Public-Coin Statistical Zero-Knowledge Batch Verification against Malicious Verifiers

Inbar Kaslasi[*] Ron D. Rothblum[*] Prashant Nalini Vasudevan[†]

March 1, 2021

Abstract

Suppose that a problem Π has a statistical zero-knowledge (SZK) proof with communication complexity m. The question of batch verification for SZK asks whether one can prove that k instances $x_1, \dots, x_k$ all belong to Π with a statistical zero-knowledge proof whose communication complexity is better than $k \cdot m$ (which is the complexity of the trivial solution of executing the original protocol independently on each input).

In a recent work, Kaslasi *et al.* (TCC, 2020) constructed such a batch verification protocol for any problem having a *non-interactive* SZK (NISZK) proof-system. Two drawbacks of their result are that their protocol is private-coin and is only zero-knowledge with respect to the honest verifier.

In this work, we eliminate these two drawbacks by constructing a public-coin malicious-verifier SZK protocol for batch verification of NISZK. Similarly to the aforementioned prior work, the communication complexity of our protocol is $\big(k + \mathsf{poly}(m)\big) \cdot \mathrm{polylog}(k, m)$.

[*]Technion. {inbark,rothblum}@cs.technion.ac.il.
[†]UC Berkeley. {prashvas}@berkeley.edu.

Contents

1 Introduction

The concept of zero knowledge proofs, introduced by Goldwasser, Micali and Rackoff [GMR89], is an incredibly deep and fascinating notion that has proven to be a fundamental component in the construction and design of cryptographic protocols (see, e.g., [GMW87]). A zero-knowledge proof allows a prover to convince an efficient verifier that a given statement is true without revealing anything else to the verifier. This is formalized by requiring that for any (possibly malicious) verifier that participates in such a proof, there is an efficient simulation algorithm that simulates its interaction with the prover.

In this work we focus on *statistical zero-knowledge proofs*. In this variant, both the verifier and the prover are guaranteed information-theoretic (rather than computational) security. On the one hand, the verifier knows that even an all-powerful prover could not convince it to accept a false statement (other than with negligible probability). On the other hand, the prover knows that any polynomial-time cheating strategy of the verifier can only reveal a negligible amount of information beyond the validity of the statement being proven.

The class of languages having a statistical zero-knowledge protocol is denoted by SZK. This class contains several natural problems like Graph Nonisomorphism, and many of the problems that are central to cryptography such as quadratic residuosity [GMR89], discrete logarithm [GK93, CP92], and various lattice problems [GG00, MV03, PV08, APS18]. It has been found to possess extremely rich structure [For89, AH91, Oka00, SV03, GSV98, GV99, NV06, OV08] and to have fundamental connections to different aspects of cryptography [BL13, KMN+14, LV16, Ost91, OW93, BDRV18, KY18, BBD+20] and complexity theory [For87, AH91, Aar12, GR14, Dru15, AV19, BCH+20].

In a recent work, Kaslasi *et al.* [KRR+20] raised the question of *batch verification for statistical zero-knowledge proofs*: Suppose Π has a statistical zero-knowledge proof (SZK). Can we prove that $x_1, \dots, x_k \in \Pi$ with communication complexity that beats the naive approach of separately proving that each $x_i \in \Pi$, while still maintaining zero-knowledge? Beyond being of intrinsic interest and teaching us about the structure of SZK, such protocols have potential cryptographic applications such as the batch verification of cryptographic signatures [NMVR94, BGR98, CHP12] or well-formedness of public-keys [GMR98].

The main result of [KRR+20] was such a generic batch verification result for a subclass of languages in SZK – specifically for problems having a *non-interactive* statistical zero-knowledge proof system (NISZK). Kaslasi *et al.* construct an (interactive) SZK protocol for batching k instances of $\Pi \in \mathsf{NISZK}$, with communication complexity $\big(k + \mathsf{poly}(n)\big) \cdot \mathrm{polylog}(k, n)$, where n is the length of each of the k inputs, and poly refers to a fixed polynomial that depends only on the specific problem (and not on k). Their result should be contrasted with the naive approach of simply executing the NISZK protocol separately on each input (which has communication complexity $k \cdot \mathsf{poly}(n)$).

Two major drawbacks of the protocol of [KRR+20] are the fact that it is *private-coin* and only *honest-verifier* statistical zero-knowledge (HVSZK). These drawbacks are significant. Recall that private-coin protocols can only be verified by a designated verifier, in contrast to *public-coin* protocols that can be verified by anyone (as long as they can ensure that the coins were truly unpredicatable to the prover, e.g., they were generated by some physical phenomenon or a public randomness beacon). Further, public-coin protocols have the added benefit that they can be transformed into *non-interactive* arguments via the Fiat-Shamir transform (either heuristically [FS86], in the random-oracle model [PS96], or, more recently, under concrete cryptographic assumptions (see, e.g., [CCH+19])).

The second drawback is arguably even more significant. Recall that honest-verifier zero-knowledge is a relatively weak privacy guarantee which, in a nutshell, only guarantees that verifiers that follow the protocol to the letter learn nothing in the interaction. Usually this weak privacy guarantee is only used as a stepping stone towards getting full-fledged zero-knowledge (i.e., zero-knowledge that holds even against arbitrary polynomial-time cheating verifiers).

At first glance it may seem straightforward to overcome both drawbacks of the protocol of [KRR+20] by employing the known generic transformations from *private-coin honest-verifier statistical zero-knowledge* to *public-coin malicious-verifier statistical zero-knoweldge* [Oka00, GSV98, HRV18]. Unfortunately, these tranformations incur a large polynomial overhead in communication that we cannot afford in our context (see also Remark 1.4 below).

1.1 Our Results

In this paper we eliminate the two major drawbacks mentioned above by constructing a *public-coin malicious-verifier* SZK batch verification protocol for every problem in NISZK. The communication complexity in our protocol is similar to that of [KRR+20].

Theorem 1.1 (Informally Stated, see Theorem 7.2)**.** *Let* $\Pi \in \mathsf{NISZK}$*. There exists a public-coin* SZK *protocol for verifying that* $x_1, \dots, x_k \in \Pi$*, with communication complexity* $\big(k + \mathsf{poly}(n)\big) \cdot \mathrm{polylog}(n, k)$*. The verifier's running time is* $k \cdot \mathsf{poly}(n, \log k)$*, and the number of rounds is* $k \cdot \mathrm{polylog}(n, k)$*.*

Our high-level approach for proving Theorem 1.1 follows a classical approach for constructing malicious-verifier zero-knowledge proofs: first construct a public-coin honest-verifier zero-knowledge batching protocol, and then show how to transform it to be *malicious-verifier* zero-knowledge. The main challenge that we must overcome is in actually implementing these two steps without incurring the exorbitant price of the generic transformations for SZK [Oka00, SV03, GV99, GSV98, HRV18]. Thus, our two main steps are:

1. Construct an efficient *public-coin* HVSZK batch verification protocol.

2. Transform it to be zero-knowledge against malicious verifiers, while preserving its efficiency.

Our first main technical contribution is in implementing Step 1.

Theorem 1.2 (Informally Stated, see Theorem 7.1)**.** *Let* $\Pi \in \mathsf{NISZK}$*. There exists a public-coin* HVSZK *protocol for verifying that* $x_1, \dots, x_k \in \Pi$*, with communication complexity* $\big(k + \mathsf{poly}(n)\big) \cdot \mathrm{polylog}(n, k)$*. The verifier's running time is* $k \cdot \mathsf{poly}(n, \log k)$*, and the number of rounds is* $O(k)$*.*

Theorem 1.2 already improves on the main result of [KRR+20], since it gives a *public-coin* batch verification protocol. However, we would like to go further and obtain security even against malicious verifiers. It is tempting at this point to apply the generic transformations of [GSV98, HRV18] from public-coin *honest-verifier* zero-knowledge, to *malicious-verifier*. Unfortunately, the overhead introduced in these transformations is too large and applying them to the protocol of Theorem 1.1 would yield a trivial result (see Remark 1.4 for details).

Rather, as our second technical contribution (which may be of independent interest), we show that the communication complexity of the [GSV98] transformation can be significantly improved for protocols satisfying a strong notion of soundness. Specifically, we refer to the notion of *round-by-round soundness*, introduced in a recent work of Canetti *et al.* [CCH+19].

Theorem 1.3 (Informally Stated, see Theorem 6.1)**.** *Any public-coin* HVSZK *protocol with negligible round-by-round soundness error can be efficiently transformed into a public-coin* SZK *protocol. In particular, a message of length ℓ in the original protocol grows to length* $\mathsf{poly}(\ell)$ *in the transformed protocol.*

Note that the growth of each message in the transformation above depends only on its own length and not on n or k – this allows us to take advantage of the fact that all but one of the messages in the protocol of Theorem 1.2 have poly-logarithmic length. We show that the protocol of Theorem 1.2 indeed has round-by-round soundness which, in combination with Theorem 1.3, yields Theorem 1.1 .

Remark 1.4 (On Generic Transformations from the Literature)**.** *We discuss here a few known generic transformations for the class* SZK *from the literature, and why they are not applicable in our context.*

Okamoto [Oka00] showed how to transform any private-coin HVSZK *protocol into a public-coin one. Unfortunately, we cannot use his transformation to derive Theorem 1.1 from the private-coin batching protocol of [KRR+20], due to the overhead involved. In more detail, Okamoto's protocol starts by taking a t-fold parallel repetition of the private-coin protocol, where $t = \ell^9 \cdot r^9$, where ℓ is the round complexity and r is the randomness complexity of the simulator. In our context $\ell = k$ and $r =$* $\mathsf{poly}(n)$ *and so the overhead from Okamoto's transformation would yield a trivial result (as a matter of fact, we could not even afford an overhead of $t = \ell$, which seems inherent to Okamoto's approach).*

Similarly, we cannot derive Theorem 1.1 from Theorem 1.2 by applying the generic transformation of [GSV98, HRV18] for transforming honest-verifier public-coin SZK *proofs to malicious-verifier ones. In more detail, the transformation of [GSV98] starts by applying an ℓ-fold parallel repetition of the honest-verifier protocol (where again ℓ is the number of rounds). In the context of Theorem 1.2, $\ell = \Theta(k)$, and so, applying the [GSV98] transformation yields a protocol with communication complexity $k \cdot \mathsf{poly}(n)$, which we cannot afford.*

The more recent work of Hubácek et al. [HRV18] gives an efficiency-preserving transformation from honest-verifier to malicious-verifier. This transformation also incurs a polynomial overhead in the communication complexity. In particular, [HRV18] rely on the instance-dependent commitments of Ong and Vadhan [OV08], which in turn use the SZK *completeness of the Entropy Difference (or* ED*) problem [GV99]. The known reduction from* SZK *to* ED *(see [Vad99, Theorem 3.3.13]) generates circuits whose input size is roughly $\ell \cdot r$. This size would correspond to the size of the decommitments in the [HRV18] protocol and again would lead to an overhead that we cannot incur.*

Indeed, our work motivates the study of communication-preserving *transformations for* SZK *protocols. In particular, obtaining a* generic *communication-preserving transformation from honest-verifier to malicious-verifier* SZK *is an interesting open question.*

1.2 Technical Overview

First, in Section 1.2.1, we describe the public-coin honest-verifier statistical zero-knowledge (HVSZK) batching protocol. Then, in Section 1.2.2, we show how to compile honest-verifier protocols *efficiently* to be secure against malicious verifiers.

1.2.1 Public-coin HVSZK Batching

Our starting point is the aforementioned recent work of Kaslasi *et al.* [KRR+20], which gave a *private-coin* HVSZK batching protocol. As their first step, they introduced a new (promise) problem called *approximate injectivety* (AI) and showed that it is NISZK-complete. They then designed a private-coin HVSZK batch verification protocol for AI. We follow a similar route, except that we construct a *public-coin* HVSZK batch verification protocol for AI.

The inputs to the AI problem, which is parameterized by a real number $\delta \in [0,1]$, are circuits $C : \{0,1\}^n \to \{0,1\}^m$. YES instances are circuits C for which all but a δ fraction of the inputs are mapped injectively to their corresponding outputs (i.e., $\Pr_x \left[|C^{-1}(C(x))| > 1\right] < \delta$). NO instances are circuits for which at most a δ fraction of the inputs are mapped injectively (i.e., $\Pr_x \left[|C^{-1}(C(x))| > 1\right] > 1 - \delta$).

Since AI_δ was shown to be NISZK complete, to prove Theorem 1.2 it suffices to show a *public-coin* HVSZK batching protocol for AI_δ. Our main technical result is precisely such a protocol achieving communication roughly $k + \mathsf{poly}(n)$, where k is the number of instances being batched. For simplicity, we first focus on the case that $\delta = 0$ – namely, distinguishing circuits that are injective from those in which no input is mapped injectively to its output. A discussion of the case of $\delta > 0$ is deferred to the end of this overview.

In order to present our approach, following [KRR+20], we will first consider a drastically easier case in which the goal is to distinguish between circuits that are *permutations* (rather than merely being injective) or are far from such.

Warmup: Batch Verification for Permutations. We first consider a variant of AI, called PERM. The inputs to PERM are length-preserving circuits $C : \{0,1\}^n \to \{0,1\}^n$. YES instances are circuits that compute permutations over $\{0,1\}^n$, whereas NO instances are circuits that are *far* from being permutations in the sense that no input is mapped injectively (i.e., every image has at least two preimages).

Consider the following batch verification protocol for PERM. Given common input $C_1, \dots, C_k : \{0,1\}^n \to \{0,1\}^n$ the parties proceed as follows. The verifier V samples $y_k \in \{0,1\}^n$ and sends it to the prover P. Then, P responds with an x_1 s.t. $C_k(C_{k-1}(...C_1(x_1)) = y_k$. The verifier checks that indeed $C_k(C_{k-1}(...C_1(x_1)) = y_k$ and accepts or rejects accordingly.

To argue that completeness holds, observe that when all of the C_i's are permutations, there exists a *unique* sequence $x_1, y_1, x_2, y_2, \dots, x_{k-1}, y_{k-1}, x_k$, where $x_i = y_{i-1}$ for every $i \in [2,k]$, that is consistent with y_k. That is, $x_i = C_i^{-1}(y_i)$, for every $i \in [k]$). The prover can thus make the verifier accept (with probability 1) by sending x_1 as its message.

For soundness, let $i^* \in [k]$ denote the maximal index i s.t. C_i is a NO instance. Since C_{i^*} is a NO instance, each of its images has at least two preimages and so the size of its image is at most 2^{n-1}. Since y_k is sampled uniformly and $C_{i^*+1}, \dots, C_k$ are permutations, we have that y_{i^*} is also random in $\{0,1\}^n$. In particular this means that with probability at least half it has no preimage under C_{i^*}, and the verifier will eventually reject no matter what value of x_1 the prover sends. Note that the soundness error, which is $1/2$ here, can be amplified by repetition.

For zero-knowledge, consider the simulator that first samples $x_1 \in \{0,1\}^n$ and then computes $y_k = C_k(C_{k-1}(...C_1(x_1))$. Since all circuits are permutations, y_k is distributed uniformly over $\{0,1\}^n$ as in the real interaction between the honest-verifier and the prover.

The above batch verification protocol for PERM, while simple, will be the basic underlying idea also for our batch verification protocol for AI_0.

Public-coin Batch Verification for AI_0. Let $C_1, \dots, C_k : \{0,1\}^n \to \{0,1\}^m$ be instances of AI_0. Since the output size of each circuit C_i is not compatible with the input size for the following circuit C_{i+1}, we cannot directly compose the circuits as we did for PERM.

A natural idea that comes to mind is to use hashing. Namely, choose a hash function[1] g to map C_i's output $y_i \in \{0,1\}^m$ to the next circuit input $x_{i+1} \in \{0,1\}^n$, for every $i \in [k]$. Based on this idea it is natural to consider a minor modificition of the batch verification protocol for PERM, where the only difference is that we interleave applications of g as we compose the circuits. Note that we have to hash the last circuit output y_k as well so that we can specify $x_{k+1} = g(y_k)$ to the prover as a genuine unstructured random string.

If we could guarantee that the hash function g maps the image of each circuit *injectively* into the domain of the subsequent circuit, then a similar analysis as in the protocol for PERM could be applied and we would be done. However, finding such a hash function in general seems incredibly difficult. Thus, instead, we choose g to be a random function.[2]

In what follows, for every $i \in [k]$, denote the image of C_i by $S_i \subseteq \{0,1\}^m$. Note that if C_i is a YES instance (i.e., injective), the size of S_i is exactly 2^n, and if C_i is a NO instance (entirely non-injective) the size of S_i is at most 2^{n-1}.

When using a random function g as our hash function, we run into two key challenges that did not exist in the protocol for PERM. The first of these two challenges arises from the fact that for any YES instance circuit C_i, with high probability over the choice of $g : \{0,1\}^m \to \{0,1\}^n$, a constant fraction of the elements in $\{0,1\}^n$ have no preimages (under g) in the set S_i.[3] If such a situation occurs for *any* of the circuits, a situation which is exceedingly likely, then even the honest prover will not be able to find a suitable preimage x_1 and we lose completeness.

The way we solve this difficulty is relatively simple: we add to the hash function g a short random auxiliary input that will be chosen independently for each of the k applications of g. We denote the auxiliary input for the i^{th} application by z_i and its length by d (which we set to $\mathrm{polylog}(n, k)$). Observe that if $g : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^n$ is chosen at random, then we expect all x's in the domain of C_{i+1} to have roughly the same number of preimages (under g) that lie in the set $S_i \times \{0,1\}^d$.[4]

This brings us to the second challege in using a random hash function g, which is slightly more subtle. When considering a YES instance circuit C_i, even ignoring the additional auxiliary input, a constant fraction of the domain $\{0,1\}^n$ of C_{i+1} will have *more than one* preimage (under g) which falls in S_i. Needless to say, this issue is further exacerbated by the addition of the auxiliary input. At first glance this may not seem like much of an issue when we consider a YES circuit. However, on further inspection, we observe that we may very well be in the case that all of the circuits except for the first circuit C_1 are YES instances (i.e., injective) and only C_1 is a NO instance (i.e,

[1]The specific type of hash function that we use is left vague for now and will be discussed in detail shortly.

[2]Jumping ahead, we note that we cannot afford for g to be entirely random, and will have to settle for some de-randomization. For the moment we ignore this issue.

[3]This is similar to the fact that when throwing N balls into N bins, in expectation, a constant fraction of the bins remain empty. Here the images of C_i play the role of the balls and the elements in the domain $\{0,1\}^n$ of C_{i+1} play the role of the bins.

[4]Here we rely on the fact that when throwing $N \cdot \mathrm{polylog}(N)$ balls into N bits, with high probability, all bins will contain very close to the expected $\mathrm{polylog}(N)$ balls.

non-injective).

If such is the case, due to the collisions that occur in g, it is likely that y_k will have an exponential (in k) number of preimages x_2 that are consistent with it. If the prover has so much flexibility in its choice of x_2 then it is likely that, even though C_1 is non-injective, the prover will be able to find a consistent preimage x_1 and we lose soundness.

Borrowing an idea from [KRR+20], we solve this challenge using interaction. The high-level approach is for the prover to *commit* to x_i *before* we reveal the auxiliary information for the next circuit. Thus, the protocol proceeds iteratively, where in each iteration first the prover commits to x_i, and then the verifier reveals the auxiliary input, which the prover uses to recover y_{i-1}, and so on. The commitment has the property that as long as we are processing YES input circuits, with high probability, there is a *unique* x_i that is consistent with the interaction. In particular, when we reach the first NO instance circuit C_{i^*} (recall that i^* denotes the maximal i s.t. C_i is a NO instance) there is a unique x_{i^*+1} that is consistent with the transcript.

Distinguishing Injective Circuits from Non-Injective Circuits. Recall that our goal is to distinguish an injective circuit from a highly non-injective circuit. Following our approach thus far, assume that we have processed the circuit up to the circuit C_i and moreover, that there is a unique x_{i+1} that is consistent with the interaction up to this point.

Recall also that if C_i is injective then $|S_i| = 2^n$, whereas if it is non-injective then $|S_i| \leq 2^{n-1}$. Thus, our approach is to employ a set lower bound protocol (a la [GS89]) as follows. The verifier chooses a "filter" function $h_i \in H_n$, where H_n is a family of pairwise independent hash functions from domain $\{0,1\}^m \times \{0,1\}^d$ to an appropriately chosen range size, as well as a target element α_i. If C_i is injective then we expect each x_{i+1} to have very close to 2^d preimages $(y_i, z_i) \in S_i \times \{0,1\}^d$ under g. On the other hand, if C_i is non-injective, then we expect each x_{i+1} to have roughly 2^{d-1} such preimages or less. Thus, by setting the range size to be roughly 2^d, the probability that one of these preimages hashes correctly via h_i to α_i is larger (by a constant) in the YES case than in the NO case.

Balancing Completeness, Soundness and Zero-Knowledge. At this point we observe that even if g and h_i were random functions, the set lower bound approach only yields a small constant gap between completeness and soundness. This is insufficient since the completeness error is accumulating across the k different circuits. Note that we cannot afford a k-fold repetition of the set lower bound protocol (since it would be prohibitively expensive in our parameter setting). Moreover, since we cannot generically amplify zero-knowledge, we also need the zero-knowledge error accumulated by the YES instance circuits to be negligible.

We resolve this issue by considering a new variant of the approximate injectivity problem which we denote by $\mathsf{AI}_{L,\delta}$. The YES instances in this variant are identical to the YES instance in AI_δ – namely circuits that are injective on all but a δ fraction of their domain. However, a circuit C is a NO instance if at least $1-\delta$ fraction of its inputs have at least L "siblings" (i.e., inputs that are mapped to the same output). Thus, the standard AI_δ problem corresponds to the case $L = 2$. Using a large enough L increases the gap between the number of preimages in YES and NO instances, letting us set the range size of the h_i's to obtain a larger gap between completeness and soundness.

We show that $\mathsf{AI}_{2,\delta}$ reduces to $\mathsf{AI}_{L,\delta'}$, where $L = 2^{\text{polylog}(n,k)}$ and δ' is related to δ. The idea for the reduction is to simply concatenate sufficiently many copies of the input circuit.[5] Thus, in the

[5]In more detail, we transform an instance C of $\mathsf{AI}_{2,\delta}$ to an instance C' of $\mathsf{AI}_{2^\ell,\delta\cdot\ell}$ by concatenating ℓ copies of C.

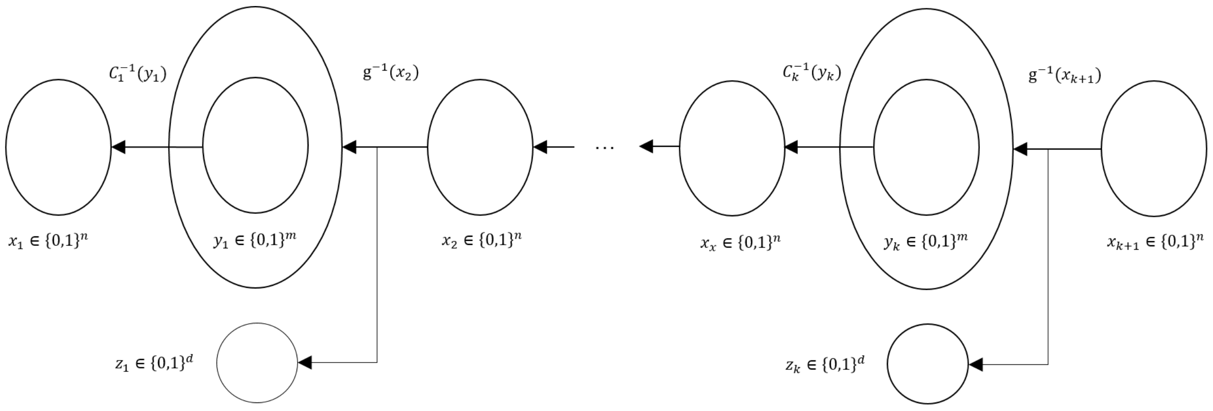

Figure 1: Sampling process of the Protocol

sequel it suffices to consider batch verification for k instances of $\mathsf{AI}_{L,0}$ (and the case that $\delta > 0$ will be discussed later on).

Over-Simplified Batch Verification Protocol for $\mathsf{AI}_{L,0}$. With the foregoing insights in mind, consider the following over-simplified batching protocol for $\mathsf{AI}_{L,0}$ (see also the diagram in Fig. 1 which gives a bird's eye view of the flow of the protocol).

1. V samples g and $x_{k+1} \leftarrow \{0,1\}^n$ and sends both to P.
2. For $i = k, ..., 1$:
   (a) V samples a filter function $h_i \leftarrow H_n$ and target value α_i (of appropriate length) and sends both to P.
   (b) P selects at random a pair (x_i, z_i) s.t.
      i. $g(C_i(x_i), z_i) = x_{i+1}$
      ii. $h_i(C_i(x_i), z_i) = \alpha_i$

      and sends (x_i, z_i) to V.
3. V sets $x'_1 = x_1$ and for $i = 1, \ldots, k$:
   (a) Computes $y'_i = C_i(x'_i)$.
   (b) Verifies that $h_i(y'_i, z_i) = \alpha_i$.
   (c) Computes $x'_{i+1} = g(y'_i, z_i)$.
   (d) Verifies that $x'_{i+1} = x_{i+1}$.
4. If all of V's checks pass then she accepts. Otherwise she rejects.

Note that the communication complexity in this protocol is $\Omega(k \cdot n)$ (since the prover sends $x_1, \ldots, x_k$), which is more than we can afford. This issue will be resolved shortly and so we ignore it for now.

It is not hard to see that if C were (almost) injective) then C' is (almost) injective. But, if (almost) every image of C has at least two preimages then (almost) every image of C' has at least 2^ℓ preimages.

For completeness, if all C_i's are injective, in every iteration i, given that there exists a consistent x_i (i.e., x_i for which there exists z where $g(C_i(x_i), z) = x_{i+1}$ and $h_i(C_i(x_i), z) = \alpha_i$), w.h.p. there exists also a consistent x_{i-1}. Hence, it is not hard to show, by induction, that with high probability, after the last iteration there exists an x_1 that passes the verifier's checks , and so V accepts .

For soundness, consider the iteration i^* which, as defined in the protocol for PERM, denotes the maximal index of a NO instance circuit. Since the number of preimages of C_{i^*} is less than $2^n/L$, by the foregoing discussion it is very likely that there does not exist a pair (x_{i^*}, z_{i^*}) that is consistent with the rest of the messages, i.e., s.t. $g(C_{i^*}(x_{i^*}), z_{i^*}) = x_{i^*+1}$ and $h_{i^*}(C_{i^*}(x_{i^*}), z_{i^*}) = \alpha_{i^*}$, and therefore V will eventually reject.

Before arguing why the protocol is zero-knowledge, we first discuss the hash function family that g is sampled from.

The Hash Function g. One important consideration that arises when derandomizing g is that a cheating prover P^* has some flexibility in the choice of the x_i that she sends for rounds $i > i^*$. Indeed, by design there will be many such x_i's. While the honest prover should choose x_i at random (from this set), a cheating prover may try to cleverly choose x_i's that help her cheat. Since all these choices are made after the function g was revealed, we cannot assume that $x_{i^*+1}, \dots, x_k$ are uniformly random relative to g. Thus, for our analysis it does not suffice that a random x_{i+1} has a suitable number of preimages under g.

Instead, we will seek a much stronger, worst-case guarantee, from g. Specifically, that for *every* $x_{i+1} \in \{0,1\}^n$, the number of preimages $(y, z) \in g^{-1}(x_{i+1})$, where $y \in S_i$, is close to its expectation, i.e., around 2^d for $C_i \in$ YES and around $2^d/L$ for $C_i \in$ NO.

As shown by Alon *et al.* [ADM$^+$99], this type of guarantee is not offered by pairwise independent hash function or even more generally by randomness extractors. On the other hand, what gives us hope is that a totally random function does have such a worst-case guarantee. Thus, we wish to construct a small hash function family G where with high probability over the choice of $g \leftarrow G$, every image has roughly the same number of preimages from a predetermined set.

Following a work of Celis *et al.* [CRSW11], we refer to such functions as *load-balancing hash functions*. A function in this family maps the set $\{0,1\}^m$ together with some auxiliary input $\{0,1\}^d$ to the set $\{0,1\}^n$. We require that for any set $S \subseteq \{0,1\}^n$ s.t. $|S| \geq 2^n/L$ and for every $x \in \{0,1\}^n$, w.h.p over $g \leftarrow G_n$, it holds that $\left|\{(y,z) : y \in S, g(y,z) = x\}\right|$ is roughly $\frac{|S| \cdot 2^d}{2^n}$. We construct a suitable load-balancing functions based on $\mathsf{poly}(n)$-wise independent hash functions (combined with almost pairwise independent hash functions).

Honest Verifier Statistical Zero-Knowledge. To argue zero-knowledge, consider a simulator that first samples an initial x_1. Then, in each iteration i the simulator samples z_i, computes $y_i = C_i(x_i)$ and $x_{i+1} = g(y_i, z_i)$. Then it samples h_i and computes $\alpha_i = h_i(y_i, z_i)$.

Statistical zero-knowledge is not obvious since the protocol and the simulator progress in opposite directions. The simulation progress direction is from circuit C_1 up to circuit C_k, while the protocol progress direction is from circuit C_k down to circuit C_1, as shown in Fig. 2. However, due to the special property of the the load-balancing function g, we manage to achieve statistical zero-knowledge.

We define the hybrid distribution H_i that is sampled as follows:

1. Sample $g \in G_n$ and $x_i \in \{0,1\}^n$.

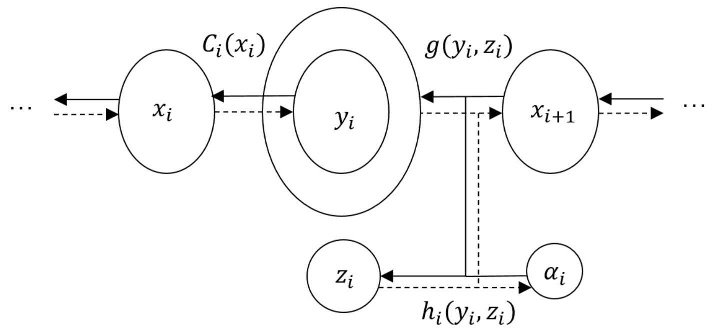

Figure 2: Protocol progress direction vs Simulator progress direction – the solid lines represent the protocol, and the dashed line the simulator.

2. Generate $(x_j, z_j, h_j, \alpha_j)_{j\in\{1,\dots,i-1\}}$ according to the protocol.
3. Generate$(z_j, h_j, \alpha_j, x_{j+1})_{j\in\{i,\dots,k\}}$ according to the simulator.
4. Output g, x_{k+1} and $(x_j, z_j, h_j, \alpha_j)_{j\in\{1,\dots,k\}}$.

Note that the simulator distribution is identical to H_1 while the protocol distribution is identical to H_{k+1}. We bound the statistical difference between H_i and H_{i+1} and use the hybrid argument.

Note that conditioned on $(g, x_i, z_i, h_i, \alpha_i, x_{i+1})$ the rest of the variables in H_i,H_{i+1} are distributed identically. Therefore it is enough to bound the statistial differences between those variables as sampled in H_i and H_{i+1}. Since g is sampled identically in both hybrids, we can fix it and bound the statistical difference of those variables conditioned on some g.

For the hybrid H_{i+1}, the variables $(x_i, z_i, h_i, \alpha_i, x_{i+1})$ are sampled according to the protocol whereas for the hybrid H_i, the variables $(x_i, z_i, h_i, \alpha_i, x_{i+1})$ are sampled according to the simulator. Let X_S, X_P and X_U denotes the distributions of any random variable X according to the simulator, protocol and the uniform distribution respectively.

Consider the distribution of $(x_i, z_i, h_i, \alpha_i, x_{i+1})_P$ which is sampled according to the protocol, i.e., (h_i, α_i, x_{i+1}) are sampled uniformly at first, and then (x_i, z_i) are chosen uniformly from the set of (x, z) s.t. $g(C_i(x), z) = x_{i+1}$ and $h_i(C_i(x), z) = \alpha_{i+1}$. On the other hand, consider the distribution of $(x_i, z_i, h_i, \alpha_i, x_{i+1})_S$ which is sampled according to the simulator, i.e., (x_i, z_i, h_i) are sampled uniformly at first and then it sets $x_{i+1} = g(C_i(x_i), z_i)$ and $\alpha_i = h_i(C_i(x_i), z_i)$. Note that the distributions $(x_i, z_i)_P$ and $(x_i, z_i)_S$ are identical conditioned on specific values of (h_i, α_i, x_{i+1}), and therefore, it is enough to bound $\Delta\big((h_i, \alpha_i, x_{i+1})_P, (h_i, \alpha_i, x_{i+1})_S\big)$.

We define the function $\varphi_i(x_i, z_i, h_i) = (h_i, \alpha_i, x_{i+1})$, where $x_{i+1} = g(C_i(x_i), z_i)$ and $\alpha_i = h_i(C_i(x_i))$. Note that

$$\Delta\left((h_i, \alpha_i, x_{i+1})_P, (h_i, \alpha_i, x_{i+1})_S\right) = \Delta\Bigg((h_i, \alpha_i, x_{i+1})_U, \varphi_i\big((x_i, z_i, h_i)_U\big)\Bigg).$$

Therefore, what we have left is to show that φ_i's output on uniform input is close to uniform.

Consider uniform (x_i, z_i, h_i) and set $x_{i+1} = g(C_i(x_i), z_i)$ and $\alpha_i = h_i(C_i(x_i), z_i)$. x_{i+1} is close to uniform due to the special property of g that every x_{i+1} has roughly the same number of preimages (y_i, z_i), and due to the fact that each image of C_i has exactly one preimage. Now fix x_{i+1}. the number of pairs (x_i, z_i) that are mapped to x_{i+1} is roughly 2^d, i.e, there are d bits of entropy on which h_i is applied. Therefore, since h_i is pairwise independent hash function and as such, also a strong extractor, we get that (h_i, α_i) is also close to uniform.

Reducing Communication via Hashing. The foregoing soundness analysis relied on the fact that the prover sent the values $(x_1, \ldots, x_k)$ during the interaction. However, as noted above, this requires $n \cdot k$ communication which we cannot afford. In a nutshell, we resolve this inefficiency by having the prover only send short hashes of the x_i's, details follows.

At the beginning of each round, the verifier V sends a description of a pairwise independent hash function f_i in addition to the filter function h_i and target value α_i. Then, in addition to sending z_i, the prover P in her turn also sends a hash value $\beta_i = f_i(x_i)$ (rather than sending x_i explicitly). In order for the hash value to commit the prover to x_i, we would like for the hash function f_i to be injective on the set of consistent (x, z)'s (i.e., those for which $g(C_i(x), z) = x_{i+1}$ and $h_i(C_i(x), z) = \alpha_i$). This set is of size roughly 2^d where $d = \text{polylog}(n, k)$. Therefore, setting the output size of f_i to be $\mathsf{poly}(d)$ ($= \text{polylog}(n, k)$) bits is sufficient. At the very end of the protocol, the prover P still needs to explicitly send x'_1, so that V can compute $y'_i = C_i(x'_i)$ and $x'_{i+1} = g(y'_i, z_i)$, for every $i \in [k]$, and verify that $h_i(y'_i, z_i) = \alpha_i$, and lastly that $x'_{k+1} = x_{k+1}$. Note that the verifier can only perform these checks at the end of the interaction (as she did in the simplified protocol) since she must obtain the value of x'_1 in order to generate $x'_2, \ldots, x'_k$.

Overall, the communication is dominated by the values x_{k+1} and g, which are sent by the verifier, and the value x'_1 sent by the prover. Each of these messages has length $\mathsf{poly}(n, \log(k))$. All the rest of the messages including the specification of the hash functions h_i, f_i as well as the values α_i, β_i and z_i have length $\text{polylog}(n, k)$. Overall we obtain the desired communication complexity $\big(k + \mathsf{poly}(n)\big) \cdot \text{polylog}(n, k)$.

When $\delta > 0$. For the case where $\delta > 0$, the arguments we made for completeness and zero-knowledge still hold in a straightforward manner, but we need to be more careful about soundness. More specifically, for some YES instance circuit C_i (where $i > i^*$) and $x_{i+1} \in \{0,1\}^n$, there potentially can exist a pair $(y_i, z_i) \in g^{-1}(x_{i+1})$ s.t. $h_i(y_i, z_i) = \alpha_i$ and y_i has an exponential number of preimages. Therefore, the set of consistent (x, z) is of exponential size and therefore, in order to fix the chosen x_i by the prover, β_i must consist polynomial number of bits, which is of course too expensive for us.

However, since there is only a small number of images y_i that have more then one preimage (δ fraction of 2^n), there is also only a small number of pairs $(y_i, z_i) \in g^{-1}(x_{i+1})$ where y_i is such an image. Therefore, w.h.p over (h_i, α_i), none of those problematic pairs satisfy the condition that $h_i(y_i, z_i) = \alpha_i$, and therefore, their preimages are inconsistent which allows the earlier setting of the output size of f_i to work.

Comparison to the [KRR+20] Protocol. Our public-coin batching protocol bears some resemblance to the *private-coin* batching protocol of [KRR+20]. We highlight here the similarity and differences between our approaches. We note that readers who are unfamiliar with [KRR+20] can safely skip this discussion and proceed directly to Section 1.2.2.

In the protocol of [KRR+20] the verifier V first samples x_1 and auxiliary randomnesses $(z_1, \ldots, z_k)$ as part of her setup. She computes $y_i = C_i(x_i)$ and determines the next circuit input as $x_{i+1} = g(y_i, z_i)$, for every $i \in [k]$, where in contrast to our protocol, the function g is simply a (strong) seeded randomness extractor (where y_i is the min-entropy source and z_i is the extractor's seed).

The actual interaction starts by having the verifier V send y_k to P. The parties then proceed in iterations where in each iteration i, given x_{i+1}, the prover needs to guess x_i using some additional hints that the verifier supplies. The prover's guesses are communicated by sending a short hash.

While their protocol bears some similarity to ours, we emphasize several fundamental ways in which our approach differs from that of [KRR+20] (beyond the fact that our protocol is public-coin).

- In [KRR+20] the verifier herself samples $(z_1, \dots, z_k)$ and using it computes $(x_1, \dots, x_k)$ as part of her setup, and then she reveals these gradually. This means that in the [KRR+20] there is a ground truth that the verifier can compare. In contrast, in our protocol, each x_i is chosen via an interactive process that involves both parties and happens "online". In particular, and as discussed above, this means that a cheating prover can bias the distribution of the x_i's as we process the circuit.

- On a related note, if the prover commits to a wrong x_i in some iteration i, then, in [KRR+20], the verifier V can immediately detect this and reject. In contrast, in our protocol we are unable to do so and must wait to detect this at the very end of the interaction.

- In our protocol the x_i's are computed in reverse order starting from x_k down to x_1, whereas in the protocol of [KRR+20] the x_i's are computed in order, starting from x_1 up to x_k. This may seem like a minor difference but turns out to complicate matters significantly when considering zero-knowledge. Indeed, both the [KRR+20] simulator as well as our simulator compute the x_i's in order. This means that in the current work the protocol and the simulator, operate in *different* order. This makes the analysis of the simulation significantly more challenging.

- Lastly, [KRR+20] use extractors in order to map each circuit output y_i to the next circuit input x_{i+1}. As discussed in detail in the above technical overview, the average-case guarantee provided by extractors are not good enough for us and we rely on the stronger notion of load-balancing hash functions.

1.2.2 Efficient Transformation to Public-Coin Malicious Verifier SZK

Goldreich, Sahai, and Vadhan [GSV98] showed that any public-coin HVSZK protocol can be transformed into a public-coin SZK protocol. Applying their transformation to the public-coin honest-verifier batch verification protocol described above would indeed result in a malicious-verifier SZK batch verification protocol for AI, and thus NISZK. This transformation, however, starts by repeating the HVSZK protocol several times in parallel in order to make the soundness error exponentially small in the number of rounds. This would incur a blowup in communication by a factor of $\Omega(k)$, which we cannot afford.

In order to get around this, we show that the transformation of [GSV98], when used on protocols with a stronger soundness guarantee called *round-by-round soundness* [CCH+19], can be performed without the initial repetition step, and thus achieve a much smaller blowup in communication. Then we show that our honest-verifier batching protocol does indeed provide this guarantee, and thus the transformation can be applied to it with this better blowup, giving us the desired result. We now briefly describe the transformation, the round-by-round soundness property, and how they fit together.

The GSV Transformation. Recall that in a public-coin HVSZK protocol, the honest verifier's messages consist of uniformly random strings. What breaks when the verifier is malicious is that it might choose these strings arbitrarily rather than uniformly at random. In the GSV transformation, the prover and verifier essentially run the given HVSZK protocol, but instead of the verifier sending these random strings, they are sampled by the prover and the verifier together using a Random Selection (RS) protocol. This protocol, which is constructed by [GSV98], uses four messages, is public-coin, and produces as output a string r of some desired length ℓ. It provides, very roughly, the following guarantees when run with security parameter λ:

1. If the prover is honest, the distribution of r is $2^{-\lambda}$-close to uniform over $\{0,1\}^{\ell}$, and the transcript of the protocol is simulatable given output r.
2. If the verifier is honest, for any set $T \subseteq \{0,1\}^{\ell}$, we have $\Pr[r \in T] \leq 2^{\lambda} \cdot |T| / 2^{\ell}$.

The first property above ensures that the resulting protocol is complete and zero-knowledge. The second property ensures that the prover cannot skew the distribution of r, and thus soundness is maintained. Following the analysis in [GSV98], however, it turns out that if the original protocol has soundness error s and r rounds, the bound obtained on the soundness error of the new protocol is roughly $2^{r\lambda} \cdot s$. Thus, we would need to start by decreasing the soundness error of the HVSZK protocol to less than $2^{-r\lambda}$. The only way we know to do this generically is by repetition, which results in a multiplicative blowup of at least $\Omega(r)$ in communication – in our case this is $\Omega(k)$, which is too large for us. We get around this by showing that our batch verification protocol has a stronger soundness property that results in a much better bound on the soundness error when this transformation is applied.

Round-by-Round Soundness. Typically, the soundness property in an interactive proof places requirements on how likely it is that a verifier accepts on a NO input. Round-by-round soundness, introduced by Canetti et al [CCH+19], instead places requirements on intermediate stages of the protocol. It involves a mapping State from partial transcripts of the protocol to the set $\{\texttt{alive}, \texttt{doomed}\}$. A protocol is *$\epsilon$-round-by-round sound* if there exists a mapping State such that for any input x and partial transcript τ with $\mathsf{State}(x,\tau) = \texttt{doomed}$, and any subsequent prover message α, the probability over the next verifier message β that $\mathsf{State}(x,\tau,\alpha,\beta) = \texttt{alive}$ is at most ϵ. Further, any complete transcript that is $\texttt{doomed}$ always results in a rejection by the verifier, and for any NO instance x, it is the case that $\mathsf{State}(x,\perp) = \texttt{doomed}$.

In other words, at a point where a protocol is $\texttt{doomed}$, irrespective of what the prover does, the set of "bad" verifier messages that make the protocol $\texttt{alive}$ in the next round has relative size at most ϵ. To see its implications for standard soundness, consider a protocol that has r rounds and is ϵ-round-by-round sound. The probability that the verifier accepts on a NO instance is at most the probability that the complete transcript is $\texttt{alive}$. Thus, since the protocol has to go from $\texttt{doomed}$ to $\texttt{alive}$ in at least one of the r rounds, the probability that the verifier accepts is at most $r\epsilon$.

Putting it Together. Now, consider passing an ϵ-round-by-round sound protocol through the GSV transformation described above. Here, each verifier message is replaced by the output of the RS protocol. On a NO instance, in order to successfully cheat, the prover has to make at least one of the r verifier messages fall into the "bad" set for that round. Each of these bad sets, however, has relative size at most ϵ, and thus the RS protocol's output falls in each with probability at most $\epsilon 2^{\lambda}$. Thus, that total probability that the prover successfully cheats is at most $\epsilon r 2^{\lambda}$.

So, if the original protocol had round-by-round soundness error somewhat smaller than $(r2^{\lambda})^{-1}$, the resulting protocol would still be sound. This is a much more modest requirement than before, and can be achieved with a multiplicative blowup of at most $O(\lambda \log r)$ in communication. Completeness and zero-knowledge follow from the other properties of the RS protocol, and setting λ to be polylog(n, k) gives us the desired SZK protocol.

Round-by-Round Soundness of our HVSZK Batching Protocol. To show that our protocol described earlier has round-by-round soundness, we define the State function as follows. Consider the partial transcript τ_j that corresponds to its j'th iteration – this consists of g, x_{k+1}, $(h_i, \alpha_i, f_i, z_i, \beta_i)_{i \in \{k,\dots,j+1\}}$, and (h_j, α_j, f_j).

- For $j > i^*$: Output `doomed` on the transcript τ_j if, for any prover message (z_j, β_j) that follows, there exists at most one preimage x_j that is consistent with τ_j and (z_j, β_j). Further, if there is no such x_j, output `doomed` on all future transcripts that extend τ_j.
  - Consider a `doomed` transcript τ_{j+1}, a prover message (z_{j+1}, β_{j+1}), and the unique x_{j+1} that is consistent with them as promised (if it doesn't exist, future transcripts are already `doomed`). By our earlier discussion, w.h.p., there are very few x_j's which have a z_i such that $g(C_j(x_j), z_j) = x_{j+1}$ and $h_j(C_j(x_j), z_j) = \alpha_j$. Therefore, w.h.p., f_j maps these x_j's injectively. Hence for any prover message (z_{j+1}, β_{j+1}), we have that w.h.p. $\tau_{j+1}, (z_{j+1}, \beta_{j+1}), (h_j, \alpha_j, f_j)$ is also `doomed`.
- For $j = i^*$: We define the State function to output `doomed` on the transcript τ_{i^*} if for any prover message (z_{i^*}, β_{i^*}), there does not exist x_{i^*} that is consistent with τ_{i^*} and (z_{i^*}, β_{i^*}).
  - Consider any `doomed` transcripts τ_{i^*+1}, a prover message $(z_{i^*+1}, \beta_{i^*+1})$, and the unique x_{i^*+1} that is consistent with them. As discussed earlier, w.h.p., there does not exist (x_{i^*}, z_{i^*}) s.t. $g(C_{i^*}(x_{i^*}), z_{i^*}) = x_{i^*+1}$ and $h_{i^*}(C_{i^*}(x_{i^*}), z_{i^*}) = \alpha_{i^*}$ and therefore for every prover message $(z_{i^*+1}, \beta_{i^*+1})$, it holds w.h.p. that $\tau_{i^*+1}, (z_{i^*+1}, \beta_{i^*+1}), (h_{i^*}, \alpha_{i^*}, f_{i^*})$ is `doomed` too.
- For $j < i^*$: We define the State function to answer according to the partial transcript τ_{i^*}, and therefore round-by-round soundness in this case is immediate.

We set $\mathsf{State}(x, \perp)$ to be `doomed` if x is a NO instance, and anything that is not `doomed` is set to be `alive`. Lastly, consider a complete transcript that is `doomed`; there does not exists x_{i^*} that is consistent with the beginning of the transcript, and therefore V must reject. This function now witnesses the round-by-round soundness of our protocol.

1.3 Organization

We start with preliminaries in Section 2. In Section 3 we formalize our notion of a *load-balancing* hash function and provide a construction based on k-wise independent hash functions. In Section 4 we introduce our variant of the approximate injectivity (AI) problem and show that it is NISZK-complete. In Section 5 we construct our *public-coin honest-verifier* batch verification for AI. In

Section 6 we show a generic, and efficient, transformation from public-coin HVSZK (having round-by-round soundness) to public-coin full-fledged SZK. Lastly, in Section 7 we use the results obtained in the prior sections to obtain our public-coin SZK batch verification protocol for NISZK.

2 Preliminaries

For a finite non-empty set S, we denote by $x \leftarrow S$ a uniformly distributed element in S. We also use U_ℓ to denote a random variable uniformly distributed over $\{0,1\}^\ell$.

For a distribution X over a finite set U and a (non-empty) event $E \subseteq U$, we denote by $X|_E$ the distribution obtained by conditioning X on E: namely, $\Pr[X|_E = u] = \Pr[X = u \mid E]$, for every $u \in U$. The *support* of X is defined as $\mathsf{supp}(X) = \{u \in U : \Pr[X = u] > 0\}$.

Definition 2.1. *Let $\Pi = (\mathrm{YES}, \mathrm{NO})$ be a promise problem , where $\mathrm{YES} = (\mathrm{YES}_n)_{n\in\mathbb{N}}$ and $\mathrm{NO} = (\mathrm{NO}_n)_{n\in\mathbb{N}}$, and let $k = k(n) \in \mathbb{N}$. We define the promise problem $\Pi^{\otimes k} = (\mathrm{YES}^{\otimes k}, \mathrm{NO}^{\otimes k})$, where $\mathrm{YES}^{\otimes k} = (\mathrm{YES}_n^{\otimes k})_{n\in\mathbb{N}}$, $\mathrm{NO}^{\otimes k} = (\mathrm{NO}_n^{\otimes k})_{n\in\mathbb{N}}$ and*

$$\mathrm{YES}_n^{\otimes k} = (\mathrm{YES}_n)^k$$

and

$$\mathrm{NO}_n^{\otimes k} = (\mathrm{YES}_n \cup \mathrm{NO}_n)^k \setminus (\mathrm{NO}_n)^k.$$

2.1 Statistical Difference

The statistical distance between two distributions P and Q over a finite set U is defined as $\Delta(P,Q) = \max_{S\subseteq U}(P(S) - Q(S)) = \frac{1}{2}\sum_{u\in U} |P(u) - Q(u)|$.

We proceed to describe several basic facts about the statistical distance (omitting proofs of standard facts which can be found in any textbook).

Fact 2.2 (Data processing inequality for statistical distance)**.** *For any two distributions X and Y, and every (possibly randomized) process f:*

$$\Delta\left(f(X), f(Y)\right) \leq \Delta\left(X, Y\right).$$

Fact 2.3. *Let (X_0, Y_0) and (X_1, Y_1) be two pairs of jointly distributed random variables s.t. $\mathsf{supp}(X_0) = \mathsf{supp}(X_1)$ and for every $x \in \mathsf{supp}(X_0)$, the distributions $Y_0|_{X_0=x}, Y_1|_{X_1=x}$ are identically distributed. Then,*

$$\Delta\big((X_0, Y_0), (X_1, Y_1)\big) = \Delta(X_0, X_1).$$

Proof. Consider the random process f that, given input $x \in \mathsf{supp}(X_0)$, samples from the conditional distribution $Y_0|_{X_0=x}$, or equivalently, from $Y_1|_{X_1=x}$. The fact follows by applying Fact 2.2 with respect to the foregoing random process and observing that the joint distribution (X_0, Y_0) (resp., (X_1, Y_1)) is identical to $(X_0, f(X_0))$ (resp., $(X_1, f(X_1))$).

□

Fact 2.4 (See, e.g., [KRR+20, Fact 2.2])**.** *For any two distributions X and Y, and an event E:*

$$\Delta\left(X, Y\right) \leq \Delta\left(X|_E, Y\right) + \Pr_X[\neg E].$$

Fact 2.5. *Let X and Y be distributions s.t. $\Delta(X,Y) \le \delta$ and let E be an event s.t. $\Pr_Y[E], \Pr_X[E] \ge 1-\epsilon$ with $\epsilon \le \frac{1}{2}$. Then,*

$$\Delta\left(X|_E, Y|_E\right) \le 2 \cdot (\epsilon + \delta)\,.$$

Proof. Assume wlog that $\Pr_Y[E] \ge \Pr_X[E]$. Define as shorthands $p_u = \Pr[X=u]$, $q_u = \Pr[Y=u]$, $p_{(u,E)} = \Pr_X[X=u,E]$, $q_{(u,E)} = \Pr_Y[Y=u,E]$, $p_{u|E} = \Pr_X[X=u|E]$ and $q_{u|E} = \Pr_Y[Y=u|E]$. Note that

$$\sum_u \left|p_{(u,E)} - q_{(u,E)}\right| = \sum_{u\in E}\left|p_{(u,E)} - q_{(u,E)}\right| = \sum_{u\in E}\left|p_u - q_u\right| \le \sum_u \left|p_u - q_u\right| \le \delta. \tag{1}$$

Therefore

$$\begin{aligned}
\Delta\left(X|_E, Y|_E\right) &= \frac{1}{2}\sum_u \left|p_{u|E} - q_{u|E}\right| \\
&= \frac{1}{2}\cdot\sum_u\left|\frac{p_{(u,E)}}{\Pr_X[E]} - \frac{q_{(u,E)}}{\Pr_Y[E]}\right| \\
&= \frac{1}{2}\cdot\frac{1}{\Pr_Y[E]}\cdot\sum_u\left|\frac{\Pr_Y[E]}{\Pr_X[E]}\cdot p_{(u,E)} - q_{(u,E)}\right| \\
&\le \frac{1}{2}\cdot\frac{1}{\Pr_Y[E]}\cdot\left(\sum_u\left|\left(\frac{\Pr_Y[E]}{\Pr_X[E]} - 1\right)\cdot p_{(u,E)}\right| + \sum_u\left|p_{(u,E)} - q_{(u,E)}\right|\right) \\
&\le \frac{1}{2}\cdot\frac{1}{\Pr_Y[E]}\cdot\left(\frac{1}{1-\epsilon} - 1 + \sum_u\left|p_{(u,E)} - q_{(u,E)}\right|\right) \\
&\le \frac{1}{2}\cdot\frac{1}{1/2}\cdot\left(\frac{1}{1-\epsilon} - 1 + \delta\right) \\
&= \frac{\epsilon}{1-\epsilon} + \delta \\
&\le 2\cdot(\epsilon+\delta)\,,
\end{aligned}$$

where the first inequality is by the triangle inequality, the second inequality is by our setting of parameters that $1-\epsilon \le \Pr_Y[E], \Pr_X[E] \le 1$, and since $\sum_u \left|p_{(u,E)}\right| \le \Pr_X[E] \le 1$, the third inequality is by plugging Eq. (1) and by our setting of parameters that $\Pr_Y[E] \ge 1-\epsilon \ge \frac{1}{2}$, and the last inequality by our setting of parameters that $\epsilon \le \frac{1}{2}$.

□

Fact 2.6. *For any jointly distributed X_0, X_1, and Y it holds that*

$$\Delta\Big(\big(X_0, Y\big), \big(X_1, Y\big)\Big) = \mathop{\mathrm{E}}_{y\leftarrow Y}\Big[\Delta\big(X_0|_{Y=y}, X_1|_{Y=y}\big)\Big].$$

Proof.

$$\begin{aligned}
\Delta\Big((X_0, Y), (X_1, Y)\Big) &= \frac{1}{2}\sum_{(x,y)} \Big| \Pr[(X_0,Y) = (x,y)] - \Pr[(X_1,Y) = (x,y)] \Big| \\
&= \frac{1}{2}\sum_{(x,y)} \Pr[Y = y] \cdot \Big| \Pr[X_0 = x|Y = y] - \Pr[X_1 = x|Y = y] \Big| \\
&= \sum_{y} \Pr[Y = y] \cdot \frac{1}{2} \cdot \sum_{x} \Big| \Pr[X_0 = x|Y = y] - \Pr[X_1 = x|Y = y] \Big| \\
&= \mathop{\mathrm{E}}_{y \leftarrow Y}\Big[\Delta\big(X_0|_{Y=y}, X_1|_{Y=y} \big) \Big].
\end{aligned}$$

□

2.2 Statistical Zero-knowledge

We use $(\mathsf{P},\mathsf{V})(x)$ to refer to the *transcript* of an execution of an interactive protocol with prover P and verifier V on common input x. The transcript includes the input x, all messages sent by P to V in the protocol and the verifier's random coin tosses. We say that the transcript $\tau = (\mathsf{P},\mathsf{V})(x)$ is accepting if at the end of the corresponding interaction, the verifier accepts.

Definition 2.7 (Interactive proof). *Let $c = c(n) \in [0,1]$ and $s = s(n) \in [0,1]$. An interactive proof with completeness error c and soundness error s for a promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$, consists of a probabilistic polynomial-time verifier V and a computationally unbounded prover P such that following properties hold:*

- **Completeness:** *For any $x \in \Pi_{\mathrm{YES}}$:*

$$\Pr[(\mathsf{P},\mathsf{V})(x) \textit{ is accepting}] \geq 1 - c(|x|).$$

- **Soundness:** *For any (computationally unbounded) cheating prover P^* and any $x \in \Pi_{\mathrm{NO}}$:*

$$\Pr[(\mathsf{P}^*,\mathsf{V})(x) \textit{ is accepting}] \leq s(|x|).$$

We denote this proof system by the pair (P,V).

In this paper we focus on *public-coin* interactive proofs. An interactive proof (P,V) is public-coin if the verifier's messages are selected independently and uniformly at random (and their lengths are fixed independently of the interaction).

We next define *honest-verifier* statistical zero-knowledge proofs, in which zero-knowledge is only guaranteed wrt the *honest* (i.e., prescribed) verifier's behavior.

Definition 2.8 (HVSZK). *Let $z = z(n) \in [0,1]$. An interactive proof system (P,V) is an* Honest Verifier SZK Proof-System (HVSZK)*, with zero-knowledge error z, if there exists a probabilistic polynomial-time algorithm* Sim *(called the* simulator*) such that for any $x \in \Pi_{\mathrm{YES}}$:*

$$\Delta\left((\mathsf{P},\mathsf{V})(x), \mathsf{Sim}(x)\right) \leq z(|x|).$$

For the malicious verifier SZK definition, we allow the verifier access to non-uniform advice. Therefore, we also provide the simulator with the same advice. Let $\mathsf{Sim}_{[a]}$ denote the simulator Sim given access to the some advice string $a \in \{0,1\}^*$.

Definition 2.9 (SZK)**.** *Let $z = z(n) \in [0,1]$. An interactive-proof (P, V) is a* statistical zero-knowledge proof-system (SZK)*, with zero-knowledge error z, if for every probabilistic polynomial-time verifier V^* there exists an algorithm Sim (called the* simulator*) that runs in (strict) polynomial time such that for any $x \in \Pi_{\mathrm{YES}}$ and $a \in \{0,1\}^*$:*

$$\Delta\left((\mathsf{P}, \mathsf{V}^*_{[a]})(x), \mathsf{Sim}_{[a]}(x)\right) \leq z(|x|).$$

If the completeness, soundness and zero-knowledge (resp., honest-verifier zero-knoweldge) errors are all negligible, we simply say that the interactive proof is an SZK (resp., HVSZK) protocol. We also use SZK (resp., HVSZK) to denote the class of promise problems having such an SZK (resp., HVSZK) protocol.

2.2.1 Non-Interactive Statistical Zero-Knowledge Proofs

We also define the *non-interactive* variant of SZK as follows.

Definition 2.10 (NISZK)**.** *Let $c = c(n) \in [0,1]$, $s = s(n) \in [0,1]$ and $z = z(n) \in [0,1]$. An* non-interactive statistical zero-knowledge proof (NISZK) with completeness error c, soundness error s and zero-knowledge error z *for a promise problem $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$, consists of a probabilistic polynomial-time verifier V, a computationally unbounded prover P and a polynomial $\ell = \ell(n)$ such that following properties hold:*

- **Completeness:** *For any $x \in \Pi_{\mathrm{YES}}$:*

$$\Pr_{r \in \{0,1\}^{\ell(|x|)}}[\mathsf{V}(x, r, \pi) \text{ accepts}] \geq 1 - c(|x|),$$

  where $\pi = \mathsf{P}(x, r)$.

- **Soundness:** *For any $x \in \Pi_{\mathrm{NO}}$:*

$$\Pr_{r \in \{0,1\}^{\ell(|x|)}}[\exists \pi^* \text{ s.t. } \mathsf{V}(x, r, \pi^*) \text{ accepts}] \leq s(|x|).$$

- **Honest Verifier Statistical Zero Knowledge:** *There is a probabilistic polynomial-time algorithm Sim (called the* simulator*) such that for any $x \in \Pi_{\mathrm{YES}}$:*

$$\Delta\left((U_\ell, \mathsf{P}(x, U_\ell)), \mathsf{Sim}(x)\right) \leq z(|x|).$$

(Note that the zero-knowledge property in Definition 2.10 is referred to as "honest-verifier" simply because the verifier does not send any messages to the prover and so it is meaningless to consider malicious behavior.)

As above, if the errors are negligible, we say that Π has a NISZK protocol and use NISZK to denote the class of all such promise problems.

We note that *sequential* repetition reduces the completeness and soundness errors of zero-knowledge proofs at an exponential rate, while (roughly) preserving its zero-knowledge. The former fact is easy to see, whereas the second was shown by Goldreich and Oren [GO94] while relying on the auxiliary-input definition of zero-knowledge.[6]

Lemma 2.11 ([GO94])**.** *Let Π be a statistical zero-knowledge proof-system with constant completeness and soundness errors and with zero-knowledge error z. Let $\Pi^{\otimes k}$ denote the protocol obtained by repeating Π sequentially k times and having the verifier decide based on majority vote. Then, $\Pi^{\otimes k}$ has completeness and soundness errors $2^{-\Omega(k)}$ and has zero-knowledge error at most kz.*

2.3 Many-wise independence

Definition 2.12 (δ-almost ℓ-wise Independent Hash Functions)**.** *For $\ell = \ell(n) \in \mathbb{N}$, $m = m(n) \in \mathbb{N}$ and $\delta = \delta(n) > 0$, a family of functions $\mathcal{F} = (\mathcal{F}_n)_n$, where $\mathcal{F}_n = \left\{ f : \{0,1\}^m \to \{0,1\}^n \right\}$ is* δ-almost ℓ-wise independent *if for every $n \in \mathbb{N}$ and distinct $x_1, x_2, \ldots, x_\ell \in \{0,1\}^m$ the distributions:*

- *$(f(x_1), \ldots, f(x_\ell))$, where $f \leftarrow \mathcal{F}_n$; and*
- *The uniform distribution over $(\{0,1\}^n)^\ell$,*

are δ-close in statistical distance.

When $\delta = 0$ we simply say that the hash function family is ℓ-wise independent. Constructions of (efficiently computable) many-wise hash function families with a very succinct representation are well known. In particular, when $\delta = 0$ we have the following well-known construction:

Lemma 2.13 (See, e.g., [Vad12, Section 3.5.5])**.** *For every $\ell = \ell(n) \in \mathbb{N}$ and $m = m(n) \in \mathbb{N}$ there exists a family of ℓ-wise independent hash functions $\mathcal{F}^{(\ell)}_{n,m} = \{f \colon \{0,1\}^m \to \{0,1\}^n\}$ where a random function from $\mathcal{F}^{(\ell)}_{n,m}$ can be selected using $O\big(\ell \cdot \max(n,m)\big)$ bits, and given a description of $f \in \mathcal{F}^{(\ell)}_{n.m}$ and $x \in \{0,1\}^m$, the value $f(x)$ can be computed in time $\mathsf{poly}(n,m,\ell)$.*

For $\delta > 0$, the seminal work of Naor and Naor [NN93] yields a highly succinct construction.

Lemma 2.14 ([NN93, Lemma 4.2])**.** *For every $\ell = \ell(n) \in \mathbb{N}$, $m = m(n) \in \mathbb{N}$ and $\delta = \delta(n) > 0$, there exists a family of δ-almost ℓ-wise independent hash functions $\mathcal{F}^{(\ell)}_{n,m} = \{f \colon \{0,1\}^m \to \{0,1\}^n\}$ where a random function from $\mathcal{F}^{(\ell)}_{n,m}$ can be selected using $O\big(\ell \cdot n + \log(m) + \log(1/\delta)\big)$ bits, and given a description of $f \in \mathcal{F}^{(\ell)}_{n.m}$ and $x \in \{0,1\}^m$, the value $f(x)$ can be computed in time $\mathsf{poly}(n,m,\ell,\log(1/\delta))$.*

2.4 Tail Bounds

We will also use a tail bound for random variables with limited independence. We first recall Chebyshev's inequality.

Lemma 2.15 (Chebyshev's inequality)**.** *Let X be a random variable, and let $\mu = E[X]$. Then, for every $A > 0$ it holds that*

$$\Pr\left[|X - \mu| \geq A\right] \leq \frac{\mathrm{Var}\left[X\right]}{A^2}.$$

[6]To be more precise, [GO94] prove this statement for CZK but their proof extends readily to SZK.

Chebyshev's inequality is often used to derive a tail bound for sums of pairwise independent random variables. This application readily extends to *almost* pairwise independent random variables.

Lemma 2.16. *Suppose that $X_1, \dots, X_n$ are δ-almost pairwise independent identically distributed Bernoulli random variables, with success probability p. Then,*

$$\Pr\left[|X-\mu| \geq A\right] \leq \frac{n \cdot p \cdot (1-p) + n^2 \cdot \delta}{A^2},$$

where $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X] = n \cdot p$.

Proof. Since the random variables for δ-almost pairwise independent, for every distinct $i, j \in [n]$ it holds that

$$\mathrm{E}[x_i \cdot x_j] = \Pr[x_i = 1 \text{ and } x_j = 1] \leq p^2 + \delta.$$

Therefore,

$$\begin{aligned}
\mathrm{Var}(X) &= \mathrm{E}[X^2] - \mathrm{E}[X]^2 \\
&= \mathrm{E}\left[\big(\sum_{i=1}^{n} X_i\big)^2\right] - \mathrm{E}\left[\big(\sum_{i=1}^{n} X_i\big)\right]^2 \\
&= \sum_{i} \mathrm{E}[x_i^2] + \sum_{i \neq j} \mathrm{E}[x_i \cdot x_j] - n^2 \cdot p^2 \\
&\leq n \cdot p + (n^2 - n) \cdot (p^2 + \delta) - n^2 \cdot p^2 \\
&\leq n \cdot p \cdot (1-p) + n^2 \cdot \delta,
\end{aligned}$$

and the lemma follows from Chebyshev's inequality (Lemma 2.15). □

We now continue to the tail bound from [BR94] of ℓ-wise independent functions.

Theorem 2.17. *Let $\ell \geq 4$ be an even integer. Suppose $X_1, ..., X_n \in \{0,1\}$ are ℓ-wise independent random variables. Let $X = X_1 + ... + X_n$, $\mu = E[X]$, and let $A > 0$. Then*

$$\Pr\left[|X-\mu| \geq A\right] \leq 8 \left(\frac{\ell \cdot \mu + \ell^2}{A^2}\right)^{\frac{\ell}{2}}.$$

2.5 Extractors

Recall that a random variable X is a k-source if $H_\infty(X) \geq k$, i.e., if $\Pr[X = x] \leq 2^{-k}$ for every x. Recall that U_ℓ denotes a random variable uniformly distributed over $\{0,1\}^\ell$.

Definition 2.18. *The function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* strong (k, ϵ)-extractor *if for every k-source X supported on $\{0,1\}^n$, the distribution $(U_d, \mathsf{Ext}(X, U_d))$ is ϵ-close to (U_d, U_m).*

We now continue to show a variant of the leftover hash lemma [HILL99], for *almost* pairwise independent hash functions. The proof follows [Vad12, Theorem 6.18].

Lemma 2.19 ((Almost) Leftover Hash Lemma)**.** *If $H = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is a δ-almost pairwise independent family of hash functions for some $\delta \in [0,1]$, where $m \leq k - 2\log(1/\epsilon)$ for some ϵ, then $\mathsf{Ext}(x,h) \overset{def}{=} h(x)$ is a strong (k,ϵ')-extractor, where $\epsilon' = \frac{1}{2} \cdot \sqrt{\epsilon^2 + \delta \cdot 2^m}$.*

Proof. Let X be a k-source, and let d be the description size of $h \in H$. Also, let $D = 2^d$, $M = 2^m$ and $K = 2^k$. Recall that for a probability distribution X, the collision probability of X, denoted by $\mathsf{CP}(X)$, is defined to be $\Pr_{x_1,x_2 \leftarrow X}[x_1 = x_2]$.

By definition, $CP(H, H(X)) = \Pr[(H, H(X)) = (H', H(X'))]$, where (H', X') is identically distributed and independent of (H, X). It holds that $(H', H'(X')) = (H, H(X))$ if and only if $H' = H$ and $H(X') = H(X)$. Thus,

$$\begin{aligned} CP(H, H(X)) &= CP(H) \cdot \big(CP(X) + \Pr[H(X) = H(X') | X \neq X']\big) \\ &\leq \frac{1}{D} \cdot \left(\frac{1}{K} + \frac{1}{M} + \delta\right) \\ &\leq \frac{1+\epsilon^2}{D \cdot M} + \frac{\delta}{D}, \end{aligned}$$

where the first inequality is since X is a k-source and since H is a family of δ-almost pairwise independent hash functions.

Using the fact that $CP(X) = \|X - u\|_2^2 + \frac{1}{N}$, where u is the uniform distribution on $[N]$ (see [Vad12, Lemma 4.8]), we have

$$\begin{aligned} \|(H, H(X)) - U_d \times U_m\|_2^2 &= CP(H, H(X)) - \frac{1}{D \cdot M} \\ &\leq \frac{1+\epsilon^2}{D \cdot M} + \frac{\delta}{D} - \frac{1}{D \cdot M} \\ &= \frac{\epsilon^2}{D \cdot M} + \frac{\delta}{D}, \end{aligned} \tag{2}$$

Therefore,

$$\begin{aligned} \Delta((H, H(X)), U_d \times U_m) &= \frac{1}{2} \|(H, H(X)) - U_d \times U_m\|_1 \\ &\leq \frac{\sqrt{D \cdot M}}{2} \cdot \|(H, H(X)) - U_d \times U_m\|_2 \\ &\leq \frac{\sqrt{D \cdot M}}{2} \cdot \sqrt{\frac{\epsilon^2}{D \cdot M} + \frac{\delta}{D}} \\ &= \frac{1}{2} \cdot \sqrt{\epsilon^2 + \delta \cdot M}, \end{aligned}$$

where the first inequality is by a standard norm inequality, and the second is by Eq. (2). □

2.6 Round-By-Round Soundness

In this section we define the notion of *round-by-round soundness* of interactive proofs, as introduced in the recent work of Canetti *et al.* [CCH+19].

Let (P, V) be a public-coin interactive proof. We denote by $V(x, \tau)$ the distribution of the next message (or output) of V on the input x and partial transcript τ.

Definition 2.20. *Let* (P, V) *be a public-coin interactive proof for the promise problem* $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$.

We say that (P, V) *has a* round-by-round soundness error $\epsilon = \epsilon(n)$ *if there exists some (possibly inefficient) function* State *that takes as input the main input* x *and a partial transcript* τ *and outputs either* `alive` *or* `doomed` *and has the following properties:*

1. *If* $x \in \text{NO}$*, then* $\mathsf{State}(x, \bot) = \texttt{doomed}$ *(where* $\bot$ *denotes the empty transcript).*
2. *For any transcript prefix* τ*, if* $\mathsf{State}(x, \tau) = \texttt{doomed}$*, then for any prover message* α *it holds that*
$$\Pr_{\beta \leftarrow V(x,\tau,\alpha)}[\mathsf{State}(x, \tau, \alpha, \beta) = \texttt{alive}] \leq \epsilon(n).$$
3. *For any full transcript* τ *(i.e, a transcript in which the verifier halts) such that* $\mathsf{State}(x, \tau) = \texttt{doomed}$*, it holds that* $V(x, \tau)$ *is rejecting.*

Canetti *et al.* [CCH+19] also show the following simple fact (which follows from the union bound).

Fact 2.21. *Let* (P, V) *be a* $2r$*-message interactive proof with round-by-round soundness error* ϵ*. Then,* (P, V) *has standard soundness error* $r \cdot \epsilon$*.*

3 Load-balancing Functions

We now define *load-balancing hash functions*, a central tool in our construction. Loosely speaking, a load balanching hash function is a function mapping a set $\{0,1\}^m$ together with a short auxiliary random string $\{0,1\}^d$ to a range $\{0,1\}^n$. The key property that we seek is that for every subset of $\{0,1\}^m$ of size roughly 2^n it holds that every element $x \in \{0,1\}^n$ has roughly the same number of preimages $(y, z) \in S \times \{0,1\}^d$.

Definition 3.1 (Load Balancing Hash Function Family)**.** *Let* $m = m(n) \in \mathbb{N}$*,* $d = d(n) \in \mathbb{N}$ *and* $\epsilon : \mathbb{N}^4 \to [0,1]$*. We say that a family of hash functions* $G = (G_n)_n$*, where* $G_n = \left\{g : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^n\right\}$*, is* (d, ϵ)-load-balancing*, if for every* $n \in \mathbb{N}$*, number of elements* $v \in \mathbb{N}$*, and set* $S \subseteq \{0,1\}^{m(n)}$ *of size* $|S| \leq 2^n$ *it holds that:*
$$\Pr_{g \leftarrow G}\left[\exists x \in \{0,1\}^n : \left| L_{S,g}(x) - \frac{|S| \cdot 2^d}{2^n} \right| > v\right] \leq \epsilon(n, |S|, v, d),$$
where $L_{S,g}(x) = \left|\left\{(y, z) \in S \times \{0,1\}^d \ : \ g(y, z) = x\right\}\right|$*.*

Lemma 3.2. *For any values* $n, \lambda \in \mathbb{N}$ *and function* $,m = m(n), d = d(n)$ *, there is an explicit family of hash functions* $G = (G_n)_n$*, where* $G_n = \left\{g : \{0,1\}^{m(n)} \times \{0,1\}^{d(n)} \to \{0,1\}^n\right\}$*, that is* $\left(d, \lambda, \epsilon\right)$*-load-balancing, where*
$$\epsilon_\lambda(n, |S|, v, d) = 2^n \cdot \left(64 \cdot (n + \lambda) \cdot \frac{\mu + (n + \lambda)}{v^2}\right)^{n+\lambda+4} + 2^{-\lambda-1},$$
where $\mu = \frac{|S| \cdot 2^d}{2^n}$*, s.t. a random function in the family can be sampled using* $O(n^2 + \lambda^2 + d \cdot (n + \lambda))$ *uniformly random bits, and each such function can be evaluated in time* $\mathsf{poly}(n, m, d, \lambda)$*.*

Proof. We define a function family $G = (G_n)_n$ where each function $g \leftarrow G_n$ is described by a pair of functions (h, g'). The function g is defined as $g(y, z) = g'(h(y), z)$ and the functions h and g' are sampled as follows:

- $h \leftarrow H$, where $H = \left\{h : \{0,1\}^m \to \{0,1\}^{2n+\lambda+2}\right\}$ is a family of $\frac{1}{2^{2n+\lambda+2}}$-almost pairwise independent functions,
- $g' \leftarrow G'$ and $G' = \left\{g' : \{0,1\}^{2n+\lambda+2} \times \{0,1\}^d \to \{0,1\}^n\right\}$ is a family of r-wise independent functions, where $r = 2 \cdot (n + \lambda + 4)$.

Let $S \subseteq \{0,1\}^m$ s.t. $2^{n-\ell} \le |S| \le 2^n$. We first show that whp, h is injective on S.

Claim 3.2.1. *With probability at least $1 - 2^{-(\lambda+1)}$. it holds that $h \leftarrow H$, restricted to the domain S, is injective.*

Proof. For every distinct $x_1, x_2 \in S$, since H is $\frac{1}{2^n}$-almost pairwise independent, the probability that they collide is at most $2 \cdot 2^{-2n-\lambda-2}$. Claim 3.2.1 follows by taking a union over all $\binom{|S|}{2} \le 2^{2n}$ such pairs. □

Thus, we may assume that h is injective on S.

We proceed to bound the probability that the number of elements that are mapped to a certain image is in the range of $(1 \pm \epsilon) \cdot \frac{|S| \cdot 2^d}{2^n}$.

Let $x \in \{0,1\}^n$, and let $X = |g'^{-1}(x)|$ be a random variable (over the probability space defined by choosing $g' \in G'$). Observe that X can be expressed as a sum of $|S| \cdot 2^d$ random variables that are r-wise independent and $\mu = E[X]$. Therefore, By the tail inequality of Theorem 2.17

$$\begin{aligned}\Pr\left[|X - \mu| \ge v\right] &\le 8 \cdot \left(\frac{r \cdot \mu + r^2}{v^2}\right)^{\frac{r}{2}} \\ &= 8 \cdot \left(\frac{2 \cdot (n+\lambda+4) \cdot \mu + 4 \cdot (n+\lambda+4)^2}{v^2}\right)^{n+\lambda+4} \\ &\le \left(64 \cdot (n+\lambda) \cdot \frac{\mu + (n+\lambda)}{v^2}\right)^{n+\lambda+4}. \qquad (3)\end{aligned}$$

Taking a union bound over all the 2^n images gives us that the probability of this condition to happen over all bins is less than 2^n times Eq. (3).

In total, the required probability is a result of a union bound over the events that h is injective on S and that there is a satisfying amount of preimages in every image.

Representation Length. By Lemma 2.14, the representation length of $h \in H : \{0,1\}^m \to \{0,1\}^{2n+\lambda+2}$, an $\frac{1}{2^n}$-almost pairwise independent function is $\log\log(2^m) + 2 \cdot \log(2^{2n+\lambda+2}) = O(\log m + n + \lambda)$. The representation length of $g' \in \{0,1\}^{2n+\lambda+2} \times \{0,1\}^d \to \{0,1\}^n$, an r-wise independent hash function, is $\log(2^{2n+\lambda+2+d}) \cdot 2(n + \lambda + 4) = O(n^2 + \lambda^2 + d \cdot (n + \lambda))$.

Evaluation Time. Evaluating h on an input of size m takes $\mathsf{poly}(m)$. Evaluating g' on an input of size $O(n \cdot d)$ takes $\mathsf{poly}(n, \lambda, d)$. In total, g can also be computed in time $\mathsf{poly}(m, d)$.

$\square$

Corollary 3.3. *For any $n, m(n), \lambda, \ell, \epsilon' \in (0,1]$ and $d \geq 3 \log\left(\frac{n+\lambda}{\epsilon'^2}\right) + \ell$, the family of hash functions from Lemma 3.2 has the following properties:*

1. *For any set $S \subseteq \{0,1\}^m$ s.t. $2^{n-\ell} \leq |S| \leq 2^n$ it holds that*
$$\Pr_{g \leftarrow G}\left[\exists x \in \{0,1\}^n : \left| L_{S,g}(x) - \frac{|S| \cdot 2^d}{2^n} \right| > \frac{|S| \cdot 2^d}{2^n} \cdot \epsilon' \right] \leq 2^{-\lambda}.$$

2. *For every ν s.t. $12 \cdot (n + \lambda) \leq \nu$ and set $S \subseteq \{0,1\}^m$ s.t. $|S| \leq 2^{n-d}$ it holds that*
$$\Pr_{g \leftarrow G}\left[\exists x \in \{0,1\}^n : L_{S,g}(x) - \frac{|S| \cdot 2^d}{2^n} > v \right] \leq 2^{-\lambda}.$$

4 Approximate Injectivity

In this section we analyze the *approximate injectivity* problem, introduced by Kaslasi *et al.* [KRR+20]. In particular, we consider a variant in which NO cases are (approximately) many-to-one and show that it is NISZK-complete.

We say that x' is a *sibling* of x, with respect to the circuit C, if $C(x) = C(x')$. We omit C from the notation if it is clear from the context.

Definition 4.1. *The problem $\mathsf{AI}^{n,m}_{L,\delta}$ is defined as the promise problem of circuits with n input bits and m output bits, where*
$$(\mathsf{AI}^{n,m}_{L,\delta})^Y = \left\{ \text{circuit } C : \Pr_x \left[|C^{-1}(C(x))| > 1 \right] < \delta \right\},$$
and
$$(\mathsf{AI}^{n,m}_{L,\delta})^N = \left\{ \text{circuit } C : \Pr_x \left[|C^{-1}(C(x))| < L \right] < \delta \right\}.$$
We omit m and n from the notation when they are clear from the context.

To show that $\mathsf{AI}_{L,\delta}$ is NISZK-hard, we rely on the fact that it is known to be NISZK-hard in the special case when $L = 2$.

Lemma 4.2 ([KRR+20])**.** *Let $\delta = \delta(n) \in [0,1]$ be a non-increasing function such that $\delta(n) > 2^{-o(n^{1/4})}$. Then, $\mathsf{AI}_{2,\delta}$ is NISZK-hard.*

Thus, to show that $\mathsf{AI}_{L,\delta}$ is NISZK-hard, it suffices to reduce $\mathsf{AI}_{2,\delta}$ to $\mathsf{AI}_{L,\delta}$.

Lemma 4.3. *For every parameter $\ell = \mathsf{poly}(n)$, there exists a polynomial time Karp-reduction from $\mathsf{AI}^{n,m}_{2,\delta}$ to $\mathsf{AI}^{n \cdot \ell, m \cdot \ell}_{2^\ell, \ell \cdot \delta}$.*

Before proving Lemma 4.3, we observe that Lemma 4.2 together with Lemma 4.3 immediately implies that $\mathsf{AI}_{L,\delta}$ is NISZK-hard. Since $\mathsf{AI}_{2,\delta}$ is a special case of $\mathsf{AI}_{L,\delta}$, we get also that $\mathsf{AI}_{L,\delta}$ is NISZK-complete.

Corollary 4.4. *Let $\delta = \delta(n) \in [0,1]$ be a non-increasing function and $\ell = \mathsf{poly}(n)$ such that $\delta(n) > \frac{2^{-o(n^{1/4})}}{\ell}$. Then, there exist constants $c, d \in \mathbb{N}$ such that $\mathsf{AI}_{2^\ell,\delta}^{n^c,m^d}$ is NISZK-complete.*

We proceed to prove Lemma 4.3.

Proof of Lemma 4.3. Given a circuit C, the reduction outputs the circuit $\bar{C}(x_1,\dots,x_\ell) = \big(C(x_1),\dots,C(x_\ell)\big)$. The new input size is $n \cdot \ell$, and the new output size is $m \cdot \ell$. We show that this reduction maps YES instances to YES instances and NO instances to NO instances.

YES Instances. Let $C \in \mathsf{AI}_{2,\delta}^{n,m}$ be a YES instance. We denote the set of preimages in C (resp. $\bar{C}$) that have no siblings other than themselves by $X \subseteq \{0,1\}^n$ (resp. $\bar{X} \subseteq \{0,1\}^{\ell \cdot n}$). Since C is a YES instance, it holds that $|X| \geq (1-\delta)\cdot 2^n$.

Claim 4.4.1. *$X^\ell \subseteq \bar{X}$.*

Proof. Suppose not. Then there exists $\bar{x} = (x_1,\dots,x_\ell) \in X^\ell$ s.t. $\bar{x} \notin \bar{X}$. Since $\bar{x} \notin \bar{X}$, there exists $\bar{x}' = (x'_1,\dots,x'_\ell)$ s.t. $\bar{x} \neq \bar{x}'$ and $\bar{C}(\bar{x}) = \bar{C}(\bar{x}')$. Thus, there is at least one index i for which $x_i \neq x'_i$, but $C(x_i) = C(x'_i)$. Hence, $x_i \notin X$, in contradiction to the fact that $\bar{x} \in X^\ell$. $\square$

Using Claim 4.4.1, It holds that $\left|\bar{X}\right| \geq \left|X^\ell\right| = |X|^\ell \geq (2^n \cdot (1-\delta))^\ell$. Thus,

$$\begin{aligned}
\Pr_{\bar{x}\leftarrow\{0,1\}^{\ell\cdot n}}\left[\left|\bar{C}^{-1}(\bar{C}(\bar{x}))\right| \leq 1\right] &= \Pr_{\bar{x}\leftarrow\{0,1\}^{\ell\cdot n}}\left[\bar{x} \in \bar{X}\right] \\
&= \frac{\left|\bar{X}\right|}{2^{n\cdot\ell}} \\
&\geq \frac{|X|^\ell}{2^{n\cdot\ell}} \\
&\geq \frac{(2^n\cdot(1-\delta))^\ell}{2^{n\cdot\ell}} \\
&= (1-\delta)^\ell \\
&\geq 1-\ell\cdot\delta,
\end{aligned}$$

and so $\bar{C}$ is a YES instance of $\mathsf{AI}_{2^\ell,\ell\cdot\delta}^{n\cdot\ell,m\cdot\ell}$.

NO Instances. Let $C \in \mathsf{AI}_{2,\delta}^{n,m}$ be a NO instance. We need to lower bound the number of preimages in the new circuit $\bar{C}$ which have at least 2^ℓ siblings. We denote the set of preimages having at least one sibling (other than themselves) in the original circuit C by $X_+ \subseteq \{0,1\}^n$. Since C is a NO instance, it holds that $|X_+| \geq (1-\delta)\cdot 2^n$.

Claim 4.4.2. *Every $\bar{x} \in (X_+)^\ell$ has at least 2^ℓ siblings.*

Proof. Consider some $\bar{x} = (x_1,\dots,x_\ell) \in (X_+)^\ell$. For every $i \in [\ell]$, since $x_i \in X_+$, there is at least one other preimage $\hat{x}_i \neq x_i$ that is mapped to the same image by C. Therefore, it holds that every $\bar{x}' \in (\{x_1,\hat{x}_1\} \times \cdots \times \{x_\ell,\hat{x}_\ell\})$ has the same image in the new circuit $\bar{C}$. Since this set has cardinality 2^ℓ, we have that $\bar{x}$ has at least 2^ℓ siblings. $\square$

Thus,

$$\begin{aligned}\Pr_{x' \leftarrow \{0,1\}^{\ell \cdot n}}\left[\left|\bar{C}^{-1}(\bar{C}(x))\right| \geq 2^{\ell}\right] &\geq \Pr_{x' \leftarrow \{0,1\}^{\ell \cdot n}}\left[x' \in (X_+)^{\ell}\right] \\ &= \frac{|X_+|^{\ell}}{2^{n \cdot \ell}} \\ &\geq \frac{(2^n \cdot (1-\delta))^{\ell}}{2^{n \cdot \ell}} \\ &= (1-\delta)^{\ell} \\ &\geq 1 - \ell \cdot \delta,\end{aligned}$$

and so $\bar{C}$ is a NO instance of $\mathsf{AI}_{2^{\ell},\ell \cdot \delta}^{n \cdot \ell, m \cdot \ell}$. □

5 Public-coin Batch Verification for $\mathsf{AI}_{\mathsf{L},\delta}$

In this section we prove the following lemma by showing a *public-coin* HVSZK protocol for batch verification of $\mathsf{AI}_{L,\delta}$ (as defined in Definition 4.1).

Lemma 5.1. *Let $\delta = \delta(n) \in [0,1]$ and $k = k(n) \in \mathbb{N}$. Also, let $\lambda = \lambda(n) \in \mathbb{N}$ be a security parameter and let $\ell = \ell(n) \in \mathbb{N}$, with $\ell(n) \geq \lambda(n)$ for all n. Set $d = 7 \cdot (\log n + \log k + \lambda) + \ell$ and assume that $\delta \leq 2^{-d}$ and $d < 2\ell - 2\lambda$.*

Then, $\mathsf{AI}_{2^{\ell},\delta}^{\otimes k}$ has an HVSZK public-coin protocol with completeness error $2^{-\lambda+1}$, round-by-round soundness error $2^{-\lambda}$ and statistical zero knowledge error $k \cdot (\delta \cdot 2^{d+2} + 2^{-\lambda+6})$. The communication complexity is $O(n^2) + k \cdot \mathsf{poly}(\log n, \log k, \lambda)$ and the verifier running time is $k \cdot \mathsf{poly}(n, \log k, \lambda)$.

Furthermore, the protocol consists of k rounds. The length of the verifier's first message is $O(n^2 + \ell \cdot n \cdot \mathsf{poly}(\log n, \log k, \lambda)$ and the length of all other verifier messages is $\mathsf{poly}(\log n, \log k, \lambda)$.

The protocol establishing Lemma 5.1 is presented in Fig. 3. The rest of this section is devoted to proving that the protocol indeed satisfies the requirements of that lemma.

Notational Convention. The loop in Step 2 of the protocol operates in *reverse order*, from $i = k$ down to $i = 1$. We emphasize that throughout the proof, by "iteration $i = j$", we refer to the iteration in which the counter i is equal to j (which is the $(k - j + 1)$-th iteration).

Section Organization. First, in Section 5.1 we prove some general lemmas that will be useful throughout the analysis. Then, in the following sections we prove completeness (Section 5.2), round-by-round soundness (Section 5.3) and statistical zero-knowledge (Section 5.4). Last, in Section 5.5 we analyze the communication complexity and the verifier run time .

5.1 Useful Lemmas

Let $C_1, \ldots, C_k : \{0,1\}^n \to \{0,1\}^m$ be instances of $\mathsf{AI}_{2^{\ell},\delta}$. We denote by $i^* \in \{0, \ldots, k\}$ the maximal index i s.t. C_i is a NO instance, and $i^* = 0$ if no such index exists (the latter holds if all k circuits are YES instances).

For every $i \in [k]$, we denote by S_i the set of images of the circuit C_i, and by S_i^+ the set of images of C_i that have more than one preimage. We will use the fact that for $i > i^*$, it holds that

Public-coin HVSZK Batching Protocol for $\mathsf{AI}_{2^\ell,\delta}$.

Parameters: input length n, output length m, number of instances k, security parameter λ, arity ℓ and seed length $d = 7 \cdot (\log n + \log k + \lambda) + \ell$.

Input: Circuits $C_1, \ldots, C_k : \{0,1\}^n \to \{0,1\}^m$, where all circuits[a] have size at most N, input length n, and output length $m \le N$.

Ingredients:

- Let $G = (G_n)_n$, where $G_n = \left\{g : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^n\right\}$, be the explicit family of load-balancing functions from Lemma 3.2, with seed length d and accuracy $2^{-\lambda}$ with respect to security parameter $\lambda + \log k + 1$.
- Let $H = (H_n)_n$, where $H_n = \left\{h : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^{d/2}\right\}$, be the explicit family of $2^{-3d/2}$-almost pairwise independent hash function from Lemma 2.14.
- Let $F = (F_n)_n$, where $F_n = \left\{f : \{0,1\}^n \to \{0,1\}^{2d+\lambda+\log k}\right\}$, be the explicit family of $2^{-(2d+\lambda+\log k)}$-almost pairwise independent hash functions from Lemma 2.14.

The Protocol:

1. V samples $g \leftarrow G_n$ and $x_{k+1} \leftarrow \{0,1\}^n$ and sends both to P.
2. For $i = k, ..., 1$:
   (a) V samples $\alpha_i \leftarrow \{0,1\}^{d/2}$, $h_i \leftarrow H_n$, and $f_i \leftarrow F_n$, and sends (α_i, h_i, f_i) to P.
   (b) P generates the set
   $$XZ_i = \left\{(x,z) \in \{0,1\}^n \times \{0,1\}^d \ : \ g(C_i(x), z) = x_{i+1} \text{ and } h_i(C_i(x), z) = \alpha_i\right\}.$$
   (c) P samples $(x_i, z_i) \leftarrow XZ_i$, and sends (z_i, β_i) to V, where $\beta_i = f_i(x_i)$. In case the set XZ_i is empty, P sends an arbitrary[b] $(z_i, \beta_i) \in \{0,1\}^d \times \{0,1\}^{2d+\lambda+\log k}$.
   We denote the pair received by V by (z_i', β_i') (allegedly equal to (z_i, β_i)).
3. P sends x_1 to V.
4. V receives $x_1' \in \{0,1\}^n$ (allegedly equal to x_1) and computes:
   (a) For $i = 1, \ldots, k$:
      i. $y_i' = C_i(x_i')$
      ii. $x_{i+1}' = g(y_i', z_i')$
5. V checks that $x_{k+1} = x_{k+1}'$ and that $\forall i \in [k]$, $\beta_i' = f_i(x_i')$ and $\alpha_i = h_i(y_i', z_i')$.
6. If all of V's checks passed then she accepts. Otherwise she rejects.

[a]The circuits can be trivially modified to have the same output length $m \le N$ by padding.

[b]Alternatively, we could simply have the prover abort in this case. However, it will be more convenient for our analysis that P send an arbitrary (z_i, β_i) pair rather than sending a special abort symbol.

Figure 3: A Public-coin HVSZK Batching Protocol for $\mathsf{AI}_{2^\ell,\delta}$

$\left|S_i^+\right| \le \delta \cdot 2^n$ and also, since at least $(1-\delta)\cdot 2^n$ preimages of C_i are mapped uniquely, it also holds that $\left|S_i \setminus S_i^+\right| \ge 2^n \cdot (1-\delta) \ge 2^{n-\ell}$ (where the last inequality uses the fact that $\delta \le \frac{1}{2}$ and $\ell \ge 1$).

The following definition captures the properties that we seek from the function g.

Definition 5.2. *We say that $g : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^n$ is* good *if the following conditions hold:*

1. *For every $i \in \{i^*+1, \dots, k\}$ and every $x \in \{0,1\}^n$:*

$$\frac{2^d \cdot \left|S_i \setminus S_i^+\right|}{2^n} \cdot (1-2^{-\lambda}) \le \left|g^{-1}(x) \cap \left((S_i \setminus S_i^+) \times \{0,1\}^d\right)\right| \le \frac{2^d \cdot \left|S_i \setminus S_i^+\right|}{2^n} \cdot (1+2^{-\lambda}).$$

2. *For every $i \in \{i^*+1, \dots, k\}$ and every $x \in \{0,1\}^n$:*

$$\left|g^{-1}(x) \cap \left(S_i^+ \times \{0,1\}^d\right)\right| < 12 \cdot (n + \lambda + \log k).$$

3. *If $i^* \ge 1$, then for every $x \in \{0,1\}^n$:*

$$\left|g^{-1}(x) \cap \left(S_{i^*} \times \{0,1\}^d\right)\right| < 2^{d-\ell+2}.$$

The first condition states that for every $i > i^*$, every $x \in \{0,1\}^n$ has roughly the same number of preimages (y,z), with y having a unique image under the circuit C_i. The second condition states the complentary fact: every $x \in \{0,1\}^n$ does not have many preimages (y,z) such that y has more than one preimage under g. Lastly, the third condition gives an upper bound on the number of preimages (y,z) where y has a unique preimage, but this time under the last NO instance circuit (i.e., C_{i^*}).

Lemma 5.3. *A random $g \in G_n$ is good with probability at least $1-2^{-\lambda}$.*

Proof. For every $i > i^*$, the circuit C_i is a YES instance and so $\left|S_i \setminus S_i^+\right| \ge 2^{n-\ell}$ and $\left|S_i^+\right| \le \frac{\delta}{2} \cdot 2^n \le 2^{n-d}$ (where we use that fact that $\delta \le 2^{-d}$). Therefore, by Corollary 3.3 and the fact that g is a load-balancing hash function, for every $i > i^*$, the probability that Conditions 1 and 2 hold is at least $1 - 2 \cdot 2^{-\lambda - \log k - 1}$.

We proceed to the third condition and recall that it is only required to hold if $i^* \ge 1$. For every $x \in \{0,1\}^n$, we need to bound the number of pairs (y,z) s.t. y is an image of C_{i^*} and $g(y,z) = x$. Since C_{i^*} is a NO instance, it holds that $|S_{i^*}| \le 2^{n-\ell} + \delta \cdot 2^n$. We add additional arbitrary pairs of the form $(y,z) \in \{0,1\}^m \times \{0,1\}^d$ to S_{i^*} in order to form a set S'_{i^*} of size exactly $2^{n-\ell} + \delta \cdot 2^n \ge 2^{n-\ell}$. Since g is load-balancing, by Corollary 3.3, with probability at least $1 - 2^{-\lambda - \log k - 1}$, for every $x \in \{0,1\}^n$ there are at most $\frac{2^d \cdot (2^{n-\ell} + \delta \cdot 2^n)}{2^n} \cdot (1 + 2^{-\lambda}) \le 2^{d-\ell+2}$ pairs from S'_{i^*} (and therefore also from S_{i^*}) mapped to every x (where we used the fact that $\delta \le 2^{-d} \le 2^{-\ell}$).

Overall, by taking a union bound over the three conditions and at most k circuits, we get that g is good with probability at least

$$1 - (k - i^*) \cdot 2 \cdot 2^{-\lambda - \log k - 1} - 2^{-\lambda - \log k - 1} > 1 - 2^{-\lambda}.$$

□

We proceed to show a couple of useful properties of good function, which will be instrumental for our analysis in the subsequent subsections.

Proposition 5.4. *Assume that g is good, then $\Delta\left(g\big(S_i \setminus S_i^+, U_d\big), U_n\right) \leq 2^{-\lambda}$, for every $i \in \{i^* + 1, \ldots, k\}$.*

Proof. By Condition 1 of Definition 5.2, for every $x \in \{0,1\}^n$ it holds that $\frac{1-2^{-\lambda}}{2^n} \leq \Pr[g\big(S_i \setminus S_i^+, U_d\big) = x] \leq \frac{1+2^{-\lambda}}{2^n}$. Therefore, by definition of statistical difference it holds that

$$\begin{aligned}
\Delta\left(g\big(S_i \setminus S_i^+, U_d\big), U_n\right) &= \sum_x \left|\Pr[g\big(S_i \setminus S_i^+, U_d\big) = x] - \Pr[U_n = x]\right| \\
&\leq \sum_x \left|2^{-n} + 2^{-\lambda-n} - 2^{-n}\right| \\
&= \sum_x 2^{-\lambda-n} \\
&= 2^{-\lambda}.
\end{aligned}$$

□

Lemma 5.5. *Assume that g is good. Then, for every YES instance circuit C_i where $i > i^*$ and for every $x_{i+1} \in \{0,1\}^n$, with probability at least $1 - 2^{-\lambda - \log k}$ over uniformly distributed $h_i \in H_n$ and $\alpha_i \in \{0,1\}^{d/2}$, the set*

$$XZ_i = \left\{(x,z) \in \{0,1\}^n \times \{0,1\}^d \ : \ g(C_i(x), z) = x_{i+1} \text{ and } h_i(C_i(x), z) = \alpha_i\right\}$$

is non-empty.

Proof. Since g is good, by the discussion in the beginning of the subsection, the size of the set $\left(S_i \times \{0,1\}^d\right) \cap g^{-1}(x_{i+1})$ is at least $2^d \cdot (1-\delta) \cdot (1 - 2^{-\lambda}) \geq 2^{d-1}$. Fix $\alpha_i \in \{0,1\}^{d/2}$. We denote by X the number of pairs (y_i, z_i) in the above set that are mapped by h_i to α_i.

Let $\nu = |(S_i \times \{0,1\}^d) \cap g^{-1}(x_{i+1})|$ and observe that X can be expressed as a sum of $\nu \geq 2^{d-1}$ Bernoulli random variables $X_1, \ldots, X_\nu$, with parameter $p = 2^{-d/2}$, which are $2^{-3d/2}$-almost pairwise independent. Therefore, by Lemma 2.16 and using the fact that h_i is a $2^{-3d/2}$-almost pairwise independent function, we have that

$$\begin{aligned}
\Pr[X = 0] &= \Pr[\nu \cdot 2^{-\frac{d}{2}} - X = \nu \cdot 2^{-\frac{d}{2}}] \\
&\leq \Pr[|X - \nu \cdot 2^{-\frac{d}{2}}| \geq \nu \cdot 2^{-\frac{d}{2}}] \\
&\leq \frac{\nu \cdot 2^{-\frac{d}{2}} \cdot (1 - 2^{-\frac{d}{2}}) + \nu^2 \cdot 2^{-3d/2}}{(\nu \cdot 2^{-\frac{d}{2}})^2} \\
&\leq \frac{\nu \cdot 2^{-\frac{d}{2}}}{(\nu \cdot 2^{-\frac{d}{2}})^2} + \frac{2^{-\frac{3d}{2}}}{2^{-d}} \\
&= \frac{2^{\frac{d}{2}}}{\nu} + 2^{-\frac{d}{2}} \\
&\leq 2^{-\frac{d}{2}+1} + 2^{-\frac{d}{2}} \\
&\leq 2^{-\lambda - \log k},
\end{aligned}$$

and the lemma follows. □

5.2 Completeness

Assume that the circuits $C_1, ...C_k : \{0,1\}^n \to \{0,1\}^m$ that were fixed in Section 5.1 are all YES instances.

Proposition 5.6. *Assume that g is good. Then, for every $i \in [k]$, with probability at least $1-(k-i+1)\cdot 2^{-\lambda-\log k}$, the sets $XZ_k, \dots, XZ_i$ (as defined in Step 2b of Fig. 3) are non-empty .*

Proof. The proposition follows from Lemma 5.5 and an inductive argument. □

By Lemma 5.3, g is good with probability at least $1-2^{-\lambda}$. Given that g is good, by Proposition 5.6, the sets $XZ_k, \dots XZ_1$ are non-empty with probability at least $1-k\cdot 2^{-\lambda-\log k} = 1-2^{-\lambda}$. In this case the honest prover never sends an arbitrary choice of (z_i, β_i) (which should be interpreted as an abort).

Thus, by construction, all of the verifier's checks pass. Overall we obtain that the verifier accepts with probability at least $1-2^{-\lambda+1}$.

5.3 Round-By-Round Soundness

To show that round-by-round soundness holds, we need to construct a suitable State function as per Definition 2.20. Our State function is defined in Fig. 4, where we break the definition into parts based on the type of partial transcript given as input.

Let $C_1, \dots, C_k : \{0,1\}^n \to \{0,1\}^m$ be a NO instance (i.e., at least one of these circuits is a NO instance for AI). Recall that $i^* \in [k]$ denotes the maximal i such that C_i is a NO instance. Observe first that, as required (see Definition 2.20), the state function $\mathsf{State}((C_1, \dots, C_k), \perp)$ on the empty transcript is $\texttt{doomed}$.

Next, we need to show that for every doomed transcript τ, and additional prover message m_p, with high probability over the choice of the next verifier's message m_v, the transcript (τ, m_p, m_v) is still doomed. We do so by analyzing the different types of partial transcripts in the protocol.

The First Message. Observe that:

$$\Pr_{g \leftarrow G_n, x_{k+1} \leftarrow \{0,1\}^n}\left[\mathsf{State}\big((C_1, \dots, C_k), g, x_{k+1}\big) = \texttt{doomed}\right] = \Pr_{g \leftarrow G_n}[g \text{ is good}] \geq 1-2^{-\lambda},$$

where the inequality follows from Lemma 5.3. Thus, the transcript is doomed after the first verifier message with all but $2^{-\lambda}$ probability.

Fix $j \in \{k, \dots, i^*+1\}$ and a partial transcript[7]

$$\tau_{j+1} = \Big((C_1, \dots, C_k), g, x_{k+1}, \big(h_i, \alpha_i, f_i, z_i, \beta_i\big)_{i \in \{k, \dots, j+2\}}, h_{j+1}, \alpha_{j+1}, f_{j+1}\Big).$$

Note that this transcript includes all messages up till (and including) the verifier's message in iteration $j+1$. Assume that $\mathsf{State}(\tau_{j+1}) = \texttt{doomed}$ and fix the prover's message (z_{j+1}, β_{j+1}) for

[7]Recall that h_{k+1}, α_{k+1} and f_{k+1} were all defined as empty strings to handle the edge case $j = k$.

A. $\underline{\mathsf{State}\big((C_1,\dots,C_k),\perp\big)\text{ (recall that }\perp\text{ denotes the empty transcript) :}}$

If one of circuits C_i is a NO instance output `doomed`. Otherwise output `alive`.

B. $\underline{\mathsf{State}\big((C_1,\dots,C_k),g,x_{k+1}\big)}$:

If g is good and one of the circuits C_i is a NO instance, output `doomed`. Otherwise output `alive`.

C. $\underline{\mathsf{State}\Big((C_1,\dots,C_k),g,x_{k+1},\big(h_i,\alpha_i,f_i,z_i,\beta_i\big)_{i\in\{k,\dots,j+1\}},h_j,\alpha_j,f_j\Big)\text{, where } j\in\{k,\dots,i^*+1\}\text{:}}$

1. Set h_{k+1},α_{k+1} and f_{k+1} to be empty strings.[a]
2. If $\mathsf{State}\big((C_1,\dots,C_k),g,x_{k+1},(h_i,\alpha_i,f_i,z_i,\beta_i)_{i\in\{k,\dots,j+2\}},h_{j+1},\alpha_{j+1},f_{j+1}\big) =$ `alive`, output `alive`.
3. (a) If $j=k$, set $x'_{k+1}=x_{k+1}$.
   (b) Else (i.e., $j<k$): if $\left|A^{(j+1)}_{z_{j+1},\beta_{j+1}}\right|\neq 1$ (as defined in Step C.4), then output `doomed`. Otherwise, denote the unique element in $A^{(j+1)}_{z_{j+1},\beta_{j+1}}$ by x'_{j+1}.
4. For every $\beta\in\{0,1\}^{2d+\lambda+\log(k)}$ and $z\in\{0,1\}^d$, let $A^{(j)}_{z,\beta}$ be the set of all $x\in\{0,1\}^n$ s.t.
   (a) $h_j(C_j(x),z)=\alpha_j$; and
   (b) $g(C_j(x),z)=x'_{j+1}$; and
   (c) $f_j(x)=\beta$.
5. If for every (z_j,β_j), it holds that $\left|A^{(j)}_{z_j,\beta_j}\right|\leq 1$, output `doomed`. Otherwise output `alive`.

[a]The values $h_{k+1},\alpha_{k+1},f_{k+1}$ are defined as empty strings in order to handle the edge case $j=k$.

D. $\underline{\mathsf{State}\Big((C_1,\dots,C_k),g,x_{k+1},\big(h_i,\alpha_i,f_i,z_i,\beta_i\big)_{i\in\{k,\dots,j+1\}},h_j,\alpha_j,f_j\Big)\text{, where } j\in\{i^*,\dots,1\}\text{:}}$

1. Set $h_{k+1}=\alpha_{k+1}=f_{k+1}=\perp$.
2. If $\mathsf{State}\big((C_1,\dots,C_k),g,x_{k+1},(h_i,\alpha_i,f_i,z_i,\beta_i)_{i\in\{k,\dots,i^*+2\}},h_{i^*+1},\alpha_{i^*+1},f_{i^*+1}\big) =$ `alive`, output `alive`.
3. (a) If $i^*=k$, set $x'_{i^*+1}=x_{k+1}$.
   (b) Else (i.e., $i^*<k$): if $\left|A^{(i^*+1)}_{z_{i^*+1},\beta_{i^*+1}}\right|\neq 1$ (as defined in Step C.4), then output `doomed`. Otherwise, denote the unique element in $A^{(i^*+1)}_{z_{i^*+1},\beta_{i^*+1}}$ by x'_{i^*+1}.
4. If there exist $x_{i*}\in\{0,1\}^n$ and $z_{i*}\in\{0,1\}^d$ such that
   (a) $h_{i^*}(C_{i^*}(x_{i^*}),z_{i^*})=\alpha_{i^*}$; and
   (b) $g(C_{i^*}(x_{i^*}),z_{i^*})=x'_{i^*+1}$,

   output `alive`. Otherwise output `doomed` .

Figure 4: Definition of the State function, separated by the type of partial transcript

iteration $j+1$. We need to show that with high probability over the choice of (h_j, α_j, f_j) it holds that $\mathsf{State}\big(\tau_{j+1}, (z_{j+1}, \beta_{j+1}), (h_j, \alpha_j, f_j)\big) = \texttt{doomed}$. We do so separately depending on the value of j:

When $j \in \{k, \ldots, i^* + 1\}$. Suppose first that $j < k$. Since $\mathsf{State}(\tau_{j+1}) = \texttt{doomed}$, the state function does not output $\texttt{alive}$ in Step C.2. We may also assume that $|A^{(j+1)}_{z_{j+1},\beta_{j+1}}| = 1$ since otherwise the function outputs $\texttt{doomed}$ in Step C.3.b. Denote the unique element in $A^{(j+1)}_{z_{j+1},\beta_{j+1}}$ by x'_{j+1}. Likewise, for the case $j = k$, define $x'_{k+1} = x_{k+1}$.

The following lemma shows that with high probability over the choice of random h_j, α_j, f_j and z_j, there is at most one "consistent" x_j.

Lemma 5.7. *With probability at least $1 - 2^{-\lambda - \log k + 4}$ over $(h_j, \alpha_j, f_j) \leftarrow H_n \times \{0,1\}^{d/2} \times F_n$, the following holds: for every $(z_j, \beta_j) \in \{0,1\}^d \times \{0,1\}^{2d+\lambda+\log k}$ there exists at most one value $x_j \in \{0,1\}^n$ such that (1) $f_j(x_j) = \beta_j$, (2) $h_j(C_j(x_j), z_j) = \alpha_j$, and (3) $g(C_j(x_j), z_j) = x_{j+1}$.*

Proof. Recall that S_j is the set of images of the circuit C_j, and S_j^+ is the set of images of C_i having more than one preimage. We say that a pair $(y, z) \in \{0,1\}^m \times \{0,1\}^d$ is *bad* if $y \in S_j^+$. We first argue that with high probability, no bad pair is mapped (via h_j) to α_j.

Since g is good, by definition (see Definition 5.2), at most $12 \cdot (n + \lambda + \log k)$ bad pairs are mapped via g to x_{j+1}. Consider a fixed h_j. Since α_j is chosen uniformly at random, the probability that a particular pair (y, z) is mapped to α_j by h_j is $2^{-d/2}$. Thus, by a union bound, with probability at least $1 - \frac{12\cdot(n+\log k+\lambda)}{2^{d/2}}$, no bad pair is mapped to α_j.

Denote by X the set of all $x \in \{0,1\}^n$ such that (1) x has no sibling (other than itself) and (2) there exists $z \in \{0,1\}^d$ such that $g(C_j(x), z) = x_{j+1}$. Since g is good, by definition (see Definition 5.2), there are at most $\frac{2^d \cdot |S_i \setminus S_i^+|}{2^n} \cdot (1 + 2^{-\lambda}) \leq 2^d \cdot (1 + 2^{-\lambda}) < 2^{d+1}$ pairs (y, z) s.t. $y \in S_j \setminus S_J^+$ and $g(y, z) = x_{j+1}$. Note that exactly one $x \in X$ is mapped to each of these y's, and therefore $|X| < 2^{d+1}$.

For every distinct $x, x' \in X$, since f_j is $2^{-2d-\lambda-\log k}$-almost pairwise independent with a range of size $2^{2d+\lambda+\log k}$, the probability that they collide under f_j is at most $2^{-2d-\lambda-\log k+1}$. Taking a union bound over at most 2^{2d+2} such pairs, it holds that f_j is injective when restricted to the domain X, with probability at least $1 - 2^{-\lambda-\log k+3}$.

To summarize (and using a union bound), we have that (1) no bad pair is mapped to x_{j+1} and (2) f_j is injective on X, with probability at least

$$1 - 2^{-d/2+\log(n+\log k+\lambda)+\log 12} - 2^{-\lambda-\log k+3} \leq 1 - 2^{-\lambda-\log k+4},$$

where the inequality follows from the setting of d.

Note that if the latter two conditions hold then for every $\beta \in \{0,1\}^{2d+\lambda+\log k}$ there exists at most one pair (x, z) such that $f_i(x) = \beta$, $h_i(C_i(x), z) = \alpha_i$ and $g(C(x_i), z) = x_{i+1}$, and in particular, for every pair (z, β) s.t. the same conditions apply as required. $\square$

Lemma 5.7 implies that with all but $2^{-\lambda-\log k+4}$ probability, for every $z \in \{0,1\}^d$ and $\beta \in \{0,1\}^{2d+\lambda+\log k}$ it holds that $\left|A^{(j)}_{z,\beta}\right| \leq 1$, and therefore the State function outputs $\texttt{doomed}$ in Step D.4 as required.

When $j = i^*$. Suppose first that $i^* < k$. As above, since $\mathsf{State}(\tau_{i^*+1}) = \texttt{doomed}$, the state function does not output `alive` in Step D.2. We may also assume that $|A^{(i^*+1)}_{z_{i^*+1},\beta_{i^*+1}}| = 1$ since otherwise the function outputs `doomed` in Step D.3.b. Denote the unique element in $A^{(i^*+1)}_{z_{i^*+1},\beta_{i^*+1}}$ by x'_{i^*+1}. Likewise, for the case $i^* = k$, define $x'_{k+1} = x_{k+1}$.

The following lemma shows that with high probability over the choice of h_{i^*} and α_{i^*}, there do not exist any "consistent" x_{i^*}.

Lemma 5.8. *With probability at least $1 - 2^{-\lambda - \log k + 4}$ over $(h^*_i, \alpha_{i^*}) \leftarrow H_n \times \{0,1\}^{d/2}$, the following holds: for every $(x_{i^*}, z_{i^*}) \in \{0,1\}^n \times \{0,1\}^d$ such that $g(C_{i^*}(x_{i^*}), z_{i^*}) = x_{i^*+1}$ it holds that $h_{i^*}(C_{i^*}(x_{i^*}), z_{i^*}) \neq \alpha_{i^*}$.*

Proof. Consider the message $x_{i^*+1} \in \{0,1\}^n$. Since g is good, by definition (see Definition 5.2), there are at most $2^{d-\ell+2}$ pairs (y_{i^*}, z_{i^*}) mapped to x_{i^*+1}, where y_{i^*} is in the image of C_{i^*}. The probability for any such pair to be mapped to α_{i^*} by h_{i^*} is $2^{-d/2}$. Therefore, by a union bound, the probability that one of the pairs is mapped to α_{i^*} by h_{i^*} is at most $\frac{2^{d-\ell+2}}{2^{d/2}} = 2^{d/2-\ell+2} \leq 2^{-\lambda+2}$, where the inequality follows by our setting of parameters. Thus, with probability at least $1 - 2^{-\lambda+2}$, no such pair is mapped to α_{i^*}. □

Therefore, with all but $2^{-\lambda - \log k + 4}$ probability, the State function outputs `doomed` in Step D.4.

When $j \in \{i^*-1, \ldots, 1\}$. For every $j \in \{i^*-1, \ldots, 1\}$, by construction, it holds that $\mathsf{State}(\tau_{j+1}) = \mathsf{State}(\tau_{i^*})$, where τ_{i^*} is the partial transcript corresponding to iteration $k, \ldots, i^*$ (this is evident by the fact that the code of Part D. of the State function does not depend on j, other than the requirement that $j \leq i^*$). Therefore, for every iteration $j \in \{i^* - 1, \ldots, 1\}$ it holds that if τ_{j+1} is `doomed` then τ_{i^*} is `doomed` and so $\mathsf{State}\big(\tau_{j+1}, (z_{j+1}, \beta_{j+1}), (h_j, \alpha_j, f_j)\big) = \texttt{doomed}$.

Verifier rejects full `doomed` transcripts. To establish round-by-round soundness, it remains to be shown that full `doomed` transcripts make the verifier reject.

Fix a full transcript τ and assume toward a contradiction that τ makes the verifier accept, but $\mathsf{State}(\tau) = \texttt{doomed}$. By construction, the transcript τ can be described as follows:

$$\tau = \Big((C_1, \ldots, C_k), g, x_{k+1}, \big(h_i, \alpha_i, f_i, z_i, \beta_i\big)_{i \in \{k, \ldots, 1\}}, x_1\Big),$$

where $g \in G_n$, $x_1, x_{k+1} \in \{0,1\}^n$, $h_1, \ldots, h_k \in H_n$, $f_1, \ldots, f_k \in F_n$, $\alpha_1, \ldots, \alpha_k \in \{0,1\}^{d/2}$, $\beta_1, \ldots, \beta_k \in \{0,1\}^{2d+\lambda+\log k}$ and $z_1, \ldots, z_k \in \{0,1\}^d$.

Since τ is accepting there exist $x'_1, \ldots, x'_{k+1}$ such that

1. $x'_1 = x_1$.
2. $x'_{k+1} = x_{k+1}$.
3. For every $j \in [k]$:
   (a) $x'_{j+1} = g(C_j(x'_j), z_j)$.
   (b) $\beta'_j = f_j(x'_j)$.
   (c) $\alpha_j = h_j(C_j(x'_j), z_j)$.

For every $j \in [k]$, denote by τ_j the prefix of τ corresponding to all iterations up to (and including) iteration j. More specifically,

$$\tau_j = \Big((C_1, \dots, C_k), g, x_{k+1}, (h_i, \alpha_i, f_i, z_i, \beta_i)_{i \in \{k, \dots, j+1\}}, h_j, \alpha_j, f_j \Big).$$

As discussed above, since $\mathsf{State}(\tau) = \mathsf{State}(\tau_{i^*})$, and $\mathsf{State}(\tau) = \mathtt{doomed}$, we have that $\mathsf{State}(\tau_{i^*})$ is also doomed.

Claim 5.8.1. *For every $j \in \{k, \dots, i^*\}$ it holds that $\mathsf{State}(\tau_j) = \mathtt{doomed}$.*

Proof. Suppose that $\mathsf{State}(\tau_j) = \mathtt{alive}$ for some $j \in \{k, \dots, i^*\}$. Then, by the recursive construction (see Steps C.2 and D.2), it holds that $\mathsf{State}(\tau_{i^*}) = \mathtt{alive}$ — a contradiction. □

Claim 5.8.2. *For every $j \in \{k, \dots, i^* + 1\}$ it holds that $A^{(j)}_{z_j, \beta_j} = \{x'_j\}$.*

Proof. We prove by reverse induction on j. For the base case $j = k$, since $\mathsf{State}(\tau_k)$ is doomed, by construction (see Step C.5) it must be the case that for every (z, β) it holds that $\left|A^{(k)}_{z,\beta}\right| \leq 1$. The base case follows by observing that $x'_k \in A^{(k)}_{z_k, \beta_k}$.

Assume that the claim holds for $j+1$. By the inductive hypothesis we have that $\left|A^{(j+1)}_{z_{j+1}, \beta_{j+1}}\right| = 1$ and therefore the State function does not output $\mathtt{doomed}$ in Step C.3.b.

Since $\mathsf{State}(\tau_j) = \mathtt{doomed}$ it therefore must be the case that the function outputs $\mathtt{doomed}$ in Step C.5. This means that for every (z, β) it holds that $\left|A^{(j)}_{z,\beta}\right| \leq 1$. The claim follows by observing that $x'_j \in A^{(j)}_{z_j, \beta_j}$. □

Since $\mathsf{State}(\tau_{i^*}) = \mathtt{doomed}$, by construction either:

- $i^* < k$ and $\left|A^{(j+1)}_{z_{j+1}, \beta+1}\right| \neq 1$ — in contradiction to Claim 5.8.2.
- There does not exist a pair (x, z) such that
  (a) $h_{i^*}(C_{i^*}(x), z) = \alpha_{i^*}$; and
  (b) $g(C_{i^*}(x), z) = x'_{i^*+1}$.

  However such a pair *does* exist - namely (x'_{i^*}, z_{i^*}).

Both cases let to a contradiction and round-by-round soundness follows.

5.4 Honest Verifier Statistical Zero-Knowledge

The simulator establishing that the protocol is honest-verifier statistical zero-knowledge is presented in Fig. 5. We proceed to show that the simulation is indeed statistically close to the real interaction.

Let $(C_1, ... C_k)$ be a YES instance for $\mathsf{AI}_{2^\ell, \delta}$. Consider the following hybrid distributions:

Definition 5.9. *For every $i \in [k+1]$, we define the distribution H_i as the output of the following random process:*

1. *Sample $g \in G_n$.*

Simulator

THE SIMULATOR:

1. Sample $g \leftarrow G_n$.
2. Sample $x_1 \leftarrow \{0,1\}^n$.
3. For $i = 1, \ldots, k$:
   - Compute $y_i = C_i(x_i)$.
   - Sample $z_i \leftarrow \{0,1\}^d$.
   - Sample $h_i \leftarrow H_n$ and $f_i \leftarrow F_n$.
   - Compute $\alpha_i = h_i(y_i, z_i)$.
   - Compute $\beta_i = f_i(x_i)$.
   - Compute $x_{i+1} = g(y_i, z_i)$.
4. Output $\left(g, x_1, x_{k+1}, (h_i, f_i, \alpha_i, \beta_i, z_i,)_{i\in[k]}\right)$.

Figure 5: Simulator for Public-coin Batch SZK Protocol for AI

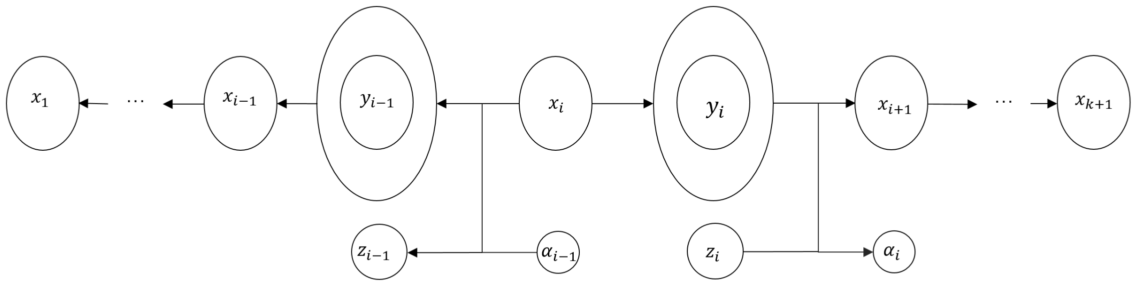

Figure 6: The hybrid H_i sampling process

2. *Choose uniformly at random $x_i \in \{0,1\}^n$.*

3. *Generate $\left(x_1, (h_j, f_j, \alpha_j, \beta_j, z_j)_{j\in\{1,\ldots,i-1\}}\right)$ as follows: Emulate Step 2 from Fig. 3 (i.e., the real protocol) for the iterations $i-1$ down to 1, using g and x_i.*

4. *Generate $\left(x_{k+1}, (h_j, f_j, \alpha_j, \beta_j, z_j,)_{j\in\{i,\ldots,k\}}\right)$ as follows: Emulate Step 3 from Fig. 5 (i.e., the simulator) for the iterations i up to k, using g and x_i.*

5. *Output $\left(g, x_1, x_{k+1}, (h_j, f_j, \alpha_j, \beta_j, z_j)_{j\in[k]}\right)$*

Observe that the hybrid H_{k+1} is distributed identically to the protocol's transcript, whereas the hybrid H_1 is distributed identically to the simulator's output. Thus, by a standard hybrid argument, to prove zero-knowledge it suffices to prove the following lemma.

Lemma 5.10. *For every $i \in [k]$, it holds that $\Delta(\mathsf{H}_i, \mathsf{H}_{i+1}) \leq \delta \cdot 2^{d+2} + 2^{-\lambda+6}$.*

Lemma 5.10 is proved in Section 5.4.1, which follows.

5.4.1 Proof of Lemma 5.10

Fix $i \in [k]$ and a function $g \in G_n$ which is good (as per Definition 5.2). We denote by $\mathsf{H}_i|_g$ the distributions H_i conditioned on the fixed function g being selected. We denote by $\left(X_i^{(P)}, Z_i^{(P)}, H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)}\right)$ the joint distribution of the random variables $(x_i, z_i, h_i, \alpha_i, x_{i+1})$ in the random process that generates the hybrid H_{i+1} (conditioned on the function g being selected). Similarly, we denote by $\left(X_i^{(S)}, Z_i^{(S)}, H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)}\right)$ the joint distribution of $(x_i, z_i, h_i, \alpha_i, x_{i+1})$ in the hybrid H_i (conditioned on the function g being selected).[8]

In order to bound $\Delta(\mathsf{H}_i, \mathsf{H}_{i+1})$ we first observe that it suffices to consider only a subset of the random variables. This is captured by the following proposition.

Proposition 5.11.

$$\Delta(\mathsf{H}_i|_g, \mathsf{H}_{i+1}|_g) \leq \Delta\left(\left(X_i^{(P)}, Z_i^{(P)}, H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)}\right), \left(X_i^{(S)}, Z_i^{(S)}, H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)}\right)\right).$$

Proof. Conditioned on the values $(x_i, z_i, h_i, \alpha_i, x_{i+1})$, the rest of the output values of $\mathsf{H}_i|_g$ and $\mathsf{H}_{i+1}|_g$ are identical distributed. This is because the rest of the values are generated by the same random process applied to $(x_i, z_i, h_i, x_{i+1}, \alpha_i)$ in both hybrids (namely, iterations $j \in \{1, \ldots, i-1\}$ in Step 3 and iterations $j \in \{i+1, \ldots, k\}$ in Step 4). The proposition follows by the data processing inequality for statistical distance (see Fact 2.2). $\square$

Thus, it suffices to bound the statistical distance between $\left(X_i^{(P)}, Z_i^{(P)}, H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)}\right)$ and $\left(X_i^{(S)}, Z_i^{(S)}, H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)}\right)$.

Definition 5.12. *We say that the tuple $(h_i, \alpha_i, x_{i+1}) \in H_n \times \{0,1\}^{d/2} \times \{0,1\}^n$ is* consistent with respect to *C_i if there exists $(x_i, z_i) \in \{0,1\}^n \times \{0,1\}^d$ such that $g(C_i(x_i), z_i) = x_{i+1}$ and $h_i(C_i(x_i), z_i) = \alpha_i$. We omit C_i from the notation, and simply say that the tuple is* consistent, *when C_i is clear from the context.*

Proposition 5.13. *For every consistent tuple (h_i, α_i, x_{i+1}), the following two random variables are identically distributed:*

- $\left(X_i^{(P)}, Z_i^{(P)}\right)$ *conditioned on $H_i^{(P)} = h_i$, $A_i^{(P)} = \alpha_i$ and $X_{i+1}^{(P)} = x_{i+1}$.*
- $\left(X_i^{(S)}, Z_i^{(S)}\right)$ *conditioned on $H_i^{(S)} = h_i$, $A_i^{(S)} = \alpha_i$ and $X_{i+1}^{(S)} = x_{i+1}$.*

Proof. Let $(h_i, \alpha_i, x_{i+1}) \in H_n \times \{0,1\}^{d/2} \times \{0,1\}^n$ be a consistent tuple. Recall that XZ_i is the set of pairs $(x_i, z_i) \in \{0,1\}^n \times \{0,1\}^d$ such that $g(C_i(x_i), z_i) = x_{i+1}$ and $h_i(C_i(x_i), z_i) = \alpha_i$.

Conditioned on $H_i^{(P)} = h_i$, $A_i^{(P)} = \alpha_i$ and $X_{i+1}^{(P)} = x_{i+1}$, by construction, the prover chooses the pair (x_i, z_i) uniformly from the set XZ_i (see Step 2c in Fig. 3). Thus, the distribution $\left(X_i^{(P)}, Z_i^{(P)}\right)$ is uniform over the set XZ_i.

On the other hand, the distribution $\left(X_i^{(S)}, Z_i^{(S)}\right)$ is uniform over $\{0,1\}^n \times \{0,1\}^d$. However, by conditioning on $H_i^{(S)} = h_i$, $A_i^{(S)} = \alpha_i$ and $X_{i+1}^{(S)} = x_{i+1}$, we have that the conditional distribution is also uniform over the set XZ_i. $\square$

[8] The superscripts P and S are meant to distinguish between variables generated by the protocol and simulation, respectively.

We denote by $\left(\hat{X}_i^{(P)}, \hat{Z}_i^{(P)}, \hat{H}_i^{(P)}, \hat{A}_i^{(P)}, \hat{X}_{i+1}^{(P)}\right)$ the distribution of $\left(X_i^{(P)}, Z_i^{(P)}, H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)}\right)$, conditioned on $\left(H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)}\right)$ being consistent. Similarly, we denote by $\left(\hat{X}_i^{(S)}, \hat{Z}_i^{(S)}, \hat{H}_i^{(S)}, \hat{A}_i^{(S)}, \hat{X}_{i+1}^{(S)}\right)$ the distribution of $\left(X_i^{(S)}, Z_i^{(S)}, H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)}\right)$, conditioned on $\left(H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)}\right)$ being consistent. By Proposition 5.13 and Fact 2.3, it holds that

$$\Delta\Big(\big(\hat{X}_i^{(P)}, \hat{Z}_i^{(P)}, \hat{H}_i^{(P)}, \hat{A}_i^{(P)}, \hat{X}_{i+1}^{(P)}\big), \big(\hat{X}_i^{(S)}, \hat{Z}_i^{(S)}, \hat{H}_i^{(S)}, \hat{A}_i^{(S)}, \hat{X}_{i+1}^{(S)}\big)\Big) = \Delta\Big(\big(\hat{H}_i^{(P)}, \hat{A}_i^{(P)}, \hat{X}_{i+1}^{(P)}\big), \big(\hat{H}_i^{(S)}, \hat{A}_i^{(S)}, \hat{X}_{i+1}^{(S)}\big)\Big). \tag{4}$$

We denote by $X_i^{(U)}$, $Z_i^{(U)}$, $H_i^{(U)}$, $A_i^{(U)}$ and $X_{i+1}^{(U)}$ the uniform distributions (independent of everything else) over the sets $\{0,1\}^n$, $\{0,1\}^d$, H_n, $\{0,1\}^{d/2}$ and $\{0,1\}^n$, respectively. Define the function $\varphi_i(x_i, z_i, h_i) = (h_i, \alpha_i, x_{i+1})$, where $x_{i+1} = g(C_i(x_i), z_i)$ and $\alpha_i = h_i(C_i(x_i))$.

Note that by definition of the hybrids, we have that $\left(H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)}\right) \equiv \left(H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}\right)$ and also $\left(H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)}\right) \equiv \varphi_i\Big(X_i^{(U)}, Z_i^{(U)}, H_i^{(U)}\Big)$. Therefore,

$$\Delta\Big(\big(H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)}\big), \big(H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)}\big)\Big) = \Delta\left(\big(H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}\big), \varphi_i\big(X_i^{(U)}, Z_i^{(U)}, H_i^{(U)}\big)\right). \tag{5}$$

The following proposition bound the RHS of Eq. (5).

Proposition 5.14. *It holds that*

$$\Delta\left(\big(H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}\big), \varphi_i\big(X_i^{(U)}, Z_i^{(U)}, H_i^{(U)}\big)\right) \le \delta \cdot (2^d + 1) + 2^{-\lambda+2}.$$

We momentarily defer the proof of Proposition 5.14 and show how to use it to prove Lemma 5.10.

Claim 5.14.1. *$(H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)})$ are consistent with probability at least $1-2^{-\lambda-\log k}$ and $(H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)})$ are consistent with probability 1.*

Proof. It holds that $(H_i^{(P)}, A_i^{(P)}, X_{i+1}^{(P)})$ are consistent with probability at least $1 - 2^{-\lambda-\log k}$ by Lemma 5.5, and $(H_i^{(S)}, A_i^{(S)}, X_{i+1}^{(S)})$ are consistent with probability 1 by construction. □

By Eq. (5), Proposition 5.14, Fact 2.5, and Claim 5.14.1, it holds that

$$\Delta\Big(\big(\hat{H}_i^{(P)}, \hat{A}_i^{(P)}, \hat{X}_{i+1}^{(P)}\big), \big(\hat{H}_i^{(S)}, \hat{A}_i^{(S)}, \hat{X}_{i+1}^{(S)}\big)\Big) \le 2 \cdot (\delta \cdot (2^d + 1) + 2^{-\lambda+2} + 2^{-\lambda-\log k}) \le \delta \cdot 2^{d+2} + 2^{-\lambda+4}. \tag{6}$$

Thus, by Eqs. (4) and (6), it holds that

$$\Delta\Big(\big(\hat{X}_i^{(P)}, \hat{Z}_i^{(P)}, \hat{H}_i^{(P)}, \hat{A}_i^{(P)}, \hat{X}_{i+1}^{(P)}\big), \big(\hat{X}_i^{(S)}, \hat{Z}_i^{(S)}, \hat{H}_i^{(S)}, \hat{A}_i^{(S)}, \hat{X}_{i+1}^{(S)}\big)\Big) \le \delta \cdot 2^{d+2} + 2^{-\lambda+4}. \tag{7}$$

Therefore, By Proposition 5.11, Eq. (7), Fact 2.4, and Claim 5.14.1, it holds that

$$\Delta(\mathsf{H}_i|_g, \mathsf{H}_{i+1}|_g) \le \delta \cdot 2^{d+2} + 2^{-\lambda+5}.$$

We denote by G^* the set of good g. Then,

$$\begin{aligned}\Delta(\mathsf{H}_i, \mathsf{H}_{i+1}) &= \mathop{\mathrm{E}}_{g \leftarrow G}\left[\Delta(\mathsf{H}_i|_g, \mathsf{H}_{i+1}|_g)\right] \\ &\leq \mathop{\mathrm{E}}_{g \leftarrow G^*}\left[\Delta(\mathsf{H}_i|_g, \mathsf{H}_{i+1}|_g)\right] + 2^{-\lambda} \\ &\leq \delta \cdot 2^{d+2} + 2^{-\lambda+6},\end{aligned}$$

where the first inequality is by Lemma 5.3 (and the law of total expectation).

Thus, to complete the proof of Lemma 5.10 we only need to prove Proposition 5.14, which we do in the following subsection.

5.4.2 Proof of Proposition 5.14

Recall $S_i \setminus S_i^+$, is the set of images having unique preimage under C_i. Denote by X_i' the distribution $X_i^{(U)}$ conditioned on $x_i \in C_i^{-1}(S_i \setminus S_i^+)$, i.e., the uniform distribution over the set $C_i^{-1}(S_i \setminus S_i^+)$.

We denote by $\left(H_i', A_i', X_{i+1}'\right)$ the distribution of $\varphi_i(X_i', Z_i^{(U)}, H_i^{(U)})$. Since X_i' is uniform over a set that is injectively mapped to $S_i \setminus S_i^+$, $C_i(X_i')$ is uniform over $S_i \setminus S_i^+$. Therefore, and since g is good, by Proposition 5.4 we have that $X_{i+1}' = g(C_i(X_i'), Z_i^{(U)})$ is $2^{-\lambda}$-close to the uniform distribution $X_{i+1}^{(U)}$. We denote by $\left(\widetilde{H}_i, \widetilde{A}_i\right)$ the random variables generated by the following random process:

1. Sample $x_{i+1} \leftarrow X_{i+1}^{(U)}$.
2. Sample $(h_i, \alpha_i) \leftarrow (H_i', A_i')|_{X_{i+1}'=x_{i+1}}$.
3. Output (h_i, α_i).

Note that the conditioned random variable $(H_i', A_i')|_{X_{i+1}'=x_{i+1}}$ is defined for every x_{i+1} since g is good, and therefore by Condition 1 of Definition 5.2, there exists a pair $(y, z) \in \{0,1\}^n \times \{0,1\}^d$ where $y \in (S_i \setminus S_i^+)$ s.t. $g(y, z) = x_{i+1}$.

It holds that

$$\begin{aligned}\Delta\left((H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}), \varphi_i(X_i', Z_i^{(U)}, H_i^{(U)})\right) &= \Delta\left((H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}), (H_i', A_i', X_{i+1}')\right) \\ &\leq \Delta\left((H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}), (\widetilde{H}_i, \widetilde{A}_i, X_{i+1}^{(U)})\right) \\ &\quad + \Delta\left((\widetilde{H}_i, \widetilde{A}_i, X_{i+1}^{(U)}), (H_i', A_i', X_{i+1}')\right) \\ &\leq \Delta\left((H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}), (\widetilde{H}_i, \widetilde{A}_i, X_{i+1}^{(U)})\right) + 2^{-\lambda}, \quad (8)\end{aligned}$$

where the first inequality is due to the triangle inequality, and the second is due to the fact that X_{i+1}' is $2^{-\lambda}$ close to $X_{i+1}^{(U)}$ and Fact 2.2.

Definition 5.15. *For every $i \in \{2, \ldots, k+1\}$, We say that $x_i \in \{0,1\}^n$ is* nice *if for every (y_{i-1}, z_{i-1}) s.t. $g(y_{i-1}, z_{i-1}) = x_i$ it holds that y_{i-1} has a* unique *preimage with respect to C_{i-1}.*

Claim 5.15.1. *A uniformly chosen $x_i \in \{0,1\}^n$ is nice with probability at least $1 - \delta \cdot 2^d$.*

Proof. Since C_{i-1} is a YES instance, there are at most $2^{n+d} \cdot \delta$ pairs (y_i, z_i) s.t. y_i has more than one preimage under C_{i-1}. For each such pair, the probability it is mapped via g to a uniformly chosen x_i is 2^{-n}. Therefore, the probability that no such pair is mapped to a uniformly chosen x_i is at least $1 - \delta \cdot 2^d$. $\square$

Now fix a nice $x_{i+1} \in \{0,1\}^n$. Since g is good, conditioned on $X'_{i+1} = x_{i+1}$, the number of pairs (y_i, z_i) that are mapped to x_{i+1} via g is at least $2^d \cdot (1 - 2^{-\lambda}) \geq 2^{d-1}$. Since x_{i+1} is nice, each such y_i has only one preimage under C_i, and therefore X'_i is uniform on these preimages. Since $(X'_i, Z_i^{(U)})$ is uniform, it holds that $(C_i(X'_i), Z_i^{(U)})$ is a $(d-1)$-source over $\{0,1\}^{m+d}$ conditioned on $X'_{i+1} = x_{i+1}$.

By definition of φ_i, it holds that H'_i is uniform and independent from X'_i and X'_{i+1}. Therefore, it holds that $\widetilde{H}_i$ is uniform over a family of $2^{-3d/2}$-almost pairwise independent hash functions. Hence, by Lemma 2.19, with $\epsilon = 2^{-d/4+1/2}$, we have that the function $\mathsf{Ext}\big((y_i, z_i), h_i\big) = h_i(y_i, z_i)$ is a strong $(d-1, 2^{-d/4+1/2})$-extractor. Therefore, given an nice x_{i+1}, conditioned on $X_{i+1}^{(U)} = x_{i+1}$, it holds that $(\widetilde{H}_i, \widetilde{A}_i)$ is $2^{-d/4+1/2}$-close to uniform. Denote the set of nice x_i as X_A. Thus:

$$\begin{aligned}
\Delta\Big((H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}), (\widetilde{H}_i, \widetilde{A}_i, X_{i+1}^{(U)})\Big) &= \mathop{\mathrm{E}}_{x \leftarrow X_{i+1}^{(U)}}\left[\Delta\Big((H_i^{(U)}, A_i^{(U)})_{|X_{i+1}^{(U)}=x}, (\widetilde{H}_i, \widetilde{A}_i)_{|X_{i+1}^{(U)}=x}\Big)\right] \qquad (9)\\
&\leq \mathop{\mathrm{E}}_{x \leftarrow X_A}\left[\Delta\Big((H_i^{(U)}, A_i^{(U)})_{|X_{i+1}^{(U)}=x}, (\widetilde{H}_i, \widetilde{A}_i)_{|X_{i+1}^{(U)}=x}\Big)\right] + \delta \cdot 2^d\\
&\leq 2^{-d/4+1/2} + \delta \cdot 2^d,
\end{aligned}$$

where the equality is by the law of total expectation and Fact 2.6, and the first inequality is by Claim 5.15.1.

In total by Eqs. (8) and (9), we have

$$\Delta\left((H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}), \varphi_i(X'_i, Z_i^{(U)}, H_i^{(U)})\right) \leq 2^{-d/4+1/2} + \delta \cdot 2^d + 2^{-\lambda} \leq \delta \cdot 2^d + 2^{-\lambda+2}.$$

Since C_i is a YES instance, with probability at least $(1-\delta)$, it holds that $x_i \leftarrow X_i^{(U)}$ is in the set $S_i \setminus S_i^+$. Therefore, by Fact 2.4 it holds that

$$\Delta\left((H_i^{(U)}, A_i^{(U)}, X_{i+1}^{(U)}), \varphi_i(X_i^{(U)}, Z_i^{(U)}, H_i^{(U)})\right) \leq \delta \cdot (2^d + 1) + 2^{-\lambda+2}.$$

This concludes the proof of Proposition 5.14.

5.5 Communication Complexity and Verifier Run Time

We analyze the communication complexity of the protocol by accounting for each message that is sent.

- By Lemma 3.2, the cost of sending g is $O(n^2 + \lambda^2 + (\log k + \ell) \cdot (\lambda + n))$ and the cost of sending $(z_1, ... z_k)$ is $O(k \cdot (\log n + \log k + \lambda + \ell))$.
- By Lemma 2.14, the cost of sending $(h_1, \ldots, h_k)$ is $k \cdot \mathsf{poly}(\log n, \log k, \lambda)$ and the cost of sending $(\alpha_1, ... \alpha_k)$ is $O(k \cdot (\log n + \log k + \lambda))$.

- By Lemma 2.14, the cost of sending $(f_1, \ldots, f_k)$ is $O(k \cdot (\log n + \log k + \lambda)$ and the cost of sending $(\beta'_2, \ldots \beta'_k)$ is $O(k \cdot (\log n + \log k + \lambda))$.
- Sending x_1 costs n bits.

In total, the communication complexity is $O(n^2) + k \cdot \mathsf{poly}(\log n, \log k, \lambda)$. Lastly, we analyze the verifier's run time. For each iteration i the verifier's running time is as follows.

- By Lemma 3.2, the cost of computing g is $\mathsf{poly}(n, \log k, \lambda)$.
- By Lemma 2.14, the cost of computing h_i is $\mathsf{poly}(n, \log k, \lambda)$.
- By Lemma 2.14, the cost of computing f_i is $\mathsf{poly}(n)$.
- The cost of computing C_i is $\mathsf{poly}(n)$.

In total, the verifier running time is $k \cdot \mathsf{poly}(n, \log k, \lambda)$

6 From Honest to Malicious Verifier

In this section, we show how efficiently to transform an *honest-verifier* SZK protocol with round-by-round soundness, into a *malicious-verifier* SZK protocol. Our transformation builds on the prior work of Goldreich, Sahai and Vadhan [GSV98] who showed a generic transformation from honest to malicious verifiers for SZK, which unfortunately (and as discussed in more detail in the introduction) is not efficient enough for our purposes.

Theorem 6.1. *Suppose a problem Π has a public-coin honest-verifier* SZK *proof system. This protocol can be transformed into a public-coin malicious-verifier* SZK *proof system for Π with the following properties when given security parameter λ:*

1. *Suppose the original protocol has r rounds and the prover and verifier communication in its i^{th} round are s_i and ℓ_i, respectively. Then the transformed protocol has $2r$ rounds, and its total communication is $\left(\sum_{i\in[r]} s_i + O\left(\sum_{i\in[r]} \ell_i^4\right)\right)$.*
2. *The completeness and statistical zero-knowledge errors are at most $\mathsf{poly}(r, \ell_{\max}) \cdot 2^{-\Omega(\lambda)}$ more (additively) than the respective errors in the original protocol, where $\ell_{\max} = \max_i \ell_i$.*
3. *If the original protocol has round-by-round soundness error ϵ, then this protocol has soundness error $\left(\epsilon r 2^\lambda + \frac{1}{2\lambda}\right)$.*
4. *The verifier runs in time polynomial in the input length, r, $\ell_{\max}$, and λ, as does the prover, if given oracle access to the prover from the original protocol.*

The transformation that we use to prove Theorem 6.1 is almost exactly the same as the one of Goldreich, Sahai and Vadhan [GSV98]. The main difference is that [GSV98] first perform an $O(r)$-fold parallel repetition of the underlying HVSZK protocol, where r is its round complexity. This increases the communication complexity by a factor of r, which we cannot afford.

In contrast, in our analysis we avoid the use of parallel repetition and instead rely on the underlying protocol satisfying a stronger notion of soundness - namely, round-by-round soundness (Definition 2.20).

Key to the [GSV98] transformation is a random selection protocol described in Lemma 6.2. For ease of description, we refer to the two parties in this protocol as Merlin (M) and Arthur (A). When we eventually invoke this protocol in our transformation, the prover will play the part of M and the verifier the part of A. The following description is from Vadhan's PhD thesis [Vad99], and we have made some of the accounting more precise since this will be significant to us.

Lemma 6.2 ([Vad99, Proposition 6.3.3]). *There is an interactive protocol (M, A) that takes inputs $(1^\ell, 1^q, 1^p)$, outputs an element of $\{0,1\}^\ell$, and has the following properties:*

- **Efficiency:** *The protocol is public-coin (for both parties), and both M and A run in polynomial time (in ℓ, p, and q). It consists of four messages, with A sending the first message. It has communication complexity $O(\ell^4)$.*[9]

- **Soundness:** *For all Merlin strategies M^* and all sets $T \subseteq \{0,1\}^\ell$, the probability that the output lies in T is at most:*

$$2^p \cdot \frac{|T|}{2^\ell} + \frac{1}{q}$$

- **Strong Simulablity:** *There exists a polynomial-time black-box simulator S such that for any Arthur strategy A^*, the statistical difference between the following two distributions is at most $\mathsf{poly}(q, \ell) \cdot 2^{-\Omega(p)}$:*

  1. *Execute (M, A^*) on input (ℓ, q, p), let $\alpha \in \{0,1\}^\ell$ be the output of the protocol, and v the view of A^*. Output (v, α).*
  2. *Sample $\alpha \leftarrow \{0,1\}^\ell$ uniformly at random. Output $(S^{A^*}(1^\ell, 1^q, 1^p, \alpha), \alpha)$.*

Using Lemma 6.2 we are now ready to prove Theorem 6.1.

Proof of Theorem 6.1. Given a public-coin honest-verifier SZK protocol (P_H, V_H) for a problem Π, consider the protocol (P, V) from Fig. 7. We show that this protocol satisfies the requirements of Theorem 6.1.

Efficiency. The increase in the number of rounds and message sizes follow from the corresponding efficiency guarantees of the Random Selection (RS) protocol from Lemma 6.2, noting that the β_i's may be sent by P concurrently with the last message of the RS protocol. The efficiency of V and of P given oracle access to P_H follow from the efficiency of V_H, and of A and M in the RS protocol.

Completeness and Zero-Knowledge. The increases in the completeness and statistical zero-knowledge errors follow from arguments identical to the corresponding parts of the proof of Theorem 6.3.5 in [Vad99], which states the properties of the transformation in Fig. 7 when applied to protocols that may not be round-by-round sound. We omit these arguments here and refer the reader to [Vad99, Section 6.3.3].

[9]The bound of ℓ^4 is not stated in the original proposition in [Vad99], but follows from inspecting the construction presented there.

<u>Honest-to-Malicious Verifier Transformation</u>.

Background: Problem Π has a public-coin honest-verifier SZK protocol (P_H, V_H) with r rounds of interaction (that is, $2r$ messages), in which V_H sends the first message. We denote also by P_H the function that takes as input a partial transcript ending with a message of V_H, and outputs the next message of P_H according to the protocol. For $i \in [r]$, denote by ℓ_i the length of V_H's message in the i^{th} round.

Input: $x \in \{0,1\}^n$, and security parameter λ

Ingredients:

- The Random Selection protocol (M, A) from Lemma 6.2, which takes as input a tuple $(1^\ell, 1^q, 1^p)$, and outputs a string $\alpha \in \{0,1\}^\ell$ to both parties.

The Protocol:

1. For i from 1 to r:
   - Run the Random Selection protocol with P playing the part of M and V as A, with input $(1^{\ell_i}, 1^{2\lambda r}, 1^\lambda)$ to get output $\alpha_i \in \{0,1\}^{\ell_i}$.
   - P computes $\beta_i \leftarrow P_H(x, \alpha_1, \beta_1, \ldots, \alpha_i)$ and sends it to V.
2. V accepts iff V_H would accept on the transcript $(x, \alpha_1, \beta_1, \ldots, \alpha_r, \beta_r)$.

Figure 7: Honest-to-Malicious Verifier Transformation

Soundness. Let State be the state function that witnesses the protocol (P_H, V_H) having round-by-round soundness error at most ϵ. In our protocol, V accepts the "sub-transcript" $(\alpha_1, \ldots, \beta_r)$ if and only if V_H would have accepted with this as the transcript. By the definition of State, this transcript is accepted by V_H only if $\mathsf{State}(x, (\alpha_1, \ldots, \beta_r)) = \mathtt{alive}$. We bound the probability of this happening when x is a NO instance of Π.

Fix any NO instance x. For any $i \in [r]$, any partial sub-transcript $\tau_i = (\alpha_1, \beta_1, \ldots, \alpha_{i-1})$, and any potential prover message β, denote by $T_{\tau_i,\beta}$ the set of $\alpha \in \{0,1\}^{\ell_i}$ such that $\mathsf{State}(x, (\tau_i, \beta, \alpha)) = \mathtt{alive}$. Since (P_H, V_H) is a public-coin protocol, its round-by-round soundness immediately implies the following claim.

Claim 6.2.1. *For any $i \in [r]$, τ_i, and β, if $\mathsf{State}(x, \tau_i) = \mathtt{doomed}$, then $|T_{\tau_i,\beta}| \le \epsilon \cdot 2^{\ell_i}$.*

Consider a sub-transcript $(\alpha_1, \ldots, \beta_r)$ generated by the protocol (P, V). Note that, since x is a NO instance, we have $\mathsf{State}(x, \emptyset) = \mathtt{doomed}$. Thus, in order for us to have $\mathsf{State}(x, (\alpha_1, \ldots, \beta_r)) = \mathtt{alive}$, there needs to be an $i \in [r]$ and a partial sub-transcript $\tau_i = (\alpha_1, \beta_1, \ldots, \alpha_{i-1})$ such that $\mathsf{State}(x, \tau_i) = \mathtt{doomed}$ and $\mathsf{State}(x, (\tau, \beta_{i-1}, \alpha_i)) = \mathtt{alive}$. In other words, it has to be the case for some i that $\alpha_i \in T_{\tau_i, \beta_{i-1}}$.

By the soundness of the RS protocol from Lemma 6.2 and the bound from Claim 6.2.1, for any i, this happens with probability at most $\left(2^\lambda \cdot \epsilon + \frac{1}{2\lambda r}\right)$. Thus, by the union bound, the probability that there exists such an $i \in [r]$ is at most $\left(\epsilon r 2^\lambda + \frac{1}{2\lambda}\right)$.

□

7 Public-coin Malicious Verifier SZK Batching for NISZK

In this section we state and prove our main theorems. We first state our public-coin HVSZK batch verification protocol for NISZK.

Theorem 7.1. *Let $\Pi \in \mathsf{NISZK}$ and $k = k(n) \in \mathbb{N}$ such that $k(n) \leq 2^{n^{0.01}}$ and let $\lambda = \lambda(n) \in \mathbb{N}$ be a security parameter such that $\lambda(n) \leq n^{0.1}$. Then, $\Pi^{\otimes k}$ has a public-coin* HVSZK *protocol with completeness, zero-knowledge, and round-by-round soundness errors of $2^{-\lambda}$.*

The communication complexity is $\big(k + \mathsf{poly}(n)\big) \cdot \mathsf{poly}(\log n, \log k, \lambda)$ and the verifier running time is $k \cdot \mathsf{poly}(n, \log k, \lambda)$.

Furthermore, the protocol consists of k rounds. The length of the verifier's first message is $\mathsf{poly}(n)$ and the length of each of the verifier's other messages is $\mathrm{polylog}(n, k, \lambda)$.

Proof. Let $\Pi \in \mathsf{NISZK}$. Given common input $(x_1, \ldots, x_k)$, the protocol for $\Pi^{\otimes k}$ starts by having the prover and verifier perform the reduction of Corollary 4.4 on each instance to obtain circuits $(C_1, \ldots, C_k)$ of the problem $\mathsf{AI}^{n',m'}_{2^\ell,\delta}$, with respect to $\delta = 2^{-n^{1/5}}$ and $\ell = \log(n) \cdot \log(k) \cdot \lambda'$, where $\lambda' = (\lambda + \log k + 7)$, where $n' = n^c$ and $m' = n^h$ for some constants c and h that are guaranteed to exist by Corollary 4.4. Note that indeed $\ell = \mathsf{poly}(n)$ as required.

Actually, in order to perform this reduction it must hold that $\delta(n) > \frac{2^{-o(n^{1/4})}}{\ell}$, and this indeed is the case since

$$\frac{2^{-o(n^{1/4})}}{\ell} \leq \frac{2^{-n^{1/5}}}{\ell} \leq 2^{-n^{1/5}} = \delta.$$

Next, the prover and verifier execute the batch verification protocol of Lemma 5.1 with respect to the circuits $(C_1, \ldots, C_k)$, with security parameter λ' and with respect to

$$d = 7 \cdot (\log n' + \log k + \lambda') + \ell = 7 \cdot (c \cdot \log n + \log k + \lambda') + \ell.$$

We now show that this value of d indeed satisfies the conditions of Lemma 5.1.

In order to show the second condition (i.e. that $d < 2\ell - 2\lambda'$) it is enough to show that $\ell > 7 \cdot (c \cdot \log n + \log k) + 9\lambda'$, and this indeed the case since

$$7 \cdot \big(c \cdot \log n + \log k\big) + 9\lambda' < \log(n) \cdot \log(k) \cdot \lambda' = \ell,$$

for sufficiently large n, k and λ.

We now continue to the first condition (i.e., that $\delta \leq 2^{-d}$). First note that

$$\begin{aligned} d &= 7 \cdot (c \cdot \log n + 2 \log k + 2\lambda') + \ell \\ &\leq 2 \cdot \log(n) \cdot \log(k) \cdot \lambda' \\ &\leq 2 \cdot \log(n) \cdot \log(k) \cdot (\lambda + \log k + 7) \\ &\leq 2 \cdot \log(n) \cdot n^{0.01} \cdot (n^{0.1} + n^{0.01} + 7) \\ &\leq n^{0.15}, \end{aligned} \tag{10}$$

where all inequalities hold for sufficiently large n, k and λ and the penultimate inequality follows by our setting of parameters (in particular, $k(n) \leq 2^{n^{0.01}}$ and $\lambda \leq n^{0.1}$). Now we have that

$$\delta = 2^{-n^{1/5}} \leq 2^{-n^{0.15}} \leq 2^{-d}.$$

where the last inequality is by Eq. (10).

The completeness and round-by-round soundness of the protocol are immediate from Corollary 4.4 and Lemma 5.1. To show honest-verifier statistical zero knowledge, consider a simulator that first invokes the reduction of Corollary 4.4 and then invokes the simulator establishing the (statistical) zero-knowledge property of Lemma 5.1. The zero-knowledge error is therefore at most

$$k \cdot (\delta \cdot 2^{d+2} + 2^{-\lambda'+6}) = k \cdot \delta \cdot 2^{d+2} + 2^{-\lambda-1} \leq 2^{n^{0.01} - n^{1/5} + n^{0.15} + 2} + 2^{-\lambda-1} \leq 2^{-n^{0.1}-1} + 2^{-\lambda-1} \leq 2^{-\lambda},$$

where the first inequality is by our setting of parameters and by Eq. (10).

□

Lastly, combining Theorem 7.1 with Theorem 6.1, we get a *malicious-verifier* SZK batch verification protocol.

Theorem 7.2. *Let* $\Pi \in \mathsf{NISZK}$ *and* $k = k(n) \in \mathbb{N}$ *such that* $k(n) \leq 2^{n^{0.01}}$ *and let* $\lambda = \lambda(n) \in \mathbb{N}$ *be a security parameter such that* $\lambda(n) \leq n^{0.09}$*. Then,* $\Pi^{\otimes k}$ *has a public-coin* SZK *protocol with completeness, soundness, and zero-knowledge errors of* $2^{-\Omega(\lambda)}$*, and communication complexity of* $\big(k + \mathsf{poly}(n)\big) \cdot \mathsf{poly}(\log n, \log k, \lambda)$*. The verifier running time is* $k \cdot \mathsf{poly}(n, \lambda, \log k)$ *and the number of rounds is* $O(k \cdot \lambda)$*.*

Proof. Let $\Pi \in \mathsf{NISZK}$. The protocol guaranteed by Theorem 7.1, with respect to security parameter $\lambda_1 = 2\cdot\lambda + \log^2 n + \log^2 k + \log k$, combined with the transformation of Theorem 6.1, with respect to security parameter $\lambda_2 = \lambda + \log^2 n + \log^2 k$, yields an $O(k)$-round malicious verifier public-coin SZK protocol with completeness and zero-knowledge errors of $2^{-\Omega(\lambda)}$, soundness error of $\frac{1}{\lambda}$, and communication complexity of $\big(k+\mathsf{poly}(n)\big)\cdot\mathsf{poly}(\log n, \log k, \lambda)$. We note that the conditions of Theorem 7.1 indeed hold since $k(n) \leq 2^{n^{0.01}}$ and $\lambda_1 = 2\cdot\lambda + \log^2 n + \log^2 k \leq 2\cdot n^{0.09} + \log^2 n + n^{0.02} \leq n^{0.1}$ as required.

In order to reduce the soundness error, we perform λ sequential repetitions of this protocol. By Lemma 2.11, we get a $\big(k \cdot \mathsf{poly}(\log n, \log k, \lambda)\big)$-round SZK protocol with completeness, soundness, and zero-knowledge errors of $2^{-\Omega(\lambda)}$, and with the desired communication complexity $\big(k+\mathsf{poly}(n)\big)\cdot \mathsf{poly}(\log n, \log k, \lambda)$. □

Acknowledgments

We thank the anonymous Eurocrypt 2021 reviewers for useful comments.

Inbar Kaslasi and Ron Rothblum were supported in part by a Milgrom family grant, by the Israeli Science Foundation (Grants No. 1262/18 and 2137/19), and grants from the Technion Hiroshi Fujiwara cyber security research center and Israel cyber directorate.

Prashant Vasudevan was supported in part by AFOSR Award FA9550-19-1-0200, AFOSR YIP Award, NSF CNS Award 1936826, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

References

[Aar12] Scott Aaronson. Impossibility of succinct quantum proofs for collision-freeness. *Quantum Information & Computation*, 12(1-2):21–28, 2012.

[ADM$^+$99] Noga Alon, Martin Dietzfelbinger, Peter Bro Miltersen, Erez Petrank, and Gábor Tardos. Linear hash functions. *J. ACM*, 46(5):667–683, 1999.

[AH91] William Aiello and Johan Hastad. Statistical Zero-knowledge Languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, 1991.

[APS18] Navid Alamati, Chris Peikert, and Noah Stephens-Davidowitz. New (and old) proof systems for lattice problems. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 619–643. Springer, 2018.

[AV19] Benny Applebaum and Prashant Nalini Vasudevan. Placing conditional disclosure of secrets in the communication complexity universe. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 4:1–4:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[BBD$^+$20] Marshall Ball, Elette Boyle, Akshay Degwekar, Apoorvaa Deshpande, Alon Rosen, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Cryptography from information loss. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 81:1–81:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BCH$^+$20] Adam Bouland, Lijie Chen, Dhiraj Holden, Justin Thaler, and Prashant Nalini Vasudevan. On the power of statistical zero knowledge. *SIAM J. Comput.*, 49(4), 2020.

[BDRV18] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. From laconic zero-knowledge to public-key cryptography - extended abstract. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 674–697. Springer, 2018.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250. Springer, 1998.

[BL13] Andrej Bogdanov and Chin Ho Lee. Limits of provable security for homomorphic encryption. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August*

18-22, 2013. Proceedings, Part I, volume 8042 of *Lecture Notes in Computer Science*, pages 111–128. Springer, 2013.

[BR94] Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 276–287. IEEE Computer Society, 1994.

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

[CHP12] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. *J. Cryptology*, 25(4):723–747, 2012.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 89–105, 1992.

[CRSW11] L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:68, 2011.

[Dru15] Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015.

[For87] Lance Fortnow. The complexity of perfect zero-knowledge (extended abstract). In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 204–209. ACM, 1987.

[For89] Lance Jeremy Fortnow. *Complexity-theoretic aspects of interactive proof systems.* PhD thesis, Massachusetts Institute of Technology, 1989.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 186–194. Springer, 1986.

[GG00] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000.

[GK93] Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm. *J. Cryptology*, 6(2):97–116, 1993.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMR98] Rosario Gennaro, Daniele Micciancio, and Tal Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In Li Gong and

Michael K. Reiter, editors, *CCS '98, Proceedings of the 5th ACM Conference on Computer and Communications Security, San Francisco, CA, USA, November 3-5, 1998*, pages 67–72. ACM, 1998.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987.

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

[GR14] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. *J. Cryptol.*, 27(3):480–505, 2014.

[GS89] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. *Advances in Computing Research*, 5:73–90, 1989.

[GSV98] Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *STOC*, 1998.

[GV99] Oded Goldreich and Salil P. Vadhan. Comparing entropies in statistical zero knowledge with applications to the structure of SZK. In *CCC*, 1999.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HRV18] Pavel Hubácek, Alon Rosen, and Margarita Vald. An efficiency-preserving transformation from honest-verifier statistical zero-knowledge to statistical zero-knowledge. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 66–87. Springer, 2018.

[KMN$^+$14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 374–383. IEEE Computer Society, 2014.

[KRR$^+$20] Inbar Kaslasi, Guy N. Rothblum, Ron D. Rothblum, Adam Sealfon, and Prashant Nalini Vasudevan. Batch verification for statistical zero knowledge proofs. *Electronic Colloquium on Computational Complexity (ECCC)*, 2020.

[KY18] Ilan Komargodski and Eylon Yogev. On distributional collision resistant hashing. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 303–327. Springer, 2018.

[LV16] Tianren Liu and Vinod Vaikuntanathan. On basing private information retrieval on np-hardness. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 372–386, 2016.

[MV03] Daniele Micciancio and Salil P. Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 282–298, 2003.

[NMVR94] David Naccache, David M'Raïhi, Serge Vaudenay, and Dan Raphaeli. Can D.S.A. be improved? complexity trade-offs with the digital signature standard. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 77–85. Springer, 1994.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[NV06] Minh-Huyen Nguyen and Salil P. Vadhan. Zero knowledge with efficient provers. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 287–295, 2006.

[Oka00] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *J. Comput. Syst. Sci.*, 60(1):47–108, 2000.

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Structure in Complexity Theory Conference*, pages 133–138, 1991.

[OV08] Shien Jin Ong and Salil P. Vadhan. An equivalence between zero knowledge and commitments. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, pages 482–500, 2008.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way fuctions are essential for non-trivial zero-knowledge. In *ISTCS*, pages 3–17, 1993.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 387–398. Springer, 1996.

[PV08] Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 536–553, 2008.

[SV03] Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM (JACM)*, 50(2):196–249, 2003.

[Vad99] Salil Pravin Vadhan. *A study of statistical zero-knowledge proofs.* PhD thesis, Massachusetts Institute of Technology, 1999.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.