Doubly-Efficient Batch Verification in Statistical Zero-Knowledge

Or Keret* Ron D. Rothblum† Prashant Nalini Vasudevan‡

May 21, 2024

Abstract

A sequence of recent works, concluding with Mu *et al.* (Eurocrypt, 2024) has shown that every problem Π admitting a non-interactive statistical zero-knowledge proof (NISZK) has an efficient zero-knowledge *batch verification* protocol. Namely, an NISZK protocol for proving that $x_1, \ldots, x_k \in \Pi$ with communication that only scales poly-logarithmically with k. A caveat of this line of work is that the prover runs in exponential-time, whereas for NP problems it is natural to hope to obtain a *doubly-efficient proof* – that is, a prover that runs in polynomial-time given the k NP witnesses.

In this work we show that every problem in NISZK $\cap$ UP has a *doubly-efficient* interactive statistical zero-knowledge proof with communication $\mathrm{poly}(n, \log(k))$ and $\mathrm{poly}(\log(k), \log(n))$ rounds. The prover runs in time $\mathrm{poly}(n, k)$ given access to the k UP witnesses. Here n denotes the length of each individual input, and UP is the subclass of NP relations in which YES instances have unique witnesses.

This result yields doubly-efficient statistical zero-knowledge batch verification protocols for a variety of concrete and central cryptographic problems from the literature.

1 Introduction

Suppose that a server, holding a long list of RSA public-keys $N_1, \ldots, N_k$ together with their factorizations, wants to convince an auditor that the public-keys are well formed. That is, that each N_i is a product of exactly two distinct primes. One option is to reveal all of the factorizations but this option is simultaneously extremely bad from a security perspective (as it reveals the clients' secret keys) and is highly inefficient if k is very large.

Efficient zero-knowledge proofs are protocols that allow short and easy-to-verify proofs of complex statements, in such a way that reveals nothing beyond the fact that the statement being proved is true. To solve the above problem one might employ a general purpose zero-knowledge proof from the literature. However, such proofs rely on unproven assumptions and provide only a computational soundness guarantee.

In this work, we are interested in *statistical zero-knowledge proofs* for batch verification. These are zero-knowledge proofs, in which both soundness and zero-knowledge hold information theoretically (and against unbounded adversaries) such that the communication required to prove the correctness of k statements scales sub-linearly (ideally, merely poly-logarithmically) with k. The

*Technion. Email: or.keret@campus.technion.ac.il.

†Technion. Email: rothblum@cs.technion.ac.il.

‡National University of Singapore. Email: prashant@comp.nus.edu.sg.

information theoretic nature of such proofs also means that typically they do not involve expensive cryptographic operations, and can be highly efficient.

Returning to the above question on batch verification of prime products, a protocol due to Gennaro *et al.* [GMR98] gives a (non-interactive) statistical zero-knowledge proof that N is a product of two distinct primes. Running this protocol separately on all of the alleged RSA moduli $N_1, \ldots, N_k$, we could indeed prove in statistical zero-knowledge that all of them are prime products, alas with communication that scales linearly with k.

An alternative approach, raised in a recent sequence of works [KRV21, KRR+20, MNRV24] has shown that *every* problem that has a non-interactive statistical zero-knowledge proof also has such a zero-knowledge proof for batch verification, in which the communication only scales poly-logarithmically with k. Unfortunately, these protocols have a prover that runs in *exponential-time*, whereas, for our example above, we would like for our server, who knows all of the factorizations, to run in polynomial-time. We refer to such proof-systems, in which the prover runs in polynomial-time given the NP witness, as *doubly-efficient proofs*.[1]

The question of doubly-efficient statistical zero-knowledge batch verification for prime products is interesting in its own right, but the same question is fascinating for a variety of cryptographic problems such as checking that a long list of public-keys/signatures/ciphertexts are all well-formed. This raises the natural question, posed already by Kaslasi *et al.* [KRR+20]:

> *Does every problem in* SZK $\cap$ NP *have a* doubly-efficient statistical zero-knowledge batch verification proof*: namely, a statistical zero-knowledge proof that* $x_1, \ldots, x_k \in \Pi$ *with communication that scales poly-logarithmically in* k *and an honest-prover that runs in polynomial-time given the* k NP *witnesses?*

A source of hope, especially for our prime product example, are results by Reingold, Rothblum and Rothblum [RRR21, RRR18, RR20], which give doubly-efficient batch verification protocols for any problem in UP – the class of NP problems in which YES instances have unique witnesses. This shows that a rich class of problems in NP have batch verification protocols (indeed, the prime product problem is in UP, the witness is just the prime factorization). Unfortunately, the UP batch verification protocols in the foregoing works are *not* zero-knowledge (assuming UP $\not\subseteq$ BPP) and, as a matter of fact, even explicitly reveal some of the witnesses to the verifier.

1.1 Our Results

Our main result is a doubly-efficient statistical zero-knowledge batch proof for any problem in NISZK $\cap$ UP. Recall that by [MNRV24] such problems have a statistical zero-knowledge batch verification proof in which the honest prover is inefficient, and on the other hand, by [RR20], the same set of problems have a *different* doubly-efficient interactive proof, which is not zero-knowledge. Our main result shows how to combine these two protocols to obtain the best-of-both-worlds:

Theorem 1.1. *Let* $k = k(n)$ *such that* $k(n) \leq 2^{n^{0.01}}$*. Every problem* $\Pi \in$ NISZK $\cap$ UP *has a public-coin* SZK *protocol for verifying* k *instances* $x_1, \ldots, x_k$*, each of length* n*, with the following parameters:*

[1] Doubly-efficient proofs are also studied in the context of problems in P (rather than NP) in which case we require the prover to run in polynomial-time, without any additional auxiliary information. See the recent survey by Goldreich [Gol18].

- *Communication complexity:* $\text{poly}(n, \log(k))$.
- *Number of rounds:* $\text{polylog}(n, k)$.
- *Verifier runtime:* $\tilde{O}(k) \cdot \text{poly}(n)$.
- *The honest prover, given also the k unique witnesses, runs in time* $\text{poly}(n, k)$.

We emphasize that in contrast to general purpose results for succinct arguments (a la [Kil92]), the batching protocol of Theorem 1.1 does not rely on any unproven assumptions and yields the strong guarantees of *statistical* soundness and zero-knowledge.

Combining Theorem 1.1 with existing NISZK protocols from the literature, we immediately derive doubly-efficient statistical zero-knowledge batch verification protocols for several important problems. In particular, using the NISZK for prime products of [GMR98] we obtain a batch verification protocol for prime products (as well as variants such as quasi-safe prime products), which in particular yields an efficient method for batch verification of signature verification keys in the [GMR98] undeniable signature scheme. Using the fact that quadratic residuosity (of Blum Integers) as well as quadratic non-residuosity is in NISZK [BDSMP91], we similarly obtain such batch verification protocols for these problems.[2]

We remark that in contrast to the result of [MNRV24], our batching protocol is *interactive*. Obtaining a doubly-efficient NISZK batching protocol for $\text{NISZK} \cap \text{UP}$ remains open. We elaborate on this in Section 1.3.

1.2 Technical Overview

Our starting point is the batch verification protocol for UP of Rothblum and Rothblum [RR20]. As noted above, this protocol meets all the requirements of Theorem 1.1, except that it is blatantly *not* statistical zero-knowledge (assuming $\text{UP} \cap \text{NISZK} \not\subseteq \text{BPP}$) as it explicitly reveals some of the witnesses. Our goal is to transform the protocol to be statistical zero-knowledge, while preserving its complexity.

As the UP batching protocol of [RR20] is public-coin, a natural approach to convert it to be zero-knowledge is to utilize the "commit-and-prove" framework of Ben-Or *et al.* [BOGG$^+$90]. This framework transforms public-coin protocols into zero-knowledge ones by letting the prover commit to her messages rather than sending them in the clear. Subsequently, the prover proves, using an additional zero-knowledge proof, that if she were to open the commitments, the original verifier would have accepted.

Trying to utilize this framework to our purposes we first run into the following problem: to attain a protocol that is simultaneously statistical zero-knowledge and statistically sound, we need commitments that are both statistically hiding and statistically binding. While such commitments do not exist per se, following [BMO90, IOS97, NV06, OV08] we can use an elegant relaxation of such commitments, called *instance-dependent commitments*.

An instance-dependent commitment scheme is a commitment scheme associated with an instance x of a promise problem Π. If x is a YES instance, the commitment is required to be hiding,

[2] For batch verification of QR, there is a simple reduction from k instances to 1 by taking a random modular subset product of the given integers. However, this only works in case we use the same modulus for each instance whereas our protocol works even when using different moduli. Also, we note that [BDSMP91] only give an NISZK for QNR. However, in case N is a Blum integer, QR reduces to QNR by multiplying the input by $-1 \pmod N$ (since -1 is a QNR modulo a Blum integer).

and if x is a NO instance, then it should be binding. The above sequence of works utilized the fact that we only need binding to hold for NO instances, and hiding to hold for YES instances, and so such commitments suffice for implementing zero-knowledge proofs and in particular for implementing the commitments in the framework of [BOGG+90]. In particular, Nguyen and Vadhan [NV06] used (a suitable variant of) instance dependent commitments to show that any problem in SZK $\cap$ NP has a doubly-efficient SZK proof.

Ong and Vadhan [OV08] (building on [NV06]) constructed instance-dependent commitment schemes for any problem in SZK. Thus, we would like to start with the UP batching protocol of [RR20] and compile it to be zero-knowledge by utilizing the commit-and-prove approach, while using an instance dependent commitment. For a given input $x_1, \ldots, x_k$, since each of the individual parts x_i is an instance of our problem Π, which is in SZK (in fact NISZK), it has a corresponding instance dependent commitment. Using a standard combiner[3] for commitments we could then obtain a single instance dependent commitment for the batch verification problem. Alas, the length of commitments (and decommitments) obtained in this way scales linearly with k, which we cannot afford.

A better approach is to utilize the main result of Mu *et al.* [MNRV24], which gives a *statistical zero-knowledge* batch verification protocol for Π. Using their protocol, in combination with the instance dependent commitment characterization of [NV06, OV08] we obtain a *direct* instance-dependent commitment scheme for the batch problem $\Pi^{\otimes k} = \{(x_1, \ldots, x_k) : \forall i, x_i \in \Pi\}$. But what are the lengths of commitments and decommitments in the resulting scheme? We observe that the main technical result of [MNRV24] can be interpreted as a reduction of k instances of an NISZK problem to a single instance of the *Image Density problem* [DSDCPY98]. In Image Density, the instances are circuits such that YES instances correspond to circuits that generate a distribution that is statistically close to uniform, whereas NO instances are circuits that generate a distribution with a relatively small support. While the size of the circuit generated by the [MNRV24] reduction scales linearly with k, the size of its *input and output* only grows poly-logarithmically in k. Our next observation is that the instance dependent commitment of [NV06] for Image Density has the feature that the lengths of the commitment and decommitment correspond only to *the input and output length of the circuit* rather than its size. Equipped with the above observation we obtain an instance-dependent commitment scheme for our batch verification problem with communication complexity $\mathrm{poly}(n, \log(k))$.

The above suffices for the "commit phase" of the transformation of Ben-Or *et al.* [BOGG+90] but for the "prove phase" we still need to run a zero-knowledge proof demonstrating that "had the prover opened the commitments, the verifier would have accepted". The naïve approach might be to use one of the classical zero-knowledge proofs for NP, such as the original proof due to Goldreich *et al.* [GMW91], while relying on the instance dependent commitment described above. Here however, we run into a major problem: the [GMW91] protocol, as well as all other zero-knowledge protocols in the literature, make non-blackbox use of the underlying circuit on which we prove correctness. In our case, this circuit incorporates the commitment's verification circuit, which includes the circuit C produced by the [MNRV24] reduction. As mentioned above, the circuit C has size $\mathrm{poly}(n, k)$. Therefore, this approach would result in an unacceptable communication complexity of $\mathrm{poly}(n, k)$.

To overcome this issue, we build on the recent work of Hazay *et al.* [HVW23], which used the

[3]Here we need a combiner between k commitments, which needs to be hiding if all of them are hiding, and binding if at least one of them is binding. A suitable combiner in this setting is to simply commit separately using each commitment scheme.

"MPC-in-the-head" framework of Ishai *et al.* [IKOS09] in order to compile public-coin interactive protocols to be zero-knowledge *while using the commitment scheme as a blackbox*. We emphasize that while all zero-knowledge proofs in the literature need an explicit description of the *circuit*, there is no a priori need to make a non blackbox use of the underlying commitment. Indeed, Hazay *et al.* show how to compile several public-coin interactive proofs from the literature to be zero-knowledge in such a blackbox way.

We proceed to describe the transformation in more detail. Recall that the MPC-in-the-head paradigm offers a general transformation from Secure Multiparty Computation (MPC) protocols to zero-knowledge protocols. The high-level structure is as follows:

- Let $\Pi \in \text{NP}$ be a promise problem with a corresponding witness relation $R_\Pi(x, w)$. We start with an MPC protocol for a functionality f that corresponds to verifying that the players' inputs form an additive secret sharing of a valid witness of R_Π (for some fixed x).
- The prover, "in-her-head" secret shares the witness amongst virtual parties and emulates the execution of the MPC protocol. Subsequently, she commits to the resulting views of the parties.
- The verifier selects a small subset of the views she wishes the prover to reveal.
- The prover decommits to these views.
- The verifier accepts only if the views are consistent (with one another and the protocol description) and are accepting.

We utilize the MPC-in-the-head paradigm to compile the [RR20] protocol to be zero-knowledge using the instance dependent commitment as a blackbox. The idea is to adjust our protocol so that the prover does not commit directly to her messages (as in [BOGG+90]). Rather, she commits to ℓ additive secret shares of each of her messages, for some small parameter ℓ. Then, the prover proceeds with an "MPC-in-the-head" proof, where she gives each simulated party a secret share of her messages. The computed functionality corresponds to the verification predicate used by the verifier in the UP batching protocol of [RR20].

This approach was inspired by the work of Hazay *et al.* [HVW23], and it only relies on black-box access to the circuit that verifies decommitments. Hence, since the circuit that verifies decommitments has input and output sizes $\text{poly}(n, \log(k))$, our objective of achieving communication complexity $\text{poly}(n, \log(k))$ remains feasible.

A final remaining problem is that the communication complexity of the MPC-in-the-head proof depends on the running time of the verifier for the UP batching protocol [RR20] (since the parties run an MPC protocol that emulates this computation). Unfortunately, this verifier has running time $k \cdot \text{poly}(n, \log(k))$, which is prohibitively large (and is inherent since the verifier has to, at the very least, read its input). We solve this last issue by analyzing more closely the verifier of the [RR20] UP batching protocol. We observe that the verifier in their protocol consists of two main phases:

- A "preprocessing step" which depends only on the input and verifier's coin tosses. We emphasize that this step does not depend on the prover's messages and in particular can happen before the interaction begins. The verifier needs to only keep an internal state of size $\text{poly}(n, \log(k))$ at the end of this step.

- An interactive step, in which the prover and verifier interact. The key point is that this step can be implemented in $\text{poly}(n, \log(k))$ time.

To capitalize on this observation, we let both the verifier and the prover independently execute the preprocessing step. We then utilize the "MPC-in-the-head" proof only for the second step, which is computable in time $\text{poly}(n, \log k)$, thereby obtaining communication complexity $\text{poly}(n, \log k)$.

1.3 Open Questions

We highlight several natural follow-up questions that we leave open:

1. **Generalizing to** SZK**:** The key question left open by the line of work on batch verification of statistical zero-knowledge proofs is a general batch verification protocol that works for all of SZK, rather than just NISZK. Given such a result, it seems possible that our techniques, in combination with the UP batching protocol of [RR20], could yield a doubly-efficient SZK batch protocol for $\text{SZK} \cap \text{UP}$.

2. **Generalizing to** NP**:** Our result is limited to UP languages, where we inherit this limitation from [RR20]. The work of Bitansky *et al.* [BKP+23] indicates that a general doubly-efficient batch verification protocol for NP is somewhat unlikely, as it would lead to (unconditional) *statistical witness indistinguishable proofs* for NP. However, this barrier becomes meaningless (i.e., a non-issue) when considering languages in $\text{SZK} \cap \text{NP}$. Thus, there is no fundamental barrier that we are aware of in generalizing our results from $\text{NISZK} \cap \text{UP}$ to $\text{NISZK} \cap \text{NP}$ (or even $\text{SZK} \cap \text{NP}$ for that matter).

3. **Non-interactive:** Our protocol is designed to handle problems that have non-interactive proof-systems (namely, NISZK and UP). Unfortunately however, in contrast to the protocols that we start off with, our batch verification protocol is highly interactive. This is due to two reasons: first, the UP batching protocol of [RR20] which we use is highly interactive and second, the instance dependent commitment of [NV06] that we use is interactive. Thus, making our batch-verification protocol be non-interactive seems to require using fundamentally different techniques.

1.4 Organization

Preliminaries are in Section 2. The proof of the main result (namely, Theorem 1.1) is in Section 3. Some proofs are deferred to Appendices A to C.

2 Preliminaries

A *promise problem* Π consists of two disjoint sets of strings $\Pi = (\Pi_Y, \Pi_N)$. The set Π_Y is called the set of YES instances and the set Π_N is called the set of NO instances.

Definition 2.1. *The* statistical distance *between two random variables X, Y on a finite universe U, denoted by $\Delta(X, Y)$, is defined as:*

$$\Delta(X,Y) = \max_{S \subseteq U} \left(X(S) - Y(S)\right) = \frac{1}{2} \sum_{u \in U} \left|X(u) - Y(u)\right|.$$

2.1 Interactive Proofs and Zero-Knowledge

We use $\text{view}_V\left(P\left(x\right), V(x)\right)$ to refer to the *view* of the verifier in an execution of an interactive protocol with prover P and verifier V on common input x. The view 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 view is *accepting* if, at the end of the corresponding interaction, the verifier accepts.

Definition 2.2 (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 Π, consists of a probabilistic polynomial-time verifier V and a computationally unbounded prover P such that following properties hold:*

- **Completeness:** *For any $x \in \Pi_Y$:*

$$\Pr\left[\text{view}_V\left(P\left(x\right), V(x)\right) \text{ is accepting}\right] \geq 1 - c(|x|).$$

- **Soundness:** *For any (computationally unbounded) cheating prover P^* and any $x \in \Pi_N$:*

$$\Pr\left[\text{view}_V\left(P^*\left(x\right), V(x)\right) \text{ is accepting}\right] \leq s(|x|).$$

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

An interactive proof (P, V) is *public-coin* if all the messages sent by the verifier are independent random strings (with a fixed length that is independent of the interaction). Let $\Pi = (\Pi_Y, \Pi_N) \in$ NP, with a corresponding witness relation $R_\Pi(x, w)$.

Definition 2.3 (Doubly-Efficient Interactive Proof)**.** *Let Π be a promise problem. A* doubly-efficient interactive proof *for the relation R_Π is an interactive-proof (P, V) for Π in which the prover strategy P can be implemented in probabilistic polynomial-time given any* NP *witness $w \in \{w' : (x, w') \in R_\Pi\}$, as an auxiliary input.*

Zero-Knowledge. For the SZK definition, we allow the malicious verifier to have access to an auxiliary input $a \in \{0,1\}^*$. Accordingly, we also provide the simulator with the same auxiliary input a.

Definition 2.4 (SZK)**.** *Let $z = z(n) \in [0,1]$. An interactive-proof (P, V) for Π is a* statistical zero-knowledge proof (SZK)*, with zero-knowledge error z, if for every probabilistic polynomial-time verifier V^* there exists a probabilistic polynomial-time algorithm Sim (called the* simulator*) such that for any $x \in \Pi_Y$ and $a \in \{0,1\}^*$:*

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

If the completeness, soundness, and zero-knowledge errors are all negligible in $|x|$, we say that the interactive proof is an SZK proof. We also use SZK to denote the class of promise problems having SZK proofs.

Non-Interactive Statistical Zero-Knowledge. Next, we define the *non-interactive* variant of statistical zero-knowledge, denoted NISZK. As usual in this setting, we give the prover and verifier access to a uniformly generated common random string (CRS).

Definition 2.5 (NISZK)**.** *Let $c = c(n) \in [0,1], s = s(n) \in [0,1]$ and $z = z(n) \in [0,1]$. A* non-interactive statistical zero-knowledge proof (NISZK) *with completeness error c, soundness error s and zero-knowledge error z for a promise problem Π, consists of a probabilistic polynomial-time verifier V, a computationally unbounded prover P and a polynomial $\ell = \ell(n)$ (the length of the* CRS*) such that the following properties hold:*

- **Completeness:** *For any $x \in \Pi_Y$:*

$$\Pr_{r \leftarrow \{0,1\}^{\ell(|x|)}} \left[V(x, r, P(x,r)) \text{ accepts} \right] \geq 1 - c(|x|).$$

- **Soundness:** *For any $x \in \Pi_N$:*

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

- **Zero-Knowledge:** *There exists a probabilistic polynomial-time algorithm Sim (called the* simulator*) such that for any $x \in \Pi_Y$:*

$$\Delta\Big(\big(U_\ell, P\left(x, U_\ell\right)\big), Sim(x) \Big) \leq z(|x|),$$

  where U_ℓ denotes a random variable distributed uniformly over $\{0,1\}^{\ell(|x|)}$.

Unless stated otherwise, we assume that $c(\cdot), s(\cdot)$ and $z(\cdot)$ are negligible in $|x|$. We use NISZK to denote the class of promise problems having such an NISZK protocol.

2.2 Instance-Dependent Commitments for NISZK

Next, we recall the notion of instance-dependent commitments and revisit their construction for NISZK [NV06].[4] An *instance-dependent commitment* [BMO90, IOS97, NV06, OV08], for a promise problem Π, is a commitment scheme associated with an instance $x \in \{0,1\}^*$. Unlike standard commitment schemes, an instance-dependent commitment scheme requires the hiding property to hold only when x is a YES instance, and the binding property to hold only when x is a NO instance. We now provide a formal definition (the following definitions of instance-dependent commitments and their security are based on [OV08])

Definition 2.6 (Instance-dependent commitments)**.** *An* instance-dependent commitment scheme *is a family of protocols $\{\mathrm{Com}_x\}_{x \in \{0,1\}^*}$ with the following properties:*

- *Scheme Com_x proceeds in two stages: a* commit *stage and a* reveal *stage. In both stages, the* sender *and* receiver *receive instance x as common input, and hence we denote the sender and receiver as S_x and R_x, respectively, and write $\mathrm{Com}_x = (S_x, R_x)$.*

[4] The later work of Ong and Vadhan [OV08] constructs instance-dependent commitments for all of SZK. However, their construction is more complex and the simpler construction of instance-dependent commitments for NISZK, due to [NV06], suffices for our results.

- *At the beginning of the commit stage, sender S_x receives a private input $b \in \{0,1\}$, which denotes the bit that S_x is supposed to commit to. At the end of the commit stage, both sender S_x and receiver R_x output a* commitment *c.*
- *In the reveal stage, sender S_x sends a pair (b, d), where d is the* decommitment *string for bit b. Receiver R_x accepts or rejects based on x, b, d, and c.*
- *The sender S_x and receiver R_x algorithms are computable in polynomial time (in $|x|$), given x as auxiliary input.*
- *For every $x \in \{0,1\}^*$, R_x will always accept (with probability 1) if both sender S_x and receiver R_x follow their prescribed strategy.*

The instance-dependent commitment scheme $\{\mathrm{Com}_x = (S_x, R_x)\}_{x\in\{0,1\}^*}$ is public coin *if for every $x \in \{0,1\}^*$, all messages sent by R_x are independent random coins.*

To simplify notation, we write Com_x or (S_x, R_x) to denote the instance-dependent commitment scheme $\{\mathrm{Com}_x = (S_x, R_x)\}_{x\in\{0,1\}^*}$. The hiding and binding properties of standard commitments extend in a natural way to their instance-dependent analogs.

Definition 2.7 (Hiding of instance-dependent Commitments)**.** *Let $\varepsilon = \varepsilon(n) \in [0,1]$. The instance-dependent commitment scheme $\mathrm{Com}_x = (S_x, R_x)$ is* ε-statistically hiding *on $I \subseteq \{0,1\}^*$ if for every R^*, and every $x \in I$,*

$$\Delta\left(\mathrm{view}_{R^*}\left(S_x(0), R^*\right), \mathrm{view}_{R^*}\left(S_x(1), R^*\right)\right) \leq \varepsilon(|x|),$$

where random variable $\mathrm{view}_{R^*}(S_x(b), R^*)$ denotes the view of R^* in the commit stage after interacting with $S_x(b)$. For a problem $\Pi = (\Pi_Y, \Pi_N)$, an instance-dependent commitment scheme Com_x for Π is ε-statistically hiding *on the YES instances if Com_x is ε-statistically hiding on Π_Y.*

Definition 2.8 (Binding of instance-dependent Commitments)**.** *Let $\varepsilon = \varepsilon(n) \in [0,1]$. The instance-dependent commitment scheme $\mathrm{Com}_x = (S_x, R_x)$ is* ε-statistically binding *on $I \subseteq \{0,1\}^*$ if for every S^*, and for all $x \in I$, the malicious sender S^* succeeds in the following game with probability at most $\varepsilon(|x|)$:*

1. *The sender S^* interacts with R_x in the commit stage obtaining commitment c.*
2. *Then S^* outputs pairs $(0, d_0)$ and $(1, d_1)$, and* succeeds *if in the reveal stage, $R_x(0, d_0, c) = R_x(1, d_1, c) =$* accept*.*

For a problem $\Pi = (\Pi_Y, \Pi_N)$, an instance-dependent commitment scheme Com_x for Π is ε-statistically binding *on the NO instances if Com_x is ε-statistically binding on Π_N.*

Remark 2.9. *We defined instance-dependent commitment schemes as bit commitments. However, they can be extended to string commitments with the same round complexity by executing multiple bit commitments in parallel.*

We revisit the instance-dependent commitment scheme for all of NISZK, given in [NV06]. This scheme is for the NISZK-complete promise problem Image Density[5] [DSDCPY98, GSV99]. Since

[5]The commitment scheme, as presented in [NV06], was for a different promise problem known as EA. However, it involved a preprocessing step intended to transform an EA instance into an Image Density instance. Since we will work directly with Image Density instances, we skip this preprocessing step.

Image Density is NISZK-complete, this commitment scheme can be adapted to suit any NISZK promise problem. We first define the Image Density problem.

Definition 2.10. *Let $\varepsilon = \varepsilon(n) \in [0,1]$. The* ε-Image Density *problem* (ID_ε)*, is defined as follows:*

$$\mathrm{ID}_{\varepsilon,Y} = \{C : \Delta(U_\mu, C) \leq \varepsilon\},$$
$$\mathrm{ID}_{\varepsilon,N} = \{C : |\mathrm{Supp}(C)| \leq \varepsilon \cdot 2^\mu\},$$

where C is a circuit of size n with η input bits and μ output bits. We highlight that we use the unconventional symbols η and μ, as n and m will be used later for alternative purposes.

That is, YES instances of Image Density have an output distribution that is statistically close to uniform, and NO instances of Image Density have an output distribution with a small support. The next lemma, due to [NV06], shows that Image Density has an instance-dependent commitment.

Lemma 2.11 ([NV06])**.** *Let $\varepsilon = \varepsilon(n) \in [0,1]$ and let n, η, μ denote the size, the input length, and the output length of an* ID_ε *circuit, respectively. The problem* ID_ε *has an instance-dependent commitment scheme that is 2ε-statistically hiding on the* YES *instances and $(\mathrm{poly}(\mu) \cdot \varepsilon)$-statistically binding on the* NO *instances. Furthermore, the number of bits sent between the sender and the receiver during the commit and reveal phases is $\mathrm{poly}(\eta, \mu)$, the receiver runs in time $n \cdot \mathrm{poly}(\eta, \mu, \log n)$, and the commitment scheme is public coin and constant-round.*

We emphasize that the poly in the furthermore clause above is a fixed polynomial and in particular does not depend on the *size* of the circuit. While the furthermore clause of Lemma 2.11 follows from the construction of Nguyen and Vadhan [NV06], it is not explicitly stated therein. Hence, to give precise bounds on the communication complexity, we provide a proof of Lemma 2.11 in Appendix B.

2.3 Batch Protocols

We introduce the concept of batch problems and review known batching results for UP and for NISZK [RR20, MNRV24].

Given a promise problem Π, we define the promise problem $\Pi^{\otimes k}$ that consists of k equal-sized instances of Π. We denote by n the size of each instance.

Definition 2.12. *For a given promise problem Π and an integer k, $\Pi^{\otimes k} = (\Pi_Y^{\otimes k}, \Pi_N^{\otimes k})$ is defined as follows:*

$$\Pi_Y^{\otimes k} = \{(x_1, \ldots, x_k) : \forall i \in [k], x_i \in \Pi_Y, |x_1| = \ldots = |x_k|\}, \text{ and}$$
$$\Pi_N^{\otimes k} = \left\{(x_1, \ldots, x_k) \in (\Pi_Y \cup \Pi_N)^k : \exists j \in [k], x_j \in \Pi_N, |x_1| = \ldots = |x_k|\right\}.$$

2.3.1 Batching for NISZK

We introduce a lemma from [MNRV24]. We note that this lemma is not directly mentioned in [MNRV24], but can be deduced by examining the proofs of [MNRV24, Theorem 1.1] and [MNRV24, Lemma 3.9]. We analyze these proofs to establish this lemma in Appendix A.

Lemma 2.13 ([MNRV24])**.** *Let $\Pi \in$ NISZK and $k = k(n)$ such that $k(n) \leq 2^{n^{0.01}}$. Then, $\Pi^{\otimes k}$ has a randomized Karp reduction to* ID_ε *with the following properties:*

- *The reduction is computable in time* $k \cdot \mathrm{poly}(n, \log k)$ *and uses* $\mathrm{poly}(n, \log k)$ *random coins.*
- *The reduction never errs on NO instances and errs only with probability negligible in* n *and* k *on YES instances.*
- ε *is negligible in both* n *and* k*.*
- *The* ID_ε *circuit produced by the reduction has size* $k \cdot \mathrm{poly}(n, \log k)$*, with input and output sizes* $\mathrm{poly}(n, \log k)$*.*

2.3.2 Doubly Efficient Batching for UP

Next we present the batch verification result for UP, due to [RR20].

Theorem 2.14 ([RR20, Corollary 5])**.** *For every promise problem* $\Pi \in \mathrm{UP}$*, there exists a public-coin interactive proof, with perfect completeness and soundness error* $\frac{1}{2}$*, for verifying that* k *instances* $x_1, \ldots, x_k$*, each of length* n*, are all in* Π*. The complexity of the protocol is as follows:*

- *Communication complexity:* $\mathrm{poly}(n, \log(k))$*.*
- *Number of rounds:* $\mathrm{polylog}(n, k)$*.*
- *Verifier runtime:* $\tilde{O}(n \cdot k) + \mathrm{poly}(n, \log(k))$*.*
- *The honest prover, given the* k *unique witnesses, runs in time* $\mathrm{poly}(n, k)$*.*

Furthermore, before the interaction begins, the verifier runs a pre-processing step on inputs $x_1, \ldots, x_k$ *and the public coins to be sent to the prover. The pre-processing time is* $\tilde{O}(n \cdot k)$*, and its output is a string* pp *of length* $\mathrm{poly}(n, \log(k))$*. Following the interaction, the verifier runs in time* $\mathrm{poly}(n, \log(k))$ *on inputs* pp *and the prover messages and decides whether to accept or reject.*

Remark 2.15. *The results in [RR20] are stated for languages, but readily extend to the setting of promise problems. More importantly, the "furthermore" part of Theorem 2.14 is not explicitly stated in [RR20], but can be inferred by inspecting their protocol, see Appendix C for details.*

2.4 Secure Multiparty Computation

Our protocol relies on the MPC-in-the-head framework of Ishai *et al.* [IKOS09]. In this section we provide the relevant background following [IKOS09]. Since it suffices for our purposes, we consider *semi-honest* MPC protocols, with perfect security, for functionalities of a particular form.

Let ℓ be the number of players, which will be denoted by $P_1, ..., P_\ell$. We will be interested in computations in which all players share a public input x, and each player P_i holds a local private input w_i. We consider protocols that securely realize an ℓ-party functionality f, where f maps the joint input $(x, w_1, \ldots, w_\ell)$ to a single output bit b.

A protocol Π is specified via its next message function. That is, $\Pi(i, x, w_i, r_i, (m_1, \ldots, m_j))$ returns the set of ℓ messages sent by P_i in round $j+1$ given the public input x, its local input w_i, its random input r_i, and the messages $m_1, \ldots, m_j$ it *received* in the first j rounds. The output of Π may also indicate that the protocol should terminate, in which case Π returns the local output of P_i. The *view* of P_i, denoted by view_i, includes w_i, r_i, and the messages *received* by P_i during

the execution of Π. Note that the messages sent by an uncorrupted player P_i as well as its local output can be inferred from view_i and x by invoking the next message function of Π. It will be useful to define the following natural notion of consistency between views.

Definition 2.16 (Consistent views)**.** *We say that two views* view_i *and* view_j *are* consistent *(for the protocol* Π *and the public input* x*) if the outgoing messages implicit in* (view_i, x) *are identical to the incoming messages reported in* view_j *and vice versa.*

The following lemma asserts that an ℓ-tuple of views corresponds to some honest execution of Π if and only if every pair of views is consistent.

Lemma 2.17 ([IKOS09, Lemma 2.3])**.** *Let* Π *be an* ℓ*-party protocol as above and* x *be a public input. Let* $\text{view}_1, \ldots, \text{view}_\ell$ *be an* ℓ*-tuple of (possibly incorrect) views. Then all pairs of views* $(\text{view}_i, \text{view}_j)$ *are consistent for* Π *and* x *if and only if there exists an honest execution of* Π *with public input* x *(and some choice of private inputs* w_i *and random inputs* r_i*) in which* view_i *is the view of* P_i *for every* $1 \leq i \leq \ell$*.*

In the semi-honest model, one may break the security requirements into the following correctness and privacy requirements.

Definition 2.18 (Correctness)**.** *We say that the protocol* Π *realizes a deterministic* ℓ*-party functionality* $f(x, w_1, \ldots, w_\ell)$ *with* perfect correctness *if for all inputs* $x, w_1, \ldots, w_\ell$*, the probability that the output of some player is different from the output of* f *is* 0*, where the probability is over the independent choices of the random inputs* $r_1, \ldots, r_\ell$*.*

Definition 2.19 (t-Privacy)**.** *Let* $1 \leq t < \ell$*. We say that* Π *realizes* f *with* perfect t-privacy *if there is a probabilistic polynomial-time simulator Sim such that for any inputs* $x, w_1, \ldots, w_\ell$ *and every set of corrupted players* $T \subseteq [\ell]$*, where* $|T| \leq t$*, the joint view* $\text{view}_T(x, w_1, \ldots, w_\ell)$ *of players in* T *is distributed identically to* $Sim\left(T, x, (w_i)_{i \in T}, f_T(x, w_1, \ldots, w_\ell)\right)$*.*

We now define a notion of efficiency for protocols that securely realize functionalities computable by circuits.

Definition 2.20 (Efficiency)**.** *Let* f *be an* ℓ*-party functionality computed by a circuit* C *of size* s*, and let* Π *realize* f*. We say that* Π *is* C-efficient *if every player* P_i *runs in time* $\text{poly}(|x|, |w_i|, \ell, s)$*, during the entire execution of* Π*.*

In our batch proof, we utilize an MPC protocol as part of the MPC-in-the-head paradigm. As an example, we will instantiate our batch proof with the renowned BGW MPC protocol [BGW88] to demonstrate that our proof is realizable.

Theorem 2.21 ([BGW88, Theorem 1], see also [AL17])**.** *For every function* f *computable by a circuit* C*, there exists a* C*-efficient MPC protocol* Π *with* 5 *players, that realizes* f *with perfect correctness and perfect* 2*-privacy.*

3 Doubly Efficient Zero-Knowledge Batching for UP ∩ NISZK

In this section, we prove Theorem 1.1 by showing a doubly-efficient statistical zero-knowledge batch proof for problems in UP ∩ NISZK. We begin by combining the results of [MNRV24] and [NV06], to show that batch instances of NISZK have an instance-dependent commitment scheme with communication complexity that scales poly-logarithmically with k.

Lemma 3.1. *Let* $\Pi \in$ NISZK *and* $k = k(n)$ *such that* $k(n) \leq 2^{n^{0.01}}$. *Then,* $\Pi^{\otimes k}$ *has an instance-dependent commitment scheme that is* ε*-statistically hiding on the* YES *instances and* ε*-statistically binding on the* NO *instances, with* ε *being negligible in* n *and in* k. *The number of bits sent between the sender and the receiver during the commit and reveal phases is* $\mathrm{poly}(n, \log k)$. *Furthermore, the commitment scheme is public-coin and constant-round, and the receiver runs in time* $k \cdot \mathrm{poly}(n, \log k)$.

We recall that an instance $(x_1, \ldots, x_k) \in \Pi^{\otimes k}$ is considered a YES instance if each x_i is a YES instance of Π, and is considered a NO instance if at least one the x_i is a NO instance of Π.

Proof. The lemma follows by composing the commitment scheme for Image Density of Lemma 2.11 with the randomized Karp reduction from $\Pi^{\otimes k}$ to Image Density of Lemma 2.13.

Thus, on common input $(x_1, \ldots, x_k)$, the sender and the receiver reduce $(x_1, \ldots, x_k)$ to an ID_ε circuit C. This only requires the sender to send the randomness that was used for the reduction, which is of size $\mathrm{poly}(n, \log k)$, to the receiver. Then, the sender and the receiver use C as the common input for the commitment scheme of Lemma 2.11 and proceed with the commit and reveal phases as per usual. The commitment scheme is ε-statistically hiding on the YES instances since the reduction of Lemma 2.13 fails only with negligible probability when the sender is honest. The commitment scheme is ε-statistically binding on the NO instances since the reduction of Lemma 2.13 never fails on NO instances. □

Remark 3.2. *Our main protocol involves commitments to multiple bits. We could in principle improve the efficiency of the protocol by reducing* $(x_1, \ldots, x_k)$ *to an* ID_ε *instance just once rather than with each commitment, but we avoid this optimization for the sake of simplicity.*

We now have everything we need to construct the doubly-efficient SZK batch proof for UP ∩ NISZK, thereby establishing Theorem 1.1.

3.1 Proof of Theorem 1.1

We first present the protocol and then prove why it meets all the requirements of Theorem 1.1.

Consider the UP batching protocol $(P_{\mathrm{UP}}, V_{\mathrm{UP}})$, promised by Theorem 2.14. In this protocol, the verifier V_{UP} runs in two phases. In the preprocessing phase, the verifier runs on inputs $x_1, \ldots, x_k$ and the public coins to be sent to the prover, and produces a string pp of length $\mathrm{poly}(n, \log k)$ (we emphasize that in this phase there is no interaction). In the online phase, the verifier interacts with the prover and decides whether to accept based only on pp and the prover's messages. In this phase, the verifier runs in time $\mathrm{poly}(n, \log k)$. Denote the verifier messages by $\overline{\alpha} = (\alpha_1, \ldots, \alpha_r)$, where α_i is the verifier's message in round i, and similarly denote the prover's messages by $\overline{\beta} = (\beta_1, \ldots, \beta_r)$. Let $C_{V_{\mathrm{UP}}}$ be a size $\mathrm{poly}(n, \log k)$ circuit that computes the verifier's decision predicate. That is, $C_{V_{\mathrm{UP}}}$ operates on inputs pp and the prover's messages and determines whether to accept.

Let f be the following function, that takes as input a string pp, and ℓ additive shares of $\overline{\beta}$ (the value of ℓ will be determined later):

$$f\left(pp, \overline{\beta^1}, \ldots, \overline{\beta^\ell}\right) = C_{V_{\mathrm{UP}}}\left(pp, \overline{\beta^1} \oplus \ldots \oplus \overline{\beta^\ell}\right),$$

where $\oplus$ denotes the bitwise exclusive-or operation:

$$\overline{\beta^i} \oplus \overline{\beta^j} = \left(\beta_1^i \oplus \beta_1^j, \ldots, \beta_r^i \oplus \beta_r^j\right).$$

Let Π be an ℓ-player $C_{V_{\mathrm{UP}}}$-efficient[6] MPC protocol, with a constant $\ell = 5$, that realizes f with perfect correctness and perfect (semi-honest) 2-privacy, i.e., the protocol of Theorem 2.21. The public input is pp, and $\overline{\beta^i}$ is the private input of player P_i. Upon completion of Π, the local output of all players should match the output of f.

Using the MPC protocol from above, together with the protocol of Theorem 2.14, We now describe the zero-knowledge batch verification protocol in Fig. 1, where the commitment scheme that is used throughout the protocol is the one of Lemma 3.1.

We analyze the protocol to verify that it fulfills all the requirements of Theorem 1.1.

Completeness. Suppose $x_1, \ldots, x_k \in \Pi_Y$. The perfect completeness of the underlying UP batching protocol and the perfect correctness of Π ensure that the shares $\overline{\beta^i}$ that P commits to during Step 1 would lead player P_i, running on public input pp and private input $\overline{\beta^i}$, to output 1 with probability 1. The views that P commits to are consistent with their respective players' inputs and with each other, so V would accept if P successfully revealed all the required commitments. Since the commitment scheme of Lemma 3.1 has correctness as long as the sender and the receiver follow their prescribed strategies, perfect completeness of our protocol follows.

Soundness. Suppose at least one of $x_1, \ldots, x_k$ is in Π_N and fix a (computationally unbounded) cheating prover P^*. Without loss of generality, let us assume that P^* is deterministic. By Lemma 3.1, the commitment scheme used throughout the protocol is statistically binding, with binding error negligible in both n and k. We delay addressing the binding error for now, by first analyzing the soundness for a modified protocol (P', V'), which we will now describe.

In this modified protocol, commitments are not used and the prover P' sends all her messages openly. Accordingly, the verifier V' does not verify decommitments on Step 6a, but only ensures that the messages sent in Step 5 are consistent with the previous messages sent by P'.

The prover P' simulates the adversary P^* and acts as an intermediary between P^* and V'. Since P^* is the adversary for the unmodified protocol, she is expected to use commitments, whereas V' does not. Thus, the prover P' will manipulate the communication as follows. The prover P' will act as the receiver of the commitments and decommitments made by P^*. After each commitment is done, P' uses its unbounded computational power to extract the message from the commitment. If the commitment is invalid or if it can be opened ambiguously, a default message is selected instead. After extracting the message, P' sends it in the clear to V'. Whenever P' receives a message from V', she feeds it to P^*. This describes the modifications in (P', V'). We move on to analyze the soundness of the modified protocol.

Recall $C_{V_{\mathrm{UP}}}$, a circuit that operates on inputs pp and the prover's messages and computes the V_{UP} verifier's decision predicate. Given the soundness of the underlying UP batching protocol of Theorem 2.14, the messages $\overline{\beta}$ that P' sends during Step 1, would make the verifier V_{UP} reject

[6] We slightly abuse notation here since $C_{V_{\mathrm{UP}}}$ does not compute the functionality f (Note that f has $\ell+1$ inputs while $C_{V_{\mathrm{UP}}}$ only has two). This abuse of notation is justified since the sizes of $C_{V_{\mathrm{UP}}}$ and the closely related circuit that computes f differ only by a multiplicative factor of $\mathrm{poly}(\ell)$, and ℓ is constant.

Common Inputs: $x_1, \ldots, x_k \in \{0,1\}^n$.
Prover's Additional Inputs: $w_1, \ldots, w_k$, which are the unique witnesses for $x_1, \ldots, x_k$.

The protocol employs the commitment scheme described in Lemma 3.1 and the MPC protocol of Theorem 2.21.

The Protocol:

1. The prover P and the verifier V execute the protocol of Theorem 2.14, with the following modification: Each time the prover P is supposed to send a message β_i in the clear, she instead additively secret shares her message into ℓ shares $\beta_i = \beta_i^1 \oplus \ldots \oplus \beta_i^\ell$. She then commits to each of those shares separately. Crucially, since the protocol of Theorem 2.14 is public-coin, the verifier V can disregard the modification in the prover's messages and continue to send random coins as usual.

2. The prover P first collects the shares from the previous step:

$$\forall i \in [\ell] : \overline{\beta^i} \triangleq \left(\beta_1^i, \ldots, \beta_r^i\right)$$

   Subsequently, the prover and the verifier, each on their own, run the pre-processing step of V_{UP} promised by Theorem 2.14 on inputs $x_1, \ldots, x_k$ and the verifier's messages to obtain pp. Then, the prover simulates the MPC protocol Π (defined earlier in the text) with public input pp and player P_i having private input $\overline{\beta^i}$. This yields the ℓ views of the ℓ players.

3. The prover commits separately to the view of each of the ℓ players.

4. The verifier selects two players $i \neq j$ at random and asks the prover to reveal their views.

5. The prover decommits to $\left(\left(\beta_1^i, \ldots, \beta_r^i\right), \mathrm{view}_i\right)$ and $\left(\left(\beta_1^j, \ldots, \beta_r^j\right), \mathrm{view}_j\right)$.

6. The verifier accepts if and only if all the following hold true:

   (a) The prover has successfully revealed all the required commitments.

   (b) The public input in view_i is pp, the private input is $\left(\beta_1^i, \ldots, \beta_r^i\right)$.

   (c) The public input in view_j is pp, the private input is $\left(\beta_1^j, \ldots, \beta_r^j\right)$.

   (d) The views (view_i, view_j) are consistent, and the players output 1 in both.

Figure 1: Doubly-efficient SZK batching protocol for UP $\cap$ NISZK

with probability at least $\frac{1}{2}$. That is, $\Pr\left[C_{V_{\mathrm{UP}}}\left(pp, \overline{\beta}\right) = 0\right] \geq \frac{1}{2}$. Assume we are in the case where $C_{V_{\mathrm{UP}}}\left(pp, \overline{\beta}\right) = 0$. Therefore, since the MPC protocol Π has perfect correctness, an honest execution of Π with public input pp and private inputs $\overline{\beta^i}$ results in all players outputting 0, in which case V' rejects on Step 6d with probability 1.

An honest execution of the MPC protocol Π with different inputs will lead V' to reject in Steps 6b or 6c with probability at least $\frac{1}{\ell}$. If the execution is not honest, by Lemma 2.17, there exists a pair of inconsistent views, and V' selects it with probability at least $\frac{1}{\binom{\ell}{2}}$, causing V' to reject in Step 6d.

Considering $\Pr\left[C_{V_{\mathrm{UP}}}\left(pp, \overline{\beta}\right) = 0\right] \geq \frac{1}{2}$, we get soundness error $1 - \frac{1}{2\binom{\ell}{2}}$ for (P', V'). We now factor in the binding error. If for some verifier coins of V' together with some receiver coins of P', V' rejects after interacting with P' and all the commitments that P' simulates are binding, then the same coins, when used by the verifier V, cause V to reject after interacting with P^*. Thus, a union bound yields a statistical soundness error of $1 - \frac{1}{2\binom{\ell}{2}} + \delta(n, k)$, where δ is negligible in both n and k.

Taking ℓ to be a constant, we have obtained a constant soundness error for the protocol. Later, we will reduce the error to be negligible by (sequential) repetition (see Remark 3.3).

Zero Knowledge. Let V^* be a probabilistic polynomial-time verifier. Since the MPC protocol Π is perfectly (semi-honest) 2-private, it has a simulator Sim_{MPC}. Using V^* and Sim_{MPC}, we construct a simulator Sim_{ZK} for our protocol in Fig. 2. The commitment scheme that is used throughout the simulation is the instance-dependent commitment of Lemma 3.1. We note that the simulator Sim_{ZK} only utilizes V^* in a black-box manner, and recall that black-box zero-knowledge implies auxiliary input zero-knowledge [GO94].

The zero-knowledge analysis closely follows that of [GMW91, IKOS09], and we outline it here for the sake of completeness. Suppose $x_1, \ldots, x_k \in \Pi_Y$. By Lemma 3.1, the commitment scheme used throughout the simulation will be statistically hiding, with hiding error negligible in both n and k. To demonstrate that the output distribution of the simulator is statistically close to that of real verifier views, we examine several hybrid distributions, wherein the simulator is given the witnesses to $x_1, \ldots, x_k$:

1. Distributions $A_0, \ldots, A_t$: In distribution A_i, during the first i attempts, the simulator Sim_{ZK} acts like the honest prover in Step 1, committing to shares of the prover's messages instead of committing to random shares. In the rest of the steps, the simulator proceeds as usual. In the remaining attempts, the simulator follows its standard strategy during all steps.

   Since any number of secret shares smaller than ℓ distributes uniformly and independently, and because the commitments hide all shares except the two shares that are revealed, the statistical distance between each $A_i, A_i + 1$ is negligible in n and k.

2. Distributions $B_0, \ldots, B_t$: In all of these distributions, the simulator behaves like the honest prover in Step 1, committing to shares of the prover's messages instead of committing to random shares. In distribution B_i, during the first i attempts, instead of utilizing the MPC simulator Sim_{MPC}, the simulator Sim_{ZK} executes the MPC protocol Π to obtain the views of the two randomly selected players. Then, the simulator chooses the views of the remaining

The Simulator $Sim_{\mathrm{ZK}}(x_1, \ldots, x_k)$**:**

Attempt $t = \ell^2 \cdot \left(\log^2(n) + \log^2(k)\right)$ times:

1. Simulate Step 1 of the protocol with V^*, with the following modification: Whenever the prover is supposed to commit to shares corresponding to message β_i, commit to random shares instead.
2. Compute pp according to $x_1, \ldots, x_k$ and the verifier messages $\overline{\alpha}$ obtained from the previous step.
3. Randomly select a pair of players $i \neq j$, and run the MPC simulator accordingly. Namely, compute $Sim_{\mathrm{MPC}}\left(\{i,j\}, pp, \left(\overline{\beta^i}, \overline{\beta^j}\right), 1\right)$ to get the views of the players i, j.
4. Generate arbitrary views for the other players (of the right length) and feed V^* with commitments to all of the generated views.
5. The verifier V^* responds with a request that the views of the players $\{i', j'\}$ be revealed. If $\{i,j\} = \{i',j'\}$, the attempt succeeded. Consequently, The simulator reveals the requested views and outputs the view of V^*. If $\{i,j\} \neq \{i',j'\}$, the attempt failed, and we start over.

If all attempts fail, output $\perp$.

Figure 2: Simulator for the protocol of Theorem 1.1

players arbitrarily and proceeds as usual. In the other attempts, the simulator follows its standard strategy during all steps except for Step 1 (in which it mimics the honest prover).

Note that $B_0 = A_t$. The statistical distance between each $B_i, B_i + 1$ is exactly zero due to the perfect 2-privacy of the MPC protocol Π.

3. Distribution C: In this distribution, the simulator behaves like in distribution B_t, but instead of generating random views for the players that are not in $\{i, j\}$, it assigns them the views computed when executing the MPC protocol Π. The statistical distance between B_t, C is negligible in n and k since the commitments to the views of the players not in $\{i, j\}$ are never revealed, and the commitment scheme is statistically hiding.

The only difference between distribution C and that of real verifier views is that C can output $\perp$ when all attempts fail. In the sampling of the distribution C, each attempt succeeds with a probability of at least $\frac{1}{\binom{\ell}{2}}$. Given that attempts are independent, the simulator succeeds on at least one attempt with all but negligible probability in n and k. Therefore, the statistical distance between C and that of real views is negligible in n and k.

Since A_0 is the output distribution of the simulator Sim_{ZK}, we deduce that the statistical distance between the output distribution of Sim_{ZK} and that of real verifier views is negligible in n and k, as required.

Complexity. We begin with analyzing the communication complexity of the protocol. The commitment scheme of Lemma 3.1 increases the communication complexity by a multiplicative factor of only $\text{poly}(n, \log k)$ per committed bit, both for the commit and the reveal phases. Hence, for the sake of analysis, we can conveniently overlook this overhead.

- **Step 1:** The verifier's messages and prover's messages during Step 1 of the protocol have length $\text{poly}(n, \log k)$ by Theorem 2.14.
- **The remaining steps:** The circuit $C_{V_{\text{UP}}}$ has size $\text{poly}(n, \log k)$, and the MPC protocol Π is $C_{V_{\text{UP}}}$-efficient. Therefore, the overall size of the MPC players' views is $\ell \cdot \text{poly}(n, \log(k), \ell) = \text{poly}(n, \log(k), \ell)$, and this term dominates the communication complexity of the remaining steps.

Since our parameter ℓ is a constant, the total communication complexity is $\text{poly}(n, \log k)$.

We proceed to analyze the round complexity of the protocol. The commitment scheme of Lemma 3.1 is constant-round. Therefore, in Step 1, The protocol inherits its round complexity $\text{polylog}(n, k)$ from the underlying UP batching protocol of Theorem 2.14. During the remaining steps, the protocol proceeds with an additional constant number of rounds, resulting in a total round complexity of $\text{polylog}(n, k)$.

The verifier's runtime is $k \cdot \text{poly}(n, \log(k))$ due to the efficiency of the verifier of Theorem 2.14, the efficiency of the receiver of Lemma 3.1, and the $C_{V_{\text{UP}}}$-efficiency of the MPC protocol Π.

The prover runs in time $\text{poly}(n, k)$ given the k unique witnesses, since the protocol of Theorem 2.14 is doubly-efficient, and because the MPC protocol Π is $C_{V_{\text{UP}}}$-efficient.

Lastly, the protocol inherits its public-coin nature from the underlying UP batching protocol of Theorem 2.14 and the commitment scheme of Lemma 3.1.

Remark 3.3. *In our analysis, we only achieved a constant soundness error. Our zero-knowledge simulator only makes black-box use of the verifier, so by sequential composition (see [GO94]), we can repeat our proof* $\mathrm{poly}(\log(n), \log(k))$ *times to get a negligible soundness error while preserving zero-knowledge and maintaining the complexity of the proof as previously stated.*

Acknowledgements

Or Keret and Ron Rothblum are funded by the European Union (ERC, FASTPROOF, 101041208). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Prashant Nalini Vasudevan is supported by the National Research Foundation, Singapore, under its NRF Fellowship programme, award no. NRF-NRFF14-2022-0010.

References

[AL17] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, 2017. 12

[BDSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991. 3

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988. 12

[BKP+23] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. *Electron. Colloquium Comput. Complex.*, TR23-077, 2023. 6

[BMO90] Mihir Bellare, Silvio Micali, and Rafail M. Ostrovsky. Perfect zero-knowledge in constant rounds. In *Symposium on the Theory of Computing*, 1990. 3, 8

[BOGG+90] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *Advances in Cryptology—CRYPTO'88: Proceedings 8*, pages 37–56. Springer, 1990. 3, 4, 5

[DHRS07] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. *Journal of Cryptology*, 20:165–202, 2007. 24, 25

[DSDCPY98] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image density is complete for non-interactive-SZK. In *Automata, Languages and Programming: 25th International Colloquium, ICALP'98 Aalborg, Denmark, July 13–17, 1998 Proceedings 25*, pages 784–795. Springer, 1998. 4, 9

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015. 26

[GMR98] Rosario Gennaro, Daniele Micciancio, and Tal Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *5th ACM Conference on Computer and Communication Security (CCS'98)*, pages 67–72, San Francisco, California, November 1998. ACM, ACM Press. 2, 3

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991. 4, 16

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994. 16, 19

[Gol18] Oded Goldreich. On doubly-efficient interactive proof systems. *Found. Trends Theor. Comput. Sci.*, 13(3):158–246, 2018. 2

[GR17] Tom Gur and Ron D. Rothblum. A hierarchy theorem for interactive proofs of proximity. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 39:1–39:43. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. 26

[GSV99] Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and NISZK. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 467–484. Springer, 1999. 9

[HVW23] Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, and Mor Weiss. Beyond MPC-in-the-head: Black-box constructions of short zero-knowledge proofs. In *Theory of Cryptography Conference*, pages 3–33. Springer, 2023. 4, 5

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009. 5, 11, 12, 16

[IOS97] Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. A language-dependent cryptographic primitive. *J. Cryptology*, 10:37–50, 1997. 3, 8

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732. ACM, 1992. 3

[KRR+20] Inbar Kaslasi, Guy N. Rothblum, Ron D. Rothblum, Adam Sealfon, and Prashant Nalini Vasudevan. Batch verification for statistical zero knowledge proofs.

In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 139–167. Springer, 2020. 2

[KRV21] Inbar Kaslasi, Ron D. Rothblum, and Prashant Nalini Vasudevan. Public-coin statistical zero-knowledge batch verification against malicious verifiers. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 219–246. Springer, 2021. 2

[MNRV24] Changrui Mu, Shafik Nassar, Ron Rothblum, and Prashant Nalini Vasudevan. Strong batching for non-interactive statistical zero-knowledge. *Electron. Colloquium Comput. Complex.*, pages TR24–024, 2024. 2, 3, 4, 10, 12, 22, 23

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11:87–108, 1998. 24

[NV06] Minh-Huyen Nguyen and Salil Vadhan. Zero knowledge with efficient provers. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 287–295, 2006. 3, 4, 6, 8, 9, 10, 12, 24

[OV08] Shien Jin Ong and Salil 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. Proceedings 5*, pages 482–500. Springer, 2008. 3, 4, 8

[RR20] Guy N. Rothblum and Ron D. Rothblum. Batch verification and proofs of proximity with polylog overhead. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 108–138. Springer, 2020. 2, 3, 4, 5, 6, 10, 11, 26

[RRR18] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Efficient batch verification for UP. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 22:1–22:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. 2, 26

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM J. Comput.*, 50(3), 2021. 2

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802. ACM, 2013. 26

[RW04] Renato Renner and Stefan Wolf. Smooth Rényi entropy and applications. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, page 233. IEEE, 2004. 22

A Proof of Lemma 2.13

Before we prove Lemma 2.13, we revisit the relevant definitions and results that appear in [MNRV24]. We start with some basic definitions of probability theory.

Definition A.1. *Let X be a random variable distributed over a universe U, and for every $x \in U$, denote by $p_x = \Pr[X = x]$. We recall the following notions of entropy of X:*

- $H_0(X) = \log\left(|\{x : p_x \neq 0\}|\right)$.
- $H_1(X) = -\sum_{x \in U} p_x \log\left(p_x\right)$.
- $H_2(X) = -\log(\mathrm{cp}(X))$.[7]

We also define the following smoothed notion of entropy version utilized in [MNRV24].

Definition A.2 (Smooth Entropy [RW04])**.** *For any $\varepsilon \geq 0$, the* ε-smooth H_2 entropy *of a random variable X is defined as follows:*

$$H_2^{\varepsilon}(X) = \max_{Y \in \mathcal{B}_{\varepsilon}(X)} H_2(Y),$$

where $\mathcal{B}_{\varepsilon}(X)$ is the set of all distributions within statistical distance ε of X.

We recall the definition of the *Smooth Entropy Approximation* problem considered in [MNRV24].

Definition A.3. *Let $\varepsilon = \varepsilon(n) \in [0, 1]$. The* ε-Smooth Entropy Approximation *problem* ($\mathrm{SEA}_{\varepsilon}$)*, is defined as follows:*

$$\mathrm{SEA}_{\varepsilon,Y} = \left\{(C, k) : H_2^{\varepsilon}(C) \geq k + 1\right\},$$

$$\mathrm{SEA}_{\varepsilon,N} = \left\{(C, k) : H_0(C) \leq k - 1\right\},$$

where C is a circuit with η input bits, $\mu \leq 3\eta$ output bits, and $0 < k \leq \mu$.

That is, YES instances of $\mathrm{SEA}_{\varepsilon}$ are close to a distribution that has high H_2 entropy, and NO instances of $\mathrm{SEA}_{\varepsilon}$ have low H_0 entropy (i.e., a small support). We now introduce a lemma based on the work [MNRV24]. While this lemma is not stated explicitly in [MNRV24], it follows immediately from the proof of [MNRV24, Theorem 1.1].

Lemma A.4 ([MNRV24, See Proof of Theorem 1.1])**.** *Let $\Pi \in$ NISZK and $k = k(n)$ such that $k(n) \leq 2^{n^{0.01}}$. Then, $\Pi^{\otimes k}$ has a randomized Karp reduction to $\mathrm{SEA}_{\varepsilon}$ with the following properties:*

- *The reduction is computable in time $k \cdot \mathrm{poly}(n, \log k)$ and uses $\mathrm{poly}(n, \log k)$ random coins.*

[7] Recall that the collision probability of a distribution X is defined as $\mathrm{cp}(X) = \Pr_{x, x' \leftarrow X}[x = x']$.

- *The reduction never errs on NO instances and errs only with probability negligible in n and k on YES instances.*
- *ε is negligible in both n and k.*
- *The SEA_ε circuit C generated by the reduction has size $k \cdot \mathrm{poly}(n, \log k)$, with input and output sizes $\mathrm{poly}(n, \log k)$.*

Remark A.5. *The fact that the running time of the reduction and the size of the SEA_ε circuit C are both $k \cdot \mathrm{poly}(n, \log k)$ does not follow from the proof of [MNRV24, Theorem 1.1], but rather it can be inferred from the reduction that establishes [MNRV24, Theorem 5.1].*

We now have all the necessary details from [MNRV24] to establish Lemma 2.13. The reduction from $\Pi^{\otimes k}$ to $\mathrm{ID}_{\varepsilon'}$ operates as follows (note that we use ε' for the ID circuits and ε for the SEA circuits since these two parameters will be different). It first efficiently reduces the tuple $(x_1, \ldots, x_k)$ to an SEA_ε instance using the reduction described in Lemma A.4. Subsequently, it reduces the SEA_ε instance to an $\mathrm{ID}_{\varepsilon'}$ circuit $\widehat{C}$.

We now present the reduction from SEA_ε to $\mathrm{ID}_{\varepsilon'}$. This is a deterministic Karp reduction, and its analysis closely follows [MNRV24, Lemma 3.9]. Denote by (C, κ) the SEA_ε instance. The circuit C has η input bits and μ output bits. Let $H_{\mu,\kappa} = \{h : \{0,1\}^\mu \to \{0,1\}^\kappa\}$ be a pairwise-independent family of hash functions as in [MNRV24, Lemma 2.9]. Each hash function from this family is described by $O(\max(\mu, \kappa)) = O(\mu)$ bits. Construct the circuit C' that corresponds to $r = 20(\log^2 n + \log^2 k)$ copies of C evaluated independently. Its input length is $\eta' = \eta \cdot r$, and its output length is $\mu' = \mu \cdot r$. Similarly, let $\kappa' = \kappa \cdot r$. The reduction, on input (C, κ), outputs a circuit $\widehat{C}$ that works as follows:

- It takes as input a description h of a hash function in $H_{\mu',\kappa'}$ and an $x \in \{0,1\}^{\eta'}$.
- It outputs $\Big(h, h\big(C'(x)\big)\Big)$.

The output length of $\widehat{C}$ is $\hat{\mu} = O(\mu') + \kappa' < O(\max(\eta', \mu'))$. Its input length is also $O(\max(\eta', \mu'))$. Suppose (C, κ) is a YES instance of SEA_ε. That is, $H_2^\varepsilon(C) \geq \kappa + 1$, and thus $H_2^{\varepsilon'}(C') \geq \kappa' + r$, where $\varepsilon' = \varepsilon \cdot r$. This implies that there is a distribution Y that is at most ε'-far from C' that has $\mathrm{cp}(Y) \leq 2^{-(\kappa'+r)}$. Let H denote the random variable corresponding to a uniformly random $h \in H_{\mu',\kappa'}$. By the leftover hash lemma (see [MNRV24, Lemma 2.10] for the exact formulation), the statistical distance between $(H, H(Y))$ and $(H, U_{\kappa'})$ is at most $2^{(-r)/2}$. Thus, the distance between $(H, H(C'))$ and $(H, U_{\kappa'})$ is at most $\varepsilon' + 2^{(-r)/2} = \varepsilon \cdot 20\left(\log^2 n + \log^2 k\right) + 2^{-10\left(\log^2 n + \log^2 k\right)}$. Since ε is negligible in n and k, $\Delta\left(\widehat{C}, U_{\hat{\mu}}\right)$ is also negligible in both n and k.

On the other hand, suppose (C, κ) is a NO instance of SEA_ε, that is, $H_0(C) \leq \kappa - 1$. This means that C has support of size at most $2^{\kappa-1}$, and C' has support of size at most $2^{\kappa'-r}$. This implies that the support size of $(H, H(C'))$ is at most $|H_{\mu',\kappa'}| \cdot 2^{\kappa'-r} \leq 2^{-r} \cdot 2^{\hat{\mu}}$. Therefore, $\left|\mathrm{Supp}\left(\widehat{C}\right)\right| \cdot 2^{-\hat{\mu}} = 2^{-r}$ is negligible in both n and k.

Since the hash functions have a succinct description and are efficiently computable, the running time of the reduction and the size of the circuit $\widehat{C}$ are both $k \cdot \mathrm{poly}(n, \log k)$, and $\widehat{C}$ has input and output lengths $O(\max(\eta', \mu')) = \mathrm{poly}(n, \log k)$, as required. This completes the analysis of the reduction from SEA_ε to $\mathrm{ID}_{\varepsilon'}$. By combining the reduction of Lemma A.4 with the reduction from SEA_ε to $\mathrm{ID}_{\varepsilon'}$, we have proved Lemma 2.13.

B Proof of Lemma 2.11

We prove Lemma 2.11 by first presenting the instance-dependent commitment scheme of [NV06], and then demonstrating that it satisfies the conditions of Lemma 2.11.

The commitment scheme makes use of *interactive hashing* [NOVY98, DHRS07], specifically employing an information-theoretically secure protocol from [DHRS07]. We begin by defining interactive hashing.

Interactive Hashing. In an interactive hashing protocol, two players are participating, A and B. Player A receives an input W, while B has no input. Upon executing the protocol, both A and B output a pair (W_0, W_1), such that one of W_0, W_1 equals W. Informally, the protocol is secure for A if, when W is uniformly distributed, even a computationally unbounded B cannot determine which one of (W_0, W_1) equals W. The protocol is secure for B if, for any sufficiently sparse set S, even a computationally unbounded A cannot force both W_0 and W_1 to reside in S. We now provide the formal definitions, based on [NV06].

Definition B.1 (Interactive Hashing)**.** *A protocol (A, B) is called an* interactive hashing protocol *if it is an* efficient *two-party protocol with the following properties:*

- **Inputs:** *A has an input string $W \in \{0,1\}^\mu$ and B has no input.*
- **Outputs:** *A and B output two distinct values $W_0, W_1 \in \{0,1\}^\mu$ (in lexicographic order) such that one of W_0, W_1 equals W.*

Definition B.2. *Let D denote the distribution of the index $d \in \{0,1\}$ such that the string W_d corresponds to the input of A in the interactive hashing protocol. An interactive hashing protocol is* secure *for A if for every unbounded B^* the distributions $\{\text{view}_{B^*}(A(W), B^*), D\}$ and $\{\text{view}_{B^*}(A(W), B^*), U_1\}$ are identical when $W \equiv U_\mu$.*

*An interactive hashing protocol is (δ, ρ)-*secure *for B if for every $S \subseteq \{0,1\}^\mu$ of density at most δ and every computationally unbounded strategy A^*, it holds that $\Pr[W_0, W_1 \in S] < \rho$.*

We will rely on an elegant interactive hashing protocol due to Ding *et al.* [DHRS07]:

Lemma B.3 ([NV06, Theorem 4.3], based on [DHRS07])**.** *For every $0 < \delta < 1$, there exists a constant-round public-coin interactive hashing protocol (A, B) that is secure for A and $(\delta, \text{poly}(\mu)\cdot\delta)$-secure for B.*

Using Lemma B.3, we now present in Fig. 3 the construction of the instance-dependent commitment for ID_ε. We proceed to analyze the construction to show that it satisfies Lemma 2.11. (Although our focus is on the communication complexity, for completeness we provide a full analysis).

Complexity. During the commit phase, the sender samples a random output x from the circuit C and engages in an interactive hashing protocol with the receiver on input x. Since both of the parties in the interactive hashing protocol are efficient and C has output size μ, the number of exchanged bits is $\text{poly}(\mu)$. Subsequently, the sender sends one additional bit to the receiver. During the reveal phase, the sender sends the revealed bit along with the input r to C that was

Common Input: Circuit $C : \{0,1\}^\eta \to \{0,1\}^\mu$.
Sender's Additional Input: a bit $b \in \{0,1\}$.

Commit phase:

1. The sender S generates a random $r \in \{0,1\}^\eta$ and computes $x = C(r)$
2. (S, R) run the interactive hashing protocol (A, B) of Lemma B.3 with the parameter $\delta = \varepsilon$, and with S playing $A(x)$ and R playing B. Their common output is a pair (x_0, x_1).
3. The sender S finds $d \in \{0,1\}$ such that $x_d = x$ and sends $c = d \oplus b$.
4. The commitment z is defined as (x_0, x_1, c).

Reveal phase:

1. S reveals b and r.
2. The receiver R accepts if $C(r) = x_{c \oplus b}$ and otherwise it rejects.

Figure 3: Instance-dependent commitment scheme for ID_ε

used to generate x. The circuit C has input size η, thus poly (η, μ) bits are exchanged in both the commit and the reveal phases.

During the commit phase, the receiver runs in time $\mathrm{poly}(\eta, \mu)$, due to the interactive hashing protocol of [DHRS07] being efficient. During the reveal phase, the receiver runs in time $|C| \cdot \mathrm{poly}(\eta, \mu, \log(|C|))$ to evaluate the circuit C on input r. Hence, the receiver has runtime $|C| \cdot \mathrm{poly}(\eta, \mu, \log(|C|))$ in both phases.

Also note that the commitment scheme is constant-round and public-coin since the interactive hashing protocol of [DHRS07] is constant-round and public-coin.

Correctness. Given any input circuit and bit b, if the sender and the receiver follow their prescribed strategies, then the receiver always accepts. This is ensured by the underlying interactive hashing protocol of Lemma B.3 which guarantees that one of the output strings (x_0, x_1) will be equal to the input x.

Hiding error. In case C is a YES instance of ID_ε, then $\Delta(C, U_\mu) \leq \varepsilon$. For every receiver R^*, we denote by B^* the interactive hashing strategy induced by R^*. Additionally, we denote by $d \in \{0,1\}$ the index of the interactive hashing output that equals the input (to A). For a random variable X

and an index $b \in \{0,1\}$, we write as a shorthand $v(X,b) \triangleq (\text{view}_{B^*}(A(X), B^*), b)$. We then have:

$$\begin{aligned}\Delta(\text{view}_{R^*}(S(0),R^*), \text{view}_{R^*}(S(1),R^*)) &= \Delta(v(C,d\oplus 0), v(C,d\oplus 1))\\ &\le \Delta(v(C,d), v(U_\mu,d))\\ &\quad + \Delta(v(U_\mu,d), v(U_\mu,d\oplus 1))\\ &\quad + \Delta(v(U_\mu,d\oplus 1), v(C,d\oplus 1)))\\ &\le \Delta(C,U_\mu) + 0 + \Delta(C,U_\mu)\\ &\le 2\varepsilon.\end{aligned}$$

Therefore, the hiding error $\Delta\big(\text{view}_{R^*}(S(0),R^*), \text{view}_{R^*}(S(1),R^*)\big)$ is at most 2ε.

Binding error. In case C is a NO instance of ID_ε, the density of $\text{Supp}(C)$ is at most ε. Let S^* be a malicious sender participating in the game defining the binding property. The sender S^* can succeed only if both x_0, x_1 are in the support of C. Because we chose the parameter δ for the interactive hashing protocol to match the density of $\text{Supp}(C)$, by the $(\delta, \text{poly}(\mu)\cdot\delta)$-security of the interactive hashing we have $\Pr[x_0, x_1 \in \text{Supp}(C)] < \text{poly}(\mu)\cdot\delta$. Since $\delta = \varepsilon$, the probability of violating the binding condition is at most $\text{poly}(\mu)\cdot\varepsilon$.

C More Details on Theorem 2.14

The [RR20] batching proof (similarly to many interactive proofs in the literature, such as the sumcheck and [GKR15] protocols) is "holographic". This means that the verifier runs in time $\text{poly}(n, \log(k))$ given oracle access to the *low degree extension* (LDE) of the input $(x_1, \ldots, x_k)$ (see [GR17] for a formal treatment). Moreover, the locations of the oracle queries (to the LDE) of the input only depend on the verifier's internal randomness. Thus, to see that the furthermore part of Theorem 2.14 holds, observe that the verifier can compute these values during its pre-processing step (i.e., prior to its interaction with the prover).

To see that the [RR20] protocol is indeed holographic we briefly recall the construction. Their protocol (building on an idea from [RRR18]) is recursive, where in each step we reduce the task of batch proving k instances, to batch proving $k/2$ related instances. The reduction relies on an *interactive proof of proximity* (IPP) which is developed in the same work (improving on a prior result of [RVW13]). The reduction step proceeds by running an IPP whose input is a list of the k witnesses concatenated with an LDE of the k instances. The IPP verifier queries some points from this LDE (which is why we view the protocol as holographic) and needs to additionally query some points of the witnesses. The parameters are set so that the verifier only needs to query at most $k/2$ of the witnesses, so rather than actually performing these queries, we recursively check these via an additional batch verification protocol.[8]

[8] To facilitate the recursion, we need to rely on a protocol that does batch verification and additionally checks some (small-depth) predicate on the k witnesses.