# ZK-PCPs from Leakage-Resilient Secret Sharing

Carmit Hazay
Bar-Ilan University
carmit.hazay@biu.ac.il

Muthuramakrishnan Venkitasubramaniam
University of Rochester
muthuv@cs.rochester.edu

Mor Weiss
Bar-Ilan University
mor.weiss@biu.ac.il

**Abstract**

Zero-Knowledge PCPs (ZK-PCPs; Kilian, Petrank, and Tardos, STOC '97) are PCPs with the additional zero-knowledge guarantee that the view of any (possibly malicious) verifier making a bounded number of queries to the proof can be efficiently simulated up to a small statistical distance. Similarly, ZK-PCPs of Proximity (ZK-PCPPs; Ishai and Weiss, TCC '14) are PCPPs in which the view of an adversarial verifier can be efficiently simulated with few queries to the input.

Previous ZK-PCP constructions obtained an exponential gap between the query complexity $q$ of the honest verifier, and the bound $q^*$ on the queries of a malicious verifier (i.e., $q = \mathsf{poly}\log(q^*)$), but required either exponential-time simulation, or adaptive *honest* verification. This should be contrasted with standard PCPs, that can be verified non-adaptively (i.e., with a single round of queries to the proof). The problem of constructing such ZK-PCPs, *even when* $q^* = q$, has remained open since they were first introduced more than 2 decades ago. This question is also open for ZK-PCPPs, for which no construction with non-adaptive honest verification is known (not even with exponential-time simulation).

We resolve this question by constructing the *first* ZK-PCPs and ZK-PCPPs which *simultaneously* achieve *efficient* zero-knowledge simulation and *non-adaptive* honest verification. Our schemes have a square-root query gap, namely $q^*/q = O(\sqrt{n})$ where $n$ is the input length.

Our constructions combine the "MPC-in-the-head" technique (Ishai et al., STOC '07) with leakage-resilient secret sharing. Specifically, we use the MPC-in-the-head technique to construct a ZK-PCP variant over a large alphabet, then employ leakage-resilient secret sharing to design a new alphabet reduction for ZK-PCPs which preserves zero-knowledge.

# Contents

# 1 Introduction

Probabilistically Checkable Proofs (PCPs) [ALM+92, AS92] allow a probabilistic verifier to check the validity of a given statement by only querying few proof bits. Zero-Knowledge (ZK) proofs [GMR85] allow a prover to convince a verifier of the validity of a statement, without revealing any additional information to the verifier. This work focuss on *Zero-Knowledge Probabilistically Checkable Proofs (ZK-PCPs)* (and ZK-PCPs *of proximity*), which combine the advantages of these two types of proof systems. Before describing our main results, we first give a short overview of these proof systems.

**Probabilistically Checkable Proofs (PCPs)** [ALM+92, AS92] allow a randomized efficient verifier $\mathcal{V}$ with oracle access to a purported proof $\pi$ to verify an NP-statement of the form "$x \in \mathcal{L}$" by reading only few bits of $\pi$. The proof can be efficiently generated given the NP witness, and the verifier accepts true claims with probability 1, whereas false claims are accepted with low probability (which is called the soundness error). The celebrated PCP theorem [ALM+92, AS92, Din06] states that any NP language has a PCP system with soundness error $1/2$, in which the verifier reads only $O(1)$ bits from a polynomial-length proof (soundness can be amplified through repetition). An attractive feature of these PCP systems is that the verifier is *non-adaptive*, namely it makes a single round of queries to the proof. *PCPs of Proximity (PCPPs)* [DR04, BGH+04, Din06] are a generalization of PCPs in which the verifier does not read its entire input. Instead, $\mathcal{V}$ has oracle access to $x, \pi$, and wishes to check whether $x$ is close to $\mathcal{L}$ (in relative Hamming distance). The best PCPP constructions for NP [BS08, Mie09] obtain comparable parameters to the PCP systems described above, where any $x$ which is $\delta$-far from $\mathcal{L}$ in relative Hamming distance is rejected with high probability, and $\delta$ is a constant or even inverse polylogarithmic ($\delta$ is called the *proximity parameter*).

**Zero-Knowledge (ZK) proofs** [GMR85] allow a randomized efficient prover to prove an NP-statement of the form "$x \in \mathcal{L}$" to a randomized efficient verifier, while guaranteeing that true claims are accepted with probability 1, false claims are rejected with high probability, and the verifier learns no information about the corresponding NP-witness. This last property, known as *zero-knowledge*, is formalized by requiring that for any (possibly malicious) efficient verifier $\mathcal{V}^*$, there exists an efficient *simulator* machine that has access only to the statement $x$, and can simulate the interaction of $\mathcal{V}^*$ with the honest prover.

**Zero-Knowledge PCPs (ZK-PCPs)** [KPT97] combine the attractive features of PCPs and ZK proofs. Specifically, ZK-PCPs are PCPs in which the prover $\mathcal{P}$ is randomized, and the proof $\pi$ has the following zero-knowledge guarantee: the view of every (possibly malicious, possibly unbounded) verifier $\mathcal{V}^*$ that makes an a-priori bounded number of queries to the proof, can be efficiently simulated up to a small statistical distance. We remark that restricting $\mathcal{V}^*$ to making an a-priori bounded number of queries is inherent to obtaining ZK with polynomial-length proofs.

The first ZK-PCP constructions for NP [KPT97, IMS12] obtain ZK against any verifier $\mathcal{V}^*$ that is restricted to querying at most $q^* = q^*(|x|)$ proof bits, with proofs of length $\mathsf{poly}(q^*)$ that can be verified with $\mathsf{polylog}(q^*)$ queries and have a negligible soundness error. In particular, the *query gap* $q^*/q$ — the ratio between the query complexities of the malicious and honest verifiers — obtained by these constructions is exponential.[1] Unfortunately, obtaining ZK in [KPT97, IMS12] did not come without a cost: it required the *honest* verifier to be adaptive, namely to make *several* rounds of queries to the proof (where the queries of each round depend on the answers to previous queries).

[1] We stress that a larger gap is preferable to a smaller one, since it means the proof can be verified with few queries, while guaranteeing zero-knowledge even when a malicious verifier makes many more queries (compared to the honest verifier).

In cryptographic applications of ZK-PCPs (e.g., in [IWY16]) this blows-up the round complexity of resultant protocols. In particular, every round of queries which the verifier makes to the ZK-PCP induces two communication rounds in the interactive protocols which rely on ZK-PCPs.

Ishai and Weiss [IW14] introduce the notion of *Zero-Knowledge PCPPs (ZK-PCPPs)*. Similar to ZK-PCPs, the ZK-PCPP prover is randomized, and zero-knowledge means that the view of any verifier $\mathcal{V}^*$ making $q^*$ queries to *the input* and the proof can be efficiently simulated, up to a small statistical distance, *by making only $q^*$ queries to the input*. They use similar techniques to [KPT97, IMS12] to obtain ZK-PCPPs for NP with comparable parameters to the ZK-PCPs of [KPT97, IMS12], where the proximity parameter $\delta$ is constant or inverse polylogarithmic. These ZK-PCPPs also require adaptive verification, which increases the round complexity in their cryptographic applications [IW14, Wei16].

As discussed in Sections 2.1.4 and 2.2.2 below, adaptive verification is in fact inherent to the constructions of [KPT97, IMS12, IW14]. Indeed, these schemes are obtained by combining a PCP or PCPP with a weak zero-knowledge guarantee that only holds against the *honest* verifier, with an information-theoretic commitment primitive called locking schemes [KPT97]. This latter primitive requires adaptive opening, which causes the resultant ZK-PCP verifier to be adaptive.

**Can ZK-PCPs be verified non-adaptively?** Motivated by the goal of obtaining ZK for PCPs at no additional cost, Ishai et al. [IWY16] gave a partial answer to this question. Specifically, they construct ZK-PCPs with similar parameters to the schemes of [KPT97, IMS12] in which the honest verifier is *non-adaptive*, but with a weaker zero-knowledge guarantee compared to standard ZK-PCPs: the zero-knowledge simulator is inefficient (this is also known as *witness-indistinguishability*). Alternatively, they obtain ZK with efficient simulation against *computationally-bounded* verifiers, assuming the existence of one-way functions and a common random string. The techniques of [IWY16] diverge from the standard method of designing ZK-PCPs [KPT97, IMS12] discussed above. Specifically, the ZK-PCP of [IWY16] is based on a novel connection to *leakage-resilient circuits*, which are circuits that operate over encoded inputs, and resist certain "side channel" attacks in the sense that such attacks reveal nothing about the input other than the output. Unfortunately, the weaker ZK guarantee of the ZK-PCPs of [IWY16] carries over to any application in which these systems are used. Moreover, [IWY16] give evidence that inefficient simulation is inherent to their technique of using leakage-resilient circuits.

**Non-adaptive honest vs. malicious verification.** It is instructive to note that while having non-adaptive *(honest)* verification is a feature of the system (since it guarantees that the honest verifier can achieve soundness with a single round of queries), having zero-knowledge against *non-adaptive malicious* verifiers is a restriction of the system, since there is no ZK guarantee against *adaptive* malicious verifiers, that make several rounds of queries to the proof.

We note that in [IWY16], leakage-resilient circuits fall short of yielding ZK-PCPs with full-fledged zero-knowledge not only because the simulation is inefficient, but also because *zero-knowledge holds only against non-adaptive (malicious) verifiers.* Ishai et al. [IWY16] obtain ZK (with inefficient simulation) against adaptive verifiers by combining leakage-resilient circuits with techniques of [CDD+01]. These techniques incur an exponential blowup in the complexity of the ZK simulator, but did not pose a problem for [IWY16] since their simulator (even against *non-adaptive malicious* verifiers) was already inefficient.

The current landscape of ZK-PCPs is unsatisfying. Current ZK-PCP constructions either require adaptive verification [KPT97, IMS12], or guarantee only a weak form of ZK with an inefficient

simulator [IWY16]. This holds regardless of the query gap, i.e., even if we restrict malicious verifiers to making *the same* number of queries as the honest verifier. For ZK-PCPPs, the situation is even worse: *no* constructions with non-adaptive verification are known (not even with inefficient simulation). This state of affairs gives rise to the following natural question:

*Do there exist ZK-PCPs (and ZK-PCPPs) with non-adaptive verification and efficient simulation?*

As we discuss in Sections 2.1.4 and 2.2.2 below, the limitations of existing ZK-PCP and ZK-PCPP constructions seem to be inherent to the respective techniques they employ to obtain ZK. This seems to imply that obtaining both non-adaptive verification and efficient simulation requires new techniques. Or maybe such objects do not even exist?

## 1.1 Our results

In this work, we answer our research question in the affirmative: we construct ZK-PCPs and ZK-PCPPs that can be verified non-adaptively and have efficient zero-knowledge simulation. Unlike the schemes of [KPT97, IMS12, IW14, IWY16], which obtain an exponential gap between the query complexities of the malicious and honest verifiers, we are only able to obtain a *polynomial* query gap ($q^*$ vs. $(q^*)^\epsilon$, for some constant $\epsilon \in (0,1)$).

In the following, we say that a PCP (PCPP, resp.) system is a non-adaptive $q$-query $q^*$-ZK-PCP ($q^*$-ZK-PCPP, resp.) if it is perfectly ZK against a (possibly malicious, possibly adaptive) verifier making $q^*$ queries, and achieves a $\mathsf{negl}(q^*)$ soundness error where the honest verifier makes $q$ non-adaptive queries to the proof.

Specifically, we obtain the following results:

**Theorem 1** (Non-Adaptive ZK-PCPs with Efficient Simulation)**.** *There exists a constant $\epsilon \in (0,1)$ such that for any ZK parameter $q^* \in \mathbb{N}$ there exists a non-adaptive $(q^*)^\epsilon$-query $\Omega(q^*)$-ZK-PCP for NP.*

**Theorem 2** (Non-Adaptive ZK-PCPPs with Efficient Simulation)**.** *Let $n \in \mathbb{N}$ be an input length parameter. Then there exists a constant $c > 0$ such that for any proximity parameter $\delta \geq 1/\sqrt{n}$, there exists a non-adaptive $q$-query $q^*$-ZK-PCPP for NP with proximity parameter $\delta$, $q^* = \Omega\left(n^{c+1}\right)$, and $q = \tilde{O}\left(n^{c+1/2}\right)$.*

Our non-adaptive ZK-PCPs and ZK-PCPPs can be plugged-into the applications described in [IW14, IWY16, Wei16], and will reduce the round complexity of the resultant protocols.[2]

Our constructions show that leakage-resilience techniques *can* be used to construct ZK-PCPs (and ZK-PCPPs) with both non-adaptive (honest) verification and efficient simulation. Specifically, we circumvent the negative result of [IWY16] on the limitations of using leakage-resilient *circuits*, by relying on leakage-resilient *secret sharing* [DP07, DDV10] secure against local leakage [GK18, BDIR18, ADN+19]. Compared to leakage-resilient circuits, leakage-resilient secret sharing has the weaker guarantee of only protecting *information* from leakage, whereas leakage-resilient circuits also protect *computation*. However, this weaker guarantee suffices for our needs, and admits leaner and more efficient constructions compared to those of leakage-resilient circuits (and applications

[2] In this context, we note that if one only requires ZK against the *honest* verifier, then non-adaptive ZK-PCPs and ZK-PCPPs are known. (This is implicit in [KPT97] and [IW14] for ZK-PCPs and ZK-PCPPs respectively, via standard soundness amplification.) Consequently, our non-adaptive ZK-PCPs and ZK-PCPPs (with ZK against *malicious* verifiers) do not improve the round complexity in applications that only require ZK against the honest-verifier (e.g., the ZK arguments of [IMS12], and the commit-and-prove protocols of [IW14]).

using them). Specifically, we use leakage resilient secret sharing to design a new alphabet reduction procedure that transforms a ZK-PCP over a large alphabet to a ZK-PCP over bits, while preserving zero-knowledge.

# 2 Our Techniques

We now give more details about our ZK-PCP and ZK-PCPP constructions.

## 2.1 ZK-PCPs with Non-Adaptive Verification and Efficient Simulation

Our starting point is a ZK-PCP implicit in the work of [IKOS07]. They use secure Multi-Party Computation (MPC) protocols to construct a ZK-PCP variant *over a large (poly-sized) alphabet* with efficient ZK simulation, that can be verified non-adaptively. Their ZK-PCP suffers from two disadvantages. First, strictly speaking it is not a ZK-PCP, since in standard ZK-PCPs the proof is a bit string, whereas the ZK-PCP of [IKOS07] is over a large alphabet. Second, their construction has no query gap, namely the proof is ZK against verifiers querying $q^*$ proof symbols, but to get soundness the honest verifier must also make $q^*$ queries.

### 2.1.1 Amplifying the Query Gap in the ZK-PCP of [IKOS07]

To prove that $x \in \mathcal{L}$ for some NP-language $\mathcal{L}$ with a corresponding NP relation $\mathcal{R} = \mathcal{R}(x, w)$, Ishai et al. [IKOS07] employ an $n$-party protocol that computes the function $f_{\mathcal{R}}(x, w_1, \ldots, w_n) = \mathcal{R}(x, \oplus w_i)$, where $x$ is a common input, and $w_i$ is the input of the $i$th party $P_i$. The prover executes the MPC protocol "in its head", obtaining the *views* of all parties $P_1, \ldots, P_n$ in the execution (the view of party $P_i$ consists of its input, random input, and all messages it received during the execution). The proof consists of all these views, where each view is a symbol in the resultant proof. To verify the proof, the verifier reads several views, and checks that: (1) the output reported in all views is 1; and (2) the views are *consistent*, namely for every pair $V_i, V_j$ of queried views of $P_i, P_j$ (respectively), the incoming messages from $P_i$ reported in $V_j$ are the messages $P_i$ would send in the protocol given its view $V_i$, and vice versa.

To get $q^*$-ZK, the protocol should be *private* against $q^*$ (semi-honest) parties, in the sense that they learn nothing from the execution except their inputs and the output. For soundness, [IKOS07] rely on a notion of correctness against $q^*$ corrupted parties (known as *robustness*, see Section 3.2 for the exact definition), guaranteeing that even if $q^*$ parties arbitrarily deviate from the protocol, they cannot cause an honest party to output 1 in a protocol execution on $x \notin \mathcal{L}$. We revisit their analysis, and show that general MPC protocols yield a square root query gap. That is, given a $q^*$-private and $q^*$-robust MPC protocol, the resultant ZK-PCP over a large alphabet is ZK against a (possibly malicious) verifier querying $q^*$ proof symbols, and can be non-adaptively verified with only $\sqrt{q^*}$ queries, with a negligible soundness error. This already yields a non-trivial ZK-PCP *over a large alphabet*. The analysis appears in Section 4, and is black-box in the underlying MPC protocol.

### 2.1.2 Alphabet Reduction for ZK-PCPs

Next, we address the fact that the ZK-PCP of [IKOS07] is over a large alphabet. For standard PCPs, one can easily reduce the alphabet $\Sigma$ over which the proof $\pi$ is defined to $\{0,1\}$ by simply replacing each alphabet symbol with a bit string, thus obtaining a new proof $\pi'$ over $\{0,1\}$. This

would increase the proof length and the query complexity of the honest verifier by a multiplicative $\log|\Sigma|$ factor, but would not otherwise affect the system.[3]

Unfortunately, applying this transformation to *zero-knowledge* PCPs might render the resultant scheme totally insecure. Indeed, while the system would still be ZK against verifiers making $q^*$ queries, the query gap now reduces since the query complexity of the *honest* verifier (i.e., the number of queries it *must* make to obtain soundness) increases. Specifically, depending of $|\Sigma|$, the honest verifier might now need to make $> q^*$ queries, but $\pi'$ might not be ZK even against $q^*+1$ malicious queries. As a result, $\pi'$ might not be ZK even against *malicious* verifiers that make *fewer* queries than the honest verifier! Indeed, a malicious verifier $\mathcal{V}^*$ with oracle access to $\pi'$ is not restricted to querying "whole" symbols of $\pi$, i.e., reading the entire substring of $\pi'$ that corresponds to a symbol of $\pi$. On the contrary, $\mathcal{V}^*$ might read *"parts"* of symbols, thus potentially gaining (partial) information on $q^*+1$ symbols of $\pi$, and possibly violating the ZK guarantee of the original system.

The trivial alphabet reduction for PCPs described above fails because querying *even a single bit* in the bit string $s_\sigma$ representing a symbol $\sigma \in \Sigma$ might reveal information about $\sigma$. Therefore, to make this alphabet reduction work for *zero-knowledge* PCPs, we must guarantee that querying few bits of $s_\sigma$ reveals nothing about $\sigma$. We do so using leakage-resilient secret sharing.

At a high level, a ($t$-threshold) *Secret Sharing Scheme (SSS)* is a method of distributing a secret $\mathsf{s}$ among $n$ parties by giving each party $P_i$ a share $\mathsf{Sh}_i$, such that any $t$ shares reveal no information about $\mathsf{s}$, but any $t+1$ shares completely determine $\mathsf{s}$. A *Leakage-Resilient Secret Sharing Scheme (LR-SSS)* against local leakage [GK18, BDIR18, ADN+19] has the added feature of resisting leakage on the shares, in the following sense. The secret $\mathsf{s}$ remains hidden given $t$ shares, *as well as few leakage bits computed separately on every other share.*

Given a ZK-PCP system $(\mathcal{P},\mathcal{V})$ over a large alphabet $\Sigma$, and a LR-SSS for secrets in $\{0,1\}^{\log|\Sigma|}$, our alphabet reduction works as follows. The prover $\mathcal{P}'$ uses $\mathcal{P}$ to generate a proof $\pi = \sigma_1 \dots \sigma_N$ over $\Sigma$, replaces every $\sigma_i$ with its bit-representation $s_{\sigma_i}$, which it secret shares using the LR-SSS. The proof $\pi'$ consists of the secret sharings of $s_{\sigma_1}, \dots, s_{\sigma_N}$. To verify the proof, the verifier $\mathcal{V}'$ emulates $\mathcal{V}$, where a query $Q$ of $\mathcal{V}$ to its proof $\pi$ is answered as follows. $\mathcal{V}'$ uses the secret sharing of $s_{\sigma_Q}$ (from its own proof oracle $\pi'$) to reconstruct $\sigma_Q$, which it then provides to $\mathcal{V}$.[4]

The PCP system obtained through the reduction preserves the completeness and soundness of $(\mathcal{P},\mathcal{V})$, and guarantees ZK against *non-adaptive* (possibly malicious) verifiers that are restricted to making (roughly) $q^{**} = q^*\ell t$ queries to the proof, where $(\mathcal{P},\mathcal{V})$ is ZK against verifiers querying $q^*$ proof symbols, and the LR-SSS is private against any $t$ shares as well as $\ell$ leakage bits from every other share.

To see why ZK holds, we split the proof $\pi'$ into $N$ "sections", where the $i$th section contains the secret sharing of $s_{\sigma_i}$, and $s_{\sigma_i}$ is the bit-representation of the $i$th symbol $\sigma_i$ of the original proof $\pi$. Roughly (and somewhat inaccurately), the queries of any non-adaptive (possibly malicious) verifier $\mathcal{V}^*$ querying at most $q^{**}$ proof bits divide the sections of $\pi'$ into two groups.

1. "Light" sections, from which $\mathcal{V}^*$ reads at most $\ell t$ bits. In particular, for each such section containing the secret shares $\mathsf{Sh}_1, \dots, \mathsf{Sh}_n$ of the bit-representation $s_\sigma$ of a symbol $\sigma$, there can be at most $t$ shares from which $\mathcal{V}^*$ reads more than $\ell$ bits, and each other share is queried only $\ell$ times. Therefore, the leakage-resilience of the LR-SSS guarantees that $s_\sigma$ (and consequently also $\sigma$) remains entirely hidden.

[3] We note that several PCP constructions (e.g. [Din06]) use more elaborate alphabet reduction techniques *for efficiency reasons* (in particular, their goal is to achieve quasi-linear length proofs with $O(1)$ query complexity and a constant soundness error). A $\log|\Sigma|$ blowup is less significant in the context of *zero-knowledge* PCPs, where the query complexity is anyway $\omega(1)$ since we wish to have a negligible soundness error.

[4] Due to some technical issues, the construction is actually somewhat more involved, see Section 5 for the construction and further details.

2. "Heavy" sections, from which $\mathcal{V}^*$ queries more than $\ell t$ bits. Notice that there can only be at most $q^*$ such sections, and $\mathcal{V}^*$ obtains no information about the symbols of $\pi$ encoded in "light" sections, so the queries to the "heavy" sections can be simulated by the ZK of $(\mathcal{P}, \mathcal{V})$.

(The full — and accurate — analysis appears in the proof of Theorem 6 in Section 5.)

In summary, combining a ZK-PCP over a large alphabet with a SSS that resists probing leakage, we obtain a ZK-PCP over $\{0,1\}$. Instantiating the transformation with the ZK-PCP of [IKOS07] (with our improved analysis) together with the LR-SSS of [SV19] yields a ZK-PCP with the parameters of Theorem 1that is ZK against (possibly malicious and unbounded) verifiers that only make *non-adaptive* queries to their proof oracle.

### 2.1.3 Amplifying to ZK Against Adaptive Verifiers

The analysis of our alphabet reduction for ZK-PCPs crucially relied on the fact that the malicious verifier $\mathcal{V}^*$ was *non-adaptive*. Indeed, the queries to "light" and "heavy" sections are simulated differently (using the LR-simulator of the LR-SSS, and the ZK-simulator of $(\mathcal{P}, \mathcal{V})$, respectively), meaning the simulator for $(\mathcal{P}', \mathcal{V}')$ needs to know at the onset of the simulation which sections are "heavy" and which are "light". Obtaining ZK against *adaptive* verifiers seems to require a stronger leakage-resilience guarantee with a two-phase flavor similar to the locking schemes of [KPT97, IMS12]. Specifically, in the first phase of the simulation, the LR-simulator $\mathsf{Sim}_{\mathsf{LR}}$ should be able to answer adaptive leakage queries as in a standard (adaptively-secure) LR-SSS. However, unlike standard LR-SSSs, at some point the simulation may move to a second phase. In the second phase the simulator $\mathsf{Sim}_{\mathsf{LR}}$ is given a secret $\mathsf{s}$, and should be able to "explain" $\mathsf{s}$ by providing an entire secret sharing of $\mathsf{s}$ which is random subject to being consistent with the previously-simulated answers to leakage queries. We formalize this notion, introducing *equivocal SSSs* as a generalization of standard LR-SSSs, and provide a construction based on codes with leakage-resilience guarantees (see Section 2.3 for further details).

Applying our alphabet reduction to $(\mathcal{P}, \mathcal{V})$ and an *equivocal* SSS now yields a ZK-PCP with ZK against any — possibly malicious *and adaptive* — verifier $\mathcal{V}^*$ making at most $q^{**}$ queries. Indeed, the ZK-PCP simulator $\mathsf{Sim}$ starts the simulation by answering all queries using the LR-simulator $\mathsf{Sim}_{\mathsf{LR}}$ of the equivocal SSS. Whenever the number of queries to a certain proof section passes some threshold, $\mathsf{Sim}$ uses the ZK-simulator $\mathsf{Sim}_{\mathsf{ZK}}$ to simulate the underlying symbol $\sigma$ of $\pi$. Then, $\mathsf{Sim}$ provides the bit-representation of $\sigma$ to $\mathsf{Sim}_{\mathsf{LR}}$ as the secret-shared secret. The equivocation property of the SSS guarantees that $\mathsf{Sim}_{\mathsf{LR}}$ can now "explain" this secret by providing an entire secret sharing which is consistent with the previous leakage. These secret shares can be used to answer any further queries to that section of the proof. The analysis appears in Section 5.1.

To sum up, combining a ZK-PCP over a large alphabet with an *equivocal* SSS against probing leakage yields a ZK-PCP over $\{0,1\}$, where zero-knowledge holds against *adaptive* verifiers. Instantiating the transformation with the ZK-PCP of [IKOS07] and the equivocal scheme described in section 2.3.2 yields a ZK-PCP with the parameters of Theorem 1.

### 2.1.4 Why Do Previous Approaches of Constructing Non-Adaptive ZK-PCPs Fail?

It is instructive to discuss why our approach of combining alphabet reduction with LR secret sharing succeeds in simultaneously obtaining non-adaptive verification and efficient simulation, whereas previous approaches [KPT97, IMS12, IWY16] could only achieve one of these properties.

As noted above, the ZK-PCPs of [KPT97, IMS12] are obtained by combining PCPs that are ZK against the honest verifier with locking schemes. In effect, the locking schemes are used to "force" the queries of a malicious verifier to be distributed (almost) as the queries of *the honest verifier*. This

transformation causes adaptive verification due to two reasons: first, the original proof is altered in such a way that necessitates adaptive queries to verify it. Second, the locking schemes themselves require adaptive opening. We are faced with a similar challenge, where the queries of the malicious verifier might drastically deviate from those of an honest verifier (namely, $\mathcal{V}^*$ might query "parts" of symbols, whereas the honest verifier always queries whole symbols). However, instead of "forcing" the queries of $\mathcal{V}^*$ to "look" honest, we allow $\mathcal{V}^*$ to make any set of queries, but guarantee that queries to "parts" of symbols reveal no information on the underlying symbol.

The ZK-PCP of [IWY16] uses a different approach. Their starting point is a *non-ZK* PCP, and they use leakage-resilient circuits to protect *the entire PCP generation*. That is, the queries of the verifier are interpreted as leakage on the process of generating the PCP from the NP-witness, and by protecting this entire computation from leakage, they obtain ZK. More specifically, they change the NP statement which is being verified: instead of verifying that $(x, w)$ is a satisfying input for the verification circuit $C$ of the NP-relation $\mathcal{R}$, the honest verifier checks whether the leakage-resilient version $\widehat{C}$ of $C$ is satisfiable. Therefore, soundness of the resultant ZK-PCP system crucially relies on the fact that if there exists no $w$ such that $C(x, w) = 1$, then there exists no $w'$ such that $\widehat{C}(x, w') = 1$,[5] a notion which they call *SAT-respecting*. Ishai et al. [IWY16] give evidence that SAT-respecting leakage-resilient circuits with efficient LR-simulation (for the leakage classes needed to construct ZK-PCPs) exist only for languages in BPP. The inefficient LR-simulation is the cause of ZK with inefficient simulation in their ZK-PCPs. We circumvent their negative results by using LR-*SSSs* to protect *information* — instead of using LR *circuits* to protect *computation* — and apply the LR-SSS to PCPs with ZK guarantees (whereas [IWY16] use standard PCPs).

## 2.2 ZK-PCPPs with Non-Adaptive Verification and Efficient Simulation

We extend our techniques to apply to PCPs *of Proximity*. Specifically, our alphabet reduction could also be applied to ZK-PCPPs, which reduces the task of designing ZK-PCPPs with non-adaptive verification to designing such ZK-PCPPs over a large alphabet.

### 2.2.1 A ZK-PCPP with Non-Adaptive Verification Over Large Alphabets

The first step is to design a ZK-PCPP over a large alphabet. We do so by presenting a variant of the system of [IKOS07] in which the verifier does not read its entire input. Recall that the proof in the ZK-PCP of [IKOS07] consists of the views of all parties in an execution of an MPC protocol for the function $f_{\mathcal{R}}(x, w_1, \ldots, w_n) = \mathcal{R}(x, \oplus w_i)$, where $x$ is a common input, and $w_i$ is the input of the $i$th party $P_i$. In particular, a single proof symbol (i.e., a single view) reveals the entire input $x$. This is problematic in the context of ZK-PCPPs, in which the proof is required to hide not only the NP-witness $w$, but also most of the input $x$, in the sense that a verifier making few queries learns only few physical bits of $x$.

Thus, no single party can hold the entire input $x$. Instead, following [IW14] we introduce $m$ additional "input parties" (where $m = |x|$), such that the MPC protocol is over $m + n$ parties $P_1, \ldots, P_{m+n}$. The inputs of parties $P_1, \ldots, P_m$ are $x_1, \ldots, x_m$ (respectively), and the inputs of parties $P_{m+1}, \ldots, P_{m+n}$ are $w_1, \ldots, w_n$ (respectively). Proof generation from this revised protocol is similar to the original construction of [IKOS07], and verification is also similar, except that whenever the verifier queries a view of $P_i, i \in [m]$, it also queries $x_i$ from its input oracle, and checks that $x_i$ is the input of $P_i$ reported in the view.

[5] In fact, $\widehat{C}$ operates on encoded inputs, however to simplify the discussion we disregard this at this point, and provide a more accurate discussion in Section 2.2.2 below.

If the MPC protocol is $q^*$-private, then we are guaranteed that $q^*$ queries to the proof reveal at most $q^*$ bits of $x$. Indeed, the $q^*$-privacy of the protocol guarantees that the views of $q^*$ parties reveal no information other than their inputs (which is at most $q^*$ bits of $x$) and their output (which is 1).

As for soundness, we show that our improved analysis (discussed in Section 2.1.1 above) extends to PCPPs. Specifically, we show that if the MPC protocol is $q^*$-robust then the resultant ZK-PCPP is sound with proximity parameter $\delta \geq 1/\sqrt{|x|}$, namely the verifier rejects (with high probability) inputs that are $\delta$-far from the language. Indeed, let $x$ be $\delta$-far from $\mathcal{L}$, and notice that any (possibly ill-formed) proof $\pi^*$ for $x$ determines an effective input $x^*$ for the underlying MPC protocol ($x^*$ is obtained by concatenating the inputs reported in the views of $P_1, \ldots, P_m$). We show that if $x^*$ is $\delta$-far from $x$ then with overwhelming probability the verifier will query a view on which $x, x^*$ differ, in which case it rejects with probability 1. Otherwise, $x^*$ is $\delta$-close to $x$, implying $x^* \notin \mathcal{L}$, in which case our improved analysis of Section 2.1.1 essentially shows that the PCPP verifier rejects with overwhelming probability. This yields the first ZK-PCPP with non-adaptive verification and efficient simulation, but the proof is over a large alphabet. The analysis of this ZK-PCPP system appears in Section 6.1.

Combining this ZK-PCPP over a large alphabet with our alphabet reduction, we obtain *the first* ZK-PCPPs over $\{0,1\}$ with non-adaptive verification. Moreover, the system has efficient ZK simulation. Specifically, combining the ZK-PCPP over a large alphabet with a LR-SSS gives a ZK-PCPP with ZK against *non-adaptive malicious* verifiers, whereas combining it with an equivocal SSS gives a full-fledged ZK-PCPP. Instantiating the equivocal SSS with the scheme of Section 2.3.2 gives a ZK-PCPP with the properties of Theorem 2. The analysis appears in Section 6.2.

### 2.2.2 Why Do Previous Approaches of Constructing Non-Adaptive ZK-PCPPs Fail?

We now give some intuition as to why LR-SSS is useful to obtaining ZK-PCPPs with non-adaptive verification, whereas ZK-PCPP constructions based on leakage-resilient circuits seem to fail. Recall that a leakage-resilient circuit operates over encoded inputs, where leakage from some function class $\mathcal{F}$ on the internal wire values of the circuit reveals no information about the input other than the output. In particular, this implies that the input encoding should resist leakage from $\mathcal{F}$, since leakage can be applied to the input wires (which carry the encoded inputs).

Recall that the verifier wishes to verify that $(x, w) \in \mathcal{R}$, namely that $(x, w)$ satisfies the verification circuit $C$ of $\mathcal{R}$. When using leakage-resilient circuits to verify this claim, the circuit $C$ is replaced with its leakage-resilient variant $\widehat{C}$, which operates over encoded inputs, where the queries of the verifier are interpreted as leakage on the wire values of $\widehat{C}$. This raises the question of how to incorporate $x$ into the computation. Syntactically, $\widehat{C}$ cannot operate directly on the unencoded input $x$, but if $\widehat{C}$ operates on an encoding of $x$, the prover can cheat by providing an encoding of some $x^* \neq x$. (The verifier will not be able to tell the difference because the input encoding is resilient against leakage, namely against the verifier queries.) The solution of [IWY16] is to first *hard-wire* $x$ into $C$, i.e. replace $C$ with $C_x = C(x, \cdot)$, and then generate the leakage-resilient version $\widehat{C_x}$ of $C_x$. The verifier will now verify that $\widehat{C_x}$ is satisfiable.

While this solves the problem for ZK-PCPs, it cannot be applied in the context of ZK-*PCPPs*. Indeed, verifying that $\widehat{C_x}$ is satisfiable requires knowing the structure of $\widehat{C_x}$. This is indeed the case for ZK-PCPs, since $x$ is known to the verifier in its entirety, so the verifier can locally construct $\widehat{C_x}$. However, the ZK-PCPP verifier does not know all of $x$, nor do we want it to — the advantage of ZK-PCPPs over ZK-PCPs (which is crucially exploited by cryptographic applications of ZK-PCPPs) is that the verifier can be sublinear in the input length, and verify the claim without learning "too much" about the input. Therefore, we cannot hard-wire $x$ into the verification circuit, and so it is

unclear how the verifier would verify consistency of its own input with the one used to evaluate $C$.

Finally, we note that even if one were to solve this issue of how to handle the input, using leakage-resilient circuits would incur inefficient ZK simulation in the resultant ZK-PCPP, due to the negative results of [IWY16].

## 2.3 Equivocal Secret Sharing

We generalize the notion of LR-SSS [DP07, DDV10] secure against local leakage [GK18, BDIR18, ADN+19] by introducing *equivocal SSSs*. We then construct a 1-party equivocal SSS based on codes with probing-resilience. We note that while a 1-party SSS is useless as a means of sharing a secret, its equivocation property gives a meaningful way of encoding a secret in a leakage-resilient (in fact, equivocal) manner. In particular, such schemes suffice for constructing ZK-PCPs and ZK-PCPPs as described in Sections 2.1 and 2.2 above. Moreover, since 1-party schemes can be more efficient than multi-party schemes, using 1-party schemes results in more efficient ZK-PCPs and ZK-PCPPs.

### 2.3.1 Equivocal SSS: Definition

Recall that a standard $t$-threshold $n$-party SSS guarantees that the secret remains entirely hidden given any $t$ of the $n$ shares. LR-SSS enhances this privacy property to hold even given leakage on the other shares. We will focus on resisting adaptive $(t, \ell)$-local probing leakage, in which the leakage reveals $t$ shares, as well as $\ell$ bits from every other share. This can be formalized by comparing the real execution with an ideal experiment in which an efficient simulator $\mathsf{Sim}$, that has no knowledge of the secret, answers the leakage queries.

An equivocal SSS generalizes the notion of (adaptive) LR-SSS by considering a two-phase ideal experiment, where the first phase is similar to the ideal experiment of LR-SSS. At the end of the first phase, the adversary can choose whether to continue to the second phase. In the second phase, the simulator is given a secret $\mathsf{s}$, and must generate an entire secret sharing of $\mathsf{s}$, which is given to the adversary. The adversary should have only a negligible advantage in distinguishing the real execution from the ideal experiment, as long as it didn't violate the leakage restrictions. That is, as long as at the onset of the second phase, the adversary obtained at most $t$ shares, and probed at most $\ell$ bits from every other share. Since the adversary can choose *not* to continue to the second phase of the simulation, this notion strictly generalizes the notion of a LR-SSS. Notice that we make no restriction on the computational power of the adversary. The definition appears in Section 3.4.

### 2.3.2 Equivocal SSS: Construction

We use a 1-party equivocal SSS that resists $(0, \ell)$-local probing leakage [GK18, BDIR18, ADN+19], where $\ell$ is a constant fraction of the share size. Considering 1-party schemes suffices for the ZK-PCP and ZK-PCPP application, and admit lean constructions that result in more efficient ZK-PCPs and ZK-PCPPs (in terms of query complexity and proof length).

**Existing leakage-resilient encodings.** A 1-party equivocal SSS gives a method of encoding data such that the resultant encoding is equivocal, and consequently also leakage-resilient. We note that leakage-resilient encodings have been considered before under different names. ZK codes [DGR97, DGR99, ISVW13] are encodings that resists *non adaptive probing* leakage, i.e., these are 0-threshold 1-party SSSs that resist non-adaptive probing leakage. Leakage-resilient storage [DP07, DDV10] encodes the data into two parts that resist *adaptive* leakage from *each part separately*. Thus, leakage-resilient storage can be though of as a ramp 2-party SSS which is private against 0 parties, reconstructible given both shares, and resists adaptive leakage. We note that the schemes

of [DP07, DDV10] resists *general* local leakage (i.e., leakage which operates on each share separately, and has short output), and not just probing. An Equivocal SSS generalizes these notions — while ZK codes and leakage-resilient storage are only secure as long as the number of leakage bits does not pass an a-priori bound, equivocal schemes guarantee security even beyond the leakage threshold. Another related notion is that of a Reconstructable Probabilistic Encoding (RPE) [CDMW08, BDKM16, CDMW18, BDG+18]. Informally, these are leakage resilient encodings that are also equivocal (with perfect leakage resilience and equivocation), with an additional error correction property. RPEs and equivocal SSSs are incomparable: while RPEs are a strengthening of *1-party* equivocal SSS (due to their error correction), (multi-party) equivocal SSSs guarantee leakage resilience even when full shares are leaked. In particular, they can potentially achieve a better leakage rate.

**A specific 1-party equivocal SSS construction.** Our constructions will employ the ZK code of [DGR97, DGR99]. Thy construct a linear code in $\{0,1\}^n$ with constant rate, and leakage resilience against a constant fraction of leaked bits. It is also equivocal, which follows from linearity using its dual distance (see [BDG+18, Lemma 2]). Therefore, it is a 1-party equivocal SSS with constant rate and security against probing of a constant fraction of codeword bits.

**Why do equivocal SSSs yield non-adaptive ZK-PCPs?** We note that the locking schemes used in the constructions of [KPT97, IMS12, IW14] posses an equivocation property which is similar to our equivocal SSS, but applying them towards constructing ZK-PCPs and ZK-PCPPs causes *the honest* verifier to be adaptive. The reason is that reconstructing (i.e., opening) the secret in a locking scheme requires making several rounds of queries to the locking scheme. This is because one should be able to recover the locked secret *without reading the entire lock*, which is needed since locking schemes are generally much longer than the locked secret. (The blowup is inherent to obtaining equivocation in locking schemes.) On the other hand, in an equivocal SSS the secret is reconstructed by reading all (or most) of the shares, which can be done non-adaptively. The fact that reconstruction requires reading many shares is not problematic in terms of efficiency, since the total length of all shares is usually relatively short compared to the secret.

### 2.4 Future Directions

Our work still leaves several open questions for future research. First, it is natural to ask whether the query gap (between the query complexity of the malicious and honest verifiers) in our constructions could be improved — to a better polynomial gap, or even exponential? This could potentially be achieved by instantiating the ZK-PCP of [IKOS07] with an MPC protocol with stringent communication requirements, in which the communication complexity (more specifically, the size of the views) is sublinear in the number of parties. Another natural research direction is to construct *multi-party* equivocal SSSs, and in particular ones that withstand *general* local leakage (and not just probing leakage). Finally, it would be interesting to find further applications of equivocal SSSs in other contexts, e.g., for adaptively-secure MPC.

## 3 Preliminaries

**Basic notations.** We denote the security parameter by $\kappa$. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large $\kappa$'s it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We denote the set of all negligible functions by $\mathsf{negl}(\kappa)$. We use the abbreviation PPT to denote

probabilistic polynomial-time, and denote by $[n]$ the set of elements $\{1,\ldots,n\}$ for some $n \in \mathbb{N}$. For a string $s$ of length $n$, and a subset $I \subseteq [n]$, we denote by $s|_I$ the restriction of $s$ to the coordinates in $I$. For an $\mathsf{NP}$ relation $\mathcal{R}$, we denote by $\mathcal{R}_x$ the set of witnesses of $x$, and by $\mathcal{L}_{\mathcal{R}}$ its associated language. That is, $\mathcal{R}_x = \{w \mid (x,w) \in \mathcal{R}\}$ and $\mathcal{L}_{\mathcal{R}} = \{x \mid \exists\ w\ s.t.\ (x,w) \in \mathcal{R}\}$.

Let $\delta \in (0,1)$, let $\Sigma$ be an alphabet, and let $x, y$ be strings over $\Sigma^n$. We say that $x, y$ are *$\delta$-close* if $\frac{|\{i\,:\,x_i \neq y_i\}|}{n} < \delta$, otherwise we say that $x, y$ are *$\delta$-far*. We say that $x$ is $\delta$-close to a language $\mathcal{L}$ if there exists $x' \in \mathcal{L}$ such that $x, x'$ are $\delta$-close. Otherwise, we say that $x$ is $\delta$-far from $\mathcal{L}$.

**Definition 1.** *Let $X_\kappa$ and $Y_\kappa$ be random variables accepting values taken from a finite domain $\Omega$. The* statistical distance *between $X_\kappa$ and $Y_\kappa$ is*

$$\mathsf{SD}(X_\kappa, Y_\kappa) = \frac{1}{2}\sum_{w \in \Omega} \left|\Pr[X_\kappa = w] - \Pr[Y_\kappa = w]\right|.$$

*We say that $X_\kappa$ and $Y_\kappa$ are* $\varepsilon$-close *if their statistical distance is at most $\varepsilon(\kappa)$. We say that $X_\kappa$ and $Y_\kappa$ are* statistically close*, denoted $X_\kappa \approx_s Y_\kappa$, if $\varepsilon(\kappa)$ is negligible in $\kappa$.*

We use the asymptotic notation $O(\cdot)$ and $\Omega(\cdot)$. We will sometimes disregard polylogarithmic factors, using $\widetilde{O}(n)$ and $\widetilde{\Omega}(n)$ to denote $n \cdot \mathsf{poly}\log n$ and $n/\mathsf{poly}\log n$ ,respectively.

### 3.1 Zero-Knowledge Probabilistically Checkable Proofs (PCPs) and PCPs of Proximity

Informally, a Probabilistically Checkable Proof (PCP) system for a language $\mathcal{L}$ consists of a probabilistic polynomial time prover that given $x \in \mathcal{L}$ and a corresponding witness generates a proof for $x$, and a probabilistic polynomial-time verifier having direct access to individual symbols of the proof. This proof string (called oracle) will be accessed only partially by the verifier. The oracle queries are determined by the verifier's input and coin tosses. Formally,

**Definition 2** (PCP)**.** *A* Probabilistically Checkable Proof (PCP) *for a language $\mathcal{L}$ consists of a PPT prover $\mathcal{P}$ and a* PPT *verifier $\mathcal{V}$ such that the following holds for some negligible function $\mathsf{negl} = \mathsf{negl}(\kappa)$.*

1. Syntax. *The prover $\mathcal{P}$ has input $1^\kappa, x, w$, and outputs a proof $\pi_x$ for $x$ over some alphabet $\Sigma$. The verifier $\mathcal{V}$ has input $1^\kappa, x$, and oracle access to $\pi$. It makes $q$ queries to $\pi$, and outputs either 0 or 1 (representing reject or accept, respectively).*

2. Completeness: *For every $x \in \mathcal{L}$, every $w \in \mathcal{R}_x$, and every proof $\pi_x \in \mathcal{P}(1^\kappa, x, w)$,*

$$\Pr[\mathcal{V}^{\pi_x}(1^\kappa, x) = 1] \geq 1 - \mathsf{negl}(\kappa)$$

*where the probability is over the randomness of $\mathcal{V}$, and $\kappa$ is the security parameter.*

3. Soundness: *For every $x \notin \mathcal{L}$ and every oracle $\pi^*$,*

$$\Pr[\mathcal{V}^{\pi^*}(1^\kappa, x) = 1] \leq \mathsf{negl}(\kappa)$$

*where the probability is over the coin tosses of the verifier, and $\kappa$ is a security parameter.* $\mathsf{negl}$ *is called the* soundness error *of the system.*

**Efficiency measures of a PCP system.** We associate with a PCP system the following efficiency measures: the alphabet size $|\Sigma|$, the query complexity $q$, and the proof length $|\pi|$. We will call such a system a $q$-query PCP over alphabet $\Sigma$. We are generally interested in obtaining PCPs with $\Sigma = \{0,1\}$, in which the proof length $|\pi|$ is polynomial in $|x|$, and $q$ is significantly smaller than $|\pi|$. We note that a PCP prover is usually deterministic, but allowing for randomized provers will be useful when discussing *zero-knowledge* PCPs, as we do next.

Next, we define *zero-knowledge* PCPs. Intuitively, these are PCPs in which the witness remains entirely hidden throughout the verification process, even when the verifier is malicious and can potentially make many more queries to the proof compared to the honest verifier. We guarantee zero-knowledge against any, possibly malicious and unbounded, verifier - the only restriction is on the number of queries the verifier makes to the proof (this restriction is inherent to obtaining zero-knowledge). Thus, we first define the notion of a *query bounded* verifier.

**Definition 3** (Query-bounded verifier)**.** *We say that a (possibly malicious) verifier $\mathcal{V}^*$ with oracle access to a proof $\pi$ is* $q^*$-query-bounded *if it makes only $q^*$ queries to $\pi$.*

**Definition 4** (Non adaptive verifier)**.** *We say that a (possibly malicious) verifier $\mathcal{V}^*$ is* non adaptive *if its queries are determined solely by its input $x$ and its randomness (in particular, $\mathcal{V}^*$ can make a single round of queries to its proof oracle).*

We will use the following notation. For a PCP system $(\mathcal{P}, \mathcal{V})$ and a (possibly malicious) verifier $\mathcal{V}^*$, we use $\mathbf{View}_{\mathcal{V}^*,\mathcal{P}}(x,w)$ to denote the view of $\mathcal{V}^*$ when it has input $x$ and oracle access to a proof that was randomly generated by $\mathcal{P}$ on input $(x,w)$. We are now ready to define zero-knowledge PCPs.

**Definition 5** (ZK-PCP)**.** *We say that a PCP system $(\mathcal{P}, \mathcal{V})$ for $\mathcal{L}$ is a* $(q^*, \varepsilon)$-Zero-Knowledge PCP *(or ZK-PCP for short) if for any (possibly malicious and adaptive) $q^*$-query-bounded verifier $\mathcal{V}^*$ there exists a* PPT *simulator $\mathsf{Sim}$, such that for any $(x,w) \in \mathcal{R}$, $\mathsf{Sim}(1^\kappa, x)$ is distributed $\varepsilon$-statistically close to $\mathbf{View}_{\mathcal{V}^*,\mathcal{P}}(x,w)$.*

*If $(\mathcal{P}, \mathcal{V})$ is a $(q^*, \varepsilon)$-ZK-PCP for $\varepsilon = \mathsf{negl}(\kappa)$ then we simply say that $(\mathcal{P}, \mathcal{V})$ is a $q^*$-ZK-PCP. If $(\mathcal{P}, \mathcal{V})$ is a $(q^*, 0)$-ZK-PCP then we say it is a* perfect *$q^*$-ZK-PCP. If the ZK property of the system only holds against* PPT *verifiers $\mathcal{V}^*$ then we say the system is a* computational *$(q^*, \varepsilon)$-ZK-PCP (or CZK-PCP for short). If the zero-knowledge property is only guaranteed against non-adaptive verifiers then we say the system is a ZK-PCP* for non-adaptive verifiers. *If the* honest *ZK-PCP verifier is non-adaptive then we say that $(\mathcal{P}, \mathcal{V})$ is a* non-adaptive ZK-PCP.

We stress that having a non-adaptive honest verifier is a desirable feature of the system, whereas having ZK against non-adaptive verifiers is a weaker form of ZK (since the system has no guarantee against malicious *adaptive* verifiers).

We remark that although this definition requires a weaker notion with a non-universal simulator, all our constructions obtain the stronger notion with a universal simulator. Furthermore, our constructions will rely on the MPC-in-the-head approach, where the quality of ZK will be inherited from the level of security of the underlying MPC protocol employed by the construction.

Next, we define the notion of PCPs of Proximity (PCPPs). In such systems, the verifier has *oracle access* to the input statement (instead of receiving it explicitly). This allows the verifier to be sublinear in the input length. As for soundness, we cannot generally expect the verifier to reject all $x \notin \mathcal{L}$, but rather only require that inputs which are "sufficiently far" from the language are rejected. The notion of distance which we use is relative Hamming distance.

**Definition 6** (PCPP). *A* Probabilistically Checkable Proof of Proximity (PCPP) *for a language $\mathcal{L}$ with proximity parameter $\delta$ consists of a PPT prover $\mathcal{P}$ and a* PPT *verifier $\mathcal{V}$ such that the following holds for some negligible function $\mathsf{negl} = \mathsf{negl}(\kappa)$.*

1. Syntax. *The prover $\mathcal{P}$ has input $1^\kappa, x, w$, and outputs a proof $\pi_x$ for $x$ over some alphabet $\Sigma$. The verifier $\mathcal{V}$ has input $1^\kappa, |x|$, and oracle access to $x, \pi$. It makes $q$ queries to $x, \pi$, and outputs either 0 or 1.*

2. Completeness: *For every $x \in \mathcal{L}$, every $w \in \mathcal{R}_x$, and every proof $\pi_x \in \mathcal{P}(1^\kappa, x, w)$,*

$$\Pr[\mathcal{V}^{x,\pi_x}(1^\kappa, |x|) = 1] \geq 1 - \mathsf{negl}(\kappa)$$

   *where the probability is over the randomness of $\mathcal{V}$, and $\kappa$ is the security parameter.*

3. Soundness: *For every $x$ that is $\delta$-far from $\mathcal{L}$ and every oracle $\pi^*$,*

$$\Pr[\mathcal{V}^{x,\pi^*}(1^\kappa, |x|) = 1] \leq \mathsf{negl}(\kappa)$$

   *where the probability is over the coin tosses of the verifier, and $\kappa$ is a security parameter.* $\mathsf{negl}$ *is called the* soundness error *of the system, and $\delta$ is called the* proximity parameter.

We also consider a zero-knowledge variant of PCPPs. Similar to ZK-PCPs, such systems guarantee that the witness remains entirely hidden throughout the verification, even if the verifier is malicious and makes many queries to the proof. Zero-knowledge PCPPs additionally guarantee that *most of the input* remains hidden from the verifier, in the sense that the view of the verifier can be simulated by making the same (total) number of queries to the input alone. Similar to the PCP setting, we use $\mathbf{View}_{\mathcal{V}^*,\mathcal{P}}(x, w)$ to denote the view of $\mathcal{V}^*$ given oracle access to $x$, and a proof that was randomly generated by $\mathcal{P}$ on input $x, w$.

**Definition 7** (ZK-PCPP). *We say that a PCPP system $(\mathcal{P}, \mathcal{V})$ for $\mathcal{L}$ is a $(q^*, \varepsilon)$-*Zero-Knowledge PCPP *(or ZK-PCPP for short) if for any (possibly malicious and adaptive) $q^*$-query-bounded verifier $\mathcal{V}^*$ there exists a* PPT *simulator $\mathsf{Sim}$, such that for any $(x, w) \in \mathcal{R}$, $(\mathsf{Sim}^x(1^\kappa, |x|), q_{\mathsf{Sim}})$ is distributed $\varepsilon$-statistically close to $(\mathbf{View}_{\mathcal{V}^*,\mathcal{P}}(x, w), q_{\mathcal{V}^*})$, where $q_{\mathsf{Sim}}$ denotes the number of queries which $\mathsf{Sim}$ makes to $x$, and $q_{\mathcal{V}^*}$ denotes the number of queries which $\mathcal{V}^*$ make to $x$ and the proof.*

*Following the terminology for ZK-PCPs, if $(\mathcal{P}, \mathcal{V})$ is a $(q^*, \varepsilon)$-ZK-PCPP for $\varepsilon = \mathsf{negl}(\kappa)$ then we simply say that $(\mathcal{P}, \mathcal{V})$ is a $q^*$-ZK-PCPP. If $(\mathcal{P}, \mathcal{V})$ is a $(q^*, 0)$-ZK-PCPP then we say it is a* perfect *$q^*$-ZK-PCPP. If the zero-knowledge property is only guaranteed against non-adaptive verifiers then we say the system is a ZK-PCPP* for non-adaptive verifiers. *If the* honest *ZK-PCPP verifier is non-adaptive then we say that $(\mathcal{P}, \mathcal{V})$ is a* non-adaptive ZK-PCPP.

### 3.2 Secure Multi-Party Computation

Following the seminal works of Ishai et al. [IKOS07, IKOS09], our constructions rely on secure multi-party computation protocols in the honest majority setting. In this context, a view $V_i$ of a party $P_i$ within a protocol execution includes its input, randomness and incoming messages. Consistent views are defined as follows.

**Definition 8** (Consistent views). *We say that a pair of views $V_i, V_j$ are consistent (with respect to a protocol $\Pi$ and some public input $x$) if the outgoing messages implicit in $V_i$ are identical to the incoming messages reported in $V_j$ and vice versa.*

We consider security of protocols in both the semi-honest (passive) and the malicious (active) models. In the former model, one may break the security requirements into the following correctness and privacy requirements.

**Definition 9** (Correctness)**.** *We say that* $\Pi$ *realizes a deterministic* $n$*-party functionality* $\mathcal{F}(x, r_1, \ldots, r_n)$ *with perfect (resp., statistical) correctness if for all inputs* $(x, r_1, \ldots, r_n)$*, the probability that the output of some party is different from the output of* $\mathcal{F}$ *is* 0 *(resp., negligible in* $\kappa$*), where the probability is over the independent choices of the random inputs* $r_1, \ldots, r_n$*.*

**Definition 10** ($t$-Privacy)**.** *Let* $1 \leq t < n$*. We say that* $\Pi$ *realizes* $\mathcal{F}$ *with perfect* $t$*-privacy if there is a* PPT *simulator* $\mathsf{Sim}$ *such that for any inputs* $(x, r_1, \ldots, r_n)$ *and every set of corrupted parties* $T \subset [n]$ *where* $|T| \leq t$*, the joint view* $\mathbf{View}_T(x, r_1, \ldots, r_n)$ *of parties in* $T$ *is distributed identically to* $\mathsf{Sim}(T, x, \{r_i\}_{i \in T}, \mathcal{F}_T(x, r_1, \ldots, r_n))$*. The relaxations to statistical or computational privacy are defined in the natural way. That is, in the statistical (resp., computational) case we require that for every distinguisher* $\mathcal{D}$ *(resp.,* $\mathcal{D}$ *with circuit size* $\mathsf{poly}(\kappa)$*) there is a negligible function* $\delta(\cdot)$ *such that*

$$\begin{aligned} &|\Pr[\mathcal{D}(\mathbf{View}_T(\kappa, x, r_1, \ldots, r_n)) = 1] \quad - \\ &\quad \Pr[\mathcal{D}(\mathsf{Sim}(\kappa, T, x, \{r_i\}_{i \in T}, \mathcal{F}_T(x, r_1, \ldots, r_n))) = 1]| \leq \delta(\kappa). \end{aligned}$$

In the malicious model, in which corrupted parties may behave arbitrarily, security cannot be generally broken into correctness and privacy as above. However, similar to [IKOS07], for our purposes we only need the protocols to satisfy a weaker notion of security in the malicious model that is implied by the standard general definition. Specifically, it suffices that $\Pi$ be $t$-private as defined above, and moreover it should satisfy the following notion of correctness in the malicious model.

**Definition 11** ($r$-Robustness)**.** *We say that* $\Pi$ *realizes* $\mathcal{F}$ *with perfect (resp., statistical)* $r$*-robustness if it is perfectly (resp., statistically) correct in the presence of a semi-honest adversary as in Definition 9, and furthermore for any computationally unbounded malicious adversary corrupting a set* $T$ *of at most* $r$ *players, and for any inputs* $(x, r_1, \ldots, r_n)$*, the following robustness property holds. If there is no* $(x, r_1, \ldots, r_n)$ *such that* $\mathcal{F}(x, r_1, \ldots, r_n) = 1$*, then the probability that some uncorrupted party outputs* 1 *in an execution of* $\Pi$ *in which the inputs of the honest parties are consistent with* $(x, r_1, \ldots, r_n)$ *is* 0 *(resp., is negligible in* $\kappa$*).*

### 3.3 Leakage-Resilient Secret Sharing Schemes (LR-SSS)

A Secret-Sharing Scheme (SSS) allows a dealer to distribute a secret among $n$ parties. Specifically, during a *sharing* phase each party receives a share from the dealer, and the secret can then be recovered from the shares during a *reconstruction* phase. The scheme is associated with an *access structure* which defines subsets of authorized and unauthorized parties, where every authorized set can recover the secret from its shares, whereas unauthorized sets learn nothing about the secret even given all their shares. A *Leakage-Resilient* SSS (LR-SSS) guarantees this latter property holds even if the unauthorized set obtains some leakage on the other shares.

We will mainly be interested in *$t$-threshold* secret sharing schemes, in which all (and only) subsets of size at least $t+1$ are authorized to reconstruct the secret. We first define secret sharing schemes.

**Definition 12** (Secret Sharing Scheme)**.** *An* $n$*-party Secret Sharing Scheme (SSS) for secrets in* $\mathcal{S}$ *consists of the following pair of algorithms.*

**Sharing algorithm** Share**:** *Takes as input a secret* $s \in \mathcal{S}$ *and outputs shares* $(s_1, \cdots, s_n) \in \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$*, where* $s_i$ *is called the* share *of party* $i$*, and* $\mathcal{S}_i$ *is the* domain of shares *of party* $i$*.*

**Reconstruction algorithm** Reconst: *Takes as input a description $\mathcal{G}$ of an authorized set, and shares $\{s_i : i \in \mathcal{G}\}$ and outputs $s' \in \mathcal{S}$.*

*The scheme is required to satisfy the following properties:*

**Correctness:** *For every $s \in \mathcal{S}$, and every authorized set $\mathcal{G}$,*

$$\Pr\left[\mathsf{Reconst}\left(\mathcal{G}, \mathsf{Share}\left(s\right)|_{\mathcal{G}}\right) = s\right] = 1$$

*where $\mathsf{Share}\left(s\right)|_{\mathcal{G}}$ denotes the shares of the parties in the authorized set $\mathcal{G}$.*

**Secrecy:** *For any pair of secrets $s, s' \in \mathcal{S}$, and any unauthorized set $\mathcal{G}$, $\mathsf{Share}\left(s\right)|_{\mathcal{G}}$ and $\mathsf{Share}\left(s'\right)|_{\mathcal{G}}$ are statistically close.*

In our constructions, we will use Shamir's secret sharing scheme [Sha79], which we review next.

**Definition 13** (Shamir's SSS)**.** *Let $\mathbb{F}$ be a field.*

**Sharing algorithm:** *For any input $s \in \mathbb{F}$, pick a random polynomial $p(\cdot)$ of degree $t$ in the polynomial-field $\mathbb{F}[x]$ with the condition that $p(0) = s$, and output $p(1), \ldots, p(n)$.*

**Reconstruction algorithm:** *For any input $(s'_i)_{i \in S}$ where none of the $s'_i$ are $\perp$ and $|S| > t$, compute a polynomial $g(x)$ such that $g(i) = s'_i$ for every $i \in S$. This is possible using Lagrange interpolation where $g$ is given by*

$$g(x) = \sum_{i \in S} s'_i \prod_{j \in S/\{i\}} \frac{x - j}{i - j} \ .$$

*Finally the reconstruction algorithm outputs $g(0)$.*

We will actually require a stronger correctness property for SSSs which, informally, guarantees unique reconstruction for any set of (possibly ill-formed) shares. This can be thought of as a weak form of unique decoding, where we only require error detection (i.e., identifying whether or not an error occurred), and not error correction. Alternatively, this is a weaker form of verifiable secret sharing, which need only be secure against a corrupted dealer (i.e., all the share holders are assumed to be honest). Formally,

**Definition 14** (Strongly-correct SSS)**.** *We say that an $n$-party secret sharing scheme* (Share, Reconst) *for secrets in $\mathcal{S}$ is* strongly correct *if* Reconst *is deterministic, and the only authorized set is $[n]$ (the set of all parties).*

We note that for any $t < n$, $t$-out-of-$n$ Shamir's scheme (with the access structure in which the only authorized set is $[n]$) is strongly correct. For this, we assume that the shares are numbered in some arbitrary way, and reconstruction always uses the "first" $t + 1$ shares, see Remark 3.1 below.

**Remark 3.1** (Strong correctness implies unique reconstruction in threshold schemes)**.** *The strong correctness property of Definition 14 implies unique reconstruction in threshold schemes, when these are thought of as ramp secret sharing schemes which are private for sets of size at most $t$, and reconstructible for the set of all parties. Indeed, we assume without loss of generality that the shares are numbered (in some arbitrary way). Given all $n$ shares, reconstruction is performed with the first $t + 1$ shares. These $t + 1$ shares determine* some *secret, and the fact that* Reconst *is deterministic guarantees its uniqueness. In particular, we note that in this case Shamir's secret sharing scheme has unique reconstruction.*

**Remark 3.2.** *We note that for our alphabet reduction for ZK-PCPs (Construction 5, Section 5) and ZK-PCPPs (Construction 13, Section 6.2) we can make due with* any *SSS with deterministic reconstruction, by having the verifier use some arbitrary* fixed *minimal authorized set for reconstruction. Notice that if such a set can be efficiently found then the verifier in Constructions 5 and 13 will be PPT. We further note that for threshold SSSs (such as the one used in our final constructions in Theorems 1 and 2), such a minimal authorized set can be found efficiently.*

Next, we define *leakage-resilient* SSS.

**Definition 15** (Leakage-resilient SSS)**.** *We say that a secret sharing scheme* $(\mathsf{Share}, \mathsf{Reconst})$ *for* $\mathcal{S}$ *is* $\varepsilon$*-leakage resilient against a family* $F$ *of leakage functions if for every* $f \in F$*, and every pair of secrets* $s, s' \in \mathcal{S}$*,* $f(\mathsf{Share}(s))$ *and* $f(\mathsf{Share}(s'))$ *are* $\varepsilon$*-statistically close.*

We will be particularly interested in the *local probing* leakage family, which consists of all functions that, given the $n$ shares, output the shares of an unauthorized set in their entirety, as well as $\ell$ bits from each of the other shares. More specifically, we will only consider the $t+1$-threshold access structure mentioned above, in which all (and only) subsets of size $\geq t+1$ are authorized. Formally:

**Definition 16** ($(t,\ell)$-local probing leakage)**.** *Let* $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ *be the domain of shares for some secret sharing scheme. For a subset* $\mathcal{G} \subseteq [n]$ *and a sequence* $(\mathcal{I}_1, \ldots, \mathcal{I}_n)$ *of subsets of* $[n]$*, the function* $f_{\mathcal{G},\mathcal{I}_1,\ldots,\mathcal{I}_n}$ *on input* $(s_1, \ldots, s_n)$ *outputs* $s_i$ *for every* $i \in \mathcal{G}$*, and outputs* $s_i|_{\mathcal{I}_i}$ *for every* $i \notin \mathcal{G}$*. The* $(t,\ell)$-local probing function family *corresponding to* $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ *is defined as follows:*

$$F_{t,\ell} = \left\{ f_{\mathcal{G},\mathcal{I}_1,\ldots,\mathcal{I}_n} \ : \ \mathcal{G} \subseteq [n], |\mathcal{G}| \leq t, \forall i \notin \mathcal{G}, |\mathcal{I}_i| \leq \ell \right\}.$$

### 3.4 Equivocal Secret Sharing

In this section we define the notion of equivocal secret sharing, compare it to leakage-resilient secret sharing, and present a 1-party equivocal SSS based on coding. We start with the definition.

At a high level, an equivocal SSS is a leakage-resilient SSS with the additional guarantee that even after some bits are leaked from the shares, one can still "open" the secret (by providing the entire secret sharing) consistently with the previous leakage. This is formalized (in Definition 17) in the simulation-based paradigm, by comparing the real world experiment with an ideal experiment, which are described in Figure 1.

The real and ideal experiments have two phases: a leakage phase and a guessing phase. This is captures by having the adversary and simulator consist of two separate algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ and $(\mathsf{Sim}_1, \mathsf{Sim}_2)$, respectively. Leakage resilience is guaranteed against a family $F$ of leakage functions and a leakage bound $\ell$.

In the real world, the secret $s$ is secret shared into $n$ shares $\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n$. The adversary $\mathcal{A}_1$ is then given oracle access to a $\mathcal{SHARE}$ oracle, and a $\mathcal{LEAK}$ oracle. The $\mathsf{Share}$ oracle, given an index $i$, returns the $i$'th share $\mathsf{Sh}_i$. Each call to $\mathcal{SHARE}$ updates the set $T$ of secret shares which the adversary has queried so far, by adding $i$ to $T$ ($T$ is initialized to $\emptyset$). The $\mathcal{LEAK}$ oracle takes as input a function $g \in F$, which specifies, for each share $\mathsf{Sh}_i, i \notin T$, a leakage function $g_i$. It applies these leakage functions to the shares $\mathsf{Sh}_i, i \notin T$, and returns the outputs $\mathsf{output}_i$. For each such share $\mathsf{Sh}_i, i \notin T$, it also updates the counter $\ell_i$ of the number of leakage bits obtained on $\mathsf{Sh}_i$, by increasing it by $|\mathsf{output}_i|$. $T$ and $\ell_1, \ldots, \ell_n$ are treated as global parameters that can be accessed and updated by all oracles.

At the end of the first phase of the experiment (the adversary $\mathcal{A}_1$ decides when to end the first phase and move to the second phase), $\mathcal{A}_1$ outputs a bit $b_R$, which specifies whether it wishes to learn the entire secret sharing of $s$. If $b_R = 1$, i.e., the adversary chose to proceed to the second phase,

then it learns the entire secret sharing of $s$ (this is done by calling the $\mathcal{REVEAL}$ oracle). Otherwise, the adversary obtains no further information beyond what it obtained during the leakage phase.

At the second phase of the game, the adversary $\mathcal{A}_2$ outputs a guess $b'_R$ as to whether it is in the real or ideal experiments. This guess depends on the leakage in the first phase of the game, and can either depend on the secret shares of $s$ (if $b_R = 1$) or not (if $b_R = 0$). The adversarial guess is only taken into account if the adversary did not violate the leakage restrictions, i.e., only if the following two conditions are satisfied. First, the set $T$ of shares which the adversary received throughout the experiments (through $\mathcal{SHARE}$ queries) is an unauthorized set. Second, for every share $i$, the number of bits $\ell_i$ leaked from $\mathsf{Sh}_i$ (through $\mathcal{LEAK}$ queries) does not exceed the leakage bound $\ell$. These checks are performed by calling the $\mathcal{VALID}$ oracle, where if the tests fail then the adversary automatically looses the game (by setting its "guess" to 0).

The ideal experiment is similar to the real experiment, except that the $\mathcal{SHARE}, \mathcal{LEAK}$, and $\mathcal{REVEAL}$ oracles are emulated by the simulator. In particular, the simulator needs to simulate shares and leakage on shares (through the $\mathcal{SHARE}$ and $\mathcal{LEAK}$ oracles). Additionally, if $b_I = 1$ (i.e., the adversary chose to learn the entire secret sharing in the ideal experiment) then the simulator is given the secret $s$, and needs to emulate the entire secret sharing of $s$ consistently with the previous leakage (this is done in the $\mathcal{REVEAL}$ oracle).

We note that by allowing the adversary to choose *not* to receive the entire secret sharing of $s$ at the end of the first phase, we capture leakage-resilient secret sharing as a special case of equivocal secret sharing. Indeed, if the adversary chooses not to learn the secret sharing, then the simulator is only required to adaptively simulate the leakage (with no knowledge of the secret). We elaborate more on this in Remark 3.3 below.

**Definition 17** (Equivocal SSS)**.** *We say that an $n$-party secret sharing scheme* $(\mathsf{Share}, \mathsf{Reconst})$ *for secrets in $\mathcal{S}$ is $\varepsilon(n)$-*equivocal *for leakage class $F$, leakage bound $\ell$ and access structure $\mathsf{Acc}$ if for every adversary $(\mathcal{A}_1, \mathcal{A}_2)$ there exists an efficient simulator $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ and a negligible function $\varepsilon(n)$ such that for every $s \in \mathcal{S}$,*

$$\left|\Pr\left[\mathcal{REAL}_{F,\ell,\mathsf{Acc}}(s) = 1\right] - \Pr\left[\mathcal{IDEAL}_{F,\ell,\mathsf{Acc}}(s) = 1\right]\right| \leq \varepsilon(n)$$

*where $\mathcal{REAL}_{F,\ell,\mathsf{Acc}}(s), \mathcal{IDEAL}_{F,\ell,\mathsf{Acc}}(s)$ are defined in Figure 1, and the probability is over the random coin tosses of $\mathcal{SETUP}^{\mathcal{R}}$, $(\mathcal{A}_1, \mathcal{A}_2)$ and $(\mathsf{Sim}_1, \mathsf{Sim}_2)$.*

*We say that the scheme is* perfectly equivocal*, if it is $\varepsilon$-equivocal with $\varepsilon = 0$.*

**Remark 3.3** (On the connection between equivocal and LR secret sharing)**.** *We note that equivocal secret sharing captures LR secret sharing as a special case, in the following sense. If a $t$-threshold secret sharing scheme is $\varepsilon$-equivocal with leakage bound $\ell$, then it is also $2\varepsilon$-leakage resilient against $(t, \ell)$-local probing leakage. Indeed, if the $\mathcal{REVEAL}$ oracle is not called in Figure 1 (which happens when $b = 0$) then the simulator never receives the secret $s$, in which case the simulated answers to the leakage queries are required to be distributed $\varepsilon$-statistically close to the real execution. As this holds for any secret, the real leakage on the shares of two different secrets must be $2\varepsilon$-statistically close.*

### 3.4.1 Equivocation from Zero-Knowledge Codes

In this section we describe a 1-party equivocal SSS that will be used in Sections 5.1 and 6 to obtain ZK-PCPs and ZK-PCPPs with ZK against adaptive verifiers. We note that while a 1-party SSS is not useful towards sharing a secret, it does provide a meaningful notion of leakage resilient *encoding*. Related notions were considered in the past under different names, among which the

**Equivocal SSS Security**

$\underline{\mathcal{SETUP}^{\mathcal{R}}(s)}$:
pick a uniformly random string $r$ for $\mathsf{Share}$
$(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n) \leftarrow \mathsf{Share}(s; r)$
output $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$

$\underline{\mathcal{SETUP}^{\mathcal{I}}()}$:
initialize $\mathsf{St}$ to the empty string
$\mathsf{St} \leftarrow \mathsf{Sim}_1(\mathsf{St})$
output $\mathsf{St}$

$\underline{\mathcal{SHARE}^{\mathcal{R}}(s, r, i)}$:
$T_1 \leftarrow T_1 \cup \{i\}$
output $\mathsf{Sh}_i$

$\underline{\mathcal{SHARE}^{\mathcal{I}}(i)}$:
$\mathsf{Sh}_i \leftarrow \mathsf{Sim}_1(\mathsf{St}, i)$
$T_1 \leftarrow T_1 \cup \{i\}$
output $\mathsf{Sh}_i$

$\underline{\mathcal{LEAK}^{\mathcal{R}}(s, r, g, T)}$:
if $g \notin F$ then return
$(\mathsf{output}_i)_{i \notin T} \leftarrow g\left((\mathsf{Sh}_i)_{i \notin T}\right)$
$T_1 \leftarrow T_1 \cup T$
for every $i \notin T$
  $\ell_i \leftarrow \ell_i + |\mathsf{output}_i|$
output $\left((\mathsf{output}_i)_{i \notin T}, (\mathsf{Sh}_i)_{i \in T}\right)$

$\underline{\mathcal{LEAK}^{\mathcal{I}}(g, T)}$:
if $g \notin F$ then return
$\left((\mathsf{output}_i)_{i \notin T}, (\mathsf{Sh}_i)_{i \in T}, \mathsf{St}\right) \leftarrow \mathsf{Sim}(\mathsf{St}, g, T)$
$T_1 \leftarrow T_1 \cup T$
for every $i \notin T$
  $\ell_i \leftarrow \ell_i + |\mathsf{output}_i|$
output $\left((\mathsf{output}_i)_{i \notin T}, (\mathsf{Sh}_i)_{i \in T}\right)$

$\underline{\mathcal{REVEAL}^{\mathcal{R}}(s, r)}$:
output $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_n)$

$\underline{\mathcal{REVEAL}^{\mathcal{I}}(s)}$:
$\mathsf{rev} \leftarrow \mathsf{Sim}_2(\mathsf{St}, s)$
output $\mathsf{rev}$

$\underline{\mathcal{VALID}(\ell, \mathsf{Acc})}$:
if $T_1 \notin \mathsf{Acc}$ and $\ell_i \leq \ell$ for every $i \in [n]$
  then output $\mathsf{true}$
else
  output $\mathsf{false}$

$\underline{\mathcal{REAL}_{F,\ell,\mathsf{Acc}}(s)}$:
$\ell_1, \ldots, \ell_n \leftarrow 0$
$T_1 \leftarrow \emptyset$
$r \leftarrow \mathcal{SETUP}^{\mathcal{R}}(s)$
$(\mathsf{St}_{\mathcal{A}}, b_R) \leftarrow \mathcal{A}_1^{\mathcal{SHARE}^{\mathcal{R}}(s,r,\cdot), \mathcal{LEAK}^{\mathcal{R}}(s,r,\cdot,\cdot)}$
if $b_R = 1$ then
  $\left(\mathsf{Sh}'_1, \ldots, \mathsf{Sh}'_n\right) \leftarrow \mathcal{REVEAL}^{\mathcal{R}}(s, r)$
  $\mathsf{St}_{\mathcal{A}} \leftarrow \mathsf{St}_{\mathcal{A}} \circ \left(\mathsf{Sh}'_1, \ldots, \mathsf{Sh}'_n\right)$
$b'_R \leftarrow \mathcal{A}_2(\mathsf{St}_{\mathcal{A}})$
if $\mathcal{VALID}(\ell, \mathsf{Acc})$ then output $b'_R$
else output 0

$\underline{\mathcal{IDEAL}_{F,\ell,\mathsf{Acc}}(s)}$:
$\ell_1, \ldots, \ell_n \leftarrow 0$
$T_1 \leftarrow \emptyset$
$\mathsf{St} \leftarrow \mathcal{SETUP}^{\mathcal{I}}()$
$(\mathsf{St}_{\mathcal{A}}, b_I) \leftarrow \mathcal{A}_1^{\mathcal{SHARE}^{\mathcal{I}}(\cdot), \mathcal{LEAK}^{\mathcal{I}}(\cdot,\cdot)}$
if $b_I = 1$ then
  $\mathsf{output} \leftarrow \mathcal{REVEAL}^{\mathcal{I}}(s)$
  $\mathsf{St}_{\mathcal{A}} \leftarrow \mathsf{St}_{\mathcal{A}} \circ \mathsf{output}$
$b'_I \leftarrow \mathcal{A}_2(\mathsf{St}_{\mathcal{A}})$
if $\mathcal{VALID}(\ell, \mathsf{Acc})$ then output $b'_I$
else output 0

Figure 1: The Security Experiments of Equivocal SSS

following notions are most relevant for us. First, the weaker notion of ZK codes [DGR97, DGR99, ISVW13] which resist only *non-adaptive* probing leakage. Second, the notion of a Reconstructable Probabilistic Encoding (RPE) [CDMW08, BDKM16, CDMW18, BDG+18] which is an incomparable

primitive that guarantees not only leakage resilience, reconstruction and equivocation, but also error correction, and where privacy and equivocation are perfect (see Remark 3.4 below for a more detailed comparison). A 1-party SSS strengthens the first notion (i.e., ZK codes) to be adaptively-secure and equivocal. It also strengthens the second notion (RPEs) to allow for a more fine-grained leakage profile (in which symbols can *partially* leak), which might allow for a higher leakage rate. Framing our constructions in the context of SSSs (instead of LR encodings) gives a unified framework for the design of ZK-PCPs and ZK-PCPPs (in Sections 5 and 6).

Concretely, we will use the following 1-party equivocal SSS, implicit in [DGR99]:[6]

**Theorem 3** (1-party equivocal SSS, implicit in [DGR99])**.** *There exists a 1-party SSS for messages in $\{0,1\}^n$ that is perfectly-equivocal against $(0,\Omega(n))$-local probing leakage, with a share of size $O(n)$.*

**Remark 3.4** (Comparison between equivocal SSSs and RPEs.)**.** *We compare equivocal SSSs to the related notion of RPEs, first introduced by [CDMW08, CDMW18]. Informally, an RPE is a randomized encoding scheme* $(\mathsf{Encode},\mathsf{Decode})$*, associated with an additional reconstruction algorithm* $\mathsf{Reconst}$*. Encodings have standard error-correction properties, and are also zero-knowledge in the sense that for any message $x$, a small fraction of codeword symbols in a random encoding of $x$ are uniformly distributed. These two properties make an RPE similar to a robust secret sharing scheme (where each codeword symbol corresponds to a share), except that secret sharing schemes might allow decoding from a subset of codeword symbols. What makes an RPE similar to* equivocal *SSSs is its reconstruction property: for any codeword $c$, any sufficiently small fraction $S$ of codeword symbols, and any message $x$, the reconstruction algorithm* $\mathsf{Reconst}$*, given $S, c|_S$ and $x$, generates an encoding $c^x$ of $x$ that is uniformly distributed (i.e., distributed identically to* $\mathsf{Encode}(x)$*) conditioned on $c^x|_S = c|_S$.*

*Equivocal SSSs and RPEs are tightly related. In particular, in the context of secret sharing we think of sharing a secret into a sequence of shares, which correspond to the message and codeword symbols, respectively, in the context of RPEs. Moreover, the leakage resilience and equivocation properties of the secret sharing correspond to the ZK and reconstruction properties of the RPE.*

*Despite these similarities, the notions are incomparable, as we now discuss. RPEs are stronger than equivocal SSSs in 3 respects. First, they guarantee error correction (whereas equivocal SSSs are not required to be robust). Second, the ZK property guarantees that few codeword symbols are* uniformly distributed*, whereas equivocal SSSs only guarantee that these symbols are distributed identically/statistically close to the symbols in the sharing of any other secret/message. Thirdly, the reconstructed encoding is identically distributed to a random encoding of the message/secret (subject to being consistent with the known codeword symbols), whereas for equivocal SSSs the simulated secret sharing is only required to be statistically close to a random sharing of the message.*

*On the other hand, equivocal SSSs are stronger since they have a more fine-grained ZK/leakage resilience guarantee - they guarantee ZK and equivocation/reconstruction even given leakage from (potentially)* all *shares/codeword symbols. This is not allowed in RPEs, in which symbols are either revealed in their entirety, or remain completely hidden.*

*Nevertheless, there are cases in which RPEs suffice. In particular, an RPE over the binary alphabet $\{0,1\}$ gives a probing-secure 1-party equivocal SSS. One particular such example, which we will use in our ZK-PCP and ZK-PCPP constructions, was already noted in Theorem 3.*

[6] Specifically, Decatur et al. [DGR99] construct a linear code $C \subseteq \{0,1\}^n$ with constant rate and probing-resilience against a constant fraction of leaked bits (as noted above, such codes are known as ZK codes). It was observed in [BDG+18, Lemma 2] that linear ZK codes are also equivocal.

# 4 A Tighter Analysis of the ZK-PCP of [IKOS07]

In this section we extend the analysis from [IKOS07], proving that an honest verifier can obtain a negligible soundness error by querying the prover's oracle with as few as $q = \tilde{O}(\sqrt{n})$ queries rather than $O(n)$ as stated in [IKOS07], where $n = \Omega(\kappa)$. A dishonest verifier that still queries the oracle with $t = O(n)$ queries on the other hand, does not violate the privacy of the prover due to the $t$-privacy of the MPC protocol $\Pi$ which implies that $\Pi$ is resilient against $t$ semi-honest corruptions.

Let $\mathcal{R}$ be the relation corresponding to the NP-language $\mathcal{L}$. Consider a perfect $t$-robust $t$-private honest-majority $n$-party MPC protocol $\Pi$ for $f$, where $f$ is the following $(n+1)$-argument functionality corresponding to $\mathcal{R}$: $f(x, w_1, \ldots, w_n) = \mathcal{R}(x, \oplus_i w_i)$. Then we prove the following theorem,

**Theorem 4** (Non-adaptive ZK-PCP with $\sqrt{n}$-gap)**.** *Let $n \geq 3$, $\mathcal{R}$ and $f$ as above. Suppose that $\Pi$ realizes the $n$-party functionality $f$ with perfect $t$-robustness (in the malicious model) and perfect, statistical or computational $t$-privacy with simulation error $\varepsilon_{ZK}$ (in the semi-honest model), where $t = \Omega(\kappa)$ and $n = ct$ for some constant $c > 1$. Then there exists a non-adaptive $q$-query ZK-PCP over some alphabet $\Sigma$ for $q = \max\{O(\sqrt{n\kappa}), \kappa\}$, with $(t, \varepsilon_{ZK})$-ZK, soundness error $2^{-\Omega(\kappa)}$, and proofs of length $n$. Moreover, $|\Sigma| = \mathsf{poly}(m, \kappa)$, where $m$ denotes the input length. Furthermore, if $\Pi$ is private against* adaptive *adversaries, then the resultant ZK-PCP is ZK against adaptive verifiers.*

*In particular, by setting $\kappa = \log^2 n$ and using an MPC protocol with perfect adaptive privacy,*[7] *we get a non-adaptive $\log n\sqrt{n}$-query ZK-PCP over $\Sigma$ with $\Omega(n)$-ZK, $\mathsf{negl}(n)$ soundness error, and proofs of length $n$ over an alphabet of size $\mathsf{poly}(m, \log n)$.*

**Proof:**

**Oracle $\pi_x$ generation.** On input statement $x$ and witness $w$, the prover first generates input shares $w_1, \ldots, w_n$ for paries $P_1, \ldots, P_n$. It next emulates $\Pi$ on these virtual parties to construct their views $V_1, \ldots, V_n$. The proof consists of the views $V_1, \ldots, V_n$.

**Verification.** The honest verifier queries $q$ out of the $n$ symbols of $\pi_x$ and accepts if:

1. The final output within all queried views is 1.
2. For any pair of views $V_i$ and $V_j$ that the verifier queries, the views are consistent (see Definition 8).

**Analysis.** Completeness follows from the correctness of the MPC protocol. ZK against $t$-query bounded (possibly malicious) verifiers follows from the $t$-privacy of the MPC (with the same simulation error). Regarding the alphabet size, it depends on the size of the MPC views, which is polynomial in the input length $m$ and the security parameter $\kappa$.

**Soundness analysis.** Let $V_1, \ldots, V_n$ be the views of the $n$ parties within an execution of $\Pi$ as above. Define the inconsistency graph $G$ as the graph on $n$ nodes where there is an edge between node $i$ and $j$ if the views $V_i$ and $V_j$ are inconsistent. Soundness is analyzed by considering the following two cases

[7] An example of such a protocol is the variant of the BGW protocol [BGW88] presented in [CDN15, Theorem 5.2].

**Case 1: There exists a vertex cover set $B$ in $G$ of size at most $t$.** In this case, by perfect $t$-robustness we have that every view $V_i$ for $i \notin B$ must have its output as 0. The analysis of this case follows similarly to [IKOS07] where the probability of not hitting a node from $B$ is $(t/n)^q \leq c^{-q} = 2^{-\Omega(\kappa)}$ and implies that the corrupted prover can successfully convince the verifier with negligible probability.

**Case 2: The smallest vertex cover set in $G$ is of size bigger than $t$.** In this case, there must be a matching of size $> t/2$ and the soundness error is bounded by the probability that the randomly chosen subset $\Gamma$ of size $q$ misses all pairs of matched nodes.

Let $B$ be a set of size $t$ containing $t/2$ pairs of matched nodes, i.e., $|B| = t$. We first compute the probability that fewer than $\frac{qt}{2n}$ nodes are chosen from $B$. Let $X_i$ denote the event that the $i^{th}$ chosen sample falls in $B$. Then we have $E[X_i] = |B|/n = t/n$. Now, applying the Hoeffding bound with replacement.

$$\begin{aligned}\Pr\left[\sum_i X_i < qt/2n\right] &\leq \Pr\left[\left|\sum_i X_i - qt/n\right| > qt/2n\right] \\ &\leq e^{-2qt^2/4n^2} \\ &= e^{-\Omega(q)}\end{aligned}$$

where the last equality follows from setting $t = \Omega(n)$.

Conditioned on selecting at least $k = qt/2n$ nodes from $B$, the probability that the verifier misses all edges is given by $\frac{2^k\binom{t/2}{k}}{\binom{t}{k}}$.[8] Now, we have

$$\begin{aligned}\frac{2^k\binom{t/2}{k}}{\binom{t}{k}} &= \frac{2^k(t/2)!(t-k)!}{t!(t/2-k)!} \\ &= \frac{2^k(t/2)(t/2-1)\cdots(t/2-k+1)}{t(t-1)\cdots(t-k+1)}.\end{aligned}$$

Next, we apply the AM-GM inequality $(t/2-a)(t/2-k+1+a) \leq \left(\frac{t-k+1}{2}\right)^2$ for $a = 0, 1, \ldots, k/2-1$. Therefore, (assuming $k$ is even) the probability can be bounded by

$$\begin{aligned}\frac{(t-k+1)^k}{t(t-1)\cdots(t-k+1)} &= \left(\frac{t-k+1}{t}\right)\left(\frac{t-k+1}{t-1}\right)\cdots\left(\frac{t-k+1}{t-k+1}\right) \\ &< \left(\frac{t-k+1}{t-k/2}\right)^{k/2} \\ &= \left(1-\frac{k/2-1}{t-k/2}\right)^{k/2} \\ &< \left(1-\frac{k/4}{t}\right)^{k/2} \\ &\approx e^{-(k/4t)\cdot(k/2)} \\ &= e^{-k^2/8t}\end{aligned}$$

[8] Namely, there are $\binom{t/2}{k}$ ways of choosing $k$ edges among the $t/2$ edges. Then we choose either of the two vertices incident on the selected edges.

where the first inequality is obtained by only considering the first $k/2$ fractions (rounding the others upwards to 1) and bounding each of these $k/2$ fractions by $\frac{t-k+1}{t-k/2}$.

In summary, the soundness error in case 2 is at most

$$e^{-\Omega(q)}+e^{-k^2/8t} \leq^{(1)} e^{-\Omega(\kappa)}+e^{-\left(\frac{qt}{2n}\right)^2\frac{1}{8t}} = e^{-\Omega(\kappa)}+e^{-\frac{q^2t}{32n^2}} \leq^{(2)} e^{-\Omega(\kappa)}+e^{-\Omega\left(q^2/n\right)} \leq^{(3)} 2e^{-\Omega(\kappa)} = e^{-\Omega(\kappa)}$$

where the inequality denoted by (2) holds because $t = \Omega(n)$, and the inequalities denoted by (1) and (3) follow from the definition of $q$.

We conclude by noting that $e^{-\Omega(\kappa)} = \mathsf{negl}(n)$ when $\kappa = \log^2 n$, in which case $q = \log n \cdot \sqrt{n}$.

■

# 5 Alphabet Reduction for ZK-PCPs

In this section we describe a reduction for ZK-PCPs over any alphabet $\Sigma$ into a ZK-PCP over $\{0,1\}$. In particular, this reduction preserves the zero-knowledge property. We note that for standard (non-ZK) PCPs, one can easily transform a PCP over any alphabet $\Sigma$ into a PCP over $\{0,1\}$ by simply representing every symbol of $\Sigma$ as a binary string. However, this reduction *does not* preserve zero-knowledge since a malicious verifier given access to the binary proof can read "parts" of symbols of the original proof, and thus potentially violate the zero-knowledge guarantee of the underlying ZK-PCP over $\Sigma$ (which only guarantees zero-knowledge when most symbols are not accessed at all).

We begin by describing a general reduction, then instantiate it to obtain ZK-PCPs over $\{0,1\}$ with a square root gap.

**A General Transformation.** Our starting point is the trivial transformation described above, in which every proof symbol is replaced with a corresponding bit-string. As discussed above, this alone does not guarantee zero-knowledge since a malicious verifier may read parts of symbols of the original proof. The high-level idea of preventing such malicious strategies from leaking additional information is to "protect" each bit-string by secret-sharing it (equivalently, encoding it) using a *leakage-resilient* secret sharing scheme (equivalently, leakage-resilient encoding). Recall that, very roughly, a probing-resilient secret sharing scheme hides the secret from an adversary that sees several secret shares, and can probe few bits in each of the other shares. The zero-knowledge property of the new PCP system now follows from a combination of leakage-resilience and the zero-knowledge property of the original ZK-PCP. To see why, given a malicious query-bounded verifier $\mathcal{V}^*$, we partition the symbols of the original proof into two groups, based on the number of bits $\mathcal{V}^*$ reads from the secret-sharing of the bit-string representing the symbol. Since $\mathcal{V}^*$ is query-bounded, there are only few symbols from whose secret shares $\mathcal{V}^*$ can read many bits (having many such symbols would have violated the query bound). The zero-knowledge property of the original ZK-PCP system guarantees that $\mathcal{V}^*$ learns nothing about the witness even if it is given all these symbols in their entirety. For the rest of the symbols, since $\mathcal{V}^*$ reads only few bits from their secret shares, the leakage-resilience of the secret sharing scheme guarantees that the secret shared symbol remains entirely hidden. The actual analysis is slightly more involved, see the proof of Theorem 6 below for details.

We now formally describe the transformation.

**Construction 5** (Alphabet reduction for ZK-PCPs)**.** *Let $\kappa$ be a security parameter. The system $(\mathcal{P}', \mathcal{V}')$ is over alphabet $\{0,1\}$.*

**Building blocks:**

- *A PCP system* $(\mathcal{P}, \mathcal{V})$ *over alphabet* $\Sigma$ *of size* $|\Sigma| = 2^m$.
- *A strongly-correct secret sharing scheme* $(\mathsf{Share}, \mathsf{Reconst})$ *for secrets in* $\{0,1\}^m$.

**Prover algorithm.** $\mathcal{P}'$ *has input* $1^\kappa, x, w$. *It runs* $\mathcal{P}$ *with input* $1^\kappa, x, w$ *to obtain a proof* $\pi$ *over* $\Sigma$. *For every proof symbol* $\sigma$, *it uses* $\mathsf{Share}$ *to secret-share the bit-representation of* $\sigma$. *(That is, the length-*$m$ *bit representation of* $\sigma$ *is treated as the secret.) Then,* $\mathcal{P}'$ *outputs the concatenation of all secret shares. We denote the proof generated by* $\mathcal{P}'$ *by* $\pi'$.

**Verifier algorithm.** $\mathcal{V}'$ *is given input* $1^\kappa, x$ *and oracle access to* $\pi'$. *It runs* $\mathcal{V}$ *with input* $1^\kappa, x$, *and emulates the oracle* $\pi$ *for* $\mathcal{V}$ *as follows. Whenever* $\mathcal{V}$ *reads a symbol* $\sigma$ *from* $\pi$, $\mathcal{V}'$ *reads the entire secret sharing of* $\sigma$ *from* $\pi'$. *Then, it uses* $\mathsf{Reconst}$ *to recover the symbol* $\sigma$, *and provides* $\sigma$ *to* $\mathcal{V}$ *as the answer of the oracle.*

The following theorem summarizes the properties of Construction 5.

**Theorem 6** (Non-adaptive ZK-PCPs for non-adaptive verifiers)**.** *Assume Construction 5 is instantiated with:*

- *A non-adaptive* $q$*-query* $(q^*, \epsilon)$*-ZK-PCP* $(\mathcal{P}, \mathcal{V})$ *over alphabet* $\Sigma$ *for a language* $\mathcal{L}$, *with proofs of length* $N$.[9]
- *A strongly-correct* $k$*-party secret sharing scheme* $(\mathsf{Share}, \mathsf{Reconst})$ *for secrets in* $\{0,1\}^m$ *with secret shares in* $\{0,1\}^M$ *which is* $\epsilon'$*-leakage-resilient against* $(t, \ell)$*-local probing leakage.*

*Then Construction 5 is a non-adaptive* $q'$*-query* $(q^{**}, \epsilon'')$*-ZK-PCP for non-adaptive verifiers, where:*

$$q' = q \cdot M \cdot k \qquad q^{**} = (q^* + 1)(\ell + 1)(t + 1) - 1 \qquad \epsilon'' = \epsilon + \epsilon' \cdot (N - q^*).$$

*Moreover, the transformation preserves the soundness and completeness of* $(\mathcal{P}, \mathcal{V})$.

**Proof:** Completeness. The completeness of the original PCP system is preserved due to the perfect reconstruction of the secret sharing scheme, which guarantees that $\mathcal{V}'$ perfectly emulates the proof oracle for $\mathcal{V}$. The claim regarding $q'$ follows directly from the construction, noting that for strong correctness only the set of all $n$ parties is authorized.

Soundness. Let $x^* \notin \mathcal{L}$, and let $\pi^*$ be a proof oracle. We show that $\mathcal{V}'$, with oracle access to $\pi^*$, rejects $x^*$ with the same probability as $\mathcal{V}$. We divide $\pi^*$ into $N$ sections: $\pi^* = \pi^{*,1} \cdots \pi^{*,N}$, where the $i$'th section $\pi^{*,i}$ contains the bits in locations $(i-1)\,kM + 1, \ldots, ikM$. (We note that in an honestly-generated proof, the $i$'th section contains the secret shares of the $i$'th symbol in a proof generated by $\mathcal{P}$.) Let $i_1, \ldots, i_q$ denote the locations which $\mathcal{V}$ asks to query from its oracle. The strong correctness of the SSS guarantees that there exist secrets $\sigma_1, \ldots, \sigma_q$ that will be reconstructed from the secret sharings in sections $i_1, \ldots, i_q$ i.e.,

$$\forall 1 \le j \le q : \exists \sigma_j \text{ s.t. } \sigma_j = \mathsf{Reconst}\left(\pi^{*,i_j}\right)$$

Indeed, this holds because strong correctness implies unique reconstruction (see Remark 3.1), even if the secret sharings in the queried sections are ill-formed. Consequently, $\mathcal{V}$ is emulated with a proof oracle $\pi$ such that $\pi_{i_j} = \sigma_j$ for every $1 \le j \le q$. Therefore, $\mathcal{V}'$ rejects with the same probability as $\mathcal{V}$.

Zero-knowledge. We prove ZK by a double averaging argument. Let $\mathcal{V}^*$ be a (possibly malicious, possibly unbounded) non-adaptive $q^{**}$-query-bounded verifier, and we describe a simulator $\mathsf{Sim}'$ for

[9] In fact, as will be evident from the proof, it suffices that $(\mathcal{P}, \mathcal{V})$ is ZK against *non-adaptive* malicious verifiers.

$\mathcal{V}^*$, which takes as input $1^\kappa, x$. Let $\mathsf{Sim}$ denote the simulator for the underlying ZK-PCP system $(\mathcal{P}, \mathcal{V})$ (whose existence is guaranteed from the $q^*$-ZK of $(\mathcal{P}, \mathcal{V})$), let $\pi'$ denote the proof oracle of $\mathcal{V}^*$, and let $\pi = \sigma_1 \cdots \sigma_N$ denote the underlying proof oracle (over $\Sigma$) which $\mathcal{P}'$ generated by emulating $\mathcal{P}$. We partition $\pi'$ into $N$ "sections", where each section corresponds to a symbol of the original proof $\pi$, and contains the $k$ secret shares (of length $M$) of the binary representation of the symbol. That is, the $i$'th section of $\pi'$ consists of the $k$ secret shares of the bit representation of $\sigma_i$.

Notice first that there are at most $q^*$ sections such that $\mathcal{V}^*$ reads at least $(\ell+1)(t+1)$ bits from each of them. Indeed, if there were $q^*+1$ sections from which $\mathcal{V}^*$ reads at least $(\ell+1)(t+1)$ bits, then the total number of bits it queries would be at least $(q^*+1)(\ell+1)(t+1) > (q^*+1)(\ell+1)(t+1)-1 = q^{**}$. Let $\mathcal{G}$ denote the set of symbols of $\pi$ corresponding to these sections:

$$\mathcal{G} = \{\sigma_i \ : \ \mathcal{V}^* \text{ reads} \geq (\ell+1)(t+1) \text{ bits from the } i\text{th section}\}$$

and notice that $\mathcal{G}$ is well defined (and known to $\mathsf{Sim}'$) at the onset of the simulation because $\mathcal{V}^*$ is non-adaptive. If there are less than $q^*$ such symbols, $\mathsf{Sim}'$ will arbitrarily add symbols to $\mathcal{G}$ until it has exactly size $q^*$. $\mathsf{Sim}'$ runs $\mathsf{Sim}$ with input $1^\kappa, x$ to simulate the symbols in $\mathcal{G}$ (this is possible because $(\mathcal{P}, \mathcal{V})$ is $(q^*, \epsilon)$-ZK). Then, $\mathsf{Sim}'$ replaces each symbol with its length-$m$ bit representation, and secret-shares it using $\mathsf{Share}$. Finally, $\mathsf{Sim}'$ uses the secret shares to answer the queries of $\mathcal{V}^*$ to these sections of the proof.

For every symbol $\sigma_i \notin \mathcal{G}$, notice that $\mathcal{V}^*$ reads at most $(\ell+1)(t+1)-1$ bits from the $i$th section. Recall that the $i$th section consists of $k$ secret shares of length $M$. By an averaging argument there are at most $t$ shares in the $i$th section such that $\mathcal{V}^*$ reads at least $\ell+1$ bits from each of them, and these shares are treated as if the verifier knows them in full. Furthermore, since from all other shares $\mathcal{V}^*$ reads at most $\ell$ bits per share, the leakage resilience property of the secret sharing can be used. Specifically, the $(t, \ell)$-local probing leakage resilience of the secret sharing scheme guarantees that the bits which $\mathcal{V}^*$ reads from the $i$th section are $\epsilon'$-statistically close to the bits in a secret sharing of an arbitrary symbol $\alpha \in \Sigma$. Thus, $\mathsf{Sim}'$ can simulate the bits read from the section by randomly secret sharing the binary representation of $\alpha$, and providing the corresponding bits to $\mathcal{V}^*$ as the answers of the oracle.

Finally, we analyze the simulation error, using a union bound. The simulated symbols in $\mathcal{G}$ are (jointly) $\epsilon$-statistically close to the symbols in the honestly-generated (original) proof $\pi$. Consequently, so are the bits simulated in the corresponding sections (because applying a randomized function such as $\mathsf{Share}$ doesn't increase the statistical distance). Moreover, for each of the remaining $N - q^*$ symbols of the original proof, the bits simulated in the corresponding section are $\epsilon'$-statistically close to the bits in $\pi'$, by the $\epsilon'$-leakage resilience of the secret sharing scheme. This implies a simulation error of $\epsilon'' = \epsilon + \epsilon' \cdot (N - q^*)$ as stated in the theorem. ■

**A ZK-PCP with square root gap.** The following corollary obtains a $\sqrt{n}$-gap between the query complexity of the honest and malicious verifiers, where $n$ is the input length. It follows from Theorem 6 by an appropriate instantiation of the building blocks.

**Corollary 7** (ZK-PCP with $\sqrt{n}$ gap for non-adaptive verifiers)**.** *There exists a constant $c > 0$ such that there exists a non-adaptive $q$-query $q^*$-ZK-PCP against non-adaptive verifiers with $q^* = \Omega\left(n^{c+1}\right)$ and $q = \tilde{O}\left(n^{c+1/2}\right)$ and $\mathsf{negl}(n)$ soundness error, where $n$ denotes the input length.*

The proof of Corollary 7 will use the following theorem, which is implicit in [SV19], and obtained by applying their compiler to Shamir's secret sharing scheme with appropriate parameters.[10]

[10] We note that [SV19] do not consider strongly-correct secret sharing schemes, but their Shamir-based scheme is strongly-correct because Shamir's scheme is strongly correct (see discussion in Section 3.3).

**Theorem 8** (Leakage resilient secret sharing – implicit in [SV19]). *Let $t < k$ and $\ell$ be natural numbers. Then there exists a strongly-correct $k$-party secret sharing scheme for secrets in $\{0,1\}^{\ell}$, which is $\epsilon$-leakage resilient against $(t,\ell)$-local leakage, where $\epsilon = 2\left(1.1\ell+\sigma\right)\cdot 2^{-\sigma-1}$ and $\sigma$ is a security parameter. Moreover, the secret shares have length $4.1\ell+(\alpha+1)\sigma$, for some constant $\alpha$.*

We now prove Corollary 7.

**Proof of Corollary 7:** We instantiate Theorem 6 with the ZK-PCP system of Theorem 4 and the LR-SSS of Theorem 8. Specifically, Theorem 4 is instantiated with parameter $n$ (the input length) and security parameter $\log^2 n$, in which case the proof has length $n$ over an alphabet of size $N = n^{\alpha}$ for some constant $\alpha$, is $q^*$-ZK for $q^* = \Omega(n)$ (i.e., with simulation error $\epsilon = 0$), and has $\mathsf{negl}(n)$ soundness error with a non-adaptive verifier that makes $q = \log n \cdot \sqrt{n}$ queries. We instantiate Theorem 8 with $k = n$ parties, $\ell = n^{\alpha}$, $\sigma = \log^2 n$ and $t = k-1 = n-1$, in which case the resultant $n$-party scheme is $\epsilon''$-equivocal against $(n-1,\ell)$-local probing leakage for $\epsilon'' = \mathsf{negl}(n)$, with shares of length $M = O(n^{\alpha})$.

Therefore, Theorem 6 guarantees that the resultant ZK-PCP has a negligible soundness error with a non-adaptive honest verifier whose query complexity is

$$q \cdot M \cdot k = \log n\sqrt{n} \cdot O\left(n^{\alpha}\right) \cdot n = O\left(\log n \cdot n^{\alpha+3/2}\right)$$

and has $(q^{**},\epsilon'')$-ZK against *any* (possibly malicious and adaptive) verifier, where

$$q^{**} = \left(q^*+1\right)\left(\ell+1\right)\left(t+1\right)-1 = \Omega(n)\cdot n^{\alpha}\cdot n = \Omega\left(n^{\alpha+2}\right)$$

and

$$\epsilon'' = \epsilon' + \epsilon''\cdot N = 0 + \mathsf{negl}(n)\cdot n = \mathsf{negl}(n).$$

The corollary now follows by setting $c = \alpha+1$. ■

### 5.1 Upgrading to ZK Against Adaptive Verifiers

Our ZK-PCPs (Theorem 6 and Corollary 7) obtained through the alphabet reduction of Construction 5 can be verified non-adaptively, but guarantee ZK only against *non-adaptive* verifiers. Ideally, we would like a ZK-PCP which can be verified non-adaptively, but guarantees ZK even against *adaptive* malicious verifiers.

In this section, we show that when Construction 5 is instantiated with an equivocal SSS (see Definition 17) instead of a leakage-resilient SSS then the resultant ZK-PCP retains its ZK even when the malicious verifier is adaptive. Concretely, we prove the following:

**Theorem 9.** *Assume Construction 5 is instantiated with:*

- *A non-adaptive $q$-query $(q^*,\epsilon)$-ZK-PCP $(\mathcal{P},\mathcal{V})$ over alphabet $\Sigma$ for a language $\mathcal{L}$, with proofs of length $N$.*[11]
- *A strongly-correct $k$-party $\epsilon'$-equivocal (ramp) secret sharing scheme against $(t,\ell)$-local probing leakage, for secrets in $\{0,1\}^m$ with secret shares in $\{0,1\}^M$.*

[11] We stress that $(\mathcal{P},\mathcal{V})$ is non-adaptive in the sense that the *honest* verifier is non-adaptive, but ZK holds against possibly *adaptive* verifiers.

*Then Construction 5 is a non-adaptive $q'$-query $(q^{**}, \epsilon'')$-ZK-PCP, where*

$$q' = q \cdot M \cdot k \qquad q^{**} = (q^* + 1)(\ell + 1)(t + 1) - 1 \qquad \epsilon'' = \epsilon + \epsilon' \cdot N.$$

*Moreover, the transformation preserves the soundness and completeness of $(\mathcal{P}, \mathcal{V})$.*

**Proof:** Completeness, soundness, and the analysis of the query complexity $q$ are identical to the proof of Theorem 6. We proceed to prove ZK. Let $\mathcal{V}^*$ be a (possibly malicious, possibly adaptive) $q^{**}$-query bounded verifier. We can assume without loss of generality that $\mathcal{V}^*$ makes its queries one at a time. We describe a simulator $\mathsf{Sim}$ for $\mathcal{V}^*$, which uses as building blocks the simulator $\mathsf{Sim}_{\mathsf{ZK}}$ of the underlying ZK-PCP over $\Sigma$, and the simulator $\mathsf{Sim}_{\mathsf{LR}}$ of the equivocal SSS. $\mathsf{Sim}$ operates as follows:

1. $\mathsf{Sim}$ interprets its proof as consisting of $N$ sections, where the $i$'th section contains the bits in locations $(i-1)kM + 1, \ldots, ikM$. Moreover, $\mathsf{Sim}$ interprets each section as containing $k$ secret shares, where the $j$'th secret share in the $i$'th section contains the bits in locations $(i-1)km + (j-1)M + 1, \ldots, (i-1)kM + jM$. (Recall that in an honestly-generated proof, the $i$'th section corresponds to the secret shares of the $i$'th symbol in a proof generated by $\mathcal{P}$.)

2. $\mathsf{Sim}$ initializes counters $(\ell_i)_{i\in[N]}$, $(\ell_{i,j})_{i\in[N],j\in[n]}$ to 0, and bit strings $(\mathsf{Sh}_{i,j})_{i\in[N],j\in[n]}$ to be empty. Then, $\mathsf{Sim}$ initializes an execution of $\mathsf{Sim}_{\mathsf{ZK}}$ and $N$ independent executions of $\mathsf{Sim}_{\mathsf{LR}}$ (one for each section of the proof). We denote these executions by $\mathsf{Sim}^1_{\mathsf{LR}}, \ldots, \mathsf{Sim}^N_{\mathsf{LR}}$. (intuitively, $\ell_i, \ell_{i,j}$ count the number of queries made to the $i$'th section, and the $j$'th secret share in the $i$'th section, respectively; and $\mathsf{Sh}_{i,j}$ holds the (partially-determined) values of the $j$'th secret share in the sharing of the $i$'th symbol of the proof $\pi$.)

3. For every query $Q$ of $\mathcal{V}^*$:

   (a) Let $i, j$ denote the section, and secret share within the section, to which $Q$ belongs.

   (b) If the $Q$'th bit in $\mathsf{Sh}_{i,j}$ has already been determined during the simulation, then $\mathsf{Sim}$ uses this bit as the oracle answer.

   (c) Otherwise:

      i. $\mathsf{Sim}$ increases $\ell_{i,j}$ and $\ell_i$ by 1.

      ii. If $\ell_i \geq (\ell + 1)(t + 1)$ then $\mathsf{Sim}$ sends $i$ to $\mathsf{Sim}_{\mathsf{ZK}}$, and obtains a simulated symbol $\sigma$.[12] Then $\mathsf{Sim}$ provides $\sigma$ to $\mathsf{Sim}^i_{\mathsf{LR}}$, and obtains simulated secret shares $(\mathsf{Sh}^1, \ldots, \mathsf{Sh}^n)$. It overwrites $\mathsf{Sh}_{i,j}$ with $\mathsf{Sh}^j$, and answers $Q$ according to $\mathsf{Sh}_{i,j}$.

      iii. Else, if $\ell_{i,j} > \ell$, then $\mathsf{Sim}$ makes a $\mathcal{SHARE}(j)$ query to $\mathsf{Sim}^i_{\mathsf{LR}}$ and obtains a simulated secret share $\mathsf{Sh}$. It overwrites $\mathsf{Sh}_{i,j}$ with $\mathsf{Sh}$, and answers $Q$ according to $\mathsf{Sh}_{i,j}$.

      iv. Else, it sends the query $Q$ to $\mathsf{Sim}^i_{\mathsf{LR}}$ and obtains a simulated bit $b$. It writes $b$ in the appropriate location in $\mathsf{Sh}_{i,j}$, and returns $b$ as the oracle answer.

We now prove that the simulated and real views of $\mathcal{V}^*$ are $\epsilon + \epsilon' N$ statistically close, using a hybrid argument.

**$\mathcal{H}_0$:** This is the view of $\mathcal{V}^*$ in the simulation described above.

**$\mathcal{H}^0_1$:** $\mathcal{H}^0_1$ is obtained from $\mathcal{H}_0$ by replacing the simulated answers of $\mathsf{Sim}_{\mathsf{ZK}}$ to $\mathsf{Sim}$ with the actual proof symbols of $\pi$.

[12] Notice that this step uses the fact that $(\mathcal{P}, \mathcal{V})$ is ZK against possibly *adaptive* verifiers.

*We claim that $\mathcal{H}_0$ and $\mathcal{H}_1^0$ are $\epsilon$-statistically close by the $\epsilon$-ZK of $(\mathcal{P},\mathcal{V})$.*

Indeed, since $\mathsf{Sim}_{\mathsf{ZK}}$ is used to simulate the $i$'th proof symbol only when $\ell_i \geq (\ell+1)(t+1)$, and since $q^{**} = (q^*+1)(\ell+1)(t+1)-1$, $\mathsf{Sim}_{\mathsf{ZK}}$ is only used to simulate at most $q^*$ symbols, and so the (adaptive) ZK of $(\mathcal{P},\mathcal{V})$ can be used.

$\mathcal{H}_1^i, 1 \leq i \leq N$**:** $\mathcal{H}_1^i$ is obtained from $\mathcal{H}_1^{i-1}$ by replacing the simulated answers of $\mathsf{Sim}_{\mathsf{LR}}^i$ with the actual bits in the $i$'th section of the proof $\pi'$.

*We claim that $\mathcal{H}_1^{i-1}$ and $\mathcal{H}_1^i$ are $\epsilon'$-statistically close by the $\epsilon'$-equivocation of the SSS.*

To prove this claim, we first analyze how $\mathsf{Sim}$ simulates the oracle answers depending on the counters $\ell_i$ and $\ell_{i,j}$. Notice first that if $\mathcal{V}^*$ makes at least $(\ell+1)(t+1)$ queries to the $i$'th section, then when query number $(\ell+1)(t+1)$ is made then the simulation of $\mathsf{Sim}_{\mathsf{LR}}^i$ enters its second phase. Therefore, the first phase of the simulation of $\mathsf{Sim}_{\mathsf{LR}}^i$ contains at most $(\ell+1)(t+1)-1$ queries to the secret sharing in the $i$'th section of $\pi'$, meaning at most $t$ shares of that section are queried more than $\ell$ times. When the $\ell+1$ query to a share $j$ is made, it results in a $\mathcal{SHARE}(j)$ query to $\mathsf{Sim}_{\mathsf{LR}}^i$ (and the simulated share is used to answer all further queries to the $j$'th share). In summary, during the first phase of the simulation, $\mathsf{Sim}_{\mathsf{LR}}^i$ receives at most $t$ $\mathcal{SHARE}(\cdot)$ queries, and at most $\ell$ bits are probed from each other share. Therefore, the $\epsilon'$-equivocation of the SSS against $(t,\ell)$-local probing leakage can be used, and it guarantees that $\mathcal{H}_1^{i-1}$ and $\mathcal{H}_1^i$ are $\epsilon'$-statistically close.

We conclude the proof by noting that $\mathcal{H}_1^N$ is distributed identically to the real view of $\mathcal{V}^*$ (because all the simulated answers are consistent with the real proof oracle). ■

**Corollary 10** (ZK-PCP with $\sqrt{n}$ gap)**.** *There exists a constant $c>0$ such that there exists a non-adaptive $q$-query perfect $q^*$-ZK-PCP with $q^* = \Omega\left(n^{c+1}\right)$ and $q = \tilde{O}\left(n^{c+1/2}\right)$, with $\mathsf{negl}(n)$ soundness error, where $n$ denotes the input length.*

**Proof:** We instantiate Theorem 9 with the ZK-PCP system of Theorem 4 and the equivocal secret sharing scheme of Theorem 3. Specifically, Theorem 4 is instantiated with parameter $n$ and security parameter $\log^2 n$, in which case the proof has length $n$ over an alphabet of size $\log\Sigma = n^\alpha$ for some constant $\alpha$, is $q^*$-ZK for $q^* = \Omega(n)$ (i.e., with simulation error $\epsilon = 0$), and has $\mathsf{negl}(n)$ soundness error with a non-adaptive verifier that makes $q = \log n \cdot \sqrt{n}$ queries. We instantiate Theorem 3 with input length $n^\alpha$, in which case the resultant 1-party scheme is perfectly-equivocal against $(0,\ell)$-local probing leakage for $\ell = \Omega(n^\alpha)$, with shares of length $M = O(n^\alpha)$.

Therefore, Theorem 9 guarantees that the resultant ZK-PCP has a negligible soundness error with a non-adaptive honest verifier whose query complexity is

$$q \cdot M \cdot k = \log n\sqrt{n} \cdot O(n^\alpha) \cdot 1 = O\left(\log n \cdot n^{\alpha+1/2}\right)$$

and has $(q^{**},\epsilon'')$-ZK against *any* (possibly malicious and adaptive) verifier, where

$$q^{**} = (q^*+1)(\ell+1)(t+1)-1 = \Omega(n)\cdot\Omega(n^\alpha)\cdot 1 - 1 = \Omega\left(n^{\alpha+1}\right)$$

and has perfect ZK because

$$\epsilon'' = \epsilon' + \epsilon'' \cdot N = 0 + 0 \cdot n = 0.$$

The corollary now follows by setting $c = \alpha$. ■

We are now ready to prove Theorem 1.

**Proof of Theorem 1:** We repeat the proof of Corollary 10, except we instantiate the ZK-PCP system of Theorem 4 with parameter $(q^*)^\beta$, where $\beta = 1/(\alpha+1)$, and $\alpha$ is the constant specified in the proof of Corollary 10. This gives a ZK-PCP with ZK against $\Omega\left((q^*)^\beta\right)$ queries, and the soundness error is $\mathsf{negl}(q^*)$ with a verifier that makes $O\left(\log(q^*)^\beta \cdot (q^*)^{\beta/2}\right)$ queries. Moreover, the proof has length $N = (q^*)^\beta$ over alphabet $\Sigma$ with $\log\Sigma = (q^*)^{\alpha\beta}$. We instantiate the equivocal SSS of Theorem 3 with input length $(q^*)^{\alpha\beta}$, in which case the scheme is perfectly-equivocal against $(0,\ell)$-local probing leakage for $\ell = \Omega\left((q^*)^{\alpha\beta}\right)$, with a secret share of size $O\left((q^*)^{\alpha\beta}\right)$. Then by Theorem 9, the resultant scheme has perfect ZK against $\Omega\left((q^*)^{\beta(\alpha+1)}\right) = \Omega(q^*)$ queries, and has a $\mathsf{negl}(q^*)$ soundness error with an honest verifier that makes $O\left(\log q^* \cdot (q^*)^{\beta(\alpha+1/2)}\right) = \widetilde{O}\left((q^*)^{\beta(\alpha+1/2)}\right) \leq (q^*)^\epsilon$ queries, for any $\epsilon$ which is larger than $\frac{\alpha+1/2}{\alpha+1}$ (and for a sufficiently large $q^*$). ■

# 6 ZK-PCPPs with Non-Adaptive Verification

In this section we extend our techniques to also apply to PCPs *of Proximity (PCPPs)*, thus obtaining the first ZK-PCPPs that can be verified *non-adaptively*. This is obtained in two steps. First, in Section 6.1 we extend the construction of [IKOS07] to a ZK-PCPP system (over a large alphabet), by describing a variant in which the verifier is not required to read its entire input. This construction uses ideas from [IW14]. Then, in Section 6.2 we show that our alphabet reduction of Section 5 can also be applied to PCPs of proximity.

**Remark 6.1** (A clarifying note on PCPPs over large alphabets.)**.** *We note that the "large alphabet" is used solely to write the proof. That is, in all PCPP systems that we consider, the input is always a bit string. Thus, even if a PCPP is over a large alphabet, input queries are to single bits, whereas proof queries are to proof symbols (which might be represented using a long bit string).*

## 6.1 A ZK-PCPP Based on the Scheme of [IKOS07]

In this section we describe and analyze a variant of the construction of [IKOS07] which gives a ZK-PCP *of proximity*. Recall from Section 4 that [IKOS07] associate a function $f$ with an NP-relation $\mathcal{R}$, where $f(x, w_1, \ldots, w_n) = \mathcal{R}(x, \oplus w_i)$, and $f$ is then evaluated using an $n$-party MPC protocol in which the input of party $i$ is $(x, w_i)$. In the context of ZK-PCPPs, giving the input $x$ to all parties is problematic, since then the view of any single party $P_i$ determines $x$. Instead, we consider the $(m+n)$-input function $f'$ (where $m = |x|$) defined as: $f'(x_1, \ldots, x_m, w_1, \ldots, w_n) = \mathcal{R}((x_1, \ldots, x_m), \oplus w_i)$. Given this function, proof generation works as in [IKOS07], whereas verifying the proof requires also checking its consistency with $x$ (for every queried view among the first $m$ views). This construction is formalized in Figure 2.

We now show that the system of Figure 2 is a ZK-PCPP:

**Theorem 11** (Non-adaptive ZK-PCPP over a large alphabet)**.** *Let $m$ be an input length parameter, and let $\delta \in (0,1)$ be a proximity parameter. Let $n \geq 3$ such that $m = \Omega(n)$, and let $\mathcal{R}$ and $f'$ be as above. Suppose that $\Pi$ realizes the $(m+n)$-party functionality $f'$ with perfect $t$-robustness (in the malicious model) and perfect, statistical or computational $t$-privacy (in the semi-honest model), where $t = \Omega(\kappa)$ and $m + n = ct$ for some constant $c > 1$. Then there exists a non-adaptive $2q$-query ZK-PCPP over some alphabet $\Sigma$ where $q = \max\left\{\kappa \cdot \sqrt{n+m}, \frac{\kappa}{\delta} \cdot \frac{m+n}{m}\right\}$, with $t$-ZK, proximity parameter $\delta$, and soundness error $2^{-\Omega(\kappa)}$. Moreover, the proof has length $n+m$.*

**A ZK-PCPP over a Large Alphabet**

**Oracle $\pi_x$ generation.** On input $1^\kappa$, an $m$-bit statement $x$ and a witness $w$, the prover first generates witness shares $w_1, \ldots, w_n$ for paries $P_{m+1}, \ldots, P_{m+n}$. It then emulates $\Pi$ on the virtual parties $P_1, \ldots, P_{m+n}$, where the inputs of parties $P_1, \ldots, P_m$ are $x_1, \ldots, x_m$ (respectively, where $x_i$ denotes the $i$th bit of $x$), and the inputs of parties $P_{m+1}, \ldots, P_{m+n}$ are $w_1, \ldots, w_n$ (respectively), to construct their views $V_1, \ldots, V_{m+n}$. The proof $\pi_x$ consists of the views $V_1, \ldots, V_{m+n}$.

**Verification.** The verifier has input $1^\kappa, |x|$, and oracle access to $x$ and the proof $\pi_x$. The verifier randomly picks $i_1, \ldots, i_q$ and queries these symbols from $\pi_x$. In addition, for every $1 \leq j \leq q$ such that $i_j \leq m$, the verifier also queries $x_{i_j}$. Finally, the verifier accepts if:

1. The final output within all queried views is 1.
2. For any pair of views $V_i$ and $V_j$ that the verifier queries, the views are consistent (see Definition 8).
3. For every view $V_i$ such that $1 \leq i \leq m$, the input reported in $V_i$ is $x_i$.

Figure 2: A ZK-PCPP based on [IKOS07]

The following is an immediate corollary of Theorem 11 (obtained by setting $n = m$ and $\delta = 1/\sqrt{m}$):

**Corollary 12.** *Let $m$ be an input length parameter. Then for any $\delta \geq 1/\sqrt{m}$, any NP-language $\mathcal{L}$ has a non-adaptive $4\kappa\sqrt{m}$-query $\Omega(m)$-ZK-PCPP over some alphabet $\Sigma$, with proximity parameter $\delta$, soundness error $2^{-\Omega(\kappa)}$, and proofs of length $2m$ where $\log|\Sigma| = \mathsf{poly}(m)$.*

We now prove Theorem 11.

**Proof: Completeness.** Follows directly from the perfect robustness of the underlying MPC protocol.

**$t$-ZK.** The $t$-privacy of $\Pi$ guarantees that the verifier can query $t' \leq t$ views without learning anything except the inputs and outputs of the corrupted parties. The output in this case is 1 (because the prover is honest and $x \in \mathcal{L}$). The inputs are either secret shares of $w$, which reveal no information about $w$ because $t' \leq t < n$, or bits of $x$, where each view contains a single secret share or a single bit of $x$. In particular, all these views can be (perfectly, statistically or computationally, depending on the quality of privacy guaranteed by $\Pi$) simulated given the $t' \leq t$ bits of $x$ contained in those views. Since the ZK simulator is allowed to query $t'$ bits of $x$, it will be able to simulate the oracle answers for the (possibly malicious) verifier.

**Soundness.** Let $x$ be $\delta$-far from $L$, and let $\pi^*$ be a (possibly ill-formed) proof oracle. Notice that the views $V_1, \ldots, V_m$ reported in $\pi^*$ determine an effective input $x^* = (x_1^*, \ldots, x_m^*)$. We consider two cases.

First, if $x^*$ is $\delta$-far from $x$, then let $\mathcal{I} = \{i \ : \ x_i \neq x_i^*\}$, so $|\mathcal{I}| \geq \delta m$. Notice that if the verifier queries a view $V_i$ for $i \in \mathcal{I}$ then it rejects (with probability 1). Therefore, it suffices to bound the probability that the verifier "misses" all of $\mathcal{I}$.

$$\Pr[\text{Verifier misses } \mathcal{I}] = (\Pr[\text{Single verifier query misses } \mathcal{I}])^q \leq \left(\frac{n+m-\delta m}{n+m}\right)^q = \left(1 - \delta \cdot \frac{m}{n+m}\right)^q.$$

Since $m = \Omega(n)$, there exists a constant $c > 0$ such that $m \geq c(m+n)$. Therefore, when $q \geq \frac{\kappa}{c\delta}$ then

$$\left(1 - \delta \cdot \frac{m}{n+m}\right)^q \leq (1 - c\delta)^q = (1 - c\delta)^{\frac{1}{c\delta} \cdot \kappa} \to e^{-\kappa}$$

in particular, for a large enough $\kappa$ this value is less than $2^{-\kappa}$, in which case the verifier rejects except with probability $2^{-\kappa}$.

Second, assume that $x^*$ is not $\delta$-far from $x$. In this case, $x^* \notin L$. Therefore, for *any* $w_1^*, \ldots, w_n^*$, $f'(x_1^*, \ldots, x_m^*, w_1^*, \ldots, w_n^*) = 0$, and we show that the analysis from the proof of Theorem 4 still holds. That is, in this case the first two checks which the verifier performs (checking the output, and checking consistency between pairs of views) already guarantees that the verifier will reject with high probability. We proceed with the formal argument.

Consider the inconsistency graph $G$ defined in the proof of Theorem 4, and we consider two cases. First, assume $G$ has a vertex cover $B$ of size $\leq t$. In this case, for every $i \notin B$, the output reported in $V_i$ is 0 (this follows from the perfect robustness of $\Pi$). Therefore, the verifier accepts only if all its queries are to nodes in $B$, which happens with probability $\left(\frac{t}{n+m}\right)^q = 2^{-\Omega(q)}$ where the equality holds because $t = \Omega(n+m)$. In particular, this probability is $2^{-\Omega(\kappa)}$ whenever $q = \Omega(\kappa)$.

Second, assume the minimal vertex cover of $G$ has size $> t$, in which case $G$ has a matching of size $> t/2$. Let $B$ be a set of $t/2$ pairs of nodes partaking in this matching, and notice that the verifier accepts only if it misses all the edges between nodes in $B$. Noticing that we now have $m+n$ nodes in total (whereas in Theorem 4 we only had $n$ nodes), and that $t = \Omega(n+m)$, a similar analysis to that provided in the proof of Theorem 4 shows that except with probability $2^{-\Omega(q)}$, the verifier queries at least $k = \frac{qt}{2(n+m)}$ nodes from $B$. Conditioning on the event that the verifier queried at least $k$ nodes from $B$, the analysis in the proof of Theorem 4 shows that the verifier reveals an edge of $B$, except with probability $e^{-k^2/8t} = e^{-\Omega\left(q^2/(n+m)\right)}$, where the equality holds because $t = \Omega(n+m)$. In particular, this probability is $2^{-\Omega(\kappa)}$ whenever $q = \Omega\left(\kappa \cdot \sqrt{n+m}\right)$. ■

### 6.2 Alphabet Reduction for ZK-PCPP

In this section we show that the alphabet reduction of Construction 5 (Section 5) can also be applied to ZK-PCPs *of proximity*. Combining this with the non-adaptive ZK-PCPP over large alphabets (Theorem 11), we obtain *the first* ZK-PCPP with a *non-adaptive* honest verifier.

We first describe a slight variant of the alphabet reduction of Construction 5 which works for ZK-PCPPs.

**Construction 13** (Alphabet reduction for ZK-PCPPs)**.** *The alphabet reduction for ZK-PCPPs is similar to the alphabet reduction of Construction 5, with the following modifications: $\mathcal{V}'$ has input $1^\kappa, |x|$ and oracle access to $x, \pi'$. It runs $\mathcal{V}$ with input $1^\kappa$ and gives it access to its own oracle $x$. Queries of $\mathcal{V}$ to its proof oracle are emulated as in Construction 5.*

Next, we show that Construction 13 preserves ZK against non-adaptive malicious verifiers.

**Theorem 14** (Non-adaptive ZK-PCPPs against non-adaptive malicious verifiers)**.** *Assume there exist:*

- *A non-adaptive $q$-query $(q^*, \epsilon)$-ZK-PCPP $(\mathcal{P}, \mathcal{V})$ over alphabet $\Sigma$ for a language $\mathcal{L}$, with proximity parameter $\delta$, and proofs of length $N$.*[13]
- *A strongly-correct $k$-party secret sharing scheme $(\mathsf{Share}, \mathsf{Reconst})$ for secrets in $\{0,1\}^m$ with secret shares in $\{0,1\}^M$ which is $\epsilon'$-leakage-resilient against $(t, \ell)$-local probing leakage.*

*Then $\mathcal{L}$ has a non-adaptive $q'$-query $(q^{**}, \epsilon'')$-ZK-PCPP for non-adaptive verifiers with proximity parameter $\delta$, where:*

$$q' = q \cdot M \cdot k \qquad q^{**} = (q^* + 1)(\ell + 1)(t + 1) - 1 \qquad \epsilon'' = \epsilon + \epsilon' \cdot (N - q^*).$$

[13] In fact, as will be evident from the proof, it suffices that $(\mathcal{P}, \mathcal{V})$ is ZK against *non-adaptive* malicious verifiers.

*Moreover, the transformation preserves the soundness and completeness of* $(\mathcal{P},\mathcal{V})$*. Furthermore, the resultant ZK-PCPP has the following feature: the view of any* $q^{**}$*-query bounded (possibly malicious) verifier* $\mathcal{V}^*$ *that makes* $q''$ *input queries can be simulated with only* $q''+q^*$ *queries to the input.*

**Proof (sketch):** We apply the alphabet reduction of Construction 13 to $(\mathcal{P},\mathcal{V})$, and show that the resultant system has the desired properties. The proof is very similar to the proof of Theorem 4, we focus on describing the differences.

Completeness follows identically to the proof of Theorem 6, as does the analysis for $q'$. We note that the query complexity of $\mathcal{V}'$ might actually be smaller than $q'$, because some of the queries of $\mathcal{V}$ might be to $x$, in which case these do not cause additional queries in the reduction.

Soundness follows similarly to the proof of Theorem 6. The only difference is that soundness need only hold for $x$ which is $\delta$-far from $\mathcal{L}$, in which case one can use the soundness of the underlying ZK-PCPP over $\Sigma$.

Zero-knowledge follows similarly to the proof of Theorem 6. The only difference is in how the input is treated. Specifically, the simulator $\mathsf{Sim}'$ is now given $1^\kappa, |x|$ as input, and has *oracle access* to $x$. It answers queries of $\mathcal{V}^*$ to the input oracle using its own input oracle. As for queries to the proof, $\mathsf{Sim}'$ identifies the $\leq q^*$ sections of the proof to which $\mathcal{V}^*$ makes at least $(\ell+1)(t+1)$ queries, and uses the simulator $\mathsf{Sim}$ of $(\mathcal{P},\mathcal{V})$ to simulate the symbols corresponding to these sections. (If there are $< q^*$ such sections then $\mathsf{Sim}'$ arbitrarily adds sections to get exactly $q^*$ such sections.) During its simulation, $\mathsf{Sim}$ sends $\leq q^*$ queries to $x$, which $\mathsf{Sim}'$ answers using its own oracle $x$. Given the simulated symbols generated by $\mathsf{Sim}$, $\mathsf{Sim}'$ answers the queries of $\mathcal{V}'$ as in the proof of Theorem 6. The rest of the analysis follows identically to the proof of Theorem 6.

Finally, we note that $\mathsf{Sim}'$ queries its input oracle in one of two cases: either when $\mathcal{V}^*$ queries his oracle, or when $\mathsf{Sim}$ does. The first case occurs $q''$ times (by the assumptions of the theorem), whereas the second case incurs at most $q^*$ queries (because $\mathsf{Sim}$ is asked to simulate at most $q^*$ queries), so in total $\mathsf{Sim}'$ makes at most $q''+q^*$ input queries. ■

**A ZK-PCPP with square root gap.** The following corollary obtains a $\sqrt{n}$-gap between the query complexity of the honest and malicious verifiers (where $n$ is the input length), and follows from Theorem 14 by an appropriate instantiation of the building blocks.

**Corollary 15** (ZK-PCPP with $\sqrt{n}$ gap for non-adaptive verifiers)**.** *Let* $n\in\mathbb{N}$ *be an input length parameter. Then there exists a constant* $c>0$ *such that for any proximity parameter* $\delta\geq 1/\sqrt{n}$*, there exists a non-adaptive* $q$*-query* $q^*$*-ZK-PCPP against non-adaptive verifiers with proximity parameter* $\delta$*,* $q^*=\Omega\left(n^{c+1}\right)$*, and* $q=\tilde{O}\left(n^{c+1/2}\right)$*.*

**Proof:** We instantiate Theorem 14 with the ZK-PCPP system of Corollary 12 and the LR-SSS of Theorem 8. Specifically, Corollary 12 is instantiated with input length $n$, proximity parameter $\delta$ and security parameter $\log^2 n$, in which case the proof has length $2n$ over an alphabet of size $N=n^\alpha$ for some constant $\alpha$, is $q^*$-ZK for $q^*=\Omega(n)$ (i.e., with simulation error $\epsilon=0$), and has $\mathsf{negl}(n)$ soundness error with proximity parameter $\delta$ and a non-adaptive verifier that makes $q=O\left(\log^2 n\cdot\sqrt{n}\right)$ queries. We instantiate Theorem 8 with $k=n$ parties, $\ell=n^\alpha$, $\sigma=\log^2 n$ and $t=k-1=n-1$, in which case the resultant $n$-party scheme is $\epsilon''$-equivocal against $(n-1,\ell)$-local probing leakage for $\epsilon''=\mathsf{negl}(n)$, with shares of length $M=O\left(n^\alpha\right)$.

Therefore, Theorem 14 guarantees that the resultant ZK-PCPP has a negligible soundness error with a non-adaptive honest verifier whose query complexity is

$$q\cdot M\cdot k=O\left(\log^2 n\sqrt{n}\right)\cdot O\left(n^\alpha\right)\cdot n=O\left(\log^2 n\cdot n^{\alpha+3/2}\right)$$

and has $(q^{**}, \epsilon'')$-ZK against *any* (possibly malicious and adaptive) verifier, where

$$q^{**} = (q^* + 1)(\ell + 1)(t + 1) - 1 = \Omega(n) \cdot n^{\alpha} \cdot n = \Omega\left(n^{\alpha+2}\right)$$

and

$$\epsilon'' = \epsilon' + \epsilon'' \cdot (N - q^*) = 0 + \mathsf{negl}(n) \cdot O(n) = \mathsf{negl}(n).$$

The corollary now follows by setting $c = \alpha + 1$. ■

Next, we show that when Construction 13 is instantiated with an *equivocal* SSS then the resultant ZK-PCPP is ZK against *adaptive* malicious verifiers:

**Theorem 16** (Non-adaptive ZK-PCPPs)**.** *Assume there exist:*

- *A non-adaptive $q$-query $(q^*, \epsilon)$-ZK-PCPP $(\mathcal{P}, \mathcal{V})$ over alphabet $\Sigma$ for a language $\mathcal{L}$, with proximity parameter $\delta$, and proofs of length $N$.*[14]
- *A strongly-correct $k$-party $\epsilon'$-equivocal secret sharing scheme against $(t, \ell)$-local probing leakage, for secrets in $\{0,1\}^m$ with secret shares in $\{0,1\}^M$.*

*Then $\mathcal{L}$ has a non-adaptive $q'$-query $(q^{**}, \epsilon'')$-ZK-PCPP with proximity parameter $\delta$, where:*

$$q' = q \cdot M \cdot k \qquad q^{**} = (q^* + 1)(\ell + 1)(t + 1) - 1 \qquad \epsilon'' = \epsilon + \epsilon' \cdot N.$$

*Moreover, the resultant ZK-PCPP has the same soundness and completeness guarantees as the ZK-PCPP over $\Sigma$, as well as the following feature: the view of any $q^{**}$-query bounded (possibly malicious) verifier $\mathcal{V}^*$ that makes $q''$ input queries can be simulated with only $q'' + q^*$ queries to the input.*

**Proof (sketch):** Completeness, soundness, and the calculations regarding the query complexity $q'$ are identical to the proof of Theorem 14. The proof of the ZK property is very similar to that of Theorem 9, so we only sketch the differences. Specifically, we describe a simulator $\mathsf{Sim}_a$ which operates identically to the simulator $\mathsf{Sim}$ from the proof of Theorem 9, except for the following modifications to Step 3:

1. If the query $Q$ is to the input $x$, then $\mathsf{Sim}_a$ uses its input oracle to answer it. Otherwise, $\mathsf{Sim}_a$ computes $i, j$ as $\mathsf{Sim}$ does.
2. In Step 3(c)ii, $\mathsf{Sim}_{\mathsf{ZK}}$ may query its input oracle during the simulation, and $\mathsf{Sim}_a$ answers such queries using its own input oracle.

The hybrids are defined identically to the proof of Theorem 9, and the indistinguishability proof is similar, where the claim regarding the statistical distance of $\mathcal{H}_0, \mathcal{H}_1^0$ uses the fact that $\mathsf{Sim}_a$ perfectly simulates the input oracle of $\mathsf{Sim}$.

The analysis regarding the number of input queries which $\mathsf{Sim}_a$ makes is identical to the proof of Theorem 14. ■

We are now ready to prove Theorem 2.

[14] We stress that $(\mathcal{P}, \mathcal{V})$ is non-adaptive in the sense that the *honest* verifier is non-adaptive, but ZK holds against possibly *adaptive* verifiers.

**Proof of Theorem 2:** We instantiate Theorem 16 with the ZK-PCPP system of Corollary 12 and the equivocal secret sharing scheme of Theorem 3. Specifically, Corollary 12 is instantiated with input length $n$, proximity parameter $\delta$ and security parameter $\log^2 n$, in which case the proof has length $2n$ over an alphabet of size $\log \Sigma = n^\alpha$ for some constant $\alpha$, is $q^*$-ZK for $q^* = \Omega(n)$ (i.e., with simulation error $\epsilon = 0$), and has $\mathsf{negl}(n)$ soundness error with proximity parameter $\delta$ and a non-adaptive verifier that makes $q = O\left(\log^2 n \cdot \sqrt{n}\right)$ queries. We instantiate Theorem 3 with input length $n^\alpha$, in which case the resultant 1-party scheme is perfectly-equivocal against $(0, \ell)$-local probing leakage for $\ell = \Omega(n^\alpha)$, with shares of length $M = O(n^\alpha)$.

Therefore, Theorem 16 guarantees that the resultant ZK-PCPP has a negligible soundness error with a non-adaptive honest verifier whose query complexity is

$$q \cdot M \cdot k = O\left(\log^2 n\sqrt{n}\right) \cdot O\left(n^\alpha\right) \cdot 1 = O\left(\log^2 n \cdot n^{\alpha+1/2}\right)$$

and has perfect $q^{**}$-ZK against *any* (possibly malicious and adaptive) verifier, where

$$q^{**} = (q^* + 1)(\ell + 1)(t + 1) - 1 = \Omega(n) \cdot \Omega\left(n^\alpha\right) \cdot 1 - 1 = \Omega\left(n^{\alpha+1}\right)$$

since

$$\epsilon'' = \epsilon' + \epsilon'' \cdot N = 0 + 0 \cdot n = 0.$$

The theorem now follows by setting $c = \alpha$. ■

## Acknowledgments

We thank the anonymous ITC'21 reviewers for their helpful comments, in particular for pointing out the connection to RPEs and noting that the ZK code of [DGR99, Theorem 2.2] is equivocal.

The first and third authors are supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. The first author is supported by ISF grant No. 1316/18. The first and second authors are supported by DARPA under Contract No. HR001120C0087. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

## References

[ADN+19] Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João L. Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In *CRYPTO, Proceedings, Part II*, pages 510–539, 2019.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *FOCS, Proceedings*, pages 14–23, 1992.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *FOCS, Proceedings*, pages 2–13, 1992.

[BDG+18] Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In *FOCS*, pages 826–837, 2018.

[BDIR18] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In *CRYPTO, Proceedings*, pages 531–561, 2018.

[BDKM16] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In *EUROCRYPT*, pages 881–908, 2016.

[BGH+04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *STOC, Proceedings*, pages 1–10, 2004.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[CDD+01] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *EUROCRYPT, Proceedings*, pages 262–279, 2001.

[CDMW08] 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 *TCC*, pages 427–444, 2008.

[CDMW18] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. A black-box construction of non-malleable encryption from semantically secure encryption. *J. Cryptol.*, 31(1):172–201, 2018.

[CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[DDV10] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *SCN, Proceedings*, pages 121–137, 2010.

[DGR97] Scott E. Decatur, Oded Goldreich, and Dana Ron. A probabilistic error-correcting scheme. *IACR Cryptol. ePrint Arch.*, 1997:5, 1997.

[DGR99] Scott E. Decatur, Oded Goldreich, and Dana Ron. Computational sample complexity. *SIAM J. Comput.*, 29(3):854–879, 1999.

[Din06] Irit Dinur. The PCP theorem by gap amplification. In *STOC, Proceedings*, pages 241–250, 2006.

[DP07] Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS, Proceedings*, pages 227–237, 2007.

[DR04] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP-theorem. In *FOCS, Proceedings*, pages 155–164, 2004.

[GK18] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *STOC, Proceedings*, pages 685–698, 2018.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC, Proceedings*, pages 291–304, 1985.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC, Proceedings*, pages 21–30, 2007.

[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 *TCC, Proceedings*, pages 151–168, 2012.

[ISVW13] Yuval Ishai, Amit Sahai, Michael Viderman, and Mor Weiss. Zero knowledge LTCs and their applications. In *RANDOM, Proceedings*, pages 607–622, 2013.

[IW14] Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In *TCC, Proceedings*, pages 121–145, 2014.

[IWY16] Yuval Ishai, Mor Weiss, and Guang Yang. Making the best of a leaky situation: Zero-knowledge PCPs from leakage-resilient circuits. In *TCC, Proceedings*, pages 3–32, 2016.

[KPT97] Joe Kilian, Erez Petrank, and Gábor Tardos. Probabilistically checkable proofs with zero knowledge. In *STOC, Proceedings*, pages 496–505, 1997.

[Mie09] Thilo Mie. Short PCPPs verifiable in polylogarithmic time with $O(1)$ queries. *Ann. Math. Artif. Intell.*, 56(3-4):313–338, 2009.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[SV19] Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. In *CRYPTO, Proceedings*, pages 480–509, 2019.

[Wei16] Mor Weiss. Secure computation and probabilistic checking. *PhD Thesis*, 2016.