilian, Petrank, and Tardos [KPT97] were the first to explicitly study the power of ZK-PCPs with malicious verifiers. Their work shows how to get around the above limitations, resulting in two kinds of ZK-PCPs with security against malicious verifiers. For the case of languages in **NP**, [KPT97] obtain a PCP of polynomial length over a binary alphabet which is zero-knowledge with negligible soundness error against malicious verifiers who are limited to ask only up to any *fixed* polynomial $p(|x|)$ number of queries, whereas the honest verifier only asks polylog$(|x|)$ queries to verify the PCP. (The length of the PCP string can be polynomially larger than $p(|x|)$.) We call such PCPs *bounded-query* ZK. Simpler constructions of bounded-query PCPs for **NP** can be obtained from protocols for secure multiparty computation [IKOS09], though these constructions require either a large alphabet or a large number of queries.

For the case of languages in **NEXP**, a scaled up version of the bounded-query construction from [KPT97] yields a ZK-PCP in which honest verifiers are efficient (*i.e.* run in poly$(|x|)$ time), but soundness holds against *arbitrary* polynomial time verifiers. However, the PCP oracle in this case cannot be computed in polynomial time even for languages in **NP**. (By "computable in polynomial time" we mean that the oracle outputs a polynomial-time computable function of its secret randomness, the input x, the **NP**-witness, and the verifier's query.) This is inherent to the approach of [KPT97], as it requires the entropy of the PCP oracle to be bigger than the number of queries made by a malicious verifier.

The above state of affairs leaves open the following natural question.

> **Main Question:** *Are there efficiently computable PCPs for* **NP** *which are statistically zero-knowledge against any polynomial-time verifier?*

2 Our Results and Techniques

2.1 Efficient ZK-PCPs are Limited to SZK

Our main theorem provides a negative answer to the main question above. In one of the original publications on which this work is based, Ishai, Mahmoody, and Sahai [IMS12] proved that any language or promise problem L with an efficient ZK-PCP *where the honest verifier's queries are non-adaptive* must satisfy $L \in \mathbf{coAM}$. Therefore, **NP** does not have such efficient ZK-PCPs unless the polynomial-time hierarchy collapses [BHZ87]. Thus, the main question above remained open whether there exist efficient ZK-PCPs for **NP** if we allow the verifier to be adaptive. In this paper we resolve this question in the negative; namely we prove:

Theorem 2.1 (Main Result)**.** *Any promise problem L with an efficient ZK-PCP is in* **SZK**.

This strengthens the negative result of [IMS12] in two ways: **(1)** we lift the restriction that the verifier be non-adaptive, and **(2)** we can conclude that $L \in \mathbf{SZK}$ which is stronger than $L \in \mathbf{AM} \cap \mathbf{coAM}$, since it is known that $\mathbf{SZK} \subseteq \mathbf{AM} \cap \mathbf{coAM}$ [For89a, AH91]. Finally, we emphasize that Theorem 2.1 does *not* assume that the simulation is black-box.

2.1.1 Motivation and Related Work

Basing Cryptography on Tamper-Proof Hardware. A recent line of work in cryptography [Kat07, MS08, CGS08, GKR08, GIS+10, Kol10, GIMS10, BCG+11, DMMQN13] studies the possibility of obtaining secure protocols in an extended model of interaction in which the parties are allowed to generate and exchange tamper-proof hardware tokens. More concretely, each party can locally construct a circuit, put it inside a tamper-proof token, and send it to another party. The receiver of a token is only allowed to use it as a black-box by giving inputs to the token and receiving the output. (If the token is stateful, asking the same query twice might lead to different answers.) The fact that the receiver of a token has no a-priori guarantee that the token is indeed well formed makes designing protocols in this model non-trivial. The work of Goyal *et al.* [GIS+10] showed that any two-party functionality (*e.g.* zero-knowledge proofs) can be carried out securely in this model without relying on computational assumptions. The solution of [GIS+10] uses *stateful* tokens, which makes it vulnerable to "resetting attacks" where a malicious party receiving a token resets it to its initial state, say, by cutting off its power.

Goyal, Ishai, Mahmoody, and Sahai [GIMS10] showed that unconditional (statistical) ZK proofs for **NP** exist using a *single stateless* token sent from the prover to the verifier followed by four messages exchanged between the verifier and the prover.[3] The protocol of [GIMS10] fell into the *Interactive PCP* model of Kalai and Raz [KR08] where a verifier interacts with a stateless PCP and a stateful prover simultaneously. The main question left open by [GIMS10] is the possibility of achieving similar zero-knowledge protocols without the additional interaction, namely by having the prover only send a single stateless token to the verifier. It is easy to see that the latter question is indeed equivalent to our main question above. Our Theorem 2.1 rules out the possibility of such protocols for all of **NP** unless the polynomial hierarchy collapses.

Resettable Zero-Knowledge. The notion of *resettable* zero-knowledge single prover proof systems, introduced by Canetti *et al.* [CGGM00], is comparably stronger than the notion of ZK-PCPs. A resettable-ZK proof is, essentially, a ZK-PCP where soundness is required to hold even against adaptive cheating provers who may manipulate their answers based on the queries they have already seen (rather than using answers fixed ahead of the queries being received). Therefore, one can interpret the result of Canetti *et al.* as *efficient* PCPs for **NP** that are *computational* zero-knowledge (based on computational hardness assumptions). Thus, in the case of computational zero knowledge, the question is resolved in the positive direction (under plausible intractability assumptions). Similarly, it would be possible to get statistical zero-knowledge probabilistically checkable *arguments* for **NP** (with soundness against computationally bounded stateful provers) if one can construct resettable statistical zero-knowledge arguments for **NP**. The work of [GMOS07] shows the existence of a closely related object, namely *concurrent* statistical zero-knowledge arguments for all of **NP**. But recall that in this work, both the zero-knowledge and the soundness are statistical, and so the abovementioned results do not resolve our main question.

Recently, Garg *et al.* [GOVW12] showed that *efficient* resettable *statistical* ZK proof systems exist for non-trivial languages (*e.g.* Quadratic Residuosity) based on computational assumptions. Therefore under the same assumptions, these languages also possess efficient ZK-PCPs. Garg *et al.* also showed that assuming the existence of exponentially hard one-way functions, statistical zero-knowledge proof systems can be made resettable. Unfortunately, this transformation does not

[3] The proof system of [GIMS10] is sound also against unbounded provers who might put a stateful functionality inside the token.

preserve the efficiency of the prover. Therefore, even though by the works of Micciancio, Ong, and Vadhan [MV03, OV08] we know that $\mathbf{SZK} \cap \mathbf{MA}$ has statistical zero-knowledge proofs with an efficient prover, the result of [GOVW12] does not necessarily preserve this efficiency. However, if one can transform any statistical ZK proof into a resettable statistical ZK proof without losing the efficiency of the prover, such a result together with [GOVW12] and our main result of Theorem 2.1 would imply that the problems with efficient ZK-PCPs are exactly those in $\mathbf{SZK} \cap \mathbf{MA}$.

2.1.2 Technique

In this section we describe the ideas and techniques behind the proof of Theorem 2.1 and give a comparison to the approach of [IMS12]. In the following let us assume for notational simplicity that L is a language; the idea is identical for general promise problems.

If L has a ZK-PCP, one naive approach to decide L using its simulator is to run the simulator to obtain a view $\nu = (r, (q_1, a_1), \ldots, (q_m, a_m))$, where r is the private randomness of the verifier, and q_i (resp. a_i) is the i'th PCP query (resp. answer), and then accept iff ν is an accepting view. This approach would obtain accepting views if $x \in L$ due to the zero-knowledge property, but there is no guarantee about the case $x \notin L$.

Our proof will show that if in addition to making sure that the view is accepting, we do some extra tests on the distribution of the simulated view, then this will allow us to decide L. Suppose for a moment that the ZK-PCP is deterministic, *i.e.* on an input x the prover deterministically generates a proof π. (Of course it is known that ZK with deterministic provers cannot exist for non-trivial languages [GO94]. We make this simplification here only to make our proof sketch easier to describe, and we will argue below how one can do away with this simplification.) We will show that when the ZK-PCP is deterministic, it suffices to run the simulator and check that the generated view is accepting and to check some entropy-related properties of the view which in our case happen to be a computational task in $\mathbf{SZK}$. Let $(\mathbf{r}, (\mathbf{q}_1, \mathbf{a}_1), \ldots, (\mathbf{q}_m, \mathbf{a}_m))$ be the distribution of views generated when running the simulator for the honest verifier. By the ZK property, this is statistically close to the view of an honest verifier interacting with the honest prover on YES instances (*i.e.* $x \in L$). Let $\mathbf{j}$ be uniform in $[m]$ and consider the distribution $(\mathbf{q_j}, \mathbf{a_j})$.

We argue that to decide the language it suffices to check first that the simulated transcript is accepting, and second that $\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j})$ is small. On YES instances, the simulated transcript is almost surely accepting because of the ZK property, and furthermore $\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j}) = 0$ because the simulated proof is deterministic. On the other hand, on NO instances (*i.e.* $x \notin L$), we will show that if $\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j})$ is sufficiently small, then the simulated transcript is statistically close to an interaction between an honest verifier and a *proof* sampled as follows: for each q, the corresponding answer of the proof is sampled according to the distribution $\mathbf{a_j} \mid \mathbf{q_j} = q$. Therefore, by the soundness condition of the ZK-PCP, it follows that the transcript must be rejecting with high probability. It is clear that checking that the simulated transcript is accepting is in $\mathbf{BPP}$, while checking that $\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j})$ is small is a conditional entropy approximation problem, which is in $\mathbf{SZK}$ [Vad06].

To generalize the above argument to the case of randomized ZK-PCPs, we use the following argument (which is a stronger version of an argument that first appeared in [IMS12]). Any efficiently computable PCP (as a random variable describing its truth table) has polynomial entropy. Therefore if we repeat the honest verifier ℓ times where ℓ is a polynomial sufficiently larger than the size of the circuit computing the PCP, we can essentially "exhaust" the entropy of the proof observed by the next independent verification over the same PCP. This allows us to prove that $\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j})$ is small *conditioned on* the PCP answers observed in the first ℓ verifications. Interestingly, this

argument when applied to *a random* query index $\mathbf{j}$ (which is the index distribution we use—see Lemma 4.4) is rather delicate and heavily relies on the fact that PCPs are fixed; the statement is not true for interactive proofs, where the answers may depend on, say, the order of the queries.

We finally note that even after making sure that the simulator is choosing its PCP answers close to some fixed oracle, it still might be the case that for NO instances it does not run an honest execution of the verifier against this PCP and somehow manages to generate accepting views for NO instances as well. To complete the proof, one final technicality that we check is that the random coins $\mathbf{r}$ generated by the simulator are indeed close to uniform conditioned on the ℓ previously sampled views. (They are guaranteed to be so on YES instances by ZK, but may not be on NO instances.) This latter task is also reducible to the conditional entropy approximation problem.

In contrast to our approach of deciding the language using the simulator can be done in **SZK** by a direct reduction to a problem in **SZK**, [IMS12] showed how to "extract" a PCP from the simulator and then run the honest verifier against this extracted PCP. Since the extraction process requires sampling from inefficiently samplable distributions, this task is accomplished with the aid of an all-powerful yet untrusted prover (this is how they obtain the conclusion that the language is in $\mathbf{AM} \cap \mathbf{coAM}$). This makes our approach conceptually different from the approach of [IMS12].

2.2 Simplifying Bounded-Query ZK-PCPs

Our second contribution is a simplification of the ZK-PCP construction of Kilian *et al.* [KPT97]. The construction of [KPT97] starts from the weakly-sound bounded-query ZK-PCP of [DFK+92] and compiles it into a PCP which is zero-knowledge against malicious verifiers of bounded query complexity. The weakly-sound PCP of [DFK+92] is zero-knowledge against any k (possibly adaptive) queries, but suffers from the soundness error $1 - 1/\operatorname{poly}(k)$. The main tool introduced and employed in the compiler of [KPT97] is called a "locking scheme", which is an analogue of a commitment scheme in the PCP model. In a locking scheme a sender holds a secret w and randomly encodes it into an oracle σ_w that can be accessed by the receiver R. The efficient receiver should not be able to learn any information about w through its oracle access to σ_w. On the other hand, the sender can later send a key to the receiver to decommit the value w. The protocol should guarantee that the sender is not able to change his mind about the value w after constructing the oracle σ_w.[4]

Kilian *et al.* [KPT97] showed an elegant way of using locking schemes to convert a ZK-PCP with $1 - 1/\operatorname{poly}(k)$ soundness error into a standard (inefficient) ZK-PCP of constant or even negligible error. Unfortunately, the locking scheme of [KPT97] which forms the main technical ingredient of their ZK-PCP constructions is quite complicated to describe and analyze (pages 6 to 12 there) and uses ad-hoc algebraic techniques.

In Section 5, we formally present and analyze a simple combinatorial construction of a locking scheme which can be viewed as a noninteractive implementation of Naor's commitment scheme [Nao91] in the PCP model.

[4] In other words, a locking scheme can be thought of as a commitment scheme with statistical security guarantees and minimal interaction such that during its commitment phase the sender sends only a single stateless tamper-proof token (containing the oracle σ_w) to the receiver.

2.2.1 Motivation and Related Work

Most applications of ZK-PCPs considered in this work either require the stronger unbounded variant (see Section 2.1.1) or alternatively can rely on an honest-verifier variant (see Section 2.3), which is easier to realize. However, efficient bounded-query ZK-PCPs with security against *malicious* verifiers can also be motivated by natural application scenarios. For instance, one can consider the goal of distributing an **NP**-witness among many servers in a way that simultaneously supports a very efficient verification (corresponding to the work of the honest verifier) and secrecy in the presence of a large number of colluding servers (corresponding to the query bound of a malicious verifier). One can also consider a "time-lock zero-knowledge proof" in which a stateless hardware token contains an embedded witness which can be very quickly validated but requires a lot of time to extract.

Another motivation behind our simpler locking schemes comes from the line of work aiming at simplifying PCP constructions and making them combinatorial. The main algebraic and technical components in the final PCP construction of Kilian *et al.* [KPT97] are **(1)** the PCP theorem of [ALM+98, AS98] (which comes in through the construction of [DFK+92]) and **(2)** the locking scheme of [KPT97]. The first (more important) component was considerably simplified by Dinur, and here we give a simplified version of the second component. (For a more extensive survey of this line of research see [Mei09] and the references therein.)

2.2.2 Technique

In the following we describe the main ideas which underly our construction. We start by reviewing Naor's commitment scheme. In this commitment scheme, the parties have access to a pseudorandom generator $f\colon \{0,1\}^n \mapsto \{0,1\}^{3n}$ and the protocol works as follows:

The receiver chooses a random "shift" $r \leftarrow \{0,1\}^{3n}$ and sends it to the sender. The sender, who holds a secret input bit b, chooses a random seed $s \leftarrow \{0,1\}^n$ and sends $f(s) + b \cdot r = t$ to the receiver (the addition and multiplication are componentwise over the binary field). In the decommitment phase the sender simply sends (b, s) to the receiver, and the receiver makes sure that $f(s) + b \cdot r = t$ holds to accept the decommitted value.

The binding property holds because the support set of $f(s)$ is of size at most $|f(\{0,1\}^n)| \leq 2^n$, and a random shift $r \leftarrow \{0,1\}^{3n}$ with overwhelming probability of at least $1 - 2^n \cdot 2^n \cdot 2^{-3n} = 1 - 2^{-n}$ will have the property that $f(\{0,1\}^n) \cap (f(\{0,1\}^n) + r) = \varnothing$. Thus for such "good" r, by sending t to the receiver the sender will be bound to at most one possible value of b (regardless of the structure of the function f). The hiding property of the scheme reduces in a *black-box* way to the pseudorandomness of $f(\mathbf{U}_n)$. Namely, if an efficient receiver $\widehat{R}$ can distinguish between $f(s) + r$ and $f(s) + r \cdot b$, another efficient algorithm D who uses $\widehat{R}$ internally is able to distinguish between $f(\mathbf{U}_n)$ and $\mathbf{U}_{3n}$. Thus, if the function f is random, the scheme will be *statistically* hiding against receivers who ask at most $\mathrm{poly}(n)$ oracle queries to f. The reason is that a random function f mapping $\{0,1\}^n$ to random values in $\{0,1\}^{3n}$ is statistically indistinguishable from a truly random function as long as the distinguisher is bound to ask at most $2^{o(n)}$ queries to f.

The above observation about the hiding property of Naor's commitment scheme means that if, in the second round of the commitment phase, the sender chooses f to be a truly random function and sends $f(s) + b \cdot r$ to the receiver as well as (providing oracle access to) $f(\cdot)$, then we get a secure (inefficient) commitment scheme in the *interactive* PCP model without relying on any

computational assumption.[5] In our construction of locking schemes we show how to eliminate the first initial message r of the receiver and emulate the role of this shift r by a few more queries asked by the receiver and more structure in the locking oracle.

2.3 Black-Box Sublinear ZK Arguments

Our third contribution is to obtain the first *black-box* constructions of sublinear zero-knowledge arguments for **NP**. This is achieved by using bounded-query efficient ZK-PCPs for **NP**.

Kilian [Kil92], relying on the PCP construction of [BFLS91],[6] proved that assuming the existence of exponentially-hard collision-resistant hash functions (CRHF) and 2-message statistically-hiding commitments, one can construct a (6-message) statistical ZK argument for **NP** with polylog(n) communication complexity (where n is the input length). Later on, Damgård *et al.* [DPP97] showed that 2-message statistically-hiding commitments can be obtained from any CRHF, which made the existence of exponentially hard CRHF sufficient for the construction of Kilian. Micali [Mic00] showed how to make Kilian's protocol noninteractive in the *random oracle model*. The above constructions make a non-black-box use of the underlying CRHF.

We observe that the bounded-query ZK-PCP of [DFK+92] can be employed to get an alternative to the ZK argument for **NP** of Kilian [Kil92] which uses the underlying CRHF as a black box. (Our protocols are in fact *fully* black-box [RTV04] with a non-uniform proof of security [CLMP13], in the sense that the security reduction makes a black-box use of the adversary, gets non-uniform advice about the adversary, and has a black-box simulator.)

Theorem 2.2 (Black-Box Sublinear ZK Arguments—Informal)**.** *Let $\mathcal{H}$ be any (nonuniformly secure) family of CRHFs. Using $\mathcal{H}$ only as a black-box, one can construct a constant-round zero-knowledge argument system for* **NP** *with negligible soundness error and communication complexity sublinear in the witness size. Furthermore:*

- *For the case of an honest verifier, the zero-knowledge is* statistical*, the round complexity is 4 messages, and the protocol is public coin.*
- *For the case of malicious-verifier zero knowledge, the round complexity is 5 messages.*
- *If the family of CRHF is exponentially secure, then the communication complexity can be made polylogarithmic in both settings above.*

Since a random oracle gives exponentially secure CRHFs and allows us to apply the Fiat-Shamir transformation [FS86], as a corollary to Theorem 2.2, in this model, we get unconditionally secure non-interactive statistical zero knowledge argument system for **NP** with negligible soundness error and polylogarithmic communication complexity.

We establish the above results in Section 6.

2.3.1 Motivation and Related Work

Our black-box construction of Theorem 2.2 is motivated by the recent line of work on studying the power of black-box cryptographic constructions vs. that of non-black-box ones (*e.g.*

[5] Note that the random oracle $f(\cdot)$ is *not* efficiently computable. The work of [GIMS10] presents an *efficient* construction of unconditionally secure commitments in the IPCP model.

[6] The more advanced PCP constructions of [ALM+98, AS98] were not known at that time.

[GKM+00, DI05, HIK+11, CDSMW08, CDSMW09, PW09, Wee10, Goy11]). The goal in this line of work is to understand whether the non-black-box application of an underlying primitive $\mathcal{P}$ which is used in a construction of another (perhaps more complicated) primitive $\mathcal{Q}$ is *necessary* or a black-box construction exists as well. The reason behind studying this question is that the black-box constructions are generally much more efficient (since the source of the non-black-box-ness is usually an inefficient Cook-Levin reduction to an **NP**-complete language). Moreover, black-box constructions are more modular and are capable of also incorporating any *physical* implementations of the employed primitive $\mathcal{P}$ in the implementation of $\mathcal{Q}$.

2.3.2 Technique

Kilian's argument system, when only required to be sound (and not ZK), has only four messages and uses the hash function as a black-box. The first three messages can be easily made ZK, and it is only the last message from the prover which potentially carries some knowledge. In this last message, the prover reveals some portions of the PCP. To retain the zero-knowledge property, Kilian substitutes the last message (of his 4-message protocol) by a zero-knowledge sub-protocol through which the prover convinces the verifier that he could have revealed the correct portion of the PCP in a way that would cause the verifier to accept. The latter zero-knowledge sub-protocol makes non-black box use of the code of the hash function used in the protocol. Thus, our goal is to remove the zero-knowledge sub-protocol performed at the end.[7]

In order to make Kilian's 6-message ZK argument black-box, we need to know more details about its first three rounds. The first message is simply the description of the hash function sent to the prover. Then by using the given hash function and applying a Merkel tree to the PCP the prover hashes down the PCP into a short string which is sent to the verifier as a commitment to the whole PCP. With some care, one can make the hash value carry negligible information about the PCP. The third message (from the verifier) consists of the indices of symbols which the PCP verifier chooses to read from the PCP. The prover, in the 4th message reveals the answers to the PCP queries by revealing the relevant paths of the Merkel tree to the verifier. The committed hash value of the PCP (the second message) together with the collision-resistance property of the hash function prevent the prover from changing his mind about the PCP that he committed to in the second message. Thus the soundness of the PCP implies the soundness of the argument system. To keep the last message of this protocol zero-knowledge, as we said, Kilian's prover will *not* simply reveal the relevant preimages, but instead would prove in a zero-knowledge manner, that he knows a set of preimages that would make the PCP verifier accept.

Our main intuitive observation is that if instead of using the PCP of [ALM+98, AS98] one feeds (a direct product version of) the the *bounded-query* ZK-PCP of [DFK+92] to the construction of Kilian, then the prover can safely reveal the relevant preimages in the last step of the basic 4-message argument of Kilian and this will not hurt the zero-knowledge property. The key point is that although the employed PCP is zero-knowledge only against bounded-query PCP verifiers, since we are in the prover/verifier setting, the prover can control how many queries of the PCP are read by the verifier, and therefore the bounded-query ZK property of the used PCP will suffice for the argument system to be zero-knowledge. Using the transformation of [FS86, Mic00] one can eliminate the interaction using the random oracle and obtain an *unconditional* construction of

[7] Barak and Goldreich [BG02] also employ Kilian's approach to get a 4-message universal argument without zero-knowledge. Similarly to Kilian's protocol, to make their protocol zero-knowledge (or just witness indistinguishable) [BG02] use the hash function in a non-black-box way.

sublinear-communication noninteractive ZK arguments for **NP** in the random oracle model. To obtain the result for malicious verifiers (and negligible soundness error), we employ a variant of the technique of Goldreich-Kahan [GK96] where both prover and verifier use statistically hiding commitments.

Using NIZK? A potential alternative way to get a ZK argument (without using ZK-PCPs) is to use noninteractive zero-knowledge (NIZK) proofs for **NP** [BFM88]. To do so, the prover and the verifier should perform a coin-tossing protocol along with the first 3 messages of the basic variant of Kilian's argument system, and this will allow the prover to be able to send a noninteractive zero-knowledge message to the verifier in his last message which proves to the verifier that the prover knows the right preimages of the hash function. This approach benefits from having only 4 messages exchanged, but it still uses the code of the hash function in a non-black-box way, and moreover, one needs to assume the existence of NIZK proofs for **NP** (*in addition* to the assumption that CRHFs exist).

3 Preliminaries

Basic Terminology and Notation. We use bold letters to denote random variables (*e.g.* $\mathbf{X}$ or $\mathbf{x}$). By $x \leftarrow \mathbf{x}$ we mean that x is sampled according to the distribution of the random variable $\mathbf{x}$. We write $\mathbb{E}_x[\cdot]$ to denote $\mathbb{E}_{x \leftarrow \mathbf{x}}[\cdot]$, where any x appearing inside the expression in the expectation is fixed. For any finite set $\mathcal{S}$, $x \leftarrow \mathcal{S}$ denotes that x is sampled uniformly from $\mathcal{S}$. $\mathbf{U}_n$ denotes the uniform distribution over $\{0,1\}^n$, and $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For jointly distributed random variables $(\mathbf{x}, \mathbf{y})$, and for a specific value $y \leftarrow \mathbf{y}$, by $(\mathbf{x} \mid y)$ we mean the random variable $\mathbf{x}$ conditioned on $\mathbf{y} = y$. When we say an event occurs with *negligible* probability, denoted by $\mathrm{negl}(n)$, we mean it occurs with probability $n^{-\omega(1)}$.

We call two random variables $\mathbf{x}, \mathbf{y}$ over the support set $\mathcal{S}$ (or their corresponding distributions) *ε-close* if their *statistical distance* $\Delta(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \cdot \sum_{s \in S} |\Pr[\mathbf{x} = s] - \Pr[\mathbf{y} = s]|$ is at most ε. By an *ensemble* (of random variables) $\{\mathbf{y}_x\}_{x \in \mathcal{I}}$ we denote a set of random variables indexed by a set $\mathcal{I}$. We call the two ensembles $\{\mathbf{y}_x\}_{x \in \mathcal{I}}$ and $\{\mathbf{z}_x\}_{x \in \mathcal{I}}$ with the same index set *statistically close* if there exists a negligible function $\lambda(n) = \mathrm{negl}(n)$ such that $\Delta(\mathbf{y}_x, \mathbf{z}_x) = \lambda(|x|)$ for $x \in \mathcal{I}$. We call the two ensembles $\{\mathbf{y}_x\}_{x \in \mathcal{I}}$ and $\{\mathbf{z}_x\}_{x \in \mathcal{I}}$ with the same index set *computationally close* if for every (family of) polynomial sized circuits $\{C_n\}$ there exists a negligible function $\lambda(n) = \mathrm{negl}(n)$ such that $|\Pr[C_{|x|}(\mathbf{y}_x) = 1] - \Pr[C_{|x|}(\mathbf{z}_x) = 1]| \leq \lambda(|x|)$ for $x \in \mathcal{I}$. We use the following lemma:

Lemma 3.1 (Union Bound for Statistical Distance)**.** *Suppose $\Delta(\mathbf{x}, \mathbf{y}) \leq \delta$ and $\Delta(\mathbf{x}', \mathbf{y}') \leq \delta'$, then it holds that $\Delta((\mathbf{x}, \mathbf{x}'), (\mathbf{y}, \mathbf{y}')) \leq \delta + \delta'$ where $(\mathbf{x}, \mathbf{x}')$ is the joint independent distribution of $\mathbf{x}, \mathbf{x}'$, and $(\mathbf{y}, \mathbf{y}')$ is the joint independent distribution of $\mathbf{y}, \mathbf{y}'$.*

We use the terms *efficient* and PPT to refer to any (perhaps oracle-aided and interactive) algorithm that runs in probabilistic polynomial-time over its "main" input (which is typically clear from the context and is distinguished from the randomness).

For a set S and a function f, by $f(S)$ we denote the set $\{f(x) \mid x \in S\}$. For $S \subseteq \{0,1\}^n$ and any $x \in \{0,1\}^n$, we define $x + S = \{x + s \mid s \in S\}$ where the addition is componentwise in GF(2).

For any sequence of oracles $\pi_1, \ldots, \pi_k$, by $\pi = (\pi_1 \mid \cdots \mid \pi_k)$ we mean their "concatenated" oracle which given the query (i, x) answers according to $\pi_i(x)$.

3.1 Promise Problems

A language L is a *partition* of $\{0,1\}^*$ into $L^{\mathrm{Y}} = L$ and $L^{\mathrm{N}} = \{0,1\}^* \setminus L$.

A *promise* language (or problem) $L = (L^{\mathrm{Y}}, L^{\mathrm{N}})$ generalizes the notion of a language by only requiring that $L^{\mathrm{Y}} \cap L^{\mathrm{N}} = \varnothing$ (but there could be some $x \in \{0,1\}^* \setminus (L^{\mathrm{Y}} \cup L^{\mathrm{N}})$ as well). For promise problems, we will sometimes use $x \in L$ to denote $x \in L^{\mathrm{Y}}$.

Definition 3.2 (Operations on Promise Languages)**.** We define the following three operations over promise languages.

- The *complement* $\overline{L} = (\overline{L}^{\mathrm{Y}}, \overline{L}^{\mathrm{N}})$ of a promise language $L = (L^{\mathrm{Y}}, L^{\mathrm{N}})$ is another promise language such that $\overline{L}^{\mathrm{Y}} = L^{\mathrm{N}}$ and $\overline{L}^{\mathrm{N}} = L^{\mathrm{Y}}$.
- For two promise languages L_1 and L_2 we define their *conjunction* $L = L_1 \wedge L_2$ as:
  - $x = (x_1, x_2) \in L^{\mathrm{Y}}$ iff $x_1 \in L_1^{\mathrm{Y}}$ and $x_2 \in L_2^{\mathrm{Y}}$,
  - $x = (x_1, x_2) \in L^{\mathrm{N}}$ iff $x_1 \in L_1^{\mathrm{N}}$ or $x_2 \in L_2^{\mathrm{N}}$.
- For two promise languages L_1 and L_2 we define their *disjunction* $L = L_1 \vee L_2$ as:
  - $x = (x_1, x_2) \in L^{\mathrm{Y}}$ iff $x_1 \in L_1^{\mathrm{Y}}$ or $x_2 \in L_2^{\mathrm{Y}}$,
  - $x = (x_1, x_2) \in L^{\mathrm{N}}$ iff $x_1 \in L_1^{\mathrm{N}}$ and $x_2 \in L_2^{\mathrm{N}}$.

It is easy to see that $L_1 \vee L_2 = \overline{\overline{L_1} \wedge \overline{L_1}}$.

Definition 3.3 (Karp Reduction)**.** A Karp reduction R from a promise problem L_1 to another promise problem L_2 is a deterministic efficient algorithm such that $R(x) \in L_2^{\mathrm{Y}}$ for every $x \in L_1^{\mathrm{Y}}$ and $R(x) \in L_2^{\mathrm{N}}$ for every $x \in L_1^{\mathrm{N}}$.

Definition 3.4. For a relation R, by $R(x)$ we denote the set of *witnesses* for x defined as $R(x) = \{w \mid (x,w) \in R\}$. We call R a polynomial relation, if there is a polynomial $p(\cdot)$ such that $|w| \leq p(|x|)$ for all $w \in R(x)$. We call R a relation for a promise language L if the following two holds: If $x \in L^{\mathrm{Y}}$, then $R(x) \neq \varnothing$ and if $x \in L^{\mathrm{N}}$, then $R(x) = \varnothing$.

3.2 Proof Systems and Zero Knowledge

For an oracle π and an (oracle-aided) algorithm V, by V^{π} we refer to an execution of V given access to π and by $\mathsf{View}\langle \mathsf{V}^{\pi} \rangle(x)$ we refer to the *view* of V in its execution given oracle access to π and input x which consists of x, its private randomness r and the sequence of its oracle query-answer pairs $[(q_1, a_1), (q_2, a_2), \dots]$ (having only the oracle answers and r is sufficient to know $\mathsf{View}\langle \mathsf{V}^{\pi} \rangle$). For interacting algorithms (P, V), by $\mathsf{View}\langle \mathsf{V}, \mathsf{P} \rangle(x)$ we denote the view of V in an interaction with P and common input x which consists of x, the private randomness of V, and the sequence of messages exchanged.

Definition 3.5. Let R be a relation for a promise language L. An interactive proof system (P, V) for L consists of two interactive algorithms P, V where V is PPT and:

- **Completeness:** For every $x \in L^{\mathrm{Y}}, w \in R(x)$, $\mathsf{V}(x)$ accepts interaction with $\mathsf{P}(x,w)$ with probability at least $2/3$.

- **Soundness:** For every $x \in L^{\mathrm{N}}$, $\mathsf{V}(x)$ rejects interaction with *any* prover $\widehat{\mathsf{P}}$ with probability at least 2/3.

We call P *efficient* if P is a PPT. The proof system (P, V) is called an *argument* if the soundness property is only required to hold against PPT provers $\widehat{\mathsf{P}}$.

Definition 3.6 (PCPs)**.** Let R be a relation for a promise language L. A (randomized) *probabilistically checkable proof* (PCP for short) $\Pi = (\{\boldsymbol{\pi}_{x\in L, w\in R(x)}\}, \mathsf{V})$ for L consists of an ensemble of random variables $\{\boldsymbol{\pi}_{x\in L, w\in R(x)}\}$ indexed by $x \in L, w \in R(x)$ whose values are *oracles* (also called *proofs*) and also a verifier V which is an oracle-aided PPT with randomness r. We require the following properties to hold.

- α**-Completeness:** For every $x \in L^{\mathrm{Y}}, w \in R(x)$, and every π in the support of $\boldsymbol{\pi}_{x\in L, w\in R(x)}$ it holds that $\Pr_r[\mathsf{V}^{\pi}_r(x) = 1] \geq \alpha$.
- β**-Soundness:** If $x \in L^{\mathrm{N}}$, then for *every* oracle $\widehat{\pi}$ it holds that $\Pr_r[\mathsf{V}^{\widehat{\pi}}_r(x) = 0] \geq \beta$. We call $1 - \beta$ the *soundness error*.

Whenever the completeness and soundness parameters α, β are not specified, we use the default value 2/3 for both of them. We call Π *efficient*, if there is a PPT P taking four inputs r, x, w, q such that for all $x \in L^{\mathrm{Y}}, w \in R(x)$, the random variable $\mathsf{P}(\mathbf{r}, x, w, \cdot)$ (over the randomness of $\mathbf{r}$) whose samples are *oracles*, is distributed identically to $\boldsymbol{\pi}_{x\in L, w\in R(x)}$. We call Π *nonadaptive* if the honest verifier prepares all of its queries at first (before getting any answers) and ask all in one round.

Definition 3.7. Let R be a polynomial relation for (promise) language L, and let $\Pi = (\{\boldsymbol{\pi}_{x\in L, w\in R(x)}\}, \mathsf{V})$ be a PCP for the problem L. Π is called *zero-knowledge* (ZK) if for every malicious PPT verifier $\widehat{\mathsf{V}}$, there exists a *simulator* Sim which runs in expected poly(n)-time and the following ensembles are statistically close:

$$\{\mathsf{Sim}(x)\}_{x\in L} \quad , \quad \{\mathsf{View}\langle \widehat{\mathsf{V}}^{\boldsymbol{\pi}_{x,w\in R(x)}}(x)\rangle\}_{x\in L}.$$

Note that $\widehat{\mathsf{V}}$ only has oracle access to $\pi \leftarrow \boldsymbol{\pi}_{x,w\in R(x)}$, and the statistical indistinguishability should hold for large enough $|x|$ regardless of which witness $w \in R(x)$ is being used. We call the simulator *straight-line* if it only interacts with the malicious verifier (as a black-box and without rewinding). For restricted class of malicious verifiers we define the following variants of zero-knowledge:

- We call Π *honest-verifier zero-knowledge* (HVZK) if the the simulation is required only for the honest verifier.
- We call Π *u-bounded-query zero-knowledge* (u-BQ-ZK) if the honest verifier asks at most u PCP queries and the statistical simulation is only required for all malicious verifiers who ask up to u PCP queries.
- We call a PCP Π of length $u \cdot k$ (maybe with some padding) *u-restricted-query zero-knowledge* (u-RQ-ZK) if, assuming the PCP is divided into u equal blocks $B_1, \ldots, B_u$, the honest verifier asks at most 1 query from each B_i, and the statistical simulation is also only required for malicious verifiers who ask at most 1 PCP query from from each B_i for $i \in [u]$.

Definition 3.8 (Zero Knowledge Proof Systems)**.** Let (P, V) be a proof system for promise language L. (P, V) is called *computational (resp. statistical) zero-knowledge* if for every PPT verifier $\widehat{\mathsf{V}}$,

there exists a *simulator* Sim which runs in expected $\mathrm{poly}(n)$-time and the following ensembles are computationally (resp. statistically) close:

$$\{\mathsf{Sim}(x)\}_{x\in L} \quad , \quad \{\mathsf{View}\langle \widehat{\mathsf{V}}, \mathsf{P}_w\rangle(x)\}_{x\in L}.$$

Note that the statistical indistinguishability should hold for large enough $|x|$ regardless of which witness $w \in R(x)$ is being used.

Definition 3.9 (Complexity Class **SZK**)**.** The class **SZK** consists of promise problems with a statistical zero-knowledge proof system.

Lemma 3.10. *For a constant k, let $L_1, \ldots, L_k$ be a set of promise languages all in* **SZK***, and let F be a constant-size k-input formula with operations: complement, conjunction, and disjunction as in Definition 3.2. Then $F(L_1, \ldots, L_k) \in$* **SZK***.*

In Appendix A we give a sketch of the proof for completeness. (See [Vad99] for a more general and improved statement than that of Lemma 3.10.)

3.3 Shannon Entropy and Related Computational Problems

All logarithms are in base 2. By $\mathrm{H}(\mathbf{X})$ we denote the Shannon entropy of $\mathbf{X}$ defined as $\mathrm{H}(\mathbf{X}) = \mathbb{E}_X \log(1/\Pr[\mathbf{X} = X])$. By $\mathrm{H}(\mathbf{X} \mid \mathbf{Y})$, we denote the conditional entropy as $\mathbb{E}_Y[\mathrm{H}(\mathbf{X} \mid Y)]$, and we note the conditional mutual information as $\mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) = \mathrm{H}(\mathbf{X} \mid \mathbf{Z}) - (\mathbf{X} \mid \mathbf{YZ})$.

Fact 3.11 (Basic Facts about Entropy)**.** *The following hold for any random variables $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$:*

1. $\mathrm{H}(\mathbf{X} \mid \mathbf{Y}) \leq \mathrm{H}(\mathbf{X})$.
2. $\mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) = \mathrm{H}(\mathbf{X} \mid \mathbf{Z}) - \mathrm{H}(\mathbf{X} \mid \mathbf{YZ}) = \mathrm{H}(\mathbf{Y} \mid \mathbf{Z}) - \mathrm{H}(\mathbf{Y} \mid \mathbf{XZ}) \geq 0$
3. *Data processing inequality: for any randomized function $\mathbf{F}$ (whose randomness is independent of $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$), it holds that $\mathrm{I}(\mathbf{F}(\mathbf{X}); \mathbf{Y} \mid \mathbf{Z}) \leq \mathrm{I}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z})$.*

Definition 3.12 (Conditional Entropy Approximation)**.** The promise problem CEA_ε is defined as follows. Suppose C is a $\mathrm{poly}(n)$-size circuit sampling a joint distribution $(\mathbf{X}, \mathbf{Y})$, *i.e.* this is the distribution of the output of C when run on a uniformly chosen input. Then given (C, r) we have:

- $(\mathbf{X}, \mathbf{Y}, r) \in \mathrm{CEA}^{\mathrm{Y}}_\varepsilon$ iff $\mathrm{H}(\mathbf{X} \mid \mathbf{Y}) \geq r$.
- $(\mathbf{X}, \mathbf{Y}, r) \in \mathrm{CEA}^{\mathrm{N}}_\varepsilon$ iff $\mathrm{H}(\mathbf{X} \mid \mathbf{Y}) \leq r - \varepsilon$.

Lemma 3.13. *For any $\varepsilon > 1/\mathrm{poly}(n)$, $\mathrm{CEA}_\varepsilon \in$* **SZK***.*

See Appendix A for the proof of this lemma.

In our main reduction, we will reduce problems to the following problem in **SZK**:

Definition 3.14 (Conditional Entropy Bound)**.** $\mathrm{CEB}_{\alpha,\beta}$ is the following promise problem where inputs are $\mathrm{poly}(n)$-size circuits C sampling a joint distribution $(\mathbf{X}, \mathbf{Y})$:

1. $(\mathbf{X}, \mathbf{Y}) \in \mathrm{CEB}^{\mathrm{Y}}_{\alpha,\beta}$ iff $\mathrm{H}(\mathbf{X} \mid \mathbf{Y}) \geq \alpha$.
2. $(\mathbf{X}, \mathbf{Y}) \in \mathrm{CEB}^{\mathrm{N}}_{\alpha,\beta}$ iff $\mathrm{H}(\mathbf{X} \mid \mathbf{Y}) \leq \beta$.

The following is immediate from Lemma 3.13:

Lemma 3.15. *For all functions $\alpha(n), \beta(n)$ uniformly computable in time $\mathrm{poly}(n)$ and satisfying $\alpha(n) - \beta(n) > 1/\mathrm{poly}(n)$, $\mathrm{CEB}_{\alpha,\beta} \in$* **SZK***.*

3.4 Statistical Distance vs Conditional Entropy

To prove our main result, we need to bound statistical distance using *conditional* (Shannon) entropy and vice versa. The following two lemmas state how to bound conditional entropy using statistical distance and vice versa; their proof can be found in Appendix A.

Lemma 3.16. (Conditional Entropy to Statistical Distance)**.** *Suppose* $\mathrm{Supp}(\mathbf{X}) \subseteq \{0,1\}^n$*. Then it holds that* $\mathbb{E}_{Y \leftarrow \mathbf{Y}} \Delta((\mathbf{X} \mid Y), \mathbf{U}_n) \leq \sqrt{n - \mathrm{H}(\mathbf{X} \mid \mathbf{Y})}$*.*

Lemma 3.17. (Statistical Distance to Conditional Entropy)**.** *For* $\varepsilon \in [0,1]$ *let* $\mathrm{H}(\varepsilon) = \varepsilon \log(1/\varepsilon) + (1-\varepsilon)\log(1/1-\varepsilon)$*. Suppose* $\Delta((\mathbf{X},\mathbf{Y}),(\mathbf{X}',\mathbf{Y}')) \leq \varepsilon$ *and* $\mathrm{Supp}(\mathbf{X}) \cup \mathrm{Supp}(\mathbf{X}') \subseteq \{0,1\}^n$*. Then it holds that* $|\mathrm{H}(\mathbf{X} \mid \mathbf{Y}) - \mathrm{H}(\mathbf{X}' \mid \mathbf{Y}')| \leq 4(\mathrm{H}(\varepsilon) + \varepsilon \cdot n)$*.*

3.5 Commitments from Collision Resistance

Definition 3.18 (Commitments)**.** A commitment scheme $\mathsf{Com} = (\mathsf{S}, \mathsf{R})$ between a sender S and a receiver S for the message space $\mathcal{W}$ and security parameter n proceeds as follows.

- In the *commitment* phase, the sender S gets some $w \in \mathcal{W}$ as private input, and the parties interact using common input 1^n.
- In the *decommitment* phase, the sender S sends the decommitment $D = (w, r)$ to the receiver, where r is the randomness used by S in commitment phase. The receiver accepts iff (w, r) is consistent with the transcript of the commitment phase.

1. We call Com *statistically hiding*, if for every malicious receiver $\widehat{\mathsf{R}}$, and every two $w, w' \in \mathcal{W}$, the views of $\widehat{\mathsf{R}}$ when S uses w or w' are statistically close (*i.e.* $\mathrm{negl}(n)$-close).
2. We call Com *computationally binding*, if for every (nonuniform) malicious sender $\widehat{\mathsf{S}}$ of $\mathrm{poly}(n)$ size, the probability of Com winning in the following game is at most $\mathrm{negl}(n)$.
   (a) $\widehat{\mathsf{S}}$ and R participate in the commitment phase.
   (b) $\widehat{\mathsf{S}}$ sends (w, r) and (w', r') and wins if $w \neq w'$ and the decommitments (w, r) and (w', r') are both accepted by the receiver.

The following construction, gives a commitment scheme using any shrinking hash function.

Definition 3.19. A family of compressing functions $\mathcal{H} = \{h_\alpha \colon \{0,1\}^{2n} \mapsto \{0,1\}^n\}_{\alpha \in \{0,1\}^{\mathrm{poly}(n)}}$ is called $S(n)$-hard collision-resistant, if for every *nonuniform* adversary Adv of size $S(n)$:

$$\Pr_{\alpha \leftarrow \{0,1\}^{\mathrm{poly}(n)}}[\mathsf{Adv}(\alpha) = (x, y) \text{ and } h_\alpha(x) = h_\alpha(y)] \leq \frac{1}{S(n)}.$$

We call $\mathcal{H}$ simply collision-resistant, if it is $n^{\omega(1)}$-hard, and we call it exponentially-hard collision-resistant, if it is $2^{n^{\Omega(1)}}$-hard.

Remark 3.20. It is more common to define cryptographic primitives by saying that for any polynomial time (sized) adversary Adv and any polynomial $p(\cdot)$ the advantage of Adv in breaking the primitive is at most $1/p(n)$ for large enough security parameter n. We note that when it comes to *nonuniform* security, defined by polynomial sized circuits, this convention is equivalent to the convention used in Definition 3.19 above.

Construction 3.21. Let $\mathcal{H} = \{h_\alpha \colon \{0,1\}^{2m} \mapsto \{0,1\}^m\}_{\alpha \in \{0,1\}^{\mathrm{poly}(m)}}$ be a family of shrinking hash functions. The commitment scheme $\mathsf{Com}_{\mathcal{H}}$ based on $\mathcal{H}$ for ℓ-bit message $b = b_1 \dots b_\ell$ is as follows.

- **Commitment:** The receiver samples $\alpha \leftarrow \{0,1\}^{\mathrm{poly}(m)}$ at random and sends it to the sender. The sender chooses $s_j, r_j \leftarrow \{0,1\}^{2m}$ at random for all $j \in [\ell]$, and the j'th bit of the message b_j, let $C(b_j) = (h_\alpha(r_j), s_j, \langle r, s\rangle + b)$ where the inner product operation $\langle \cdot, \cdot \rangle$ and the addition are both performed in the binary field GF(2). Send $C(b) = (C(b_1), \dots, C(b_\ell))$ as the commitment to b.
- **Decommitment:** The sender reveals bit b and $r_1, \dots, r_\ell$ to the receiver. The receiver verifies that the sender's message is consistent with its commitment message and reject otherwise.

Theorem 3.22 (Statistically Hiding Commitment from CRHFs [DPP97, DPP98, HM96])**.** *Let $\mathcal{H} = \{h_\alpha \colon \{0,1\}^{2m} \mapsto \{0,1\}^m\}_{\alpha \in \{0,1\}^{\mathrm{poly}(m)}}$ be a family of shrinking hash functions, and let $\mathsf{Com}_{\mathcal{H}}$ be the commitment scheme of Construction 3.21 with security parameter n and suppose $\ell = \mathrm{poly}(n)$.*

- *If $m = \omega(\log n)$, then for* every $\alpha \leftarrow \{0,1\}^{\mathrm{poly}(m)}$ *and every two ℓ-bit messages w_0, w_1, the distributions $C(w_0)$ and $C(w_1)$ are $2^{-\Omega(m)} = n^{-\omega(1)}$-close . Thus $\mathsf{Com}_{\mathcal{H}}$ is* statistically hiding*.*
- *For* every $\alpha \leftarrow \{0,1\}^{\mathrm{poly}(m)}$*, if an adversary can successfully open a commitment into two different ℓ bit messages $w_0 \neq w_1$ it can find a collision for $h_\alpha(\cdot)$. Therefore, if $\mathcal{H}$ is $s(m)$-hard collision-resistant and $s(m) = n^{\omega(1)}$, then $\mathsf{Com}_{\mathcal{H}}$ is computationally binding.*

4 Languages with Efficient ZK-PCPs are in SZK

In this section we prove Theorem 2.1.

Bounded Entropy PCPs. Suppose for a promise language L and a relation R for L, $\Pi = (\{\boldsymbol{\pi}_{x \in L, w \in R(x)}\}, \mathsf{V})$ is an efficient ZK-PCP for L. Suppose, for every x, we only keep the lexicographically first witness in $R(x)$, and remove all other witnesses from $R(x)$. Now we can simply talk abot $\{\boldsymbol{\pi}_{x \in L}\}$ instead of $\{\boldsymbol{\pi}_{x \in L, w \in R(x)}\}$. A property of the randomized oracle $\{\boldsymbol{\pi}_{x \in L}\}$ is that, as a random variable, $\boldsymbol{\pi}_{x \in L}$ has entropy at most $\mathrm{poly}(n)$, because $\mathrm{H}(\boldsymbol{\pi}_{x \in L})$ is bounded by the number of random bits used by the efficient algorithm P computing $\{\boldsymbol{\pi}_{x \in L, w \in R(x)}\}$. The following only relies on this "bounded entropy" property and implies theorem Theorem 2.1 as a corollary.

Theorem 4.1. *Suppose the promise problem $L = (L^{\mathrm{Y}}, L^{\mathrm{N}})$ has a ZK-PCP $\Pi = (\{\boldsymbol{\pi}_{x \in L}\}, \mathsf{V})$ of entropy at most $\mathrm{H}(\boldsymbol{\pi}_x) \leq \mathrm{poly}(|x|)$. Then $L \in$ **SZK**.*

In the rest of this section we prove Theorem 4.1. Let $\eta = \mathrm{H}(\boldsymbol{\pi}_x) \leq \mathrm{poly}(n)$.

The first step of our proof is to define a verifier who can "exhaust" all of the entropy of the ZK-PCP so that the proof behaves essentially as if it were deterministic. We use the following verifier: let $\mathsf{V}^{[\ell]} = (\mathsf{V}^1, \dots, \mathsf{V}^\ell)$ be a verifier who executes ℓ independent instances of V against the given oracle and let V^i be its i^{th} verification. (We will fix a choice of $\ell = \mathrm{poly}(n) \gg \eta$ later.) Let Sim be the simulator that simulates the view of $\mathsf{V}^{[\ell]}$ statistically well (*i.e.* $\mathsf{Sim}(x)$ is $\mathrm{negl}(|x|)$-close to the view of $\mathsf{V}^{[\ell]}(x)$ when accessing $\pi \leftarrow \boldsymbol{\pi}_x$ for $x \in L$). The view of V^i can be represented as $\nu^i = (r^i, q^i_1, a^i_1, \dots, q^i_m, a^i_m)$ where $r^i \in \{0,1\}^k$ is the randomness used by V^i, q^i_j is its j^{th} oracle query and a^i_j is the answer to q^i_j. We use the notation $\overline{a}^i = (a^i_1, \dots, a^i_m), \overline{q}^i = (q^i_1, \dots, q^i_m)$. The view of $\mathsf{V}^{[\ell]}$ consists of $(\nu^1, \dots, \nu^\ell)$.

In order to prove $L \in \mathbf{SZK}$, we show how to reduce L to a constant size formula over $\mathbf{SZK}$ languages. As we mentioned in the introduction, we need to check three conditions: the simulator generates an accepting view, the entropy of a random answer in the view has low entropy given the query, and the distribution of the random coins in the view is uniform.

To describe our reduction formally we first need to define a circuit C_x^{SIM} and a promise problem $D_{\alpha,\beta}$ as follows.

- The circuit C_x^{SIM} takes as input r_{SIM} (for input length $|x|$). The circuit C_x outputs $\mathsf{Sim}(x; r_{\text{SIM}}) = (\nu^1, \ldots, \nu^\ell)$ where for each $i \in [\ell]$, $\nu^i = (r^i, q_1^i, a_1^i, \ldots, q_m^i, a_m^i)$.
- For $\alpha > \beta$, $D_{\alpha,\beta}$ is a promise problem whose inputs are Boolean circuits C of input length n and size $|C| = \text{poly}(n)$; then:
  1. $C \in D^{\text{Y}}_{\alpha,\beta}$ iff $\Pr[C(\mathbf{U}_n) = 1] \geq \alpha$, and
  2. $C \in D^{\text{N}}_{\alpha,\beta}$ iff $\Pr[C(\mathbf{U}_n) = 1] \leq \beta$.

  The parameters α and β could be functions of n, and it is easy to see that for efficiently computable α, β (given n) it holds that $D_{\alpha,\beta} \in \mathbf{BPP}$ if $\alpha - \beta > 1/\text{poly}(n)$.

Reduction 4.2 (Main Reduction)**.** Given a parameter ℓ, we map $x \mapsto (C_1, C_2, C_3)$ as follows.

1. C_1 checks the uniformity of the random coins in the view. C_1 is a circuit sampling the joint distribution $(\mathbf{X}_1, \mathbf{Y}_1)$ defined as follows. On input (r_{SIM}, i), C_1 executes the circuit C_x^{SIM} on r_{SIM} to get $(\nu^1, \ldots, \nu^\ell) = C_x^{\text{SIM}}(r_{\text{SIM}})$ and sets:
$$X_1 = r^i \qquad \text{and} \qquad Y_1 = (\nu^1, \ldots, \nu^{i-1}).$$
2. C_2 checks that the conditional entropy of a randomly chosen answer is low conditioned on the corresponding query. C_2 is a circuit sampling the joint distribution $(\mathbf{X}_2, \mathbf{Y}_2)$ defined as follows. On input (r_{SIM}, i, j), C_2 executes the circuit C_x^{SIM} on r_{SIM} to get $(\nu^1, \ldots, \nu^\ell) = C_x^{\text{SIM}}(r_{\text{SIM}})$ and sets:
$$X_2 = a_j^i \qquad \text{and} \qquad Y_2 = (\nu^1, \ldots, \nu^{i-1}, q_j^i).$$
We emphasize the fact that while a_j^i, q_j^i appear in the output of C_2, the actual index j itself does *not* appear in the output.
3. C_3 checks that the view is accepting. C_3 operates as follows: on input (r_{SIM}, i), C_3 executes the circuit C_x^{SIM} on r_{SIM} to obtain $(\nu^1, \ldots, \nu^\ell) = C_x^{\text{SIM}}(r_{\text{SIM}})$, and output 1 iff ν^i is an accepting view of V.

Claim 4.3. *Reduction 4.2 is a Karp reduction from L (specified in Theorem 4.1) to the promise language $Z = \text{CEB}_{k-1/200,k-1/100} \wedge \overline{\text{CEB}}_{2\eta/\ell,1.1\eta/\ell} \wedge D_{0.66,\beta}$ for $\beta = 1/3 + 1/10 + 2m\eta/\ell$.*

Proving Theorem 4.1 using Claim 4.3. By taking $\ell = 40m\eta$, it holds that $2m \cdot \eta/\ell < 1/20$ in Lemma 4.8 and so $\beta < 1/2$, which implies that $D_{\alpha,\beta} \in \mathbf{BPP}$, $Z \in \mathbf{SZK}$, and so $L \in \mathbf{SZK}$. □

In the following we prove Claim 4.3 by studying each case of $x \in L^{\mathrm{Y}}$ and $x \in L^{\mathrm{N}}$ separately. We begin with a lemma that will be useful for the case $x \in L^{\mathrm{Y}}$. The following lemma bounds the conditional entropy of a single answer to a single randomly chosen verifier query by the conditional entropy of the set of *all* answers to the set of *all* verifier queries. This is non-trivial because the verifier queries may be asked adaptively.

Lemma 4.4. *Let A be any randomized algorithm that (adaptively) queries a PCP π. Let $r \in \{0,1\}^k$ denote the random coins of A. Let $\overline{q} = (q_1, \ldots, q_m)$ be the queries that $A^{\pi}(r)$ makes and let $a_j = \pi(q_j)$ be the corresponding answers. Let $\boldsymbol{\pi}$ be an arbitrary distribution over proofs, and let $\overline{\mathbf{q}}$ and $\overline{\mathbf{a}}$ be the distribution over (the vectors of) queries and answers obtained by querying $\boldsymbol{\pi}$ using algorithm A on uniform random coins $\mathbf{r}$. Let also $\mathbf{j}$ be an arbitrary distribution over $[m]$. Then $\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j}) \leq \mathrm{H}(\overline{\mathbf{a}} \mid \mathbf{r})$ where in the notation $\mathbf{q_j}$ the value of $\mathbf{j}$ is* not *explicitly revealed.*

Proof. By the definition of conditional entropy and adding $0 = \mathrm{H}(\mathbf{a_j q_j} \mid \boldsymbol{\pi}) - \mathrm{H}(\mathbf{a_j q_j} \mid \boldsymbol{\pi})$, we get

$$\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j}) = \mathrm{H}(\mathbf{a_j q_j}) - \mathrm{H}(\mathbf{a_j q_j} \mid \boldsymbol{\pi}) - (\mathrm{H}(\mathbf{q_j}) - \mathrm{H}(\mathbf{a_j q_j} \mid \boldsymbol{\pi})).$$

Since a proof π is *stateless* for any fixed π, given any query q asked at some point during the execution of A^{π}, the answer $a = \pi(q)$ is also fixed. Therefore it holds that $\mathrm{H}(\mathbf{a_j q_j} \mid \boldsymbol{\pi}) = \mathrm{H}(\mathbf{q_j} \mid \boldsymbol{\pi})$, and by the definition of mutual information, we may deduce that

$$\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j}) = \mathrm{I}(\mathbf{a_j q_j}; \boldsymbol{\pi}) - \mathrm{I}(\mathbf{q_j}; \boldsymbol{\pi}) \leq \mathrm{I}(\mathbf{a_j q_j}; \boldsymbol{\pi}).$$

Since $\mathrm{I}(\mathbf{a_j q_j}; \boldsymbol{\pi}) = \mathrm{H}(\boldsymbol{\pi}) - \mathrm{H}(\boldsymbol{\pi} \mid \mathbf{a_j q_j})$ and since $\boldsymbol{\pi}$ and $\mathbf{r}$ are independent, Item 1 of Fact 3.11 implies that

$$\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j}) \leq \mathrm{I}(\mathbf{a_j q_j}; \boldsymbol{\pi}) = \mathrm{H}(\boldsymbol{\pi}) - \mathrm{H}(\boldsymbol{\pi} \mid \mathbf{a_j q_j}) \leq \mathrm{H}(\boldsymbol{\pi} \mid \mathbf{r}) - \mathrm{H}(\boldsymbol{\pi} \mid \mathbf{a_j q_j r}) = \mathrm{I}(\mathbf{a_j q_j}; \boldsymbol{\pi} \mid \mathbf{r}).$$

Let $\mathbf{F}$ be the function that takes as input $(\overline{\mathbf{a}}, \overline{\mathbf{q}})$ and outputs $(\mathbf{a_j}, \mathbf{q_j})$ by sampling $\mathbf{j}$. By the data processing inequality (Item 3 of Fact 3.11) it holds that

$$\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j}) \leq \mathrm{I}(\mathbf{a_j q_j}; \boldsymbol{\pi} \mid \mathbf{r}) = \mathrm{I}(\mathbf{F}(\overline{\mathbf{aq}}); \boldsymbol{\pi} \mid \mathbf{r}) \leq \mathrm{I}(\overline{\mathbf{aq}}; \boldsymbol{\pi} \mid \mathbf{r}) \leq \mathrm{H}(\overline{\mathbf{aq}} \mid \mathbf{r}) = \mathrm{H}(\overline{\mathbf{a}} \mid \mathbf{r}) + \mathrm{H}(\overline{\mathbf{q}} \mid \overline{\mathbf{a}}\mathbf{r}).$$

Finally, since $\mathrm{H}(\overline{\mathbf{q}} \mid \overline{\mathbf{a}}\mathbf{r}) = 0$, this implies the proposition. □

Remark 4.5. We emphasize that if $\boldsymbol{\pi}$ was *stateful* (*i.e.* a "prover", rather than a "proof"), then Lemma 4.4 would be *false*. Even a deterministic prover can correlate his answers to the verifier's queries, and so it may be that $\mathrm{H}(\overline{\mathbf{a}} \mid \overline{\mathbf{q}}) = 0$ but $\mathrm{H}(\mathbf{a_j} \mid \mathbf{q_j}) > 0$. Namely, even given π (say for a stateful prover that π gives the random coins of the prover) and a query q, the answer to q may have entropy because π's answer to q may be different depending on whether q was asked as the first query or second query or third query, etc. In particular, the equality $\mathrm{H}(\mathbf{a_j q_j} \mid \boldsymbol{\pi}) = \mathrm{H}(\mathbf{q_j} \mid \boldsymbol{\pi})$ used in the proof of Lemma 4.4 would not hold anymore. This is one place where we crucially use the fixed nature of a PCP.

4.1 Proof of Claim 4.3: the Case $x \in L^{\mathrm{Y}}$

Here we would like to show that $(C_1 \in \mathrm{CEB}^{\mathrm{Y}}_{k-1/200,k-1/100}) \wedge (C_2 \in \overline{\mathrm{CEB}}^{\mathrm{Y}}_{2\eta/\ell,1.1\eta/\ell}) \wedge (C_3 \in D^{\mathrm{Y}}_{0.66,\beta})$. We study each of the generated instances C_i for $i \in [3]$. In all these cases, we first assume that the simulator's output is identically distributed to the view of $\mathsf{V}^{[\ell]}$ interacting with a prover and then will show how to remove this assumption.

The Instance C_1. If the simulator's outputs were identically distributed to the view of $\mathsf{V}^{[\ell]}$ interacting with a prover, then the simulated randomness $\mathbf{X}_1 = \mathbf{r}^i$ will be uniformly distributed over $\{0,1\}^k$ with entropy k *independently* of $\mathbf{Y}_1 = (\boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1})$. Since the simulator generates a view that is statistically close to the honest interaction (and since $k = \mathrm{poly}(|x|)$ and $\mathrm{H}(\mathrm{negl}(n)) = \mathrm{negl}(n)$) we may apply Lemma 3.17 to deduce that $\mathrm{H}(\mathbf{X}_1 \mid \mathbf{Y}_1) \geq k - \mathrm{negl}(n) \geq k - 1/200$. Therefore, $C_1 \in \mathrm{CEB}^{\mathrm{Y}}_{k-1/200, k-1/100}$.

The Instance C_2. Here we study the view of $\mathsf{V}^{[\ell]}$ while interacting with a proof generated according to the distribution $\boldsymbol{\pi}_x$ whose entropy is bounded by η. Suppose first that the simulator's outputs were identically distributed to the view of $\mathsf{V}^{[\ell]}$ interacting with $\boldsymbol{\pi}_x$. In this case, by an argument similar to [IMS12], one can show that

Claim 4.6. $\mathbb{E}_{i \leftarrow [\ell]} \mathrm{H}(\overline{\mathbf{a}}^i \mid \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1}, \mathbf{r}^i) \leq \eta/\ell$.

Proof.

$$\begin{aligned}
\eta + k\ell &\geq \mathrm{H}(\boldsymbol{\pi}_x) + \mathrm{H}(\mathbf{r}^1, \ldots, \mathbf{r}^\ell) \\
(\boldsymbol{\pi}_x \text{ and } \mathbf{r}^1, \ldots, \mathbf{r}^\ell \text{ are independent}) &= \mathrm{H}(\boldsymbol{\pi}_x, \mathbf{r}^1, \ldots, \mathbf{r}^\ell) \\
(\boldsymbol{\pi}_x \text{ and } \mathbf{r}^1, \ldots, \mathbf{r}^\ell \text{ determine } \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^\ell) &\geq \mathrm{H}(\boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^\ell) \\
&= \sum_{i \in [\ell]} \mathrm{H}(\boldsymbol{\nu}^i \mid \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1}) \\
(\mathbf{r}^i \text{ and } \overline{\mathbf{a}}^i \text{ determine } \overline{\mathbf{q}}^i) &= \sum_{i \in [\ell]} \mathrm{H}(\mathbf{r}^i \mid \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1}) + \mathrm{H}(\overline{\mathbf{a}}^i \mid \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1}, \mathbf{r}^i) \\
&= k\ell + \sum_{i \in [\ell]} \mathrm{H}(\overline{\mathbf{a}}^i \mid \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1}, \mathbf{r}^i).
\end{aligned}$$

Therefore, by averaging over i we obtain that $\mathbb{E}_{i \leftarrow [\ell]} \mathrm{H}(\overline{\mathbf{a}}^i \mid \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1}, \mathbf{r}^i) \leq \eta/\ell$. □

The following claim is also based on the assumption that the simulation is perfect, and thus the distribution of $(\boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^m)$ generated by the simulator is identical to the view of $\mathsf{V}^{[\ell]}$ run against $\pi \leftarrow \boldsymbol{\pi}_{x \in L, w}$.

Claim 4.7. *For each fixed value of i and $(\nu^1, \ldots, \nu^{i-1})$, it holds that*

$$\mathrm{H}(\mathbf{a}^i_j \mid \mathbf{q}^i_j, \nu^1, \ldots, \nu^{i-1}) \leq \mathrm{H}(\overline{\mathbf{a}}^i \mid \mathbf{r}^i, \nu^1, \ldots, \nu^{i-1}) \tag{1}$$

Namely, the entropy of the answers of the i^{th} verification gives an upper-bound on the entropy of the answer to a randomly chosen query of the verifier without *revealing its index.*

Proof. Let $(\boldsymbol{\pi}_x, \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1})$ be the joint distribution of an honest proof $\boldsymbol{\pi}_x$ and $i-1$ executions of the honest verifier $\mathsf{V}^1, \ldots, \mathsf{V}^{i-1}$ using proof $\boldsymbol{\pi}_x$. Apply Lemma 4.4 using the distribution over proofs given by $(\boldsymbol{\pi}_x \mid \nu^1, \ldots, \nu^{i-1})$, and with the honest verifier algorithm V^i as the query algorithm accessing the proof. □

Using Claims 4.6 and 4.7, we conclude that $\mathrm{H}(\mathbf{X}_2 \mid \mathbf{Y}_2) \leq \eta/\ell$, *assuming that* the simulator was perfect. If we only assume that the simulator's output is statistically close to the view of $\mathsf{V}^{[\ell]}$ interacting with $\boldsymbol{\pi}_x$, then we can apply Lemma 3.17 and deduce that $\mathrm{H}(\mathbf{X}_2 \mid \mathbf{Y}_2) \leq \eta/\ell + \mathrm{negl}(n) < 1.1\eta/\ell$ which implies that $C_2 \in \overline{\mathrm{CEB}}^{\mathrm{Y}}_{2\eta/\ell, 1.1\eta/\ell}$.

The Instance C_3. By the completeness of Π, when $\mathsf{V}^{[\ell]} = (\mathsf{V}^1, \ldots, V^\ell)$ interacts with a proof, for all $i \in [\ell]$, V^i accepts with probability $\geq 2/3$. Since the simulation is statistically close to the real interaction, it holds that $\boldsymbol{\nu}^i$ is accepting with probability $2/3 - \mathrm{negl}(n) \geq 0.66$, and so $C_3 \in D^{\mathrm{Y}}_{0.66,\beta}$.

4.2 Proof of Claim 4.3: the Case $x \in L^{\mathrm{N}}$

Here we would like to show that $(C_1 \in \mathrm{CEB}^{\mathrm{N}}_{k-1/200,k-1/100}) \vee (C_2 \in \overline{\mathrm{CEB}}^{\mathrm{N}}_{2\eta/\ell,1.1\eta/\ell}) \vee (C_3 \in D^{\mathrm{N}}_{0.66,\beta})$. This follows from the following lemma.

Lemma 4.8. *Suppose $x \in L^{\mathrm{N}}$, $C_1 \notin \mathrm{CEB}^{\mathrm{N}}_{k-1/200,k-1/100}$, and $C_2 \notin \overline{\mathrm{CEB}}^{\mathrm{N}}_{2\eta/\ell,1.1\eta/\ell}$. Then it holds that $C_3 \in D^{\mathrm{N}}_{0.66,\beta}$ for $\beta = 1/3 + 1/10 + 2m \cdot \eta/\ell$.*

Intuition. Since $C_2 \notin \overline{\mathrm{CEB}}^{\mathrm{N}}_{2\eta/\ell,1.1\eta/\ell}$, therefore, the oracle answers returned to the verifier in the i^{th} execution (for a random $i \leftarrow [\ell]$) all have very low entropy and thus close to a *fixed* proof. Moreover, due to $C_1 \notin \mathrm{CEB}^{\mathrm{N}}_{k-1/200,k-1/100}$, the randomness of the verifier in this execution has almost full entropy, and therefore, the i^{th} execution is close to an honest execution of the verifier against some oracle. Finally, since $x \in L^{\mathrm{N}}$ by the soundness of the PCP, the verifier would accept with probability at most $\approx 1/3$. The formal argument goes through a hybrid argument as follows.

Experiments. The outputs of all experiments described below consist of a view of $\mathsf{V}^{[i]}$ (*i.e.* the first i executions of the verifier). The distribution of $(\nu^1, \ldots, \nu^{i-1})$ in all of these executions is the same and is sampled by $\mathsf{Sim}(x)$, and they only differ in the way they sample ν^i.

- **Experiment** Real**.** Choose $i \leftarrow [\ell]$, and take the output $(\nu^1, \ldots, \nu^i)$ by running $\mathsf{Sim}(x)$.
- **Experiment** Ideal**.** Choose $i \leftarrow [\ell]$, and take the output $(\nu^1, \ldots, \nu^{i-1})$ by running $\mathsf{Sim}(x)$. To sample $\boldsymbol{\nu}^i = (\mathbf{r}^i, \overline{\mathbf{q}}^i, \overline{\mathbf{a}}^i)$ we first sample $r^i \leftarrow \{0,1\}^k$ uniformly at random, and then using r^i we run the verifier against the oracle $\widehat{\pi}$ defined as follows.

  The Oracle $\widehat{\pi}$: Suppose we have fixed $(\nu^i, \ldots, \nu^{i-1})$. Recall the distribution $((\mathbf{q}^i_{\mathbf{j}}, \mathbf{a}^i_{\mathbf{j}}) \mid \nu^i, \ldots, \nu^{i-1})$ defined above when defining the instance C_2 (*i.e.*, $(\mathbf{a}^i_{\mathbf{j}}, \mathbf{q}^i_{\mathbf{j}})$ is a randomly chosen pair of query-answer pairs from the view ν^i without revealing the index j). For every query q, the oracle $\widehat{\pi}$ gets one sample according to $a \leftarrow (\mathbf{a}^i_{\mathbf{j}} \mid \nu^i, \ldots, \nu^{i-1}, \mathbf{q}^i_{\mathbf{j}} = q)$ and sets $\widehat{\pi}(q) = a$ forever. If $\Pr[\mathbf{q}^i_{\mathbf{j}} = q \mid \nu^i, \ldots, \nu^{i-1}] = 0$, we define $\widehat{\pi}(q) = \perp$.
- **Experiment** Hyb_j **for** $j \in [m+1]$**.** These experiments are in between Real and Ideal and for larger j they become closer to Real. Here we choose $i \leftarrow [\ell]$, and take the output $(\nu^1, \ldots, \nu^i)$ by running $\mathsf{Sim}(x)$. Then we will *re-sample* parts of ν^i as follows. We will keep $(r^i, (q^i_1, a^i_1), \ldots, (q^i_{j-1}, a^i_{j-1}))$ as sampled by $\mathsf{Sim}(x)$. For the remaining queries and answers we sample an oracle $\widehat{\pi}$ as described in Ideal, and we let $(q^i_j, a^i_j), \ldots, (q^i_m, a^i_m)$ be the result of continuing the execution of V^i using r^i and the oracle $\widehat{\pi}$. Note that $\mathsf{Hyb}_{m+1} \equiv \mathsf{Real}$.

Claim 4.9. *If $x \in L^{\mathrm{N}}$, then $\Pr_{\mathsf{Ideal}}[\nu^i \text{ accepts }] \leq 1/3$.*

Claim 4.10. *If $C_1 \notin \mathrm{CEB}^{\mathrm{N}}_{k-1/200,k-1/100}$, then $\Delta(\mathsf{Ideal}, \mathsf{Hyb}_1) \leq 1/10$.*

Claim 4.11. *If $C_2 \notin \overline{\mathrm{CEB}}^{\mathrm{N}}_{2\eta/\ell,1.1\eta/\ell}$, then $\mathbb{E}_{j \in [m]} \Delta(\mathsf{Hyb}_j, \mathsf{Hyb}_{j+1}) \leq 2\eta/\ell$.*

Proving Lemma 4.8. Claims 4.9, 4.10, and 4.11 together imply that

$$\Pr_{\mathsf{Real}}[\nu^i \text{ accepts }] \leq \Pr_{\mathsf{Ideal}}[\nu^i \text{ accepts }] + \Delta(\mathsf{Ideal}, \mathsf{Hyb}_1) + \sum_{j\in[m]} \Delta(\mathsf{Hyb}_j, \mathsf{Hyb}_{j+1}) \leq 1/3 + 1/10 + 2m\eta/\ell$$

which proves that $C_3 \in D^{\mathrm{N}}_{2/3,\beta}$. In the following we prove these claims.

Proof of Claim 4.9. Since the oracle $\widehat{\pi}$ is sampled and fixed before choosing r^i and executing V^i, and because $x \in L^{\mathrm{N}}$, by the soundness property of the PCP it holds that $\Pr_{\mathsf{Ideal}}[\nu^i \text{ accepts }] \leq 1/3$. □

Proof of Claim 4.10. If $C_1 \notin \mathrm{CEB}^{\mathrm{N}}_{k-1/200,k-1/100}$, then we have $\mathbb{E}_{i\leftarrow[\ell]}[\mathrm{H}(\mathbf{r}^i \mid \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{i-1})] \geq k - 1/100$, and so by Lemma 3.16 it holds that

$$\mathop{\mathbb{E}}_{i\leftarrow[\ell],\nu^1,\ldots,\nu^{i-1}}[\Delta((\mathbf{r}^i \mid \nu^1, \ldots, \nu^{i-1}), \mathbf{U}_k)] \leq \sqrt{1/100} = 1/10.$$

But note that the only difference between Ideal and Hyb_1 is the way we sample r^i conditioned on the previously sampled parts (*i.e.* $\nu^1, \ldots, \nu^{i-1}$). Thus it holds that $\Delta(\mathsf{Ideal}, \mathsf{Hyb}_1) \leq 1/10$. □

Proof of Claim 4.11. The only difference between Hyb_j and Hyb_{j+1} is the way they answer q^i_j. In Hyb_{j+1} the original answer of the simulator is used, while in Hyb_j this answer is provided by the oracle $\widehat{\pi}$. Thus, they are different only when the answer re-sampled by $\widehat{\pi}$ differs from the original answer. Therefore, we have that:

$$\Delta(\mathsf{Hyb}_j, \mathsf{Hyb}_{j+1}) \leq \mathop{\mathbb{E}}_{\nu^1,\ldots,\nu^{i-1},i}\left[\Pr_{\overline{\mathbf{a}}^i,\overline{\mathbf{q}}^i,\widehat{\boldsymbol{\pi}}}[\mathbf{a}^i_j \neq \widehat{\boldsymbol{\pi}}(\mathbf{q}^i_j) \mid i, \nu^1, \ldots, \nu^{i-1}]\right]$$

Taking an expectation over all $j \leftarrow [\ell]$ we conclude Claim 4.11 as follows.

$$\begin{aligned}\mathop{\mathbb{E}}_{j}[\Delta(\mathsf{Hyb}_j, \mathsf{Hyb}_{j+1})] &= \mathop{\mathbb{E}}_{j,i,\nu^1,\ldots,\nu^{i-1}}\left[\Pr_{\overline{\mathbf{a}}^i,\overline{\mathbf{q}}^i,\widehat{\boldsymbol{\pi}}}[\mathbf{a}^i_j \neq \widehat{\boldsymbol{\pi}}(\mathbf{q}^i_j) \mid i, \nu^1, \ldots, \nu^{i-1}]\right] \\ &= \mathop{\mathbb{E}}_{i,\nu^1,\ldots,\nu^{i-1}}\left[\Pr_{\mathbf{j},\overline{\mathbf{a}}^i,\overline{\mathbf{q}}^i,\widehat{\boldsymbol{\pi}}}[\mathbf{a}^i_{\mathbf{j}} \neq \widehat{\boldsymbol{\pi}}(\mathbf{q}^i_{\mathbf{j}}) \mid i, \nu^1, \ldots, \nu^{i-1}]\right]\end{aligned}$$

By combining the sampling of $\mathbf{a}^i_{\mathbf{j}}, \mathbf{q}^i_{\mathbf{j}}$ directly, we have that

$$\begin{aligned}\mathop{\mathbb{E}}_{j}[\Delta(\mathsf{Hyb}_j, \mathsf{Hyb}_{j+1})] &= \mathop{\mathbb{E}}_{i,\nu^1,\ldots,\nu^{i-1}}\left[\Pr_{\mathbf{a}^i_{\mathbf{j}},\mathbf{q}^i_{\mathbf{j}},\widehat{\boldsymbol{\pi}}}[\mathbf{a}^i_{\mathbf{j}} \neq \widehat{\boldsymbol{\pi}}(\mathbf{q}^i_{\mathbf{j}}) \mid i, \nu^1, \ldots, \nu^{i-1}]\right] \\ &= \mathop{\mathbb{E}}_{i,\nu^1,\ldots,\nu^{i-1}}\left[1 - \Pr_{\mathbf{a}^i_{\mathbf{j}},\mathbf{q}^i_{\mathbf{j}},\widehat{\boldsymbol{\pi}}}[\mathbf{a}^i_{\mathbf{j}} = \widehat{\boldsymbol{\pi}}(\mathbf{q}^i_{\mathbf{j}}) \mid i, \nu^1, \ldots, \nu^{i-1}]\right] \\ &= \mathop{\mathbb{E}}_{i,\nu^1,\ldots,\nu^{i-1},q^i_j,a^i_j}\left[1 - \Pr_{\widehat{\boldsymbol{\pi}}}[a^i_j = \widehat{\boldsymbol{\pi}}(q^i_j) \mid i, \nu^1, \ldots, \nu^{i-1}]\right] \\ \text{(since } 1-\alpha \leq \log(1/\alpha) \text{ for } \alpha\in[0,1]) \quad &\leq \mathop{\mathbb{E}}_{i,\nu^1,\ldots,\nu^{i-1},q^i_j,a^i_j}\left[\log\frac{1}{\Pr_{\widehat{\boldsymbol{\pi}}}[a^i_j = \widehat{\boldsymbol{\pi}}(q^i_j) \mid i, \nu^1, \ldots, \nu^{i-1}]}\right] \\ \text{(by the definition of oracle } \widehat{\boldsymbol{\pi}}) \quad &= \mathop{\mathbb{E}}_{i}\left[\mathrm{H}(\mathbf{a}^{\mathbf{i}}_{\mathbf{j}} \mid \boldsymbol{\nu}^1, \ldots, \boldsymbol{\nu}^{\mathbf{i}-1}, \mathbf{q}^{\mathbf{i}}_{\mathbf{j}})\right] \\ \text{(since } C_2 \notin \overline{\mathrm{CEB}}^{\mathrm{N}}_{2\eta/\ell,1.1\eta/\ell}) \quad &\leq 2\eta/\ell.\end{aligned}$$

□

5 A Combinatorial Locking Scheme

In this section we formally describe our simple "combinatorial" locking scheme which can substitute the algebraic locking scheme of [KPT97] and simplify their results. See Section B for a sketch on how a locking scheme was used by [KPT97] to construct efficient bounded-query ZK-PCPs for **NP** and (inefficient) ZK-PCPs for **NEXP**.

Definition 5.1 (Locking Scheme [KPT97])**.** In a *locking scheme* (LS) $\Sigma = (S, R)$ for the message space W_n, the *sender* $S = \{\boldsymbol{\sigma}_w\}$ is an (ensemble) of distributions over (locking) oracles and R is a poly(n)-time *receiver* and the following properties hold.

- **Completeness:** The sender S is given a private input $w \in W_n$ and both parties are given 1^n as the common input. Then the interaction continues in two phase:
  1. **Commitment:** A locking oracle is sampled $\sigma \leftarrow \boldsymbol{\sigma}_w$ and fixed for the next step.
  2. **Decommitment:** The sender reveals w and a "key" Key_w to the receiver R who gets oracle access to σ. The completeness property asserts that $\Pr[R^\sigma(w, \mathsf{Key}_w) = 1] = 1$.
- **Binding:** Σ is called $(1-\delta)$-binding if for every fixed oracle σ, there is a $w \in W_n$ such that for any other $w' \in W_n, w' \neq w$ and any $\mathsf{Key}_{w'}$, the probability that the receiver accepts $(w', \mathsf{Key}_{w'})$ is at most δ.
- **Equivocality:** We call Σ (u, ε)-equivocal if there exist some (black-box straight-line) *simulator* running in time poly(n, u) such that for every $w \in W_n$ and every malicious receiver $\widehat{R}$ who asks up to u oracle queries, the view of the receiver in $\widehat{R}^{\boldsymbol{\sigma}_w}$ is (statistically) ε-close to its view in the game below. (Here Sim does not rewind $\widehat{R}$ and simply interacts with it.)
  1. In the commitment phase, Sim answers up to u oracle queries of $\widehat{R}$ (without knowing w). Before $\widehat{R}$ asks more than u queries, $\widehat{R}$ can choose to go to the decommitment phase.
  2. The simulator Sim and receiver $\widehat{R}$ are now given $w \in W_n$ and Sim continues to answer oracle queries of $\widehat{R}$ for up to a total of u queries (including both steps).

  We call a LS Σ u-secure, if it is $(1 - 1/u)$-binding and $(u, 1/u)$-equivocal.

One can define a natural feature of hiding for locking schemes as well, but since **(1)** the construction of [KPT97] requires equivocality and **(2)** hiding property is directly implied by equivocality, we do not include a separate definition.

Our binding property is a relaxed version of that of [KPT97] which suffices for the purpose of constructing ZK-PCPs. In a locking scheme according to [KPT97], by sampling the locking oracle and receiver's randomness, the sender, in addition to being bound to a unique $w \in W$ is also bound to a single key Key_w. The only place where the binding property of the locking scheme is used in the construction of ZK-PCPs is to prove the soundness of the PCP. However, by inspecting the actual construction (see Construction B.2) it can be easily seen that our relaxed (and in fact more standard) definition of binding is sufficient.

Construction 5.2. Our locking scheme $\Sigma = (\{\boldsymbol{\sigma}_b\} = S, R)$ works as follows. The queries and answers of the locking oracle will be of length $\Theta(n)$ where n is the security parameter.

- **Commitment.** Let b be the bit that the sender S wants to commit to. The oracle $\boldsymbol{\sigma}_b = (f \mid h \mid g)$ is sampled by concatenating the following three randomized functions.

1. A uniformly random expanding function $f\colon \{0,1\}^n \mapsto \{0,1\}^{3n}$.
2. A uniformly random shrinking function $h\colon \{0,1\}^{3n} \mapsto \{0,1\}^n$.
3. A randomized function $g\colon \{0,1\}^{3n} \mapsto \{0,1\}^{3n}$ which uses the functions f and h as well as a randomized parameter $s \leftarrow \{0,1\}^n$ and is defined as $g(r) = b \cdot r + f(s + h(r))$. The addition and multiplication are performed componentwise in GF(2).

Note that any $\sigma \leftarrow \boldsymbol{\sigma}_b$ can be represented as a function $\sigma\colon \{0,1\}^{3n+2} \mapsto \{0,1\}^{3n}$.

- **Decommitment.** To decommit, the sender sends the bit b and the key $\mathsf{Key}_b = s$ to the receiver. The receiver then chooses a random $r \leftarrow \{0,1\}^{3n}$, and by asking one query to each of f, h, g (total of 3 queries to σ) makes sure that $g(r) = b \cdot r + f(s + h(r))$.

In order to use our locking scheme to simplify the ZK-PCPs of [KPT97] (see Construction B.2) we need to be able to put messages of polynomial length (rather than just single bits) inside the locking oracle. This can be done in a straightforward way by by generating locking oracles $\boldsymbol{\sigma}_{w_0}, \ldots, \boldsymbol{\sigma}_{w_k}$ for a k bit message $w = w_1 \ldots w_k$ and concatenating them into a single oracle $\boldsymbol{\sigma}_w = (\boldsymbol{\sigma}_{w_0} \mid \cdots \mid \boldsymbol{\sigma}_{w_k})$. If the original scheme is u-secure, by a union-bound, the new scheme will be $(\frac{u}{k})$-secure while the receiver R asks only $3k$ oracle queries. Since $k = \mathrm{poly}(n) = 2^{o(n)}$ the new scheme remains $2^{\Omega(n)}$ secure if the original is scheme is $2^{\Omega(n)}$-secure.

We also note that even though the oracle of Construction 5.2 has exponential length its security parameter n, when used in the context of [KPT97] n will be chosen at most $\mathrm{polylog}(|x|)$ where x is the PCP input (see Remark B.3).

Theorem 5.3. *The locking scheme $\Sigma = (\{\boldsymbol{\sigma}_b\}, R)$ of Construction 5.2 is $2^{n/3}$-secure.*

Comparison with [KPT97]. The locking scheme of [KPT97] achieves only $14/15$ soundness (which is worse than ours), but their verifier asks only two bit-queries from the locking oracle. However, to reduce their soundness error to $2^{-\Omega(n)}$ they need to execute the verifier $\Omega(n)$ times, and so the verifier ends up asking $\Omega(n)$ many oracle queries (in our locking scheme the receiver reads $\Theta(n)$ bits of the locking oracle through only 3 oracle queries). Also we note that the verifier of [KPT97] is nonadaptive while ours asks its three queries in two rounds. However, we emphasize that the bounded-query ZK-PCP of [KPT97] *is* in fact adaptive (see Section B), and using our locking scheme will increase its adaptivity only by a factor of two.

In the following we prove Theorem 5.3. (Completeness holds trivially.)

Binding: Suppose the sender intends to decommit into both $b = 0$ using $\mathsf{Key}_0 = s_0$ and $b = 1$ using $\mathsf{Key}_1 = s_1$. Since $|f(\{0,1\}^n)| \leq 2^n$, the number of possible $r \in \{0,1\}^{3n}$ for which there exist $\{y_1, y_2\} \subset f(\{0,1\}^n)$ such that $y_1 + r = y_2$ is at most $2^n \cdot 2^n = 2^{2n}$. Thus, with probability at least $1 - 2^{-3n}2^{2n} = 1 - 2^{-n}$ over the choice of r by the receiver, it holds that $f(s_1 + h(r)) \neq r + f(s_2 + h(r))$ regardless of the values s_1, s_2. For such r (which will be sampled with probability at least $1 - 2^{-n}$), $g(r)$ will be different from (at least) one of $f(s_1 + h(r))$ and $r + f(s_2 + h(r))$, which makes the verifier reject for either of $b = 0$ or $b = 1$ regardless of the value Key_b provided by the sender.

Equivocality: We will first prove a hiding property by showing that if the receiver asks at most $2^{n/3}$ queries, its view is $2^{-n/3}$-close to the case that it receives all random answers (not only upon asking f and h queries, but also for g queries). The argument behind this claim will tell us how to do the simulation. Consider the following mental hybrid experiments.

- **Experiment Hyb_b for $b \in \{0,1\}$.** In this game, in addition to f, g, and h oracles, there is another random oracle f' which is accessed only internally by the g queries (instead of f). Namely, upon a g-query we return $b \cdot r + f'(s + h(r))$. The oracle f' is *not* accessible by $\widehat{R}$.

Since f' is random, the view of $\widehat{R}$ while participating in Hyb_0 or Hyb_1 has the same distribution. The following claim carries the main intuition behind the hiding and equivocality properties.

Claim 5.4. *For every (computationally unbounded) receiver $\widehat{R}$ who asks at most u oracle queries, its view in $\widehat{R}^{\boldsymbol{\sigma}_b}$ is $u^2 \cdot 2^{-n}$ close to its view in the experiment Hyb_b.*

Proof. We prove the claim for $b = 0$, and the proof for $b = 1$ is identical. Let F, G, H and F' be in order the set of queries asked to f, g, h and f' in Hyb_0. For every g-query x, we put $x \in H$ and $s + h(x) \in F'$ as well (as if the relevant h and f' queries are asked). Since f' is not directly accessible, it holds that $F' = h(G) + s$. Also, note that $|X| \leq u$ for all $X \in \{F, G, H, F'\}$.

We define the event COL (collision) as $F' \cap F \neq \varnothing$. A key point is that as long as COL does not happen the experiment Hyb_0 proceeds the same as $\widehat{R}^{\boldsymbol{\sigma}_0}$. The reason is that in this case, we can pretend that f' queries were some f queries in both experiments without falling into any inconsistency between f and f'. Thus we only need to show that $\Pr_{\mathsf{Hyb}_0}[\mathsf{COL}] \leq u^2 \cdot 2^{-n}$. The nice thing about Hyb_b is that (because f' is not available to $\widehat{R}$) we can sample s and the set F' *at the end* of the experiment, which we call it the s-oblivious Hyb experiment).

- **s-Oblivious Experiment Hyb.** Here f and h queries are answered randomly. Given any g-query x, we first ask $h(x)$; if there was any x' queried to g before such that $h(x) = h(x')$ we return $g(x')$, otherwise, we return $g(x) \leftarrow \{0,1\}^{3n}$. At the end of the experiment sample $s \leftarrow \{0,1\}^n$, go over all $x \in G$ and put $h(x)+s$ in F'. Note that COL implies that $h(G)+s \cap F \neq \varnothing$. However, since we have $|G| \leq u$ which implies $|h(G)| \leq u$, by a union bound the probability that under a random shift s some member of $h(G)$ is mapped to F is at most $u \cdot u \cdot 2^{-n}$.

□

The Simulator. The main idea behind the simulator of Construction 5.2 was already exposed through a proof for the hiding property. The formal description follows. Our simulator Sim in its first step (before getting some $b \in \{0,1\}$) acts the same way as the s-oblivious simulator above. Let F, H and G be the set of queries as defined above, but we again keep the set F' undefined. When the game is about to move to Step 2, Sim acts as follows: Sample $s \leftarrow \{0,1\}^n$ and let $F' = s + h(G)$, and abort if $F' \cap F \neq \varnothing$. But if $F' \cap F = \varnothing$ holds, proceed to the second step and act as follows. Pretend that the sets G and H are the only sample queries of g and h. For every g-query x that we have sampled $g(x) = y$ before, take $z = s + h(x)$ (which has been a f' query). At this stage Sim is given the message bit b, and it adds z to F and sets $f(z) = g(x) + b \cdot r$ (this is a legitimate query and answer pair for f since $F' \cap F = \varnothing$). After this point, Sim can simply continue answering the queries according to the oracle $\boldsymbol{\sigma}_b$. So, if the simulation manages to go to the second step, during the second step it will easily answer the queries from the right distribution. To analyze the behavior of Sim in the first step we can again define the event COL as $F \cap F' \neq \varnothing$ (which makes it abort). For both $b \in \{0,1\}^n$, as long as COL does not happen, the game of Sim proceeds the same as Hyb_b. As we saw above, assuming COL does not happen the game $\widehat{R}^{\boldsymbol{\sigma}_b}$ *also* proceeds the same as Hyb_b. Thus, conditioned on COL not happening Sim's game and $\widehat{R}^{\boldsymbol{\sigma}_b}$ are also the same. Therefore the probability $\Pr_{\mathsf{Hyb}_0}[\mathsf{COL}] \leq u^2 \cdot 2^{-n}$ also bounds the statistical distance between the simulation game of Sim and $\widehat{R}^{\boldsymbol{\sigma}_b}$. Finally note that for $u \leq 2^{n/3}$, it holds that $u^2 \cdot 2^{-n} \leq 2^{-n/3}$.

6 Black-Box Sublinear Zero-Knowledge Arguments

In this section we prove Theorem 2.2 which follows from the following theorem.

Theorem 6.1. *Let $\mathcal{H} = \{h_\alpha \colon \{0,1\}^{2m} \mapsto \{0,1\}^m\}_{\alpha \in \{0,1\}^{\mathrm{poly}(m)}}$ be any $s(m)$-hard family of CRHFs and $s(m) = n^{\omega(1)}$. Using $\mathcal{H}$ only as a black-box, one can construct a constant-round zero-knowledge argument for* **NP** *with security parameter $n = |x|$ (where x is the common input),* $\mathrm{negl}(n)$ *soundness error and communication complexity* $\mathrm{poly}(m) + m \cdot \mathrm{polylog}(n)$. *Furthermore:*

1. *For the case of an honest verifier, the zero knowledge is statistical, the round complexity is 4 messages, and the protocol is public coin.*
2. *For the case of malicious-verifier zero knowledge, the round complexity is 5 messages.*

Concluding Theorem 2.2 from Theorem 6.1. First suppose $s(m) = m^\rho$ for $\rho(m) = \omega(1)$. Then let $\rho'(m) = \sqrt{\rho(m)}$ and let $n = m^{\rho'(m)} = m^{\omega(1)}$ which implies $m = n^{o(1)}$ and $s(m) = m^\rho = n^{\rho'} = n^{\omega(1)}$. Applying Theorem 6.1, the communication complexity will be $\mathrm{poly}(m) + m \cdot \mathrm{polylog}(n) = \mathrm{poly}(n^{o(1)}) + n^{o(1)}\,\mathrm{polylog}(n) = n^{o(1)}$ which is sublinear in n. Now suppose we have an exponentially secure CRHF; namely, $s(m) = 2^{m^{\Omega(1)}}$. Let c be a constant $c > 0$ such that $s(m) \geq 2^{m^c}$ for large enough m and let $m = \log^{2/c}(n)$ which implies $s(m) \geq 2^{\log^2(n)} = n^{\log(n)} = n^{\omega(1)}$. Applying Theorem 6.1, the communication complexity will be $\mathrm{poly}(m) + m \cdot \mathrm{polylog}(n) = \mathrm{poly}(\log^{2/c}(n)) + \log^{2/c}(n)\,\mathrm{polylog}(n) \leq \mathrm{polylog}(n)$. □

To prove Theorem 6.1 we use the following zero-knowledge PCP (see Definition 3.7 for the definition of RQ-ZK PCPs).

Theorem 6.2 (Restricted-Query ZK PCPs for **NP** [DFK+92, ALM+98, AS98])**.** *For any language $L \in$* **NP***, there is a 2-RQ-ZK PCP for L with these features:*

- *polynomial length,*
- *efficiently computable,*
- $O(1)$ *alphabet size,*
- *completeness 1,*
- *soundness* $\Omega(1)$*,*
- *non-adaptive honest verifier,*
- *straight-line simulator (*i.e. *the simulator only gets the two queries q_1, q_2, asked from the first and second half of the PCP, and simulates the answers).*

Dwork *et al.* [DFK+92] proved Theorem 6.2 in the *two-prover* setting where the honest or malicious verifiers ask one query from each of the provers in a non-adaptive way. Any such zero-knowledge proof system can be directly converted into a 2-RQZK PCP by having the first (resp. second) half of the PCP encode the answers of the first (resp. second) prover.

We will use a "direct product" version of the PCP of Theorem 6.2 formalized as follows.

Definition 6.3 (Direct Product of PCPs)**.** Let R be a relation for a promise language L and let $\Pi_1 = (\{\boldsymbol{\pi}^1_{x\in L, w\in R(x)}\}, \mathsf{V}_1)$ and $\Pi_2 = (\{\boldsymbol{\pi}^2_{x\in L, w\in R(x)}\}, \mathsf{V}_2)$ be two (randomized) PCPs for L (and same relation R). We define the PCP $\Pi = \Pi_1 \times \Pi_2 = (\{\boldsymbol{\pi}_{x\in L, w\in R(x)}\}, \mathsf{V})$, as the *direct product* of Π_1, Π_2 as follows:

- Every $\pi \leftarrow \boldsymbol{\pi}_{x\in L, w\in R(x)}$ is sampled by independently sampling $\pi^1 \leftarrow \{\boldsymbol{\pi}^1_{x\in L, w\in R(x)}\}$ and $\pi^2 \leftarrow \{\boldsymbol{\pi}^2_{x\in L, w\in R(x)}\}$ and taking the concatenation $\pi = (\pi^1 \mid \pi^2)$.
- The verifier V, given oracle access to $\pi = (\pi^1 \mid \pi^2)$, runs two independent instances of V_1 and V_2 where V_i accesses π^i. V accepts if both of V_1 and V_2 accept.

We also define $\Pi^2 = \Pi \times \Pi$ and $\Pi^i = \Pi \times \Pi^{i-1}$ recursively.

Remark 6.4 (Direct Product vs. Parallel Repetition)**.** The direct product of PCPs as defined in Definition 6.3 is different from the *parallel repetition* of PCPs. The latter preserves the number of queries by "combining" the parallel queries. It is known that parallel repetition of PCPs might not decrease the soundness error optimally [For89b, FV02], since the verifier of direct product PCPs run independent verifications, as stated in the next lemma, in direct produc PCPs the soundness error decreases optimally .

Lemma 6.5 (Properties of Direct Product of PCPs)**.** *Let $L \in \mathbf{NP}$, R be a relation for L, and $k = \mathrm{poly}(n)$ where n is the length of the PCP $\Pi = (\{\boldsymbol{\pi}_{x\in L, w\in R(x)}\}, \mathsf{V})$. Then:*

- *Π^k has $k \cdot \mathrm{poly}(n) = \mathrm{poly}(n)$ length.*
- *If Π is efficient, so is Π^k.*
- *Π^k has the same alphabet as that of Π and its number of verifier queries is k times that of Π.*
- *If Π has completeness 1, so does Π^k.*
- *If Π has soundness error δ, then Π^k has soundness error δ^k.*
- *If Π is non-adaptive, so is Π^k.*
- *If Π is HV-ZK (resp. with a straight-line simulator), then Π^k is also HV-Zk (resp. with a straight-line simulator).*
- *If Π is u-RQ-ZK (resp. with a straight-line simulator), then Π^k is also $(k \cdot u)$-HV-Zk (resp. with a straight-line simulator).*

We derive the following corollary follows from Theorem 6.2 and Lemma 6.5 by taking the PCP of Theorem 6.2 to the power $\log^2(n)/2$.

Corollary 6.6 (RQ-ZK PCP for **NP** with Negligible Sourness Error)**.** *Let $u = \log^2(n)$. For any language $L \in \mathbf{NP}$, there is a u-RQ-ZK PCP for L with these features:*

- *polynomial length,*
- *efficiently computable,*
- *$O(1)$ alphabet size,*

- *completeness 1,*
- *soundness error* $(1-\Omega(1))^{u/2} = \text{negl}(n)$*,*
- *non-adaptive honest verifier.*
- *straight-line simulator (that gets the set of all queries and simulates their answers).*

Note that by the definition of u-RQ-ZK PCPs, if one divides the PCP of Corollary 6.6 into u equal blocks, the honest verifiers of this PCP only asks one query from each block. This is the case also for the malicious verifiers whose views are statistically simulated by the straight-line simulator.

6.1 Black-Box Sublinear Honest-Verifier ZK Arguments for NP

In this subsection, we prove Part 6.1 of Theorem 6.1.

Now we can describe our basic black-box sublinear HV-ZK argument whose analysis proves Part of Theorem 6.1. Our construction simply uses the PCP of Corollary 6.6 in the basic 4-message argument construction of Kilian (which does not involve ZK). We use the statistically-hiding commitment scheme of [DPP97] to first statistically hide the bits of the PCP. The formal description of our protocol follows.

Construction 6.7 (Black-Box Sublinear HV-ZK Argument for **NP**)**. Setup:** The language L is in **NP** and the prover $\mathsf{P}_{\mathsf{arg}}$ is given a witness w for $x \in L$ where x is the common input between $(\mathsf{P}_{\mathsf{arg}}, \mathsf{V}_{\mathsf{arg}})$ and the security parameter is $n = |x|$. $\mathcal{H} = \{h_\alpha \colon \{0,1\}^{2m} \mapsto \{0,1\}^m\}_{\alpha \in \{0,1\}^{\text{poly}(m)}}$ is a family of shrinking hash functions which is $s(m)$-hard and $s(m) = n^{\omega(1)}$. $\Pi = (\{\boldsymbol{\pi}_{x\in L, w\in R(x)}\}, \mathsf{V}_{\mathsf{pcp}})$ is the PCP of Corollary 6.6 of length N. We also let $u = \log^2(n)$.

1. **Sample the Hash Function.** The verifier samples $\alpha \leftarrow \{0,1\}^{\text{poly}(m)}$ as an index of $h_\alpha \in \mathcal{H}$, and sends α to the prover.
2. **Committing to PCP.** The prover behaves as follows:
   (a) **Sample the PCP.** Using the witness w and private randomness, the prover samples $\pi \leftarrow \{\boldsymbol{\pi}_{x\in L, w\in R(x)}\}$. Let $\pi = (a_1, \ldots, a_N)$ where a_i is a PCP answer to query $i \in [N]$.
   (b) **Hide the PCP Bits.** Using the sampled hash function h_α as the index of the hash function, the prover commits to each of the answers a_i *independently* to get $C_i = C(a_i)$ using Construction 3.21.
   (c) **Hash Down the PCP.** The prover uses a Merkel hash-tree to hash the Boolean string $(C_1, \ldots, C_M)$ into a single block C of length $|C| = m$ as follows: First pad $(C_1, \ldots, C_M)$ into $m \cdot 2^t$ bits and let $(C_1^t, \ldots, C_{2^t}^t)$ be the result where $|C_j^t| = m$. Then for "tree layer" $i = t, t-1, \ldots, 1$ and for every $j \in [2^i]$ let $C_j^{i-1} = h_\alpha(C_{2j-1}^i, C_{2j}^i)$. Finally take $C = C_1^0$, and send C to the receiver.
3. **Sending the PCP Queries.** The verifier $\mathsf{V}_{\mathsf{arg}}$ runs the verifier $\mathsf{V}_{\mathsf{pcp}}$ of the PCP Π to get the queries $(q_1, \ldots, q_u)$ and sends them to the prover.
4. **Answering the Challenges.** The prover rejects if q_i is not asked from the i'th block of π (when π is divided into u equal blocks). Then for each query $q \in \{q_1, \ldots, q_u\}$ the prover sends

the following pieces of information to the verifier (all in parallel). **(1)** The decommitment D_q for the commitments C_q (where $q \in [N]$) that includes a_q and the randomness used in generating C_q. **(2)** It reveals the preimages of all the hash values on the path of the Merkel tree from C_j^t to $C = C^0$ for any C_j^t that intersects C_q. This, for every such j, consists of two blocks from each layer $i \in [t]$ of the hash-tree.

5. **Final Verification.** The verifier $\mathsf{V}_{\mathsf{arg}}$ first makes sure that the revealed decommitments and the hash preimages are consistent (also with the received message C) and rejects otherwise. Then it runs $\mathsf{V}_{\mathsf{pcp}}$ on the query-answer pairs $(q_1, a_{q_1}), \ldots, (q_u, a_{q_u})$ and rejects iff $\mathsf{V}_{\mathsf{pcp}}$ rejects.

Now we analyze the properties of the argument system of Construction 6.7.

Round Complexity, Completeness, and Communication Complexity. The round complexity of the protocol is clearly 4. The perfect completeness of the protocol follows from the completeness of the used PCP and commitment scheme. The first message is of length $\mathrm{poly}(m)$. The second message of the prover is of length m. The third message of the verifier is of length $u \cdot \log(N) = u \cdot O(\log n)$. The final message of the prover is of size $O(u \cdot m \log n)$. Therefore the total communication is of size $\mathrm{poly}(m) + m \cdot \mathrm{polylog}\, n$.

Computational Soundness. The proof of the computational soundness of our scheme is identical to the proof of the soundness of the basic 4-message construction of Kilian [Kil92]. Roughly speaking, the soundness follows from **(1)** the soundness of the PCP Π, **(2)** the computational binding of the commitment scheme of Theorem 3.22, and **(3)** the collision-resistance of the hash function $\mathcal{H}$. Barak and Goldreich [BG02] and Micali [Mic00] gave gave proofs for the soundness of such protocol, but for sake of completeness here we give a proof that is similar to the "chain extraction" algorithm of [MMV13].

Lemma 6.8. *Construction 6.7 is computationally sound.*

Proof. Suppose $\widehat{P}_{\mathsf{arg}}$ is a prover that succeeds in convincing $\mathsf{V}_{\mathsf{arg}}$ with nonnegligible probability ε. We will show how to use $\widehat{P}_{\mathsf{arg}}$ as an oracle in a $\mathrm{poly}(n/\varepsilon)$-time security reduction R that either finds a collision in $\mathcal{H}$ or finds a PCP that convinces $\mathsf{V}_{\mathsf{pcp}}$ with nonnegligible probability $\varepsilon/4 - \mathrm{negl}(n)$ (which is a contradiction if $x \notin L$).

The reduction R starts by sampling and fixing $\alpha \leftarrow \{0,1\}^{\mathrm{poly}(m)}$ as the index of the hash function and getting a commitment C from the prover $\widehat{P}_{\mathsf{arg}}$. By an averaging argument, with probability at least $\varepsilon/2$ over fixing (α, C), the adversary $\widehat{P}_{\mathsf{arg}}$ still can convince the verifier $\mathsf{V}_{\mathsf{arg}}$ with probability $\geq \varepsilon/2$. In the following we will assume that the sampled (α, C) has this property.

Now the reduction tries to "extract" a PCP $\widehat{\pi}$ from $\widehat{P}_{\mathsf{arg}}$ that convinces $\mathsf{V}_{\mathsf{pcp}}$ with probability $\varepsilon/4 - \mathrm{negl}(n)$ as follows. Let $\delta = \varepsilon/(4N)$ and $k = n/\delta$. The reduction simply runs k *independent* instances of the verifier $\mathsf{V}_{\mathsf{arg}}$ using the *same* fixed first message α. For every PCP query $q \in [N]$, if any of the k executions of $\mathsf{V}_{\mathsf{arg}}$ leads to an accept by asking the query q (as one of its u queries) we store the decommitted PCP answer a_q as the answer to the query q.

First note that, the probability of ever extracting *two different* answers $a_q \neq a'_q$ for the same query q is negligible, because by the definition of a Merkle Tree and the binding property the commitment of Construction 3.21 proved in Theorem 3.22, any such inconsistent openings can be efficiently turned into a collision for the hash function $\mathcal{H}$ which is assumed to be $n^{\omega(1)}$-hard collision

resistant. So in the following we assume that there will not be any inconsistent extracted PCP answers.

For the fixed (α, C), call $q \in [N]$ a *heavy* query, if $\Pr[\mathsf{V}_{\mathsf{arg}} \text{ asks } q \text{ and accepts}] \geq \delta$. Let Q_H be the set of heavy queries and $Q_L = [N] \setminus Q_H$ be the set of *light queries*. The probability of R *not* extracting an answer for a $q \in Q_H$ is at most $1 - (1-\delta)^{n/\delta} < 1 - 2^n$. Thus, with probability at least $1 - N2^n = 1 - \mathrm{negl}(n)$, PCP answers will be extracted for all of Q_H.

Let $\widehat{\pi}$ be a PCP constructed as follows: for every $q \in Q_H$, it answers the extracted answer a_q, and for all other queries it answers $\perp$. Now, let ρ_{pcp} be the probability of $\mathsf{V}_{\mathsf{pcp}}$ accepting $\widehat{\pi}$, and let ρ_{arg} be the probability of $\mathsf{V}_{\mathsf{arg}}$ accepting the interaction with $\widehat{\mathsf{P}}$ *without* asking any query $q \in Q_L$. Since the probability of extracting inconsistent answers is negligible, it holds that $\rho_{\mathsf{pcp}} \geq \rho_{\mathsf{arg}} - \mathrm{negl}(n)$. Now we show that ρ_{arg} cannot be too small.

Claim 6.9. $\rho_{\mathsf{arg}} \geq \varepsilon/4$.

Proof. For the fixed α, C, by the definition of Q_H and Q_L, for every fixed $q \in Q_L$ the probability of $\mathsf{V}_{\mathsf{arg}}$ accepting while asking q is at most δ. Since we already assumed that $\mathsf{V}_{\mathsf{arg}}$ accepts with probability at least $\varepsilon/2$, by a union bound it follows that she will accept without asking any heavy queries is at least $\varepsilon/2 - N\delta = \varepsilon/2 - \varepsilon/4 = \varepsilon/2$. □

By the claim above we get $\rho_{\mathsf{pcp}} \geq \varepsilon/4 - \mathrm{negl}(n)$ which is nonnegligible if ε is nonnegligible. □

Statistical Honest-Verifier Zero-Knowledge. In the following we will describe the simulator $\mathsf{Sim}_{\mathsf{arg}}$ for the case of honest-verifier, and prove that it is a statistical simulator.

- **Simulator's Algorithm.** Since we are in the *honest verifier* regime, the simulator $\mathsf{Sim}_{\mathsf{arg}}$ can choose the verifier PCP queries itself from their right distribution and use the simulator $\mathsf{Sim}_{\mathsf{pcp}}$ of the PCP Π to generate their answers. Namely, $\mathsf{Sim}_{\mathsf{arg}}$ samples the index of the hash function $\alpha \leftarrow \{0,1\}^{\mathrm{poly}(n)}$ and chooses a random seed r_{pcp} for the PCP verifier V_{pcp} to get the PCP queries $(q_1, \ldots q_u)$ determined by r_{pcp}. Then it invokes the *straight-line* (statistical) PCP simulator $\mathsf{Sim}_{\mathsf{pcp}}$ to generate the answers $(a_{q_1}, \ldots, a_{q_u})$ for the sequence of queries $(q_1, \ldots, q_u)$. Then, it builds a "malformed" PCP $\widehat{\pi}$ as follows: For every query $q \in [N]$, if $q \in \{q_1, \ldots, q_u\}$, use the simulated answer a_q, otherwise use $a_q = \perp$ where $\perp$ is a special symbol in the PCP alphabet. Using the PCP $\widehat{\pi}$, the simulator follows the protocol honestly according to the honest prover $\mathsf{P}_{\mathsf{arg}}$ by first sending the commitment C to the PCP $\widehat{\pi}$ and then answering the (already fixed) sequence of challenges $q_1, \ldots, q_u$.

- **Simulator's Analysis.** We claim that the simulated transcript is statistically close to the view of the honest verifier in Construction 6.7. This, roughly speaking, follows from the statistical quality of the simulated PCP answers by $\mathsf{Sim}_{\mathsf{pcp}}$ and the statistical hiding of the commitment scheme of Theorem 3.22. More formally fix the first message α as the index of the hash function. Let

$$W_R = ((q_1, \ldots, q_u), (D_{q_1}, \ldots, D_{q_u}), (C_1, \ldots, C_N))$$

be the random variable in a *real execution* of the protocol that consists of the verifier's queries, commitments to answers of the (honestly generated) PCP π, and the decommitments to the u queries. Let W_S be another random variable that defined similarly to W_R, but based on the malformed simulated PCP $\widehat{\pi}$. By statistical simulation of the answers to $q_1, \ldots, q_u$, it follows

that $(q_1, \ldots, q_u), (D_{q_1}, \ldots, D_{q_u})$ are statistically close between W_R and W_S. Moreover, by the statistical hiding property of the commitment scheme of Theorem 3.22 for *every* hash function (as a first message) the random variables W_R and W_S remain statistically close even when we append the commitment to the un-queries PCP answers to $(q_1, \ldots, q_u), (D_{q_1}, \ldots, D_{q_u})$. Moreover, other messages that the verifier receives from the prover are uniquely determined by the random variables W_R and W_S. Since the statistical distance cannot increase by applying the same (even randomized) function to two random variables, therefore the simulator's output is indeed statistically close to the view of the verifier in an actual interaction.

Remark 6.10. We note that in the analysis of the statistical zero-knowledge property of the protocol of Construction 6.7, we did not need the full power of query-restricted ZK property of the PCP, and honest-verifier ZK PCPs (with the extra features mentioned in Corollary 6.6) are indeed sufficient.

Unconditional Sublinear ZK Arguments in the Random Oracle Model. We observe that because our construction above only uses CRHFs as a black box, by replacing the CRHF with a random oracle, we get an (unconditional) ZK arguments for **NP** with polylogarithmic communication complexity. Furthermore, since the protocol is public-coin, has negligible soundness error, and has constant rounds, by applying the Fiat-Shamir [FS86] transformation, we obtain *non-interactive statistical zero-knowledge* arguments for **NP** with polylogarithmic communication complexity and negligible soundness error in the random oracle model. Pointcheval and Stern [PS96] proved that such transformation leads to sound arguments when using a random oracle. To prove the zero-knowledge we cannot follow the footsteps of our honest-verifier simulator anymore, because the PCP challenges (queries) randomly depend on the commitment C. However, using the "programmability" of the random oracle, we can still simulate the view of the honest verifier as follows. The simulator will use a completely malformed PCP (with all answers equal to $\perp$) to compute the commitment. However, this commitment is statistically hiding the PCP answers, and so given the PCP queries, *information theoretically* it is possible to decommit into any answers (*e.g.* answers generated by the straight-line simulator $\mathsf{Sim}_{\mathsf{pcp}}$). This is efficiently possible as long as the simulator can program the random oracle and choose to open a commitments to any answer.

6.2 Black-Box Sublinear Malicious Verifier ZK Arguments for NP

In this subsection, we prove Part 2 of Theorem 6.1 to achieve (computational) zero-knowledge against *malicious verifiers* as well. Our protocol is an extension of the protocol of Construction 6.7. At a high level, the idea is to use a variant of the Goldreich-Kahan (GK) [GK96] technique: Namely, we ask the verifier to commit to its queries in advance, and then open these queries later. There is a fundamental difficulty in directly applying the same analysis as that of [GK96] which yields a *proof* system, whereas in our setting we can only hope to achieve an argument system. What we show is that, nevertheless, by having the verifier use a statistically hiding commitment to commit to its queries initially, and then having the prover use another statistically hiding commitment to commit to the PCPs, everything still works out.

Construction 6.11 (Black-Box Sublinear Malicious Verifier ZK Argument for **NP**)**.** The setup is the same as that of Construction 6.7.

1. The prover samples an index $\widetilde{\alpha} \leftarrow \{0,1\}^{\mathrm{poly}(m)}$ of $h_{\widetilde{\alpha}} \leftarrow \mathcal{H}$ and sends $\widetilde{\alpha}$ to the verifier.

2. The verifier V_{arg} executes an independent instance of the PCP verifier V_{pcp} to get the queries $(q_1, \ldots, q_u)$. Let $(b_1 \ldots b_v) = (q_1 \ldots q_u)$ be a Boolean representation of the queries (note that $|q_i| = \log N = O(\log n)$, and so $v = O(\log^3 n)$). Using the hash index $\widetilde{\alpha}$ in commitment scheme of Theorem 3.22, the verifier commits to all bits $b_1, \ldots, b_v$ *independently* and gets the commitments $(\widetilde{C}_1, \ldots, \widetilde{C}_v)$. These commitments are sent to the prover.

3. In the rest of the protocol, the verifier and the prover engage in executing the protocol of Construction 6.7 with the following modifications:

   - Instead of sampling the PCP queries in Step 3, the verifier will send the *already sampled* PCP queries $q_1, \ldots, q_u$ along with the relevant decommitments $\widetilde{D}_1, \ldots, \widetilde{D}_v$ for the commitments $\widetilde{C}_1, \ldots, \widetilde{C}_v$ to the bits $(b_1 \ldots b_v) = (q_1 \ldots q_u)$.
   - When the prover receives the queries, $q_1, \ldots, q_u$ and decommitments $\widetilde{D}_1, \ldots, \widetilde{D}_v$, in Step 4, the prover $\mathsf{P}_{\mathsf{arg}}$ first makes sure that the decommitments are consistent with the commitments $\widetilde{C}_1, \ldots, \widetilde{C}_v$, and only in this case answers the PCP challenges (by sending the relevant decommitments and hash preimages).

Round Complexity, Completeness, and Communication Complexity. The Commitments sent by the verifier in Step 2 can be sent along the first message of the following sub-protocol (of Construction 6.7). So the total number of messages are 5. The perfect completeness of the protocol follows from the completeness of the used PCP and commitment scheme. The communication complexity overhead compared to that of Construction 6.7 is the verifier's commitments to $v = O(\log^3 n)$ bits, each takes $O(m)$ plus the opening to these commitments that also takes $O(m)$ bits each. So the total communication complexity remains $\text{poly}(m) + m \times \text{polylog}(n)$.

Computational Zero-Knowledge. The zero-knowledge property of our scheme follows the original GK analysis. For sake of completeness, here we sketch the GK analysis. The simulator simply proceeds in the protocol acting as the prover, but commits to an arbitrary string instead of committing to an actual PCP. If the verifier refuses or fails to open her commitment to any of its PCP queries, then the simulator aborts (similarly to what the honest prover would do). However, if the verifier decommits her PCP queries successfully, then the simulator rewinds the verifier back to the point after she committed to her queries. The simulator now "knows" the PCP queries committed by the verifier (more precisely, by continuing the interaction, the verifier cannot decommit to anything different, or else the collision-resistance of the hash function could be broken efficiently). The simulator then runs the PCP simulator to obtain the simulated PCP answers. She can always do so, because the prover enforces the restriction on the places that the queries are being asked and for such set of queries the PCP simulator's output is statistically close to actual PCP answers. The simulator uses the simulated answers (and puts $\perp$ for other answers) in the PCP, and follows Prover's strategy using this PCP. The simulator keeps rewinding the verifier till the verifier opens the commitments successfully and the simulator can finish the simulation. Since the prover/simulator commitments are statistically hiding, the distribution of the commitments after rewinding is statistically close to the distribution of the commitments before rewinding. This means that the probability that the verifier properly opens her commitments will not change after the rewinding (by more than a negligible), so *the average* number of rewindings is $O(1)$.

Intuition behind Computational Soundness. The main novelty of our approach lies in the proof of (computational soundness). In the original GK transformation, soundness was easy to argue because the prover is statistically bound to its commitments, whereas the verifier is statistically hiding the commitment to her queries. Thus, a simple statistical argument showed that a cheating prover must get caught with high probability. In our setting, however, since both sides are only computationally bound to their commitments, in order to rule out a cheating prover that succeeds too often, we need to give a *computationally efficient* reduction that converts such a prover into an entity that breaks the binding property of our commitment scheme. However, in order to "catch" the prover, the reduction will need to be able to *efficiently* open the verifier's commitment to its query in multiple ways[8]. Of course, by assumption, it is not possible to efficiently open our commitments in multiple ways. We resolve this deadlock by using non-uniformity: We non-uniformly fix a prefix to the protocol including and up to the verifier's commitment to the bits of its query along with openings of each commitment to both 0 and 1, such that the cheating prover's overall success probability is not affected by this non-uniform fixing. Then, since the reduction can now open the query to be anything it likes, we can immediately apply the soundness argument for Kilian's protocol to finish the proof of soundness.

Analysis of Computational Soundness. To argue computational soundness, we present a non-uniform reduction of the soundness of Construction 6.11 to the soundness of the basic Construction 6.7. Let $\widetilde{\mathsf{P}}$ be a cheating prover that is able to convince the honest verifier to accept x with noticeable probability ε while $x \notin L$. We will show how to derive a malicious prover $\widehat{\mathsf{P}}$ for Construction 6.7 that succeeds with probability $\varepsilon - \mathrm{negl}(n)$.

We consider an *inefficient* verifier $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ who is the same as $\mathsf{V}_{\mathsf{arg}}$ with the following modifications:

- Instead of committing to $(b_0 \dots b_v) = (q_1 \dots q_u)$, $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ simply commits to v zeros (independently) and gets $(\widetilde{C}_1, \dots, \widetilde{C}_v)$. Then, for every $i \in [v]$, $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ samples two commitment random seeds r_i^0, r_i^1 where r_i^b is a uniformly sampled random seed that is consistent with opening $\widetilde{C}_i$ into the bit b. ($\widetilde{\mathsf{V}}_{\mathsf{arg}}$ would abort if such random seed does not exist).
- When $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ is asked to open the commitments $(\widetilde{C}_1, \dots, \widetilde{C}_v)$:
  1. $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ samples PCP queries $(q_1, \dots, q_u)$ represented in Boolean as: $(b_1 \dots b_v) = (q_1 \dots q_u)$.
  2. $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ returns the decommitments $(\widetilde{D}_1, \dots, \widetilde{D}_u)$ where $\widetilde{D}_i = (b_i, r_i^{b_i})$.

Claim 6.12. *The transcript of the transcript of the interaction when the verifier is $\mathsf{V}_{\mathsf{arg}}$ is statistically close to the case the verifier is $\widetilde{\mathsf{V}}_{\mathsf{arg}}$.*

Proof. This directly follows from the statistical hiding property of the commitment scheme Construction 3.21 as proved in Theorem 3.22. More formally, for every $i \in [v]$, the commitment $\widetilde{C}_i$

[8] We stress that if this were not the case, it could reflect a real vulnerability in the protocol and not just a problem with the proof. Namely, it could reflect a malleability problem that allows the prover to somehow maul the verifier's commitment and opening into a commitment and opening of the PCP in a way that satisfies the verifier. Our proof rules this out. At a more intuitive level (for specialists in this area), the reason such a malleability attack does not arise is because the prover must choose the hash function underlying the verifier's commitment scheme before the verifier chooses the hash function that will underly the prover's commitment. Because an honest verifier will choose this hash function at random, and by this time the prover has already chosen the other hash function, the cheating prover will no longer have the power to maul anything that the verifier provides.

where the verifier commits to 0 is statistically close to where she commits to some bit $b \in \{0,1\}$. By Lemma 3.1, the *joint* distribution of $(\widetilde{C}_1, \ldots, \widetilde{C}_v)$ is also statistically close when comparing the execution of $\mathsf{V}_{\mathsf{arg}}$ to that of $\widetilde{\mathsf{V}}_{\mathsf{arg}}$. Finally, we can always pretend (as a mental experiment) that $\mathsf{V}_{\mathsf{arg}}$ also chooses the randomness r_i needed for the decommitment $\widetilde{D}_i$, *after* sending the commitment $\widetilde{C}_i$. Thus, the statistical distance between two games cannot increase in the decommitment step. □

The above claim shows that $\widetilde{\mathsf{P}}$ succeeds in convincing $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ with probability $\varepsilon - \mathrm{negl}(n) = \widetilde{\varepsilon}$. By an averaging argument, we can always *fix* the randomness r of $\widetilde{\mathsf{P}}$ (that also fixes $\widetilde{\alpha}$), the commitments $(\widetilde{C}_1, \ldots, \widetilde{C}_v)$, and the random seeds $(r_1^0, r_1^1), \ldots, (r_v^0, r_v^1)$ so that when $\widetilde{\mathsf{P}}$ uses r as its randomness (denoted by $\widetilde{\mathsf{P}}_r$) and $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ uses the commitments $(\widetilde{C}_1, \ldots, \widetilde{C}_v)$ and random seeds $(r_1^0, r_1^1), \ldots, (r_v^0, r_v^1)$, $\widetilde{\mathsf{P}}_r$ still succeeds to convince $\widetilde{\mathsf{V}}_{\mathsf{arg}}$ with probability $\geq \varepsilon - \mathrm{negl}(n)$. Using these fixed strings as *nonuniform advice*, we efficiently construct a prover $\widehat{\mathsf{P}}$ that uses $\widetilde{\mathsf{P}}_r$ as a black-box and convinces the verifier of Construction 6.7 to accept with the same probability $\widetilde{\varepsilon}$ as follows.

- Fixing $r, (\widetilde{C}_1, \ldots, \widetilde{C}_v), (r_1^0, r_1^1), \ldots, (r_v^0, r_v^1)$ has already fixed the first message $\widetilde{\mathsf{P}}_r$ sends (*i.e.* $\widetilde{\alpha}$) and the commitment messages it receives.
- It starts $\widehat{P}$ receives the hash index α and forwards it to $\widetilde{\mathsf{P}}_r$ (along with the already fixed commitments $(\widetilde{C}_1, \ldots, \widetilde{C}_v)$).
- $\widehat{P}$ receives the commitment C from $\widetilde{\mathsf{P}}_r$ and forwards it to the verifier.
- $\widehat{P}$ receives the PCP queries $(q_1 \ldots q_u) = (b_1 \ldots b_v)$ from the verifier. For each b_i, it lets $\widetilde{D}_i = (b_i, r_i^{b_i})$ and sends the decommitments $\widetilde{D}_1, \ldots, \widetilde{D}_v$ to $\widetilde{\mathsf{P}}_r$.
- $\widehat{P}$ forwards the last message of $\widetilde{\mathsf{P}}_r$ to the verifier.

It is immediate that the behavior of $\widehat{P}$ when interacting with the external verifier of Construction 6.7 is identical to the behavior of $\widetilde{\mathsf{P}}_r$ when interacting with $\widetilde{V}_{\mathsf{arg}}$. Therefore, $\widehat{P}$ also manages to succeed with nonnegligible probability $\widetilde{\varepsilon}$.

ACKNOWLEDGEMENT. We thank Vipul Goyal for collaboration at an early stage of this work. We would also like to thank Kai-Min Chung and Rafael Pass for insightful discussions.

References

[AH91] William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, 1991. Preliminary version in *FOCS'87*.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *FOCS'92*.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *FOCS'92*.

[BCG+11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *ASIACRYPT*, pages 722–739, 2011.

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. In *FOCS*, pages 16–25, 1990.

[BFLS91] Babai, Fortnow, Levin, and Szegedy. Checking computations in polylogarithmic time. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1991.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 103–112, 1988.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. pages 194–203, 2002.

[BGG+88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *Advances in Cryptology – CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1988.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *STOC*, pages 113–131, 1988.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987.

[CDSMW08] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 427–444. Springer, 2008.

[CDSMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 387–402. Springer, 2009.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[CGS08] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In *EUROCRYPT*, pages 545–562, 2008.

[CLMP13] Kai-Min Chung, Huijia Lin, Mohammad Mahmoody, and Rafael Pass. On the power of nonuniformity in proofs of security. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 389–400, New York, NY, USA, 2013. ACM.

[DFK+92] Cynthia Dwork, Uriel Feige, Joe Kilian, Moni Naor, and Shmuel Safra. Low communication 2-prover zero-knowledge proofs for NP. In *CRYPTO*, pages 215–227, 1992.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, pages 378–394, 2005.

[DMMQN13] Nico Döttling, Thilo Mie, Jörn Müller-Quade, and Tobias Nilges. Implementing resettable uc-functionalities with untrusted tamper-proof hardware-tokens. In *TCC*, pages 642–661, 2013.

[DPP97] Damgard, Pedersen, and Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, 10, 1997.

[DPP98] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory*, 44(3):1143–1151, 1998.

[For89a] Lance Fortnow. The complexity of perfect zero-knowledge. *Advances in Computing Research: Randomness and Computation*, 5:327–343, 1989.

[For89b] Lance Jeremy Fortnow. *Complexity-theoretic aspects of interactive proof systems.* PhD thesis, Massachusetts Institute of Technology, Dept. of Mathematics, 1989.

[FRS94] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[FV02] Feige and Verbitsky. Error reduction by parallel repetition–A negative result. *COMBINAT: Combinatorica*, 22, 2002.

[GIMS10] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2010.

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2010.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy Rothblum. One-time programs. In *CRYPTO*, Lecture Notes in Computer Science, pages 39–56. Springer, 2008.

[GMOS07] Vipul Goyal, Ryan Moriarty, Rafail Ostrovsky, and Amit Sahai. Concurrent statistical zero-knowledge arguments for NP from one way functions. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2007.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version in *STOC'85*.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991. Preliminary version in *FOCS'86*.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[GOVW12] Sanjam Garg, Rafail Ostrovsky, Ivan Visconti, and Akshay Wadia. Resettable statistical zero knowledge. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 494–511. Springer, 2012.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 695–704. ACM, 2011.

[HIK+11] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM J. Comput*, 40(2):225–266, 2011.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *In CRYPTO '96*, pages 201–215. Springer-Verlag, 1996.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[IMS12] Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge PCPs. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 151–168. Springer, 2012.

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, Lecture Notes in Computer Science, pages 115–128. Springer, 2007.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC)*, pages 723–732, 1992.

[Kol10] Vladimir Kolesnikov. Truly efficient string oblivious transfer using resettable tamper-proof tokens. In *TCC*, pages 327–342, 2010.

[KPT97] Joe Kilian, Erez Petrank, and Gábor Tardos. Probabilistically checkable proofs with zero knowledge. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1997.

[KR08] Yael Tauman Kalai and Ran Raz. Interactive PCP. In *ICALP (2)*, pages 536–547, 2008.

[Mei09] Or Meir. Combinatorial PCPs with efficient verifiers. In *FOCS*, pages 463–471. IEEE Computer Society, 2009.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. Preliminary version in *FOCS'94*.

[MMV13] Mohammad Mahmoody, Tal Moran, and Salil Vadhan. Publicly verifiable proofs of sequential work. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 373–388, New York, NY, USA, 2013. ACM.

[MS08] Tal Moran and Gil Segev. David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In *EUROCRYPT*, pages 527–544, 2008.

[MV03] Daniele Micciancio and Salil Vadhan. Statistical zero-knowledge proofs with efficient provers: lattice problems and more. In *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 282–298. Springer, 2003.

[MX13] Mohammad Mahmoody and David Xiao. Languages with efficient zero-knowledge PCPs are in SZK. In *TCC*, pages 297–314, 2013.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in *CRYPTO'89*.

[Oka96] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 649–658. ACM Press, 1996.

[OV08] Shien Jin Ong and Salil P. Vadhan. An equivalence between zero knowledge and commitments. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 482–500. Springer, 2008.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proceedings of the 2nd Israel Symposium on Theory of Computing Systems*, pages 3–17. IEEE Computer Society, 1993.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology—EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 12–16 May 1996.

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 403–418. Springer, 2009.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[Vad99] Salil P. Vadhan. *A Study of Statistical Zero-Knowledge Proofs*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1999.

[Vad06] Salil P. Vadhan. An unconditional study of computational zero knowledge. *SIAM Journal on Computing*, 36(4):1160–1214, 2006. Preliminary version in *FOCS'04*.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *FOCS*, pages 531–540. IEEE Computer Society, 2010.

A Omitted Proofs

Lemma 3.10 (Restated)**.** *For a constant k, let $L_1, \ldots, L_k$ be a set of promise languages all in* **SZK***, and let F be a constant-size k-input formula with operations: complement, conjunction, and disjunction as in Definition 3.2. Then $F(L_1, \ldots, L_k) \in$* **SZK***.*

Proof Sketch. We will use the following:

Theorem A.1 ([Oka96])**.** *The class* **SZK** *is closed under complement.*

Since k is constant, we just need to prove the claim for formulas which have only a single operation, and then the lemma follows by an induction. Moreover since $L_1 \vee L_2 = \overline{\overline{L_1} \wedge \overline{L_1}}$, we just need to prove the claim for complement and conjunction operations. Theorem A.1 proves this for the complement. To obtain $L_1 \wedge L_2 \in$ **SZK**, given the input (x_1, x_2), the prover provides an (interactive) **SZK** proof that $x_1 \in L_1^{\mathrm{Y}}$ and then (if the first interaction is accepted), he also provides a **SZK** proof that $x_2 \in L_2^{\mathrm{Y}}$. (More formally, the prover and the verifier start with an *amplified* version of the original protocols with soundness error $< 1/6$, the soundness error of the sequential composition in this case remains $< 1/3$). On the other hand, if either of $x_1 \in L_1^{\mathrm{N}}, x_2 \in L_2^{\mathrm{N}}$ holds, the corresponding interaction rejects with probability at least $2/3$. □

Lemma 3.16 (Restated)**.** (Conditional Entropy to Statistical Distance)**.** *Suppose* $\mathrm{Supp}(\mathbf{X}) \subseteq \{0,1\}^n$*. Then it holds that* $\mathbb{E}_{Y \leftarrow \mathbf{Y}} \Delta((\mathbf{X} \mid Y), \mathbf{U}_n) \leq \sqrt{n - \mathrm{H}(\mathbf{X} \mid \mathbf{Y})}$*.*

Proof. We use the following definition.

Definition A.2 (Kullback-Leibler Divergence)**.** For random variables $\mathbf{X}, \mathbf{Y}$ such that $\mathrm{Supp}(\mathbf{X}) \subseteq \mathrm{Supp}(\mathbf{Y})$, the Kullback-Leibler divergence is defined as $\mathrm{KL}(\mathbf{X}, \mathbf{Y}) = \mathbb{E}_X \log(\frac{\Pr[\mathbf{X}=X]}{\Pr[\mathbf{Y}=X]})$.

It can be verified by straightforward calculation that $\mathrm{H}(\mathbf{X}) = n - \mathrm{KL}(\mathbf{X}, \mathbf{U}_n)$.

Pinsker's inequality states that for any random variables $\mathbf{X}, \mathbf{Y}$, it holds that $\Delta(\mathbf{X}, \mathbf{Y}) \leq \sqrt{\mathrm{KL}(\mathbf{X}, \mathbf{Y})}$. Applying Pinsker's inequality to $(\mathbf{X} \mid Y)$ and $\mathbf{U}_n$ for every fixed value of $Y \leftarrow \mathbf{Y}$ and using Jensen's inequality, we have:

$$\mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}} \Delta(\mathbf{X} \mid Y, \mathbf{U}_n) \leq \mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}}[\sqrt{n - \mathrm{H}(\mathbf{X} \mid Y)}] \leq \sqrt{\mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}}[n - \mathrm{H}(\mathbf{X} \mid Y)]} = \sqrt{n - \mathrm{H}(\mathbf{X} \mid \mathbf{Y})}$$

□

Lemma 3.17 (Restated)**.** (Statistical Distance to Conditional Entropy)**.** *For $\varepsilon \in [0, 1]$ let $\mathrm{H}(\varepsilon) = \varepsilon \log(1/\varepsilon) + (1 - \varepsilon) \log(1/1-\varepsilon)$. Suppose $\Delta((\mathbf{X}, \mathbf{Y}), (\mathbf{X}', \mathbf{Y}')) \leq \varepsilon$ and $\mathrm{Supp}(\mathbf{X}) \cup \mathrm{Supp}(\mathbf{X}') \subseteq \{0, 1\}^n$. Then it holds that $|\mathrm{H}(\mathbf{X} \mid \mathbf{Y}) - \mathrm{H}(\mathbf{X}' \mid \mathbf{Y}')| \leq 4(\mathrm{H}(\varepsilon) + \varepsilon \cdot n)$.*

To prove this lemma, we need the following:

Lemma A.3. *Suppose $\Delta((\mathbf{X}, \mathbf{Y}), (\mathbf{X}', \mathbf{Y}')) \leq \varepsilon$ and let $\mathbf{Z} = (\mathbf{X}'', \mathbf{Y})$ be a random variable distributed as follows. The component $Y \leftarrow \mathbf{Y}$ is sampled, and then $\mathbf{X}''$ conditioned on Y is distributed similar to $(\mathbf{X}' | \mathbf{Y}' = Y)$. Then it holds that $\Delta((\mathbf{X}, \mathbf{Y}), \mathbf{Z}) \leq 2\varepsilon$.*

Proof. It is sufficient to show that $\Delta((\mathbf{X}', \mathbf{Y}'), \mathbf{Z}) \leq \varepsilon$, which is true because the second components have statistical distance at most ε and the first components have statistical distance zero *conditioned* on the second components being equal. □

Proof of 3.17. We first prove the lemma for the case that $\mathbf{Y}, \mathbf{Y}'$ do not exist (*i.e.* $\Delta(\mathbf{X}, \mathbf{X}') \leq \varepsilon$). It is well known that in this case there is a random variable $\overline{\mathbf{X}}$ which has a measure of $1 - \varepsilon$ in *both* of $\mathbf{X}$ and $\mathbf{X}'$. Namely, one can think of a Boolean random variable $\mathbf{b}$ jointly distributed with $\mathbf{X}, \mathbf{X}'$ such that $\Pr[\mathbf{b} = 0] = \varepsilon$ and $(\mathbf{X} \mid \mathbf{b} = 1) \equiv \overline{\mathbf{X}} \equiv (\mathbf{X}' \mid \mathbf{b} = 1)$. Based on this "decomposition" we get:

$$\mathrm{H}(\mathbf{X}) \geq \mathrm{H}(\mathbf{X} \mid \mathbf{b}) \geq (1 - \varepsilon)\, \mathrm{H}(\mathbf{X} \mid \mathbf{b} = 1) = (1 - \varepsilon)\, \mathrm{H}(\overline{\mathbf{X}}).$$

On the other hand it holds that

$$\mathrm{H}(\mathbf{X}) \leq \mathrm{H}(\mathbf{b}) + \mathrm{H}(\mathbf{X} \mid \mathbf{b}) = \mathrm{H}(\varepsilon) + \varepsilon\, \mathrm{H}(\mathbf{X} \mid b = 0) + (1 - \varepsilon) \cdot \mathrm{H}(\mathbf{X} \mid b = 1) \leq \mathrm{H}(\varepsilon) + \varepsilon n + (1 - \varepsilon)\, \mathrm{H}(\overline{\mathbf{X}}).$$

Namely, both of $\mathrm{H}(\mathbf{X}), \mathrm{H}(\mathbf{X}')$ are lower-bounded by $(1 - \varepsilon)\, \mathrm{H}(\overline{\mathbf{X}})$ an upper-bounded by $\mathrm{H}(\varepsilon) + \varepsilon n + (1 - \varepsilon)\, \mathrm{H}(\overline{\mathbf{X}})$, and therefore $|\mathrm{H}(\mathbf{X}) - \mathrm{H}(\mathbf{X}')| \leq \mathrm{H}(\varepsilon) + \varepsilon n$.

Now, using the result above, we prove the conditional case through a hybrid argument. Given the two pairs of random variables $(\mathbf{X}, \mathbf{Y}), (\mathbf{X}', \mathbf{Y}')$ define the hybrid random variable $\mathbf{Z} = (\mathbf{X}'', \mathbf{Y})$ as defined in the statement of Lemma A.3. We claim that

1. $|\mathrm{H}(\mathbf{X}'' \mid \mathbf{Y}) - \mathrm{H}(\mathbf{X} \mid \mathbf{Y})| \leq 2(\mathrm{H}(\varepsilon) + \varepsilon n)$, and
2. $|\mathrm{H}(\mathbf{X}'' \mid \mathbf{Y}) - \mathrm{H}(\mathbf{X}' \mid \mathbf{Y}')| \leq 2\varepsilon n$.

Using these two bounds, Lemma 3.17 follows by a triangle inequality.

We obtain the first bound as follows.

$$
\begin{aligned}
|\,\mathrm{H}(\mathbf{X}'' \mid \mathbf{Y}) - \mathrm{H}(\mathbf{X} \mid \mathbf{Y})| &\le \mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}} \left[|\,\mathrm{H}(\mathbf{X}'' \mid Y) - \mathrm{H}(\mathbf{X} \mid Y)| \right] \\
&\le \mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}} \left[\mathrm{H}(\Delta(\mathbf{X}'' \mid Y, \mathbf{X} \mid Y)) + \Delta(\mathbf{X}'' \mid Y, \mathbf{X} \mid Y) \cdot n \right] \\
\text{(by concavity of Entropy)} \quad &\le \mathrm{H}\left(\mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}} \left[\Delta(\mathbf{X}'' \mid Y, \mathbf{X} \mid Y) \right] \right) + \mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}} \left[\Delta(\mathbf{X}'' \mid Y, \mathbf{X} \mid Y) \right] \cdot n \\
\text{(by Lemma A.3)} \quad &\le \mathrm{H}(2\varepsilon) + (2\varepsilon) \cdot n \\
\text{(by concavity of Entropy)} \quad &\le 2(\mathrm{H}(\varepsilon) + \varepsilon \cdot n).
\end{aligned}
$$

To obtain the second bound we do as follows.

$$
\begin{aligned}
|\,\mathrm{H}(\mathbf{X}'' \mid \mathbf{Y}) - \mathrm{H}(\mathbf{X}' \mid \mathbf{Y}')| &= \left| \mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}} \left[\mathrm{H}(\mathbf{X}' \mid \mathbf{Y}' = Y) \right] - \mathop{\mathbb{E}}_{Y \leftarrow \mathbf{Y}'} \left[\mathrm{H}(\mathbf{X}' \mid \mathbf{Y}' = Y) \right] \right| \\
&= \left| \sum_{Y \in \mathrm{Supp}(\mathbf{Y}) \cup \mathrm{Supp}(\mathbf{Y}')} \left(\Pr[\mathbf{Y} = Y] - \Pr[\mathbf{Y}' = y] \right) \cdot \mathrm{H}(\mathbf{X}' \mid \mathbf{Y}' = Y) \right| \\
&\le \sum_{Y \in \mathrm{Supp}(\mathbf{Y}) \cup \mathrm{Supp}(\mathbf{Y}')} \left| \Pr[\mathbf{Y} = Y] - \Pr[\mathbf{Y}' = y] \right| \cdot n \\
&\le 2\Delta(\mathbf{Y}, \mathbf{Y}') \cdot n
\end{aligned}
$$

□

Lemma 3.13 (Restated)**.** *For any $\varepsilon > 1/\operatorname{poly}(n)$, $\mathrm{CEA}_\varepsilon \in \mathbf{SZK}$.*

Proof. We give a reduction from CEA_ε to CEA_1, which is known to be **SZK**-complete [Vad06]. The reduction maps

$$(\mathbf{X}, \mathbf{Y}, r) \mapsto ((\mathbf{X}^1, \ldots, \mathbf{X}^{1/\varepsilon}), (\mathbf{Y}^1, \ldots, \mathbf{Y}^{1/\varepsilon}), r/\varepsilon)$$

where for every $i \in [1/\varepsilon]$, $(\mathbf{X}^i, \mathbf{Y}^i)$ is sampled identically to $(\mathbf{X}, \mathbf{Y})$ and independently of all other components (*i.e.* by an independent copy of the circuit C). It is easy to see that

$$\mathrm{H}((\mathbf{X}^1, \ldots, \mathbf{X}^{1/\varepsilon}) \mid (\mathbf{Y}^1, \ldots, \mathbf{Y}^{1/\varepsilon})) = \frac{1}{\varepsilon} \cdot \mathrm{H}(\mathbf{X} \mid \mathbf{Y}).$$

□

B Using Locking Schemes in Bounded-Query ZK-PCPs

In this section we briefly sketch the results of [KPT97, DFK+92] which are relevant to us.

The proof of [DFK+92] for Theorem 6.2 was used by [KPT97] to show how to convert any PCP for a language L (possibly out of **NP**) into another PCP for the same language L which is zero-knowledge against running the *honest verifier* a few times *independently*. More formally let $\Pi' = (\{\boldsymbol{\pi}'_x\}, \mathsf{V}')$ be a PCP with the following properties:

1. For $\pi'_x \leftarrow \boldsymbol{\pi}'_x, |x| = n$, the PCP length is at most $|\pi'_x| \le N$.

2. V' tosses $O(\log N)$ many random bits and asks $O(1)$ PCP queries.

3. Π' has completeness 1 and soundness error $c' < 1$.

The PCP Π' could be any PCP for **NP** of polynomial length [AS98, ALM+98], or a PCP for **NEXP** of exponential length [BFL90]. Lemma 2.1 in [KPT97] shows that using the proof of [DFK+92] for Theorem 6.2 and some elegant combinatorial tricks one can compile Π' into a new PCP $\Pi = (\{\boldsymbol{\pi}_x\}, \mathsf{V})$ with the following properties (k and m are two given parameters).

1. The length of the new PCP is at most $|\pi_x| \leq \text{poly}(N, k, m)$.

2. V uses $O(\log N + \log k + \log m)$ many random bits, and asks $O(1)$ PCP queries.

3. Π has completeness 1 and soundness error at most $1 - 1/\Theta(k^3)$.

4. Suppose V^m is a verifier who runs m independent copies of V over a given PCP and accepts if all of them accept. Then the view of V^m, denoted as $\mathsf{View}\langle\boldsymbol{\pi}_x, \mathsf{V}^m\rangle$, can be simulated by an efficient straight-line simulator Sim up to statistical error $1/m^k$.

Note that running V^m (*i.e.* m independent copies of V) will decrease the soundness error from $1 - 1/O(k^3)$ to $2^{-\Theta(m/k^3)}$. Moreover, if one set the compiler's parameter $m = n^{\omega(1)}$ the new PCP becomes statistically zero-knowledge against any $\mathsf{V}^{\text{poly}(n)}$ verifiers. [KPT97] shows how to use a locking scheme to essentially enforce *any* malicious poly(n)-query verifier to gain its information from the PCP oracle similarly to $\mathsf{V}^{\text{poly}(n)}$ (see Construction B.2 below).

Unfortunately [KPT97] does not give a full description of the exact modifications one needs to do to the proof of the main result of [DFK+92] to get the compiler above. In particular, [KPT97] works with Boolean PCPs (of binary alphabets), while the result of [DFK+92] uses $O(1)$ alphabet size. We first describe the result of [DFK+92] and then explain how it can be concluded there.

Theorem B.1 (Weakly-Sound Bounded-Query ZK-PCP for **NP** —Implicit in [DFK+92])**.** *For every language $L \in \mathbf{NP}$, there is a PCP $\Pi = (P = \{\boldsymbol{\pi}_x\}, \mathsf{V})$ for L such that:*

- *The PCP length is $|\pi_x| \leq \text{poly}(|x|, k)$ for every $\pi_x \leftarrow \boldsymbol{\pi}_x$, and its alphabet size is $O(1)$.*
- *The honest verifier V asks only two PCP queries.*
- *Completeness one: If $x \in L$, then $\Pr_{\pi \leftarrow \boldsymbol{\pi}_x}[\mathsf{V}^{\pi}(x) = 1] = 1$.*
- *Weak Soundness: For every $x \notin L$, and every oracle $\widehat{\pi}$ it holds that*

$$\Pr[\mathsf{V}^{\widehat{\pi}}(x) = 1] < 1 - O(1/k).$$

- *k-Query Zero-Knowledge: There is an efficient simulator Sim which, up to a negligible statistical distance, simulates the view of every malicious verifier $\widehat{\mathsf{V}}$ who (perhaps adaptively) asks at most k PCP queries. Moreover, this simulator is straight-line and does not need to know the randomness of V in order to simulate its view.*

In what follows we point out the minor modifications required for the proof of [DFK+92] to get Theorem B.1 above and (use its proof to get) the compiler of [KPT97]: In the proof of Theorem 6.2 the authors of [DFK+92] start from any language $L \in \mathbf{NP}$ and an input X and apply the PCP construction of [ALM+98, AS98] to get a 3-SAT instance X_{SAT} such that **(1)** if $X \in L$, then all the clauses of X_{SAT} are satisfiable, and **(2)** if $X \notin L$, then a constant fraction of the clauses of X_{SAT} are false, no matter how the Boolean variables $(x_1, \ldots, x_n)$ in X_{SAT} are set. At the beginning of their proof, [DFK+92] substitutes any Boolean variable x_i of X_{SAT} with the exclusive-or of three random Boolean variables conditioned on $x_i = x_i^1 \oplus x_i^2 \oplus x_i^3$. By this transformation, if one asks the value of any two variables (from the larger set of new variables), they gain no information about the assignment of $(x_1, \ldots, x_n)$ because the returned answers are totally random. All one has to do to make the compiler of [KPT97] work, is to add more dummy variables as follows. Recall that k is one of the parameters of the compiler. For each Boolean variable x_i, we would substitute it with the exclusive-or of k new random Boolean variables conditioned on $x_i = x_i^1 \oplus \ldots \oplus x_i^k$. This minor modification allows [KPT97] to use the proof of [DFK+92] to conclude the existence of their compiler. (The other parameter of the compiler, m, comes from the combinatorial arguments of [KPT97] itself.)

Finally, for sake of completeness we also describe the construction of [KPT97] in which they use locking schemes to convert the compiled PCP above into variants of ZK-PCPs (*i.e.* a bounded-query PCP for **NP** and an inefficient PCP for **NEXP**).

Construction B.2 ([KPT97])**.** Let $\Pi = (\{\boldsymbol{\pi}_x\}, \mathsf{V})$ be a PCP of length N where the verifier V tosses t random bits and asks q PCP queries. Also let $\Sigma = (\{\boldsymbol{\sigma}_x\}, R)$ be a locking scheme for the message space $\{0,1\}^t$ and input/output length (*i.e.* security parameter) $\ell(n)$ to be chosen later. In the new PCP $\Pi^{\mathsf{KPT}} = (\{\boldsymbol{\pi}_x^{\mathsf{KPT}}\}, \mathsf{V}^{\mathsf{KPT}})$ the PCP oracle $\boldsymbol{\pi}^{\mathsf{KPT}}$ first generates $\pi \leftarrow \boldsymbol{\pi}_x$ from scheme Π and then encodes it in three parts $(\mathsf{PCP}, \mathsf{PER}, \mathsf{MIX})$ described below:

1. PCP: Let $\pi = [\pi_1, \ldots, \pi_N]$. For each $i \in [N]$, generate a locking oracle and key $(\mathsf{Lock}_i, \mathsf{Key}_i)$ containing π_i and put $\mathsf{PCP}[i] = \mathsf{Lock}_i$ (as the i-th answer block in PCP).

2. PER: Generate a random permutation rand over $\{0,1\}^t$ and for each $r \in \{0,1\}^t$ generate a locking oracle and key $(\mathsf{Lock}_r, \mathsf{Key}_r)$ with the content $\mathsf{rand}(r)$ and put $\mathsf{PER}[r] = \mathsf{Lock}_r$.

3. MIX: If for the randomness $r \in \{0,1\}^t$, the verifier of Π chooses to read the blocks $\pi_{i_1}, \ldots, \pi_{i_q}$, then we put $\mathsf{MIX}[\mathsf{rand}(r)] = [r, \mathsf{Key}_r, \mathsf{Key}_{i_1}, \ldots, \mathsf{Key}_{i_q}]$.

Remark B.3 (Choosing $\ell(n)$)**.** Depending on whether the final goal is an efficient bounded-query ZK-PCP for $L \in \mathbf{NP}$ or that we want a ZK-PCP for some $L \in \mathbf{NEXP}$, different security parameters $\ell(n)$ for the locking scheme are employed in Construction B.2. (n was used as the security parameter of our locking scheme in Construction 5.2 should not be confused with the input length $n = |x|$ of the PCP construction here.) For the case of $L \in \mathbf{NP}$ one can use $\ell(n) = c \cdot \log(n)$ for a large enough constant c (which depends on the bounded-query parameter of the verifier), and for the case of $L \in \mathbf{NEXP}$ one can use $\ell(n) = n$.

In Section 4 of [KPT97] the authors show that if one starts from a PCP which is zero-knowledge only against m *random* executions of the verifier (this PCP comes as the result of their compiler described above), then Construction B.2 can convert it into a new PCP that is zero-knowledge against general verifiers (which are bounded-query ZK-PCPs in the case of $L \in \mathbf{NP}$ or inefficient PCPs in the case of $L \in \mathbf{NEXP}$). See Section 4 of [KPT97] for the details.