Zero-Knowledge Proofs of Proximity[*][†]

Itay Berman[1], Ron D. Rothblum[2], and Vinod Vaikuntanathan[3]

1 MIT, Cambridge MA, USA
itayberm@mit.edu

2 MIT, Cambridge MA, USA and Northeastern University, Boston MA, USA
ronr@mit.edu

3 MIT, Cambridge MA, USA
vinodv@mit.edu

Abstract

Interactive proofs of proximity (IPPs) are interactive proofs in which the verifier runs in time *sub-linear* in the input length. Since the verifier cannot even read the entire input, following the property testing literature, we only require that the verifier reject inputs that are *far* from the language (and, as usual, accept inputs that are in the language).

In this work, we initiate the study of *zero-knowledge proofs of proximity* (ZKPP). A ZKPP convinces a sub-linear time verifier that the input is *close* to the language (similarly to an IPP) while simultaneously guaranteeing a natural zero-knowledge property. Specifically, the verifier learns nothing beyond (1) the fact that the input is in the language, and (2) what it could additionally infer by reading a few bits of the input.

Our main focus is the setting of *statistical* zero-knowledge where we show that the following hold *unconditionally* (where N denotes the input length):

- Statistical ZKPPs can be sub-exponentially more efficient than property testers (or even *non-interactive* IPPs): We show a natural property which has a statistical ZKPP with a $\mathsf{polylog}(N)$ time verifier, but requires $\Omega(\sqrt{N})$ queries (and hence also runtime) for every property tester.
- Statistical ZKPPs can be sub-exponentially less efficient than IPPs: We show a property which has an IPP with a $\mathsf{polylog}(N)$ time verifier, but cannot have a statistical ZKPP with even an $N^{o(1)}$ time verifier.
- Statistical ZKPPs for some graph-based properties such as promise versions of expansion and bipartiteness, in the bounded degree graph model, with $\mathsf{polylog}(N)$ time verifiers exist.

Lastly, we also consider the computational setting where we show that:

- Assuming the existence of one-way functions, every language computable either in (logspace uniform) NC or in SC, has a *computational* ZKPP with a (roughly) $\sqrt{N}$ time verifier.
- Assuming the existence of collision-resistant hash functions, every language in NP has a *statistical* zero-knowledge *argument* of proximity with a $\mathsf{polylog}(N)$ time verifier.

1998 ACM Subject Classification F.1.2 Interactive and reactive computation, Probabilistic computation

Keywords and phrases Property Testing, Interactive Proofs, Zero-Knowledge

Digital Object Identifier 10.4230/LIPIcs.ITCS.2018.19

[*] The first and third author were supported in part by NSF Grants CNS-1350619 and CNS-1414119, Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship, the NEC Corporation, a Steven and Renee Finn Career Development Chair from MIT. This work was also sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.
The second author was partially supported by NSF MACS - CNS-1413920, DARPA IBM - W911NF-15-C-0236, a SIMONS Investigator award Agreement Dated 6-5-12 and by the Cybersecurity and Privacy Institute at Northeastern University

[†] A full version [10] of the paper is available at https://eprint.iacr.org/2017/114

9th Innovations in Theoretical Computer Science Conference (ITCS 2018).
Editor: Anna R. Karlin; Article No. 19; pp. 19:1–19:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Interactive proofs, introduced by Goldwasser, Micali and Rackoff [30] are protocols that allow a polynomial-time verifier to check the correctness of a computational statement, typically formulated as membership of an input x in a language $\mathcal{L}$, using an interactive protocol. Interactive proofs have had an incredible impact on theoretical computer science in general, and especially on cryptography and complexity theory.

Given the vast amounts of data that are available nowadays, and the ubiquity of cloud computing, in some applications polynomial-time or even *linear-time* verification may be too slow. A recent line of work, initiated by Rothblum, Vadhan and Wigderson [51] (following the earlier work of Ergün, Kumar and Rubinfeld [16]), asks whether we can construct interactive proofs in which the verifier runs in *sub-linear* time. Since the verifier cannot even read the entire input, we cannot hope to obtain sub-linear time verification in general (even for some very simple computations[1]). Thus, following the property testing literature [52, 21] (see also [20]), the verifier is given oracle access to the input, and soundness is relaxed. Namely, the verifier is only required to reject inputs that are *far* (in Hamming distance) from being in the language. Since the verifier is only assured that the input x is *close* to the language $\mathcal{L}$, these proof-systems are called *interactive proofs of proximity*, or IPPs for short. Recent results ([51, 34, 18, 41, 23, 50, 22, 35]) have demonstrated that many languages admit IPPs with sublinear-time verification.[2]

One of the main features of classical interactive proofs (over their non-interactive counterparts) is that they allow for proving statements in *zero-knowledge* [30, 24]: amazingly, it is possible to prove that $x \in \mathcal{L}$ without revealing anything other than that. Beyond being of intrinsic interest, zero-knowledge proofs have a multitude of applications, especially in cryptography.

In this work, we initiate the study of *zero-knowledge* proofs of proximity, or ZKPP for short. Specifically we ask:

> *Is it possible to prove the correctness of a computation to a verifier that reads only few bits of the input, without revealing any additional "non-local" information about the input?*

By non-local information, we mean any information that cannot be inferred by making only few queries to the input. In particular, and in contrast to the classical zero-knowledge setting, we want our notion of zero-knowledge to capture the fact that the verifier does not even learn the input string itself.

The Model of Zero-Knowledge Proofs of Proximity. As expected, we capture the desired zero-knowledge requirement using the simulation paradigm of [30].

▶ **Definition 1** (ZKPP, informally stated (see Section 2)). An IPP with prover $\mathbf{P}$ and verifier $\mathbf{V}$ is a ZKPP, if for any (possibly malicious) verifier $\widehat{\mathbf{V}}$, that given oracle access to input of length N runs in time $t(N) \ll N$, there exists a simulator $\mathbf{S}$ that runs in time roughly $t(N)$

[1] Consider for example verifying whether a given string has parity 0.

[2] Throughout this work we use the verification time as our primary complexity measure for IPPs. We could have alternatively chosen to view the total number of bits observed by the verifier (i.e., those read from the input and those communicated from the prover) as the main resource (note that the verification time is an upper bound on the latter). Focusing on verification time makes our upper bounds stronger, whereas our lower bounds also hold wrt the total number of bits observed by the verifier.

such that for every $x \in \mathcal{L}$ it holds that

$$\mathbf{S}^x \approx \left(\mathbf{P}(x), \widehat{\mathbf{V}}^x\right),$$

where $\left(\mathbf{P}(x), \widehat{\mathbf{V}}^x\right)$ denotes $\widehat{\mathbf{V}}$'s view when interacting with $\mathbf{P}$.

In particular, if the verifier cannot afford to read the entire input, then the simulator must successfully simulate the verifier's view even though it too cannot read the entire input. See Section 2 for the formal definition of the model and additional discussions.

Knowledge Tightness and Simulation Overhead. The above informal definition of zero-knowledge requires that for any possible cheating verifier that runs in (sublinear) time t, there exists a simulator, running in roughly the same time, that simulates the verifier's view. We call the running time of the simulator, viewed as a function of t, the *simulation overhead* of the protocol.[3]

In the zero-knowledge literature, the simulation overhead $s = s(t)$ is typically allowed to be any polynomially-bounded function. This is motivated by the fact that such polynomial-time simulation implies that every *polynomial-time* verifier strategy has a *polynomial-time* simulation.

In contrast, since in our setting of ZKPP the verifier runs in *sub-linear* time, we will sometimes need to be more precise. Suppose for example that we had a ZKPP with a $t = \sqrt{N}$ time verifier (where N is the input length) and with some unspecified polynomial simulation bound $s = s(t)$. In such a case, if for example $s(t) = \Omega(t^2)$, then the simulator would be able to read the *entire* input whereas the verifier clearly cannot. This leads to an undesirable gap between the power of the verifier and that of the simulator.

Thus, to obtain more meaningful results we will sometimes need to precisely specify the simulation overhead that is incurred. Nevertheless, since most (but not all) of our results deal with verifiers that run in poly-logarithmic time, unless we explicitly state otherwise, our default is to allow for *polynomial* simulation overhead. Indeed, in the poly-logarithmic regime, polynomial simulation implies that every poly-logarithmic time verifier strategy has a poly-logarithmic time simulation. In the few cases where we need to be more precise, the simulation overhead will be stated explicitly.

We remark that our quantification of the simulation overhead is closely related to Goldreich's [19, Section 4.4.4.2] notion of knowledge tightness of standard zero-knowledge proofs.

A Cryptographic Motivation from the 90's. Interestingly, the notion of ZKPP has already implicitly appeared in the cryptographic literature 20 years ago. Bellare and Yung [5] noticed that the soundness of the [17] construction of non-interactive zero-knowledge proof-system (NIZK) from trapdoor permutations breaks, if the cheating prover sends a description of a function that is not a permutation. [5] observed that to regain soundness in the [17] protocol, it suffices to verify that the given function is *close* to being a permutation.

Focusing on the case that the domain of the permutation[4] is $\{0,1\}^n$, [5] suggested the following natural non-interactive zero-knowledge proof for certifying that a function is *close*

[3] In our actual definition the simulation overhead may depend also on the input length (and proximity parameter). However, the more fundamental dependence is on the (possibly cheating) verifier's running time. Thus, we omit the dependence on these additional parameters from the current discussion.

[4] We remark that the general case (i.e., when the domain is not $\{0,1\}^n$) introduces significant difficulties. See [28] and [13] for details.

to a permutation: sufficiently many random elements $y_1, \ldots, y_k$ in $\{0,1\}^n$ are specified as part of a common random string[5] (CRS), and the prover is required to provide inverses $x_1, \ldots, x_k$ to all of these elements. Soundness follows from the fact that if the function is far from a permutation then, with high probability, one of these elements will simply not have an inverse. Zero-knowledge is demonstrated by having the simulator sample the x's at random and obtain the y's by evaluating the permutation.

Since the verifier in the [5] protocol is only assured that the function is close to a permutation, in our terminology, the [5] protocol is a non-interactive ZKPP. Notice that the verifier runs in time $\mathsf{poly}(n)$, which is *poly-logarithmic* in the input (i.e., the truth table of f).

1.1 Our Results

In this section we state our main results in an informal manner. See the full version [10] for the formal theorem statements.

As is the case for standard zero-knowledge, the results that we can obtain depend heavily on the specific notion of zero-knowledge. These notions depend on what exactly it means for the output of the simulator to be *indistinguishable* from a real interaction.

The main notion which we focus on in this work is that of *statistical* zero-knowledge proofs of proximity. Here, the requirement is that the distribution of the output of the simulator is statistically close[6] to that of the real interaction.

1.1.1 Statistical ZKPP

The first natural question to ask is whether this notion is meaningful – do there exist statistical ZKPPs?[7] More precisely, since every property tester is by itself a trivial ZKPP (in which the prover sends nothing), we ask whether statistical ZKPPs can outperform property testers.

We answer this question affirmatively. Moreover, we show that same natural problem considered by [5] (i.e., verifying that a function is a permutation) has a very efficient zero-knowledge proof of proximity. We emphasize that, in contrast to the protocol of [5] mentioned above, our protocol is zero-knowledge against arbitrary malicious verifiers (rather than only *honest-verifier* zero-knowledge as in the [5] protocol).

▶ **Theorem 2** (ZKPP for permutations). *Let* PERMUTATION$_n$ *be the set of all permutations on n-bit strings. Then:*

- **ZKPP Upper Bound:** PERMUTATION$_n$ *has a 4-round statistical* ZKPP *in which the verifier runs in* $\mathsf{poly}(n)$ *time.*
- **Property Testing Lower Bound:** *Every tester for* PERMUTATION$_n$ *must make at least $\Omega(2^{n/2})$ queries to the input (and in particular must run in time $\Omega\left(2^{n/2}\right)$).*

[5] Recall that NIZKs inherently require the use of a CRS.

[6] That is, the two distributions have negligible statistical distance. Negligible here refers to an auxiliary security parameter that is given to all parties, see further discussion in Section 2.2.1.

[7] Note that not every IPP is zero-knowledge. Suppose that we want to check whether a given input consists of two consecutive palindromes (of possibly different lengths) or is far from such. Alon *et al.* [2] showed that every tester for this property must make $\Omega(\sqrt{N})$ queries. However, Fischer *et al.* [18] observed that if the prover provides the index that separates the two palindromes, the property becomes easy to verify. The IPP of [18] is not zero-knowledge since any $o(\sqrt{N})$ time simulator can be transformed into an $o(\sqrt{N})$ time tester for the property, contradicting the [2] lower bound.

(Notice that $\mathsf{poly}(n)$ is poly-logarithmic in the input size, whereas $2^{n/2}$ is roughly the square root of the input size.)

Similarly to other results in the literature on constant-round statistical zero-knowledge (SZK), we can only bound the *expected* running time of our simulator (rather than giving a strict bound that holds with all but negligible probability). Using standard techniques, which introduce a super constant number of rounds, we can obtain a strict bound on the simulator's running time. However, in the interest of simplicity and since it is not our main focus, we avoid doing so. We also remark that Gur and Rothblum [33] give a lower bound on the complexity of *non-interactive* IPPs (i.e., IPP in which the entire interaction consists of a single message from the prover to the verifier, also known as MAPs) for PERMUTATION, and combining their result with ours yields a sub-exponential separation between the power of statistical ZKPP vs. MAPs. (Specifically, [33] show an MAP lower bound of roughly $\Omega(2^{n/4})$ for PERMUTATION.) Lastly, we mention that a variant of the permutation property was used by Aaronson [1] to give an oracle separation of SZK from QMA. However, the SZK protocol that he constructs (which is essentially the [5] protocol) is only *honest-verifier*[8] zero-knowledge.

Beyond the property of being a permutation, we also consider two additional *graph* problems, and show that they admit efficient *honest-verifier* ZKPP protocols. Both problems we consider are in the bounded degree graph model[9], which has been widely studied in the property testing literature [21, 26].

▶ **Theorem 3** (Honest Verifier ZKPP for Expansion and Bipartiteness). *There exist* honest-verifier *statistical* ZKPP *in which the verifier's running time is* $\mathsf{polylog}(N)$*, for input graphs of size N, for the following two promise problems:*

1. **Promise Expansion:** *Distinguish graphs with (vertex) expansion $\alpha \in (0,1]$ from graphs that are far from even having expansion roughly $\beta = \alpha^2/\log(N)$.*
2. **Promise Bipartiteness:** *Distinguish bipartite graphs from graphs that are both rapidly mixing and far from being bipartite.*

A few remarks are in order. We first note that the property testing complexity of both promise problems is known to be $\tilde{\Theta}(\sqrt{N})$ [26, 27, 15, 47, 42]. Second, the IPP for promise-bipartiteness is actually due to [51] and we merely point out that it is an honest-verifier ZKPP. In contrast, the promise-expansion property above was not previously known to admit an (efficient) IPP (let alone a zero-knowledge one). We also remark that both of the problems in Theorem 3 refer to *promise problems*. In particular, we leave open the possibility of a ZKPP for bipartiteness that also handles graphs that are not rapidly mixing, and a ZKPP for expansion that accepts graphs that are α-expanding and rejects graphs that are far from α-expanding (rather than just rejecting those that are far from being $\alpha^2/\log(N)$-expanding as in Theorem 3). Lastly, we also leave open the possibility of extending these protocols to be statistical ZKPP against arbitrary cheating verifiers (rather than just honest verifiers).[10]

[8] In an *honest-verifier* ZKPP, the simulator needs only to output an interaction that is indistinguishable from the interaction of the honest (original) verifier and the prover.

[9] In the bounded degree graph model we assume that the degree of all vertices is bounded by a parameter d and the input graph is represented by an adjacency list. In other words, one can request to see the i-th neighbor (for $i \in [d]$) of some vertex v using a single query.

[10] Since honest-verifier SZK protocols can be converted to be zero-knowledge against arbitrary malicious verifiers ([29], see also [53]), it is reasonable to wonder whether the same holds for statistical ZKPP. We conjecture that this is the case but leave the question of verifying this conjecture to future work.

Limitations of Statistical ZKPP. Given these feasibility results, one may wonder whether it is possible to obtain statistical ZKPP with poly-logarithmic complexity for large complexity classes (e.g., for any language in P), rather than just specific problems as in Theorems 2 and 3. The answer turns out to be negative since Kalai and Rothblum [41] constructed a language, computable in NC_1, for which every IPP (let alone a zero-knowledge one) requires $\Omega(\sqrt{N})$ verification time.[11]

Still, the latter observation raises the question of whether statistical ZKPP are as powerful as IPPs. That is, can every IPP be converted to be statistically zero-knowledge with small overhead? We show that this is not the case:

▶ **Theorem 4** (IPP $\not\subseteq$ SZKPP). *There exists a property* Π *that has an* IPP *in which the verifier runs in* $\mathsf{polylog}(N)$ *time, where* N *is the input length, but* Π *does not have a statistical* ZKPP *in which the verifier runs even in time* $N^{o(1)}$.

We emphasize that Theorem 4 is unconditional (i.e., it does not rely on any unproven assumptions as is typically the case when establishing lower bounds in the classical setting). Interestingly, if we do allow for a (reasonable) assumption, we can obtain a stronger separation: namely, of MAP from SZKPP:

▶ **Theorem 5** (MAP $\not\subseteq$ SZKPP). *Assuming suitable circuit lower bounds, there exists a property* Π *that has an* MAP *in which the verifier runs in* $\mathsf{polylog}(N)$ *time, where* N *is the input length, but* Π *does not have a statistical* ZKPP *in which the verifier runs even in time* $N^{o(1)}$.

The circuit lower bound that we assume follows from the plausible assumption that the Arthur-Merlin communication complexity of the set disjointness problem is $\Omega(n^{\varepsilon})$, where n is the input length and $\varepsilon > 0$ is some constant.

1.1.2 The Computational Setting

Unsurprisingly, we can obtain much stronger results if we relax some of our requirements to only be *computational* (rather than statistical). Specifically we will consider the following two relaxations:

1. (Computational Zero-Knowledge:) the simulated view is only required to be *computationally* indistinguishable from the real interaction.
2. (Computational Soundness aka Argument-System:) Here, we only require soundness against *efficient* cheating provers.

The following results show that under either one of these relaxations, and assuming reasonable cryptographic assumptions, we can transform many of the known results from the literature of IPPs to be zero-knowledge. Focusing on computational zero-knowledge, we can derive such protocols for any language computable in bounded-depth or in bounded-space, where the verifier runs in roughly $\sqrt{N}$ time.

▶ **Theorem 6** (Computational ZKPP for Bounded Depth). *Assume that there exist one-way functions. Then, every language in logspace-uniform* NC*, has a computational* ZKPP*, where the verifier (and the simulator) run in time* $N^{\frac{1}{2}+o(1)}$ *and the number of rounds is* $\mathsf{polylog}(N)$. *The simulation overhead is roughly linear.*

[11] This still leaves open the possibility that statistical ZKPPs with $O(\sqrt{N})$ complexity exist for large complexity classes. Actually, in the computational setting we show such results, see further discussion in Section 1.1.2.

▶ **Theorem 7** (Computational ZKPP for Bounded Space). *Assume that there exist one-way functions. Then, every language computable in* $\mathsf{poly}(N)$*-time and* $O(N^{\sigma})$*-space, for some sufficiently small constant* $\sigma > 0$*, has a computational* ZKPP*, where the verifier (and the simulator) run in time* $N^{\frac{1}{2}+O(\sigma)}$*. The simulation overhead is roughly linear.*

Note that in both results the simulation overhead is (roughly) linear which means that a verifier running in time t will be simulated in nearly the same time. See additional discussion on the notion of simulation overhead above.

Interestingly, if we only relax to *computational soundness*, we can do even better both in terms of expressive power and the running time of the verifier. The following result gives statistical zero-knowledge arguments of proximity for every language in NP, and with a verifier that runs in only *poly-logarithmic* time.

▶ **Theorem 8** (Statistical Zero-Knowledge Arguments for NP). *Assume that there exist collision-resistant hash functions. Then, every language in* NP*, has a constant-round* statistical *zero-knowledge argument of proximity, where the verifier runs in time* $\mathsf{polylog}(N)$*.*

(Here, since the verifier runs in *poly-logarithmic* time, we can and we do allow for polynomial simulation overhead.)

1.2 Related Works

In this section we discuss some related notions (and results) that have previously appeared in the literature, and how they compare with our results.

Zero-Knowledge PCPs. Zero-knowledge PCPs, introduced by Kilian, Petrank and Tardos [44] (and further studied in [32, 38]), are similar to standard PCPs with an additional zero-knowledge requirement. Namely, the oracle access that the (potentially malicious) verifier has to the PCP should not reveal anything beyond the fact that the input is in the language. Note that the verifier in a zero-knowledge PCP is given full access to the input and oracle access to the proof. In contrast, in zero-knowledge proofs of proximity (studied in this paper) the situation is reversed: the verifier is given oracle access to the input but full access to the communication line with the prover.

A more closely related notion of *zero-knowledge* PCP*s of proximity* was considered by Ishai and Weiss [37]. These are PCP systems in which the verifier gets oracle access to *both* the input and to an alleged PCP-style proof. Similarly to our notion of ZKPP, the verifier runs in sublinear time and is assured (with high probability) that the input is close to the language. ZKPPs and zero-knowledge PCPPs are incomparable — soundness is harder to achieve in the interactive case (since the prover's answers may be adaptive) whereas zero-knowledge is harder to obtain in the PCP setting. Therefore, the difference between our model and that of [37] is that we consider *interactice* proofs, whereas [37] focus on PCP-style proofs: namely soundness is guaranteed only if the PCP proof string is written in advance.

Zero-Knowledge Communication Complexity. A model of zero-knowledge in *communication complexity* was recently proposed by Göös, Pitassi and Watson [31] and further studied by Applebaum and Raykov [3]. Since there are known connections between property testing and communication complexity [11] (which holds also in the interactive setting, see [34]), it is interesting to study whether such a connection can be fruitful also in the zero-knowledge setting. We leave the study of this possibility to future work.

Zero-Knowledge Interactive PCPs and Oracle Proofs. Recent works by Ben-Sasson *et al.* [7, 8] study zero-knowledge interactive oracle proofs – a model in which the verifier receives *oracle* access to the communication tape, but full access to the input.[12] Our model of ZKPP is reversed – the verifier has oracle access to the input but full access to the communication tape. Chiesa *et al.* [14] consider zero-knowledge in the context of interactive PCPs, a model introduced by Kalai and Raz [40].

Measures of Knowledge. The notion of "simulation overhead", similarly to that of "knowledge tightness" [19] mentioned above, can be viewed as a (quantitative) security measure for the zero-knowledge of a protocol. Both notions are *worst-case* and consider the verifier and simulator's running times. Micali and Pass [45] considered a similar measure, but in an *execution-to-execution* setting. Finally, Goldreich and Petrank [25] considered other, incomparable, security measures than the verifier's and simulator's running times.

1.3 Technical Overview

We provide overview for our main conceptual results. Overviews for our other results are given in the appropriate places in the body of the paper.

1.3.1 ZKPP for PERMUTATION (see Theorem 2)

Since it is easier to argue, we begin with showing that any *property tester* for PERMUTATION must make at least $\Omega(\sqrt{N})$ queries, where $N = 2^n$. To see this, consider the following two distributions: (1) a random permutation over $\{0,1\}^n$; and (2) a random function from $\{0,1\}^n$ to $\{0,1\}^n$. The first distribution is supported exclusively on YES instances whereas it can be shown that the second is, with high probability, far from a permutation. However, if a tester makes $q \ll \sqrt{N}$ queries, then in both cases, with high probability, its view will be the same: q distinct random elements. The property testing lower bound follows.

We now turn to show a statistical ZKPP in which the verifier runs in $\mathsf{poly}(n)$ time. Consider the following simple IPP for PERMUTATION (based on the [5] protocol). Given oracle access to a function $f : \{0,1\}^n \to \{0,1\}^n$, the verifier chooses a random $r \in \{0,1\}^n$ and sends r to the prover. The prover computes $z = f^{-1}(r)$ and sends it to the verifier. The verifier checks that indeed $f(z) = r$ and if so accepts.

Clearly if f is a permutation then the verifier in this protocol accepts with probability 1, whereas if f is *far* from a permutation, then with some non-negligible probability the verifier chooses r which does not have a pre-image under f. In such a case the prover cannot make the verifier accept and so the protocol is sound.

It is also not hard to see that this protocol is *honest-verifier* zero-knowledge.[13] However, it is not *cheating-verifier* zero-knowledge: a cheating verifier could learn the inverse of some arbitrary r of its choice.

In order to make the protocol zero-knowledge, intuitively, we would like to have a way for the prover and verifier to jointly sample the element r such that both are assured that it is uniform. For simplicity let us focus on the task of just sampling a single bit σ. The specific properties that we need are

[12] Interactive proofs in which the verifier is not charged for reading the entire communication tape are called either *probabilistically checkable interactive proofs* [50] or *interactive oracle proofs* [9] in the literature.

[13] As a matter of fact, this protocol can be viewed as a non-interactive statistical zero-knowledge protocol for PERMUTATION (and is used as such in [5]).

1. If f is a permutation then the *prover* is assured that σ is random.
2. If f is far from being a permutation then the *verifier* is assured that σ is random.

In fact, the transformation of general honest-verifier statistical zero-knowledge proofs to cheating-verifier ones (see [53, Chapter 6]) implements a sub-routine achieving a generalization of the above task, assuming *full* access to the input. We give a simple solution for our specific case. That is, using only oracle access to a function that is either a permutation or far from any permutation.

We proceed to describe a simple procedure for sampling such a random bit σ. First, the verifier chooses at random $x \in \{0,1\}^n$ and a pairwise independent hash function $h : \{0,1\}^n \to \{0,1\}$ and sends $y = f(x)$ and h to the prover. The prover now chooses a random bit $r \in \{0,1\}$ and sends r to the verifier. The verifier now sends x to the prover who checks that indeed $f(x) = y$. The random bit that they agree on is $\sigma = r \oplus h(x)$.

From the prover's perspective, if f is a permutation then y fully determines x and so r (which is chosen uniformly at random after y is specified) is independent of $h(x)$. Hence, $\sigma = r \oplus h(x)$ is a uniformly random bit. On the other hand, from the verifier's perspective, if f is far from being a permutation, then, intuitively, even conditioned on the value y there still remains some entropy in x (indeed, x is essentially uniform among all the pre-images of y).[14] Now, using a variant of the leftover hash lemma, we can argue that $h(x)$ is close to random. Actually, since the leftover hash lemma implies that pairwise independent hash functions are *strong* extractors, we have that $h(x)$ is close to random even conditioned on h and therefore also conditioned on r (which is a randomized function of h). Thus, we obtain that $\sigma = r \oplus h(x)$ is close to being uniformly random and so our procedure satisfies the desired properties.

A Different Perspective: Instance-Dependent Commitments. *Instance-dependent commitments* [4, 39] are commitment schemes that depend on a specific instance of some underlying language: if the instance is in the language, the commitment is guaranteed to be statistically binding; and if the instance is not in the language the commitment is guaranteed to be statistically hiding. Instance-dependent commitments are a central tool in the study of SZK (e.g., [49, 48, 46]).

We can use PERMUTATION to construct an instance-dependent commitment as follows. Given a function $f : \{0,1\}^n \to \{0,1\}^n$, a commitment to a bit b is a tuple $(f(x), h, h(x) \oplus b)$, for a random $x \in \{0,1\}^n$ and a pairwise independent hash function $h\colon \{0,1\}^n \to \{0,1\}$. Our arguments can be adapted to show that if f is a permutation, then this commitment is statistically binding, whereas if f is *far* from a permutation, then this commitment is (weakly) statistically hiding (to amplify, we can repeat by choosing many x's).

One way to view our protocol for sampling the random string r that was described above, is as an instantiation of Blum's coin flipping protocol [12] based on the foregoing instance-dependent commitment.[15]

[14] Actually, the amount of entropy can be fairly small (and depends on how far f is from being a permutation). To obtain a sufficient amount of entropy, in our actual protocol we generate many such y's.

[15] Recall that in Blum's coin-flipping protocol, one party sends a commitment to a random bit b and the other party replies with another random bit b'. Now, the first party decommits and the parties agree on the bit $b \oplus b'$.

1.3.2 Separating IPP from SZKPP (see Theorem 4)

The proof of Theorem 4 is done in two steps. The first step is to construct a property Π which has an interactive proof of proximity with a large number of rounds and $\mathsf{polylog}(N)$-time verifier, but such that in every 2-*message* interactive proof of proximity for Π, the verifier's running time must be N^{δ}, for some constant $\delta > 0$. Actually, such a result was recently established by Gur and Rothblum [35].

The second step in proving Theorem 4 is a general round reduction transformation for any honest-verifier statistical zero-knowledge proof of proximity. Namely, we would like a procedure that takes any *many-messages* honest-verifier zero-knowledge proof of proximity and turns it into a 2-*message* honest-verifier zero-knowledge proof of proximity while only slightly deteriorating the verifier's and simulator's running times.

To establish such a procedure we apply the proof that the promise problem Entropy Difference (ED) is complete for the class SZK (see [53]). That proof takes an instance x of any promise problem $\Pi = (\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}}) \in \mathsf{SZK}$ and efficiently constructs two distributions X and Y such that if $x \in \Pi_{\mathsf{YES}}$ then $\mathrm{H}(X) \geq \mathrm{H}(Y) + 1$, and if $x \in \Pi_{\mathsf{NO}}$ then $\mathrm{H}(Y) \geq \mathrm{H}(X) + 1$. That proof goes on to show a zero-knowledge protocol to distinguish between the case that $\mathrm{H}(X) \geq \mathrm{H}(Y) + 1$ and the case that $\mathrm{H}(Y) \geq \mathrm{H}(X) + 1$. Two important points regarding that proof: (1) sampling from X and Y can be done by running (many times) the simulator for the original problem Π; (2) the protocol for ED consists of only two messages and requires only sample access to X and Y.

In our settings, we can view a property Π as a promise problem where functions possessing the property are in Π_{YES} and functions that are ε-far from possessing the property are in Π_{NO}. Then, we can have the verifier "run" the reduction to ED and apply the sample-access protocol for ED. The unbounded prover will behave as in the protocol for ED. Recall that the original simulator (i.e., the one for the property's IPP) required only oracle access to the input function. Since sampling from the distributions only requires running the original simulator, the new verifier can implement this step with only oracle access to the input function and with only polynomial overhead to the running time of the original simulator.

1.3.3 The Computational Setting (see Theorem 6-8)

The proofs of Theorem 6, Theorem 7 and Theorem 8 rely on the same basic idea: compiling existing public-coin protocols from the literature (specifically those of [51, 50, 43]) that are not zero-knowledge to ones that are. This step is based on the idea, which originates in the work of Ben-Or *et al.* [6], of having the prover commit to its messages rather than sending them in the clear. This ability to commit is where we use the assumption that one-way functions exist.

The compiler, which can only be applied to *public-coin* protocols is as follow. At every round, rather than sending its next message in the clear, the prover merely commits to the message that it would have sent in the protocol. Since the protocol is public-coin, the verifier can continue the interaction even though it does not see the actual contents of the prover's messages. After all commitments have been sent, the verifier only needs to check that there exist suitable decommitments that would have made the underlying IPP verifier accept. Since the commitment hides the contents of the messages, it cannot do so by itself and we would like to use the prover. At this point, one could try to naively argue that the residual statement is an NP statement, and so we can invoke a general purpose zero-knowledge protocol for NP (e.g., the classical [24] protocol or the more efficient [36] protocol).

Herein arises the main difficulty with this approach. While the statement that the verifier needs to check at the end of the interaction does consist of an existential quantifier applied to a polynomial-time computable predicate, the latter predicate makes oracle access to the input

x and so we do not know how to express it as an NP statement. To resolve this difficulty, we restrict our attention to verifiers that make *prover-oblivious queries*; that is, the queries that the verifier makes do not depend on messages sent by the prover. Luckily enough, in the IPPs that we rely on the verifier's queries are indeed prover-oblivious.

Thus, our verifier can actually make its queries after seeing only the commitments and we can construct an NP statement that refers to the actual values that it reads from the input. At this point we can indeed invoke a general purpose zero-knowledge protocol for NP and conclude the proof.

Lastly, we remark that the specific flavor of soundness and zero-knowledge that we obtain depends on the commitment scheme we use and the soundness of the protocol to which we apply the transformation. Loosely speaking, instantiating the above approach with a computationally hiding and statistically binding commitment scheme yields a *computational* zero-knowledge proof of proximity, whereas a statistically hiding and computationally binding one yields a statistical zero-knowledge *argument* of proximity.

Organization. In this extended abstract we include only the formal definitions of the ZKPP model, given in Section 2. See the full version [10] for the formal theorem statements and proofs.

2 ZKPP – Model and Definitions

A ZKPP is an interactive proof for convincing a *sub-linear* time verifier that a given input is close to the language, in *zero-knowledge*. Loosely speaking, by zero-knowledge we mean that if the (N-bit) input is in the language, the view of any (potentially malicious) verifier that runs in time $t \ll N$ can be simulated by reading not much more than t bits from the input.

The only non-trivial step in formalizing this intuition is in quantifying what we mean by "not much more". In the classical setting of zero-knowledge interactive proofs, we merely require that the simulator run in polynomial-time, and so "not much more" is interpreted as polynomial overhead. A natural adaptation for the sub-linear setting would therefore be to require that the running time of the simulator be polynomially related to that of the verifier. However, in some settings this requirement is problematic – e.g., suppose that the verifier runs in time $t = O(\sqrt{N})$. Here, a simulator that runs in time t^2 (and in particular can read the entire input) would be far less meaningful than one running in, say, $t^{3/2}$ time.

Thus, as pointed out in the introduction, it will be important for us to quantify more precisely what is the overhead incurred by the simulator. We refer to this as the *simulation overhead*, which we think of as a function of the verifier's running time (see precise statement below). Thus, rather than merely saying that a protocol is a ZKPP, we will say that it is a ZKPP with simulator overhead s.

We proceed to the formal definitions. A property is an ensemble $\Pi = (\Pi_n, \mathcal{D}_n, \mathcal{R}_n)_{n\in\mathbb{N}}$, where Π_n is a set of functions from $\mathcal{D}_n$ to $\mathcal{R}_n$, for every $n \in \mathbb{N}$. In certain contexts, it will be more convenient for us to view Π_n as a set of strings of length $|\mathcal{D}_n|$ over the alphabet $\mathcal{R}_n$ (in the natural way). In such cases we will also sometimes refer to properties as languages. We denote by N the bit-length of the input, i.e., $N = |\mathcal{D}_n| \cdot \log_2(|\mathcal{R}_n|)$. In the technical sections we will often measure efficiency in terms of the parameter N but in our actual definition below we will allow a direct dependence on n, $|\mathcal{D}_n|$ and $|\mathcal{R}_n|$. This makes the definitions slightly more cumbersome but allows us to capture certain auxiliary parameters that arise in specific models, e.g., the dependence on the degree of the graph in the bounded degree graph model (for details, see Section 2.2.1).

Lastly, similarly to [53], we use a security parameter k to control the quality of our soundness and zero-knowledge guarantees rather than letting these depend on the input length (although our reasons for doing so are slightly different from those in [53], see Section 2.2.1) for additional details).

Section Organization. We begin by recalling the definition of IPPs in Section 2.1, then proceed to define statistical ZKPP in Section 2.2, and finally we discuss computational ZKPP in Section 2.3.

2.1 Interactive Proofs of Proximity (IPPs)

Our definition of IPP follows [51] with minor adaptations.

▶ **Definition 9** (interactive proofs of proximity (IPP)). An r-message interactive proof of proximity (IPP), with respect to proximity parameter $\varepsilon > 0$, (in short, ε-IPP) for the property $\Pi = (\Pi_n, \mathcal{D}_n, \mathcal{R}_n)_{n \in \mathbb{N}}$ is an interactive protocol $(\mathbf{P}, \mathbf{V})$ between a prover $\mathbf{P}$, which gets *free* access to an input $f \colon \mathcal{D}_n \to \mathcal{R}_n$ as well as to ε, n, $|\mathcal{D}_n|$, $|\mathcal{R}_n|$ and k, and a verifier $\mathbf{V}$, which gets *oracle* access to f as well as free access to ε, n, $|\mathcal{D}_n|$, $|\mathcal{R}_n|$ and k. The following conditions are satisfied at the end of the protocol for every $k \in \mathbb{N}$ and large enough $n \in \mathbb{N}$:

- ***Completeness***: If $f \in \Pi_n$, then, when $\mathbf{V}$ interacts with $\mathbf{P}$, with probability $1 - \mathsf{negl}(k)$ it accepts.
- ***Soundness***: If f is ε-far from Π_n, then for every prover strategy $\widehat{\mathbf{P}}$, when $\mathbf{V}$ interacts with $\widehat{\mathbf{P}}$, with probability $1 - \mathsf{negl}(k)$ it rejects.

For $t = t(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon)$, we denote by $\mathsf{IPP}[t]$ the class of properties possessing ε-IPP in which the verifier's running time is at most $O(t)$. Finally, for a class of functions $\mathcal{C}$, we denote by $\mathsf{IPP}[\mathcal{C}(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon)]$ the class of properties Π for which there exists $t \in \mathcal{C}$ such that $\Pi \in \mathsf{IPP}[t]$.

The probabilities that the verifier rejects in the completeness condition, and accepts in the soundness condition, are called the completeness error and soundness error, respectively. If the completeness error is zero, then we say that the IPP has perfect completeness. A public-coin IPP is an IPP in which every message from the verifier to the prover consists only of fresh random coin tosses and the verifier does toss coins beyond those sent in its messages.

An IPP is said to have query complexity $q = q(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon) \in \mathbb{N}$ if for every $n, k \in \mathbb{N}$, $\varepsilon > 0$, $f : \mathcal{D}_n \to \mathcal{R}_n$, and any prover strategy $\widehat{\mathbf{P}}$, the verifier $\mathbf{V}$ makes at most $q(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon)$ queries to f when interacting with $\widehat{\mathbf{P}}$. The IPP is said to have communication complexity $c = c(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon) \in \mathbb{N}$ if for every $n, k \in \mathbb{N}$, $\varepsilon > 0$, and $f : \mathcal{D}_n \to \mathcal{R}_n$ the communication between $\mathbf{V}$ and $\mathbf{P}$ consists of at most $c(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon)$ bits.

Our main (but not exclusive) focus in this work is on properties that have IPPs in which the verifier's running time (and thus also the communication and query complexities) is poly-logarithmic in the input size and polynomial in the security parameter k and in the reciprocal of the proximity parameter ε. That is, the class $\mathsf{IPP}[\mathsf{poly}(\log(N), k, 1/\varepsilon)]$.

An IPP that consists of a single message sent from the prover (Merlin) to the verifier (Arthur) is called Merlin-Arthur proof of proximity (MAP) [34]. We extend all the above notations to MAPs in the natural way.

2.2 Statistical ZKPPs

Before defining general ZKPPs, we first consider zero-knowledge with respect to *honest verifiers*. Following [53], we require the simulator to run in strict polynomial-time but allow it to indicate a failure with probability 1/2 (which can then be reduced by repetition). The requirement is that conditioned on not failing, the simulated view is statistically close to the actual execution.

Recall that we say that an algorithm A is useful if $\Pr[\mathsf{A}(x) = \perp] \leq 1/2$ for every input x, and use $\widetilde{\mathsf{A}}(x)$ to denote the output distribution of $\mathsf{A}(x)$, conditioning on $\mathsf{A}(x) \neq \perp$. We define the view of the verifier $\mathbf{V}$ on a common input x (given as standard input or by oracle access to either of the parties) by $\mathrm{view}_{\mathbf{P},\mathbf{V}}(x) \overset{def}{=} (m_1, m_2, \ldots, m_r; \rho)$, where $m_1, m_2, \ldots, m_r$ are the messages sent by the parties in a random execution of the protocol, and ρ contains of all the random coins $\mathbf{V}$ used during this execution.

▶ **Definition 10** (honest-verifier zero-knowledge proof of proximity (HVSZKPP, HVPZKPP)). Let $(\mathbf{P}, \mathbf{V})$ be an IPP for a property $\Pi = (\Pi_n, \mathcal{D}_n, \mathcal{R}_n)_{n\in\mathbb{N}}$. The protocol $(\mathbf{P}, \mathbf{V})$ is said to be honest-verifier statistical zero-knowledge with simulation overhead s, for some function $s \colon \mathbb{N}^5 \times (0, 1] \to \mathbb{N}$ if there exists a useful probabilistic algorithm $\mathbf{S}$, which (like $\mathbf{V}$) gets oracle access to $f \colon \mathcal{D}_n \to \mathcal{R}_n$ as well as free access to ε, n, $|\mathcal{D}_n|$, $|\mathcal{R}_n|$ and k, and whose running time is at most $O(s(t_{\mathbf{V}}, n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon))$, where $t_{\mathbf{V}}(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon)$ is $\mathbf{V}$'s running time, such that for every $k \in \mathbb{N}$, every large enough $n \in \mathbb{N}$ and $f \colon \mathcal{D}_n \to \mathcal{R}_n$, if $f \in \Pi_n$, it holds that:

$$\mathrm{SD}\left(\widetilde{\mathbf{S}}^f(\varepsilon, n, |\mathcal{D}_n|, |\mathcal{R}_n|, k), \mathrm{view}_{\mathbf{P},\mathbf{V}}(\varepsilon, n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, f)\right) \leq \mathsf{negl}(k).$$

If the $\mathsf{negl}(k)$ can be replaced with 0 in the above equation, $(\mathbf{P}, \mathbf{V})$ is said to be honest-verifier perfect zero-knowledge with simulation overhead s.

For $t = t(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon)$, $\mathsf{HVSZKPP}[t, s]$ (resp., $\mathsf{HVPZKPP}[t, s]$) denotes the class of properties possessing honest-verifier statistical (resp., perfect) zero-knowledge proof of proximity with simulation overhead s in which the verifier's running time is at most $O(t)$.

We say that the query complexity of a simulator $\mathbf{S}$ is $q' = q'(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon) \in \mathbb{N}$ if for every $n, k \in \mathbb{N}$, $\varepsilon > 0$, $f : \mathcal{D}_n \to \mathcal{R}_n$, $\mathbf{S}_n^f$ makes at most $q'(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon)$ queries to f.

A typical setting (that we will focus on) is when the verifier's running time is $\mathsf{poly}(\log(N), k, 1/\varepsilon)$, namely poly-logarithmic in the input length N and polynomial in the security parameter k and in the proximity parameter $1/\varepsilon$. In this setting we often allow for *polynomial* simulation overhead, that is the simulator's running time is also $\mathsf{poly}(\log(N), k, 1/\varepsilon)$. Specifically, we denote by $\mathsf{HVSZKPP}\big[\mathsf{poly}(\log(N), k, 1/\varepsilon)\big]$ the class of properties $\Pi \in \mathsf{HVSZKPP}[t, s]$ for $t = \mathsf{poly}(\log(N), k, 1/\varepsilon)$ and $s = \mathsf{poly}(t, \log(N), k, 1/\varepsilon)$. The class $\mathsf{HVPZKPP}\big[\mathsf{poly}(\log(N), k, 1/\varepsilon)\big]$ is similarly defined.

Another setting of interest is when the verifier's running time is $N^\delta \cdot \mathsf{poly}(k, 1/\varepsilon)$, for some constant $\delta \in (0, 1)$. In this setting, unlike the previous one, allowing the simulation overhead to be polynomial will give the simulator much greater computational power than the verifier (e.g., if $\delta = 1/2$ and s is quadratic in the verifier's running time, then the simulator can run in time $O(N)$ and in particular may read the entire input). In this setting we aim for the simulation overhead to be *linear* in the verifier's running time (but it can be polynomial in k and $1/\varepsilon$).[16]

[16] This requirement is in the spirit of *constant* knowledge tightness, see [19, Section 4.4.4.2].

When the simulation overhead is clear from context we allow ourselves to say that the protocol is a ZKPP (rather than a ZKPP with specific simulation overhead as per Definition 10).

Cheating Verifier ZKPP. We will allow cheating verifiers to be non-uniform by giving them an auxiliary input. For an algorithm A and a string $z \in \{0,1\}^*$ (all auxiliary inputs will be binary strings, regardless of the properties' alphabet), let $\mathsf{A}_{[z]}$ be A when z was given as auxiliary input. Since we care about algorithms whose running time is insufficient to read the entire input, we would not want to allow the running time to depend on the auxiliary input (otherwise, we could artificially inflate z so that A would be able to read the entire input). Thus, following [53], we adopt the convention that the running time of A is independent of z, so if z is too long, A will not be able to access it in its entirety.

► **Definition 11** (cheating-verifier zero-knowledge proof of proximity (SZKPP, PZKPP))**.** Let $(\mathbf{P}, \mathbf{V})$ be an interactive proof of proximity for a property $\Pi = (\Pi_n, \mathcal{D}_n, \mathcal{R}_n)_{n\in\mathbb{N}}$. $(\mathbf{P}, \mathbf{V})$ is said to be cheating-verifier statistical zero-knowledge with simulation overhead s, for some function $s\colon \mathbb{N}^5 \times (0,1] \to \mathbb{N}$, if for every algorithm $\widehat{\mathbf{V}}$ whose running time is $O(t_{\widehat{\mathbf{V}}}(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon))$, there exists a useful probabilistic algorithm $\mathbf{S}$, which (like $\widehat{\mathbf{V}}$) gets oracle access to $f\colon \mathcal{D}_n \to \mathcal{R}_n$ as well as free access to ε, n, $|\mathcal{D}_n|$, $|\mathcal{R}_n|$ and k, and whose running time is at most $O(s(t_{\widehat{\mathbf{V}}}, n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon))$, such that for every $k \in \mathbb{N}$, large enough $n \in \mathbb{N}$, $z \in \{0,1\}^*$ and $f\colon \mathcal{D}_n \to \mathcal{R}_n$, if $f \in \Pi_n$, then

$$\mathrm{SD}\left(\widetilde{\mathbf{S}}^{f}_{[z]}(\varepsilon, n, |\mathcal{D}_n|, |\mathcal{R}_n|, k), \mathrm{view}_{\mathbf{P},\widehat{\mathbf{V}}_{[z]}}(\varepsilon, n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, f)\right) \le \mathsf{negl}(k).$$

If the $\mathsf{negl}(k)$ can be replaced with 0 in the above equation, $(\mathbf{P}, \mathbf{V})$ is said to be a cheating-verifier perfect zero-knowledge with simulation overhead s.

For $t = t(n, |\mathcal{D}_n|, |\mathcal{R}_n|, k, \varepsilon)$, $\mathsf{SZKPP}[t,s]$ (resp., $\mathsf{PZKPP}[t,s]$) denotes the class of properties possessing cheating-verifier statistical (resp., perfect) zero-knowledge proof of proximity with simulation overhead s in which the verifier's running time is at most $O(t)$.

Expected Simulation Overhead. Definition 11 requires that the running time of the simulator always be bounded. Similarly to many results in the ZK literature, in some cases we can only bound the simulator's *expected* running time. The following definition captures this (weaker) notion:

► **Definition 12** (cheating-verifier ZKPP with expected simulation (ESZKPP, EPZKPP))**.** Let $(\mathbf{P}, \mathbf{V})$ be an interactive proof of proximity for a property $\Pi = (\Pi_n, \mathcal{D}_n, \mathcal{R}_n)_{n\in\mathbb{N}}$. The protocol $(\mathbf{P}, \mathbf{V})$ is said to be cheating-verifier statistical zero-knowledge with expected simulation overhead s if it satisfies the same requirement as in Definition 11 except that we only bound the *expected* running time of the simulator (where the expectation is over the coins of the simulator).

The classes $\mathsf{ESZKPP}[t,s]$ and $\mathsf{EPZKPP}[t,s]$ are defined analogous to $\mathsf{SZKPP}[t,s]$ and $\mathsf{PZKPP}[t,s]$ from Definition 11.

Unless explicitly saying otherwise, all zero-knowledge protocols we discuss are cheating-verifier ones.

As in the honest-verifier case, a typical setting is that in which the verifier's running time is poly-logarithmic in the input size N and polynomial in the security parameter k and in $1/\varepsilon$, and the simulator's (possibly only expected and not strict) running time is polynomial in the running time of the cheating-verifier that it simulates, poly-logarithmic in N and polynomial in k and $1/\varepsilon$. Specifically, if we allow the cheating-verifier the same computational powers as the honest-verifier, then both the honest-verifier and every simulator run in time $\mathsf{poly}(\log(N), k, 1/\varepsilon)$. We let $\mathsf{ESZKPP}\big[\mathsf{poly}(\log(N), k, 1/\varepsilon)\big]$ be the class of properties $\Pi \in \mathsf{ESZKPP}[t, s]$ for $t = \mathsf{poly}(\log(N), k, 1/\varepsilon)$ and $s = \mathsf{poly}(t_{\widehat{\mathsf{V}}}, \log(N), k, 1/\varepsilon)$. The class $\mathsf{EPZKPP}\big[\mathsf{poly}(\log(N), k, 1/\varepsilon), \mathsf{poly}\big]$ is similarly defined.

2.2.1 Additional Discussions

We conclude Section 2.2 with a few remarks on statistical ZKPP.

▶ Remark (Proximity Promise Problems). Some of the protocols that we construct do not refer to a property but rather to a "proximity promise problem". Recall that a promise problem considers a pair of disjoint sets Π_{YES} and Π_{NO} and the goal is to distinguish input that are in Π_{YES} from those that are in Π_{NO} (and no requirement is given for inputs outside of $\Pi_{\mathsf{YES}} \cup \Pi_{\mathsf{NO}}$).

For some of our results we will consider *proximity* promise problems, which are also characterized by sets Π_{YES} and a family of sets $\left(\Pi_{\mathsf{NO}}^{(\varepsilon)}\right)_{\varepsilon \in (0,1)}$, and we require that for every $\varepsilon \in (0,1)$, it holds that $\Pi_{\mathsf{NO}}^{(\varepsilon)}$ is ε-*far* from Π_{YES} (rather than merely being disjoint). We extend the definitions above to handle proximity promise problems in the natural way (specifically, completeness and zero-knowledge should only hold for input in Π_{YES} whereas the soundness requirement is that if the verifier is given proximity parameter $\varepsilon > 0$ and an input in $\Pi_{\mathsf{NO}}^{(\varepsilon)}$, then it should reject with high probability).

▶ Remark (The Security Parameter). One of the original motivations for the introduction of a security parameter in the classical definitions of statistical zero-knowledge proofs was to control the error parameters (completeness, soundness and simulation deviation) independently from the input's length. Specifically, one may want to provide a high-quality proof (i.e., very small errors) for short inputs (see [53, Section 2.4]).

In our setting, the situation is somewhat reversed. We think of very large inputs that the verifier and simulator cannot even entirely read. Hence, it seems unreasonable to require errors that are negligible in the input length. Instead, we control the quality of the proof with the security parameter, independent of the input length.

▶ Remark (A Definitional Convention). Traditionally [21], a property tester gets an oracle access to a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and needs to determined if the function has the property or is ε-far from having this property (i.e., ε-far from any function that has the property). The tester gets n (or alternatively 2^n — the input length of the truth table of f) as a standard input and its complexity (e.g., running time, number of oracle queries) is measured as a function of n. As models and properties evolved (e.g., the bounded degree model [26]) Boolean functions no longer sufficed to (conveniently) describe properties. For example, in the bounded degree model graphs with n vertices and degree d are specified as a functions $G\colon [n] \times [d] \to [n] \cup \{\perp\}$ such that $G(u,i) = v$ if v is the i'th neighbor of a vertex u and $G(u,i) = \perp$ if u has less than i neighbors. Consequently, the parameter n alone no longer suffices to measure the complexity of the tester.

The situation becomes even more delicate when interaction is added. The model of *interactive proofs of proximity* (IPP), introduced by [51], considers an interaction between a prover and a verifier in which the prover is trying to convince the verifier that a function has a property. In the definition of [51], in addition to the function f, to which the verifier has only oracle access and is referred to as the *implicit* input, the verifier also has full access to an additional (shorter) input w, called the *explicit* input. For example, in the bounded degree model w might be simply d, and in "algebraic" properties w can contain a description of some underlying field. Roughly speaking, [51] chose to measure the complexity of the proof-system with respect to the length of the implicit input alone. This creates a slight inconvenience when trying to describe complexity measures. For example, in the bounded degree model, we would like the running time of the verifier to explicitly depend on the number of vertices and the degree d. However, as it is defined in [51], the function that bounds the verifier's running time gets only the input length (which has bit length $n \cdot d \cdot \log(n)$).

To avoid this minor issue, in this paper, we take a slightly different approach than that of [51] when defining IPPs. Our goal is to define a general model in which it is easy to compare properties from different domains (e.g., properties of bounded degree graphs and those considering algebra). To do so we no longer split the input to an implicit and explicit parts. We consider functions $f\colon \mathcal{D} \to \mathcal{R}$ from an arbitrary domain $\mathcal{D}$ to an arbitrary range $\mathcal{R}$. The verifier receives oracle access to f, and full access to $|\mathcal{D}|$ and $|\mathcal{R}|$. The prover receives full access the function f. Different complexity measures are now functions of the verifier's standard inputs — $|\mathcal{D}|$ and $|\mathcal{R}|$. Going back to the bounded degree graph example, we can see in this framework the function describing the verifier's running time gets $|\mathcal{D}_n| = n \cdot d$ and $|\mathcal{R}_n| = n$ as inputs, and can be easily "converted" into a function that simply gets n and d as inputs. Moreover, we can now define N to be the input length of the property (i.e., $N = |\mathcal{D}| \cdot \log(|\mathcal{R}|)$) and define complexity classes with respect to this input length. For example, we can define $\mathsf{IPP}[\mathsf{poly}(\log(N))]$ to be the class containing all properties with interactive proof of proximity in which the verifier's running time (as a function of $|\mathcal{D}|$ and $|\mathcal{R}|$) is bounded by $\mathsf{poly}(\log(N))$. Note that the above class does not depend on the domain and range of the propety, and properties of different "types" can still belong to $\mathsf{IPP}[\mathsf{poly}(\log(N))]$.

2.3 Computational ZKPP

Since our focus is on the statistical case, we do not provide explicit definitions of computational zero-knowledge proofs of proximity. Rather, these definitions can be easily extrapolated from the statistical ones in a standard way (see for example Vadhan's [53, Section 2] definition of computational zero-knowledge). Specifically, in the computational definitions one simply requires that the simulator's output and the protocol's view are computationally indistinguishable (rather than statistically close), with respect to the security parameter.

Acknowledgements. We thank Oded Goldreich for useful discussions and for his comments on an earlier version of this paper. We thank Omer Paneth for useful discussions and Prashant Vasudevan for referring us to [1].

References

1 Scott Aaronson. Impossibility of succinct quantum proofs for collision-freeness. *Quantum Information & Computation*, 12(1-2):21–28, 2012. URL: http://www.rintonpress.com/xxqic12/qic-12-12/0021-0028.pdf.

2 Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM J. Comput.*, 30(6):1842–1862, 2000. doi:10.1137/S0097539700366528.

3 Benny Applebaum and Pavel Raykov. From private simultaneous messages to zero-information arthur-merlin protocols and back. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel,*

January 10-13, 2016, Proceedings, Part II, volume 9563 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 2016. doi:10.1007/978-3-662-49099-0_3.

4 Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 482–493. ACM Press, 1990.

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

6 Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1988. doi:10.1007/0-387-34799-2_4.

7 Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. On probabilistic checking in perfect zero knowledge. *IACR Cryptology ePrint Archive*, 2016:988, 2016. URL: http://eprint.iacr.org/2016/988.

8 Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. Quasi-linear size zero knowledge from linear-algebraic pcps. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 33–64. Springer, 2016. doi:10.1007/978-3-662-49099-0_2.

9 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016. doi:10.1007/978-3-662-53644-5_2.

10 Itay Berman, Ron D. Rothblum, and Vinod Vaikuntanathan. Zero-knowledge proofs of proximity. *IACR Cryptology ePrint Archive*, 2017:114, 2017. URL: http://eprint.iacr.org/2017/114.

11 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012. doi:10.1007/s00037-012-0040-x.

12 Manuel Blum. Coin flipping by telephone. In *Advances in Cryptology – CRYPTO'81*, pages 11–15, 1981.

13 Ran Canetti and Amit Lichtenberg, 2017. Unpublished manuscript.

14 Alessandro Chiesa, Michael A. Forbes, and Nicholas Spooner. A zero knowledge sumcheck and its applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:57, 2017. URL: https://eccc.weizmann.ac.il/report/2017/057.

15 Artur Czumaj and Christian Sohler. Testing expansion in bounded-degree graphs. *Combinatorics, Probability & Computing*, 19(5-6):693–709, 2010. doi:10.1017/S096354831000012X.

16 Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Inf. Comput.*, 189(2):135–159, 2004. doi:10.1016/j.ic.2003.09.005.

17 Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *sicomp*, 1999. Preliminary version in *FOCS'90.*

18 Eldar Fischer, Yonatan Goldhirsh, and Oded Lachish. Partial tests, universal tests and decomposability. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 483–500. ACM, 2014. doi:10.1145/2554797.2554841.

19 Oded Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, 2001.

20 Oded Goldreich. *Introduction to Property Testing.* forthcoming (http://www.wisdom.weizmann.ac.il/~oded/pt-intro.html), 2016.

21 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

22 Oded Goldreich and Tom Gur. Universal locally testable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:42, 2016. URL: http://eccc.hpi-web.de/report/2016/042.

23 Oded Goldreich, Tom Gur, and Ron D. Rothblum. Proofs of proximity for context-free languages and read-once branching programs - (extended abstract). In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 666–677. Springer, 2015. doi:10.1007/978-3-662-47672-7_54.

24 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*, pages 691–729, 1991. Preliminary version in *FOCS'86*.

25 Oded Goldreich and Erez Petrank. Quantifying knowledge complexity. *Computational Complexity*, 8(1):50–98, 1999. doi:10.1007/s000370050019.

26 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.

27 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, volume 6650 of *Lecture Notes in Computer Science*, pages 68–75. Springer, 2011. doi:10.1007/978-3-642-22670-0_9.

28 Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *J. Cryptology*, 26(3):484–512, 2013. doi:10.1007/s00145-012-9131-8.

29 Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 399–408. ACM, 1998. doi:10.1145/276698.276852.

30 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *sicomp*, pages 186–208, 1989. Preliminary version in *STOC'85*.

31 Mika Göös, Toniann Pitassi, and Thomas Watson. Zero-information protocols and unambiguity in arthur-merlin communication. *Algorithmica*, 76(3):684–719, 2016. doi:10.1007/s00453-015-0104-9.

32 Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge pcps, and unconditional cryptography. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2010. doi:10.1007/978-3-642-14623-7_10.

33 Tom Gur and Ron D. Rothblum, 2015. Unpublished observation.

34 Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. *Computational Complexity*, pages 1–109, 2016. doi:10.1007/s00037-016-0136-9.

35 Tom Gur and Ron D. Rothblum. A hierarchy theorem for interactive proofs of proximity. In *Proceedings of the 2017 ACM Conference on Innovations in Theoretical Computer Science, Berkeley, CA, USA, January 9-11, 2016*, 2017.

36 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009. `doi:10.1137/080725398`.

37 Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 121–145. Springer, 2014. `doi:10.1007/978-3-642-54242-8_6`.

38 Yuval Ishai, Mor Weiss, and Guang Yang. Making the best of a leaky situation: Zero-knowledge pcps from leakage-resilient circuits. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2016. `doi:10.1007/978-3-662-49099-0_1`.

39 Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. A language-dependent cryptographic primitive. *Journal of Cryptology*, pages 37–49, 1997.

40 Yael Tauman Kalai and Ran Raz. Interactive PCP. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 536–547. Springer, 2008. `doi:10.1007/978-3-540-70583-3_44`.

41 Yael Tauman Kalai and Ron D. Rothblum. Arguments of proximity - [extended abstract]. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 422–442. Springer, 2015. `doi:10.1007/978-3-662-48000-7_21`.

42 Satyen Kale and C. Seshadhri. An expansion tester for bounded degree graphs. *SIAM J. Comput.*, 40(3):709–720, 2011. `doi:10.1137/100802980`.

43 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.

44 Joe Kilian, Erez Petrank, and Gábor Tardos. Probabilistically checkable proofs with zero knowledge. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 496–505. ACM, 1997. `doi:10.1145/258533.258643`.

45 Silvio Micali and Rafael Pass. Precise zero knowledge. `http://www.cs.cornell.edu/~rafael/papers/preciseZK.pdf`, 2007.

46 Daniele Micciancio and Salil Vadhan. Statistical zero-knowledge proofs with efficient provers: lattice problems and more. In *crypto03*, pages 282–298, 2003.

47 Asaf Nachmias and Asaf Shapira. Testing the expansion of a graph. *Inf. Comput.*, 208(4):309–314, 2010. `doi:10.1016/j.ic.2009.09.002`.

48 Minh-Huyen Nguyen and Salil Vadhan. Zero knowledge with efficient provers. In *stoc38*, pages 287–295, 2006.

49 Shien Jin Ong and Salil P. Vadhan. An equivalence between zero knowledge and commitments. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Con-*

ference, TCC 2008, New York, USA, March 19-21, 2008., volume 4948 of *Lecture Notes in Computer Science*, pages 482–500. Springer, 2008. doi:10.1007/978-3-540-78524-8_27.

50 Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62. ACM, 2016. doi:10.1145/2897518.2897652.

51 Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802, 2013.

52 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

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