Non-Interactive Zero-Knowledge from Non-Interactive Batch Arguments

Jeffrey Champion
UT Austin
jchampion@utexas.edu

David J. Wu
UT Austin
dwu4@cs.utexas.edu

Abstract

Zero-knowledge and succinctness are two important properties that arise in the study of non-interactive arguments. Previously, Kitagawa et al. (TCC 2020) showed how to obtain a non-interactive zero-knowledge (NIZK) argument for NP from a succinct non-interactive argument (SNARG) for NP. In particular, their work demonstrates how to leverage the succinctness property from an argument system and transform it into a zero-knowledge property.

In this work, we study a similar question of leveraging succinctness for zero-knowledge. Our starting point is a batch argument for NP, a primitive that allows a prover to convince a verifier of T NP statements $x_1, \ldots, x_T$ with a proof whose size scales sublinearly with T. Unlike SNARGs for NP, batch arguments for NP can be built from group-based assumptions in both pairing and pairing-free groups and from lattice-based assumptions. The challenge with batch arguments is that the proof size is only amortized over the number of instances, but can still encode full information about the witness to a small number of instances.

We show how to combine a batch argument for NP with a local pseudorandom generator (i.e., a pseudorandom generator where each output bit only depends on a small number of input bits) and a dual-mode commitment scheme to obtain a NIZK for NP. Our work provides a new *generic* approach of realizing zero-knowledge from succinctness and highlights a new connection between succinctness and zero-knowledge.

1 Introduction

In a non-interactive argument system for an NP language $\mathcal{L}$, a prover sends a single message π to try and convince an efficient verifier that an NP statement x is true (i.e., $x \in \mathcal{L}$). In an argument system [BCC88], we require soundness to hold against *computationally-bounded* provers (i.e., a computationally-bounded prover should not be able to convince the verifier of a false statement). Two of the most important properties considered in the context of cryptographic argument systems are zero-knowledge and succinctness:

- **Zero-knowledge**: We say a non-interactive argument satisfies zero-knowledge [GMR85] if the proof π for an NP statement x reveals nothing more about x other than the fact that the statement is true. We refer to such arguments as non-interactive zero-knowledge (NIZK) arguments [BFM88]. While NIZKs for NP are unlikely to exist in the plain model (unless NP $\subseteq$ BPP), a long line of works have constructed NIZKs in the common reference string (CRS) model from a broad range of algebraic assumptions including factoring [FLS90], assumptions on pairing groups [CHK03, GOS06, GOS12, LPWW20] and pairing-free groups [JJ21], lattice-based assumptions [CCH+19, PS19], and combinations of multiple assumptions [BKM20].
- **Succinctness:** A second property of interest is the length of the proof (and by correspondence, the verification complexity). We say that an argument system for $\mathcal{L}$ is succinct if the length of the proof and the running time of the verifier is sublinear (or more commonly, polylogarithmic) in the size of the NP relation associated with $\mathcal{L}$. Such arguments are referred to as *succinct non-interactive arguments*, or SNARGs [GW11]. We refer to [Mic95, Gro10, BCCT12, DFH12, Lip13, PHGR13, GGPR13, BCI+13, BCPR14, Gro16, BISW17, BCC+17, BISW18, BBHR18, COS20, CHM+20, Set20, ACL+22, BS23, CBBZ23] and the references therein for a survey of succinct arguments.

Many applications require non-interactive arguments that are simultaneously succinct and zero-knowledge (i.e., zkSNARGs). Given a SNARG and a NIZK (argument of knowledge), it is straightforward to obtain a zkSNARG by direct composition (i.e., the zkSNARG is a NIZK argument of knowledge of a SNARG proof of the statement). A natural question is whether there is a formal connection between these two fundamental properties of cryptographic arguments. Previously Kitagawa et al. [KMY20] showed that SNARGs for NP and one-way functions together imply NIZKs for NP (and by composition, a zkSNARG).[1] The intuition underlying this result is that since the length of a SNARG proof for an NP statement x is much shorter than the length of the associated witness w, the SNARG proof simply cannot reveal too many bits of the witness information-theoretically. Then, by composing the SNARG with leakage-resilient cryptography, this intuition can be leveraged to obtain a NIZK.

Batch arguments and zero knowledge. In this work, we continue this line of inquiry of studying the relationship between succinctness and zero-knowledge. Instead of focusing on SNARGs for NP, which have a very strong succinctness property and necessitates constructions in either idealized models or based on non-falsifiable assumptions [GW11], we start with the weaker notion of SNARGs for *batch* NP languages. This is a notion that has received extensive study recently [KVZ21, CJJ21a, CJJ21b, HJKS22, WW22, DGKV22, GSWW22, CGJ+22, KLVW23] and can be realized from *standard* cryptographic assumptions. At a high level, in a non-interactive batch argument (BARG), a prover can convince a verifier of a collection of T NP statements $(x_1, \ldots, x_T)$ with a proof of size $\mathsf{poly}(\lambda, s) \cdot o(T)$, where s is the size of the circuit computing the NP relation. In particular, a batch argument amortizes the cost of NP verification across multiple instances. While this amortization still confers some succinctness, it is certainly possible for a BARG proof to leak one or more of the underlying witnesses associated with the statements used to construct it. Thus, we ask the question:

Can we construct a NIZK argument for NP *from a non-interactive batch argument for* NP*?*

Our results. In this work, we give a generic construction of NIZKs for NP from a non-interactive batch argument for NP in conjunction with a dual-mode commitment scheme and a (sub-exponentially-hard) low-locality pseudorandom generator (PRG) with super-linear stretch. Dual-mode commitment schemes can be built from any lossy/dual-mode public-key encryption scheme, which is known from most standard assumptions [PW08, HLOV11, AFMP20]. A low-locality PRG is a PRG where each output bit only depends on a small number of the seed bits. PRGs with constant locality and super-linear stretch are a notable ingredient in the recent constructions of indistinguishability obfuscation from well-studied assumptions [JLS21, JLS22]. Our instantiations can rely on locality as high as $c \log \lambda$, where $c < 1$ is a constant and λ is the seed length; this is a much weaker requirement compared to the constant locality PRGs required for indistinguishability obfuscation. The local PRG can in turn be instantiated using Goldreich's family of PRGs [Gol00, CM01].

While the additional ingredients we rely on for constructing a NIZK for NP are (much) stronger than one-way functions, we emphasize that no combination of the underlying primitives by themselves are known to imply NIZKs for NP. We summarize our main result with the following theorem (see Corollary 3.15 for a formal description and parameter specification):

Theorem 1.1 (Informal). *Let $G\colon \{0,1\}^\lambda \to \{0,1\}^{\lambda^\delta}$ be a PRG with locality $c \log \lambda$ and super-linear stretch $\delta > 1$ for a (sufficiently-small) constant $c < 1$ and (sufficiently-large) $\delta > 1$. Then, assuming the existence of a dual-mode commitment scheme, a non-interactive batch argument for* NP *with (sufficiently-small) proof size,[2] and sub-exponential hardness of G, there exists a NIZK for* NP*.*

Our work highlights a new connection between the succinctness of an argument system and zero-knowledge. It also provides a new *generic* approach for constructing NIZKs for NP. Finally, by composing a batch argument satisfying certain efficiency properties (satisfied by most existing constructions [CJJ21b, WW22, CGJ+22]) with a NIZK for NP, we obtain a zero-knowledge batch argument. Thus, our approach can also be used to generically upgrade a

[1] More recently, Chakraborty et al. [CPW23] showed a similar implication holds starting from a mildly-compact computational witness map (a simpler primitive that is implied by a SNARG for NP).

[2] For instance, this is satisfied if the length of the proof scales polylogarithmically in the number of instances. More generally, we can instantiate the theorem even if the proof size scales with $T^{1/2-\varepsilon}$, where T is the number of instances and $\varepsilon > 0$ is a constant. We refer to Corollary 3.15 for the precise characterization.

non-interactive batch argument for NP into a zero-knowledge batch argument by relying only on a low-locality PRG and a dual-mode commitment (neither of which are known to imply NIZKs for NP).

Concurrent work. In a concurrent and independent work, Bitansky et al. [BKP+23a, BKP+23b] studied the relationship between batch arguments (both interactive and non-interactive) and statistical witness indistinguishability. Notably, their work also shows how to leverage batch arguments to construct statistical NIZK arguments (using a different set of techniques relative to this work). In the initial version of their work [BKP+23a], they showed how to construct a statistical NIZK argument for NP with a *non-uniform* prover from non-interactive batch arguments and lossy public-key encryption. This result is technically incomparable to ours. On the one hand, the prover in our construction is *uniform* while on the other, we require an additional assumption of sub-exponentially hard local PRGs; moreover, we achieve *computational* zero knowledge rather than statistical zero knowledge. Subsequent to the initial posting of this work and [BKP+23a], the authors have improved their construction [BKP+23b] to obtain statistical NIZK arguments with a uniform prover by only relying on non-interactive batch arguments and lossy public-key encryption. This is a strict strengthening over Theorem 1.1. An interesting open question that remains is whether NIZK arguments can be built from just non-interactive batch arguments and *vanilla* public-key encryption.

1.1 Technical Overview

Our starting point is the generic approach of Kitagawa et al. [KMY20] who show how to generically transform a SNARG for NP into a NIZK for NP. The approach of [KMY20] instantiates the hidden-bits paradigm of Feige et al. [FLS90] of combining a NIZK in the idealized hidden-bits model with a hidden-bits generator (HBG) [QRW19].

NIZKs in the hidden-bits model. The hidden-bits model is an *idealized* model for constructing non-interactive zero-knowledge proofs. In this model, a trusted party first generates a string of uniformly random bits $r_1, \ldots, r_m \xleftarrow{R} \{0,1\}$ and gives them to the prover. To construct a proof for a statement x, the prover selects a subset of indices $I \subseteq [m]$ along with a proof π. The verifier then receives $\{r_i\}_{i \in I}$ and π from the trusted party. The model ensures that the prover cannot influence the choice of bits $r_1, \ldots, r_m$ and that the verifier cannot learn the value of any unrevealed bit r_i for $i \notin I$. Feige et al. [FLS90] previously showed how to construct a NIZK with statistical soundness and perfect zero-knowledge in the hidden-bits model for the NP-complete problem of graph Hamiltonicity.

Hidden-bits generators. Given a NIZK in the idealized hidden-bits model, a number of works have shown how to transform it into a NIZK in the CRS model through a cryptographic compiler [FLS90, BY92, CHK03, GR13, CL18, QRW19, LPWW20, KMY20]. In this work, we focus on the abstraction based on hidden-bits generators introduced by Quach et al. [QRW19]. At a high-level, a hidden-bits generator is a cryptographic primitive that generates a (pseudorandom) sequence of hidden bits. The prover can then open up a subset of the bits while ensuring the unopened bits remain hidden. Moreover, the hidden-bits generator ensures that the prover has limited control over the output sequence of bits. In a sense, hidden-bits generators provide a cryptographic realization of the trusted sampling of the hidden-bits string in the hidden-bits model. Thus, combined with the (unconditional) NIZK for NP in the hidden-bits model, a hidden-bits generator immediately implies a NIZK for NP in the CRS model. We now describe the syntax of a hidden-bits generator more formally; we specifically consider the adaptation from [KMY20]:

- The setup algorithm Setup takes as input the security parameter λ and an output length m and outputs a common reference string crs.
- The generator algorithm GenBits takes the common reference string crs and outputs a bit-string $\mathbf{r} \in \{0,1\}^m$ of length m along with a generator state st. Here, $\mathbf{r}$ is the "hidden-bits string."
- The prove algorithm Prove takes the generator state st and a subset of indices $I \subseteq [m]$, and outputs a *succinct* proof π. The proof π is an "opening" to the bits of $\mathbf{r}$ indexed by I; we denote these bits by $\mathbf{r}_I \in \{0,1\}^{|I|}$.
- The verification algorithm Verify takes as input the common reference string crs, a set of indices $I \subseteq [m]$, a collection of bits $\mathbf{r}_I \in \{0,1\}^{|I|}$, and an opening π. The verification algorithms decides whether π is a valid opening or not to the bits indexed by I (with respect to crs).

The hidden-bits generator must in turn satisfy the following properties:

- **Correctness:** Correctness says that if $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^m)$ and $(\mathbf{r}, \mathsf{st}) \leftarrow \mathsf{GenBits}(\mathsf{crs})$, then for all sets of indices $I \subseteq [m]$, the opening π output by $\mathsf{Prove}(\mathsf{st}, I)$ is valid with respect to Verify.
- **Binding:** The binding property restricts the set of possible openings that can be computed by a computationally-bounded algorithm. Namely, for each crs in the support of Setup, there exists a subset $\mathcal{V}^{\mathsf{crs}} \subset \{0,1\}^m$ of "valid" hidden-bits strings. Namely, no efficient adversary can come up with an accepting proof π for a set of indices $I \subseteq [m]$ and an assignment $\mathbf{r}_I \in \{0,1\}^{|I|}$ that is inconsistent with every $\mathbf{r}' \in \mathcal{V}^{\mathsf{crs}}$ (i.e., an assignment $\mathbf{r}_I$ such that for all $\mathbf{r}' \in \mathcal{V}^{\mathsf{crs}}$, $\mathbf{r}'_I \neq \mathbf{r}_I$). Moreover, the set of possible hidden-bits strings induced by a particular CRS must be *sparse*: $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{m^\gamma \mathsf{poly}(\lambda)}$ for some constant $\gamma < 1$ and where λ is a security parameter.
- **Hiding:** The hiding property says that the *unopened* bits of $\mathbf{r}$ are pseudorandom. Namely, for any set $I \subseteq [m]$ and honestly-generated $\mathbf{r}$ and π, the distribution $(\mathsf{crs}, I, \mathbf{r}_I, \mathbf{r}_{\bar{I}}, \pi)$ is computationally indistinguishable from the distribution $(\mathsf{crs}, I, \mathbf{r}_I, \hat{\mathbf{r}}_{\bar{I}}, \pi)$ where $\hat{\mathbf{r}} \xleftarrow{R} \{0,1\}^m$ and $\bar{I} = [m] \setminus I$.

Kitagawa et al. [KMY20] show how to construct a hidden-bits generator satisfying the above properties by combining a SNARG for NP with a leakage-resilient (weak) pseudorandom function (PRF):

- The CRS contains m random points in the domain of the PRF $x_1, \ldots, x_m$ and the CRS for the SNARG.
- The hidden-bits string is constructed by sampling a PRF key k and setting $r_i \leftarrow \mathsf{PRF}(k, x_i)$ for each $i \in [m]$.
- The opening for a subset $I \subseteq [m]$ is a SNARG proof that there exists k such that for all $i \in I$, $r_i = \mathsf{PRF}(k, x_i)$.

In this case, binding follows from security of the (weak) PRF (as long as the length of the PRF key is smaller than the output length m) in conjunction with soundness of the SNARG (i.e., the only possible openings are to those consistent with an evaluation under a PRF key k on the inputs $x_1, \ldots, x_m$). The hiding property follows by treating the SNARG proof in the opening as "leakage" on the PRF key and then appealing to leakage-resilient pseudorandomness of the underlying PRF. Critically, this latter step relies on the length of the SNARG being *sublinear* in the length of the PRF key.

Replacing the SNARG with a batch argument. We first observe that the SNARG proof in the opening is *almost* a *batch* language. Namely, the proof is showing that for each index $i \in I$, the bit r_i satisfies $r_i = \mathsf{PRF}(k, x_i)$. Each instance is described by a tuple (i, x_i, r_i) and the witness is the PRF key k. The caveat is that in a batch language, there is no requirement that the prover uses the *same* witness (i.e., the PRF key k) for each instance. Namely, if we use replace the SNARG in the [KMY20] construction with a BARG, then the proof only suffices to argue "local consistency" (i.e., there exists some key k_i that explains each output bit r_i) rather than "global consistency" (i.e., there exists a *single* key k that explains each output bit r_i). Certainly, local consistency is insufficient as it is trivial to find a tuple of keys $(k_1, \ldots, k_m)$ that explains *any* candidate hidden-bits string $\mathbf{r} \in \{0,1\}^m$.

Enforcing consistency. To force the prover to use a consistent PRF key k across all of the instances when constructing the batch argument, we have the prover include a commitment c to the PRF key k as part of the opening. Each instance of the batch NP language is now

$$\exists k : c \text{ is a commitment to } k \text{ and } \mathsf{PRF}(k, x_i) = r_i.$$

In fact, we note that we can replace the PRF with a pseudorandom generator $\mathsf{PRG}\colon \{0,1\}^\lambda \to \{0,1\}^m$, and indeed, the (weak) PRF in the [KMY20] construction is essentially used as a PRG. We will write $\mathsf{PRG}_i\colon \{0,1\}^\lambda \to \{0,1\}$ to denote the function that takes as input the seed $\mathbf{s} \in \{0,1\}^\lambda$ and outputs the i^{th} bit of $\mathsf{PRG}(\mathbf{s})$. To generate the hidden-bits string, the generator now samples $\mathbf{s} \xleftarrow{R} \{0,1\}^\lambda$ and commits to $\mathbf{s}$ with a commitment c. The hidden-bits string is $\mathbf{r} \leftarrow \mathsf{PRG}(\mathbf{s})$ and the opening to $\mathbf{r}_I$ is a batch argument π for the following language:

$$\forall i \in I, \exists \mathbf{s} \in \{0,1\}^\lambda : c \text{ is a commitment to } \mathbf{s} \text{ and } \mathsf{PRG}_i(\mathbf{s}) = r_i.$$

As long as the commitment is *statistically* binding (i.e., the commitment c can be opened to at most one seed $\mathbf{s}$) and the batch argument is computationally sound, the scheme satisfies the binding requirement. In our security analysis (Theorem 3.3), we technically require a stronger extractability property on the commitment, which allows us to base binding on *semi-adaptive* soundness of the underlying BARG; this is the notion achieved by most recent constructions from standard assumptions [CJJ21b, WW22, HJKS22, DGKV22, HJKS22].[3] In contrast, the construction of [KMY20] relied on a SNARG with *adaptive* soundness. This is a stronger requirement that cannot be proven under a black-box reduction to a falsifiable assumption [GW11]. However, this approach for constructing a hidden bits generator does not satisfy hiding. There are two issues:

- **Length of the commitment:** The opening now contains a commitment c to the PRG seed $\mathbf{s}$. Since c is *statistically binding*, the length of c is at least as long as the seed $\mathbf{s}$.
- **Length of the proof:** Succinctness of the batch argument says that the length of the proof π satisfies $|\pi| = \mathsf{poly}(\lambda, |C|, \log |I|)$, where C is the circuit that takes as input $(i, \mathbf{s}, r_i)$ and checks that c is a commitment to $\mathbf{s}$ and $\mathsf{PRG}_i(\mathbf{s}) = r_i$. Unlike the case of a SNARG, the length of π scales *polynomially* with the size of the circuit $|C|$. Since C takes the PRG seed as input, $|C| \geq |\mathbf{s}|$, so the length of π is at least as long as the seed $\mathbf{s}$.

The [KMY20] construction argues hiding by relying on leakage resilience of the underlying weak PRF. In their setting, the only leakage on the PRF key is from the SNARG, whose length is *smaller* than the length of the PRF key. As such, the analysis reduces to a standard leakage-resilience argument. In our setting, both the commitment to the PRG seed and the length of the BARG proof potentially leak too much information about the PRG seed, and we cannot directly leverage leakage resilience to argue hiding.

Leveraging locality. Our first observation is that each individual instance in the batch language is checking a *single* output bit of the PRG. Since the length of the BARG proof scales with the size of the circuit checking a single instance, this means that if the circuit for validating a single output bit of the PRG is much smaller than the length of the overall PRG seed, we can rely on BARG succinctness. One way to construct PRGs with this property is by relying on locality. We say that a PRG is k-local if each output bit only depends on at most k bits of the seed. If a PRG is k-local, then each output bit can be verified with a circuit of size at most $2^k \cdot \mathsf{poly}(\lambda)$. In this case, to check that output bit i is correctly computed, the relation only needs to check (local) openings for the k bits of $\mathbf{s}$ that determine $\mathsf{PRG}_i(\mathbf{s})$. For instance, if the PRG has constant locality [Gol00, CM01], and we take the commitment c to be a bit-by-bit commitment to the bits of $\mathbf{s}$, then verifying a single output bit only requires a circuit of size λ^δ, for some fixed constant $\delta > 0$ that depends on the BARG scheme and the commitment scheme (but *not* the seed length of the PRG). Here λ is the main security parameter (for the BARG and for the commitment scheme). If we set the length of the PRG seed to be at least $n > \lambda^\delta$, then we can hope to rely on leakage resilience of the PRG to argue that the output still has high min-entropy even given the BARG proof. In our constructions (Theorem 1.1 and Corollary 3.15), we can use k-local PRGs with locality as high as $k = c \log \lambda$ for some constant $c < 1$.

We additionally require that our k-local PRGs be leakage resilient. Here, we rely on sub-exponential hardness and the Gentry-Wichs leakage-simulation lemma [GW11]. Roughly speaking, it says that if $\mathsf{PRG}(\mathbf{s})$ is computationally indistinguishable from $\mathbf{t} \xleftarrow{R} \{0,1\}^m$ against (non-uniform) adversaries of size at most s, then there exists an auxiliary distribution over strings $(\mathbf{t}, \mathsf{aux}^*)$ such that $(\mathsf{PRG}(\mathbf{s}), \mathsf{aux})$ is computationally indistinguishable from $(\mathbf{t}, \mathsf{aux}^*)$ against (non-uniform) adversaries of size at most $s/2^{|\mathsf{aux}|}$. Here aux is a string that can be arbitrarily correlated with $\mathbf{s}$. Thus, as long as the leakage aux is sufficiently short (as a function of the seed length) and the PRG satisfies sub-exponential security, we can argue that the outputs are still pseudorandom. Finally, we can apply a standard randomness extractor to $\mathbf{t}$ to obtain a sequence of bits that are statistically close to uniform (even given aux^*).

Dual-mode commitments. The only remaining challenge is to ensure that the (statistically-binding) commitment to the PRG seed $\mathbf{s}$ does not leak information about the seed. While it is tempting to rely on computational hiding of the commitment scheme and replace the commitment to the seed with a commitment to the all-zeroes string, this hybrid strategy does not work. The BARG proof (in the opening) is generated using the *openings* to the commitment

[3] Semi-adaptive soundness for a batch argument says that the adversary must first commit to the index i of the false statement in the soundness game. It can adaptively choose the statements $x_1, \ldots, x_T$ *after* seeing the CRS, with the restriction that instance x_i must be false.

scheme (i.e., the openings to the commitment are part of the witness for the BARG). Alternatively, we can apply the Gentry-Wichs leakage lemma to argue that the joint distribution $(\mathsf{Commit}(\mathbf{s}), \mathsf{PRG}(\mathbf{s}), \mathsf{aux})$ is computationally indistinguishable from $(\mathsf{Commit}(\mathbf{0}), \mathbf{t}, \mathsf{aux}^*)$. As long as the commitment scheme is hiding even for adversaries of size $2^{|\mathsf{aux}|}$, then security follows. However, there is a circular dependency here, as the length of aux is the length of the BARG proof, which is at least as long as the commitment (since the commitment is an input to the BARG relation). As a result, we cannot use complexity leveraging on the commitment as we could with the PRG.

We instead take a different "dual-mode" strategy [GOS06, PW08]. Specifically, we consider a dual-mode commitment scheme where the CRS can be sampled in one of two (computationally indistinguishable) modes: (1) an extractable mode which we use to argue binding; and (2) a statistically hiding mode where the commitments now *statistically* hide the input. Dual-mode commitments can be constructed from a lossy public-key encryption scheme, which is implied by most number-theoretic intractability assumptions [PW08, HLOV11, AFMP20].

The idea in the hiding proof then is to first switch the dual-mode commitment from binding mode into hiding mode. Observe that this step only changes the public parameters in the scheme. Once the CRS is in hiding mode, the commitments to the PRG seed $\mathbf{s}$ *statistically* hide $\mathbf{s}$, regardless of the size of the adversary. In this case, we can appeal to the Gentry-Wichs leakage lemma to argue that the joint distribution $(\mathsf{Commit}(\mathbf{s}), \mathsf{PRG}(\mathbf{s}), \mathsf{aux})$ is computationally indistinguishable from $(\mathsf{Commit}(\mathbf{0}), \mathbf{t}, \mathsf{aux}^*)$ assuming only sub-exponential hardness of the PRG. This means the unopened bits in the hidden-bits string are uniformly random and hiding holds. We provide the full details in Section 3 (Theorem 3.4).

Upgrading BARGs to zkBARGs. For completeness, we conclude with a few remarks on using a NIZK for NP to generically upgrade a batch argument to a zero-knowledge batch argument (zkBARG). First, we note that the naïve approach of giving a NIZK proof of knowledge of a BARG proof does *not* work out of the box. The issue is that the verification algorithm for the BARG needs to read the statements $(x_1, \ldots, x_T)$, and thus, the size of the verification circuit scales linear with T. Since the size of a NIZK proof can scale polynomially with the size of the verification circuit, the size of the NIZK proof of knowledge of a valid BARG proof for $(x_1, \ldots, x_T)$ can scale polynomially with T. Nonetheless, we can still apply this general approach in the following settings:

- **Index BARGs:** An index BARG for an NP language is one where the statements are always fixed to be the integers $1, \ldots, T$ [CJJ21b]. In an index BARG, the verification algorithm only takes the upper bound T as input and is required to run in time that is polylogarithmic in T. We can generically compose an index BARG with a NIZK to obtain a zero-knowledge index BARG. We can then apply the index-BARG-to-BARG transformation from [CJJ21b] to the zero-knowledge index BARG for NP to obtain a zkBARG for NP; note here that the [CJJ21b] transformation preserves zero-knowledge.
- **BARGs with split verification:** A BARG satisfies "split verification" [CJJ21b, WW22, CGJ+22] if the verification algorithm decomposes into a (non-succinct) statement-dependent preprocessing step that outputs a short verification key vk and a (succinct) online verification step that takes the preprocessed key vk and the proof π and decides whether to accept or reject the proof. Importantly, the online verification step can be implemented by a circuit whose size is polylogarithmic in the number of instances T. Given a BARG with a split verification property, it suffices to use a NIZK to prove knowledge of a BARG proof that satisfies the online verification check. This yields a zkBARG with split verification.

2 Preliminaries

We write λ to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. We use boldface letters (e.g., $\mathbf{x}$) to denote vectors. We write $\mathsf{poly}(\lambda)$ to denote a fixed function that is $O(\lambda^c)$ for some $c \in \mathbb{N}$ and $\mathsf{negl}(\lambda)$ to denote a function that is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$. We say an event occurs with overwhelming probability if its complement occurs with negligible probability. We say an algorithm on λ-bit inputs is efficient if it can be computed by a Boolean circuit of size $\mathsf{poly}(\lambda)$, or equivalently, if it can be computed by a Turing machine in $\mathsf{poly}(\lambda)$ time with $\mathsf{poly}(\lambda)$ bits of advice.

Let $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda \in \mathbb{N}}$ be two ensembles of distributions. For functions $s = s(\lambda)$ and $\varepsilon = \varepsilon(\lambda)$, we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are (s, ε)-indistinguishable if for all non-negative polynomials $\mathsf{poly}(\cdot)$ and all adversaries $\mathcal{A}$,

modeled as Boolean circuits of size at most $s(\lambda) \cdot \mathrm{poly}(\lambda)$, and all sufficiently large $\lambda \geq \lambda_{\mathcal{A}}$,

$$\left|\Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_{1,\lambda}] - \Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_{2,\lambda}]\right| \leq \varepsilon(\lambda).$$

We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable if there exists a negligible function $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$ such that $\mathcal{D}_1$ and $\mathcal{D}_2$ are $(1, \varepsilon)$-indistinguishable. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically indistinguishable if the statistical distance $\Delta(\mathcal{D}_1, \mathcal{D}_2)$ is bounded by a negligible function $\mathsf{negl}(\lambda)$.

Min-entropy. We recall some basic definitions on min-entropy. Our definitions are adapted from those in [DRS04]. For a (discrete) random variable X, we write $\mathbf{H}_\infty(X) = -\log(\max_x \Pr[X = x])$ to denote its min-entropy. For two (possibly correlated) discrete random variables X and Y, we define the average min-entropy of X given Y to be $\mathbf{H}_\infty(X \mid Y) = -\log(\mathbb{E}_{y \leftarrow Y} \max_x \Pr[X = x \mid Y = y])$. The optimal probability of an unbounded adversary guessing X given the correlated value Y is $2^{-\mathbf{H}_\infty(X|Y)}$.

Lemma 2.1 (Conditional Min-Entropy [DRS04, Lemma 2.2])**.** *Let A, B be random variables and suppose there are at most 2^λ elements in the support of B. Then $\mathbf{H}_\infty(A \mid B) \geq \mathbf{H}_\infty(A, B) - \lambda \geq \mathbf{H}_\infty(A) - \lambda$.*

Gentry-Wichs leakage lemma. Our analysis will rely on the following "leakage lemma" from [GW11]:

Lemma 2.2 (Indistinguishability with Auxiliary Information [GW11, Lemma 3.1])**.** *Let λ be a security parameter. There exists a polynomial $\mathrm{poly}(\cdot)$ such that the following property holds. Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be arbitrary distributions that are (s, ε)-indistinguishable for some $s = s(\lambda)$ and $\varepsilon = \varepsilon(\lambda)$. Let $\mathcal{X}^*_\lambda = \left\{\mathcal{X}^*_\lambda\right\}_{\lambda \in \mathbb{N}}$ be an augmented distribution where $\mathcal{X}^*_\lambda$ is a distribution on pairs (x_λ, π_λ) where $x_\lambda \leftarrow \mathcal{X}_\lambda$ and $\pi \in \{0,1\}^{\ell(\lambda)}$ can be arbitrarily correlated with x_λ. Then, there exists a distribution $\mathcal{Y}^* = \left\{\mathcal{Y}^*_\lambda\right\}_{\lambda \in \mathbb{N}}$ with the following properties:*

- *Each $\mathcal{Y}^*_\lambda$ is a distribution on tuples $(y_\lambda, \hat{\pi}_\lambda)$, where $y_\lambda \leftarrow \mathcal{Y}_\lambda$ and $\pi_\lambda \in \{0,1\}^{\ell(\lambda)}$.*
- *The distributions $\mathcal{X}^*$ and $\mathcal{Y}^*$ are (s', ε')-indistinguishable, where $s'(\lambda) = s(\lambda) \cdot \mathrm{poly}(\varepsilon(\lambda)/2^{\ell(\lambda)})$ and $\varepsilon'(\lambda) = 2\varepsilon(\lambda)$.*

Leftover hash lemma. Our construction will also rely on the generalized leftover hash lemma (LHL) from [BDK+11]:

Theorem 2.3 (LHL with Conditional Min-Entropy [BDK+11, Theorem 3.2, adapted])**.** *Let (X, Z) be random variables sampled from some joint distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Z}$. Let $\mathcal{H} = \{h \colon \mathcal{X} \to \{0,1\}^v\}$ be a family of universal hash functions, and let $L = \mathbf{H}_\infty(X \mid Z) - v$ be the entropy loss. Let $\mathcal{A}(r, h, z)$ be a (possibly probabilistic) distinguisher where*

$$\Pr[\mathcal{A}(r, h, z) = 1 : r \xleftarrow{R} \{0,1\}^v, h \xleftarrow{R} \mathcal{H}, (x, z) \leftarrow \mathcal{D}] \leq \varepsilon.$$

Then, the distinguishing advantage of $\mathcal{A}$ on the following distributions is at most $\sqrt{\varepsilon/2^L}$:

$$\left\{(h(x), h, z) : \begin{array}{c}(x,z) \leftarrow \mathcal{D} \\ h \xleftarrow{R} \mathcal{H}\end{array}\right\} \text{ and } \left\{(r, h, z) : \begin{array}{c}(x,z) \leftarrow \mathcal{D} \\ r \xleftarrow{R} \{0,1\}^v, h \xleftarrow{R} \mathcal{H}\end{array}\right\}$$

Corollary 2.4 (LHL with Conditional Min-Entropy)**.** *Let (X, Z) be random variables sampled from some joint distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Z}$. Let $\mathcal{H} = \{h \colon \mathcal{X} \to \{0,1\}^v\}$ be a family of universal hash functions. Let $L = \mathbf{H}_\infty(X \mid Z) - v$ be the entropy loss. Then the statistical distance between the following distributions is at most $2^{-L/2}$:*

$$\left\{(h(x), h, z) : \begin{array}{c}(x,z) \leftarrow \mathcal{D} \\ h \xleftarrow{R} \mathcal{H}\end{array}\right\} \text{ and } \left\{(r, h, z) : \begin{array}{c}(x,z) \leftarrow \mathcal{D} \\ r \xleftarrow{R} \{0,1\}^v, h \xleftarrow{R} \mathcal{H}\end{array}\right\}$$

Proof. Follows by setting $\varepsilon = 1$ in Theorem 2.3 (which captures *all* distinguishers). □

Pseudorandom generators. We recall the definition of a pseudorandom generator.

Definition 2.5 (Pseudorandom Generator)**.** Let λ be a security parameter. A pseudorandom generator with output length $m = m(\lambda)$ is an efficiently-computable function family $\mathsf{PRG} = \{\mathsf{PRG}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathsf{PRG}_\lambda \colon \{0,1\}^\lambda \to \{0,1\}^{m(\lambda)}$. For functions $s = s(\lambda)$ and $\varepsilon = \varepsilon(\lambda)$, we say that PRG is (s,ε)-secure if the following two distributions are (s,ε)-indistinguishable:

$$\left\{\mathsf{PRG}_\lambda(x) : x \xleftarrow{R} \{0,1\}^\lambda\right\} \quad \text{and} \quad \left\{y \xleftarrow{R} \{0,1\}^{m(\lambda)}\right\}.$$

We say that PRG is sub-exponentially secure if there exists a constant $\alpha > 0$ and a negligible function $\varepsilon = \mathsf{negl}(\lambda)$ such that PRG is $\left(2^{\lambda^\alpha}, \varepsilon\right)$-secure.

Definition 2.6 (Locality of a PRG)**.** We say a $\mathsf{PRG}\colon \{0,1\}^\lambda \to \{0,1\}^m$ has locality $k = k(\lambda)$ if each output bit of $\mathsf{PRG}(x)$ is a function of at most k bits of the seed x. We say that PRG is computable in NC^0 if PRG has constant locality $k = O(1)$.

Local PRGs constructions. Goldreich [Gol00, MST03] gave the first candidate local PRG construction (with constant locality) based on constraint-satisfiability problems over expander graphs. A long line of subsequent works have studied variants of Goldreich's construction [CM01, MST03, CEMT09, App12, ABR12, OW14, AL16, AK19]; we refer to [App15] for an excellent survey of the state of the art. Notably, PRGs with constant locality and super-linear stretch have featured prominently in constructions of indistinguishability obfuscation [Lin17, LT17, JLS21, JLS22].

There has also been an extensive line of works studying attacks and ruling out certain instantiations of local PRGs [MST03, CEMT09, BQ09, OW14, App15, AL16, CDM+18, Üna23]. For local PRGs with super-linear stretch $\lambda^{1+\delta}$, the most recent attacks [BQ09, Üna23] run in time roughly $\lambda^{O(\lambda^{1-\delta/k})}$ where k is the locality.

Dual-mode commitments. Next, we recall the notion of a "dual-mode" commitment (also called a "mixed commitment") [DN02]. At a high-level, these are non-interactive commitment schemes in the common reference string (CRS) model where the CRS can be sampled from one of two computationally indistinguishable distributions. In one distribution (or mode), the commitment scheme is extractable (i.e., given trapdoor information, one can efficiently extract the committed value from a commitment), and in the other distribution (or mode), the commitment scheme is statistically hiding.[4] We give the formal definition below:

Definition 2.7 (Dual-Mode Bit Commitment [DN02])**.** A dual-mode bit commitment scheme is a tuple of efficient algorithms $\Pi_{\mathsf{BC}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, \mathsf{mode}) \to (\mathsf{crs}, \mathsf{td})$: On input the security parameter λ and $\mathsf{mode} \in \{\mathsf{bind}, \mathsf{hide}\}$, the setup algorithm outputs a common reference string crs and a trapdoor td (possibly empty).
- $\mathsf{Commit}(\mathsf{crs}, b) \to (c, \sigma)$: On input the common reference string crs and a bit $b \in \{0,1\}$, the commit algorithm outputs a commitment c and an opening σ.
- $\mathsf{Verify}(\mathsf{crs}, c, b, \sigma) \to \{0,1\}$: On input the common reference string crs, a commitment c, a bit $b \in \{0,1\}$, and an opening σ, the verification algorithm outputs a bit $b' \in \{0,1\}$.

Moreover, Π_{BC} should satisfy the following properties:

- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all bits $b \in \{0,1\}$, all modes $\mathsf{mode} \in \{\mathsf{bind}, \mathsf{hide}\}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, c, b, \sigma) = 1 : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{mode});\\ (c, \sigma) \leftarrow \mathsf{Commit}(\mathsf{crs}, b)\end{array}\right] = 1.$$

- **Mode indistinguishability:** For all efficient adversaries $\mathcal{A}$, and sampling $(\mathsf{crs}_{\mathsf{bind}}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{bind})$, $(\mathsf{crs}_{\mathsf{hide}}, \mathsf{td}') \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{hide})$, we have that

$$\left|\Pr\left[\mathcal{A}(1^\lambda, \mathsf{crs}_{\mathsf{bind}}) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, \mathsf{crs}_{\mathsf{hide}}) = 1\right]\right| = \mathsf{negl}(\lambda).$$

[4] In some settings, we can require a stronger "equivocation" property in hiding mode where given trapdoor information, one can sample a commitment c and openings for c to *any* value. Our constructions do not require equivocation.

- **Extractable in binding mode:** There exists an efficient algorithm Extract that takes as input a trapdoor td and a string $c \in \{0,1\}^*$, and outputs a bit $b \in \{0,1\}$. Then, for all adversaries $\mathcal{A}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, c, b, \sigma) = 1 \wedge b \neq b' : \begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{bind});\\ (c, \sigma, b) \leftarrow \mathcal{A}(\mathsf{crs});\\ b' \leftarrow \mathsf{Extract}(\mathsf{td}, c)\end{array}\right] = \mathsf{negl}(\lambda).$$

- **Statistically hiding in hiding mode:** For a security parameter λ and a bit $\beta \in \{0,1\}$, we define the hiding game between an adversary $\mathcal{A}$ and a challenger as follows:

  1. The challenger starts by sampling $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{hide})$ and gives crs to $\mathcal{A}$.
  2. Algorithm $\mathcal{A}$ outputs two messages $b_0, b_1 \in \{0,1\}$.
  3. The challenger computes $(c, \sigma) \leftarrow \mathsf{Commit}(\mathsf{crs}, b_\beta)$ and replies to $\mathcal{A}$ with c.
  4. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

  Then Π_{BC} is statistically hiding in hiding mode if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all adversaries $\mathcal{A}$ in the above hiding experiment,

$$|\Pr[b' = 1 \mid \beta = 0] - \Pr[b' = 1 \mid \beta = 1]| = \mathsf{negl}(\lambda).$$

Constructions of dual-mode commitments. Dual-mode commitments (with extraction) can be built from any lossy public-key encryption scheme [BHY09], which can in turn be constructed from most standard algebraic assumptions [PW08, HLOV11, AFMP20]. In particular, a commitment to an input x is just a public-key encryption of x and the opening is the corresponding encryption randomness. In extracting mode, the extraction trapdoor is the decryption key.

2.1 Non-Interactive Zero-Knowledge Arguments for NP

We recall the notion of a non-interactive zero-knowledge argument for NP [GMR85, BFM88]. We specifically consider the NP-complete language of Boolean circuit satisfiability. Namely, for a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, we say that a statement $\mathbf{x} \in \{0,1\}^n$ is a YES instance if there exists a witness $\mathbf{w} \in \{0,1\}^h$ such that $C(\mathbf{x}, \mathbf{w}) = 1$.

Definition 2.8 (NIZK Argument for NP)**.** A non-interactive zero-knowledge argument for Boolean circuit satisfiability is a tuple of efficient algorithms $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: On input the security parameter $\lambda \in \mathbb{N}$, the setup algorithm outputs a common reference string crs.
- $\mathsf{Prove}(\mathsf{crs}, C, \mathbf{x}, \mathbf{w}) \to \pi$: On input the common reference string crs, a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $\mathbf{x} \in \{0,1\}^n$, and a witness $\mathbf{w} \in \{0,1\}^h$, the prove algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}, C, \mathbf{x}, \pi) \to b$: On input the common reference string crs, the Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $\mathbf{x} \in \{0,1\}^n$, and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.

Moreover, Π_{NIZK} should satisfy the following properties:

- **Completeness:** For all $\lambda \in \mathbb{N}$, all Boolean circuits $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, all statements $\mathbf{x} \in \{0,1\}^n$, and all witnesses $\mathbf{w} \in \{0,1\}^h$ where $C(\mathbf{x}, \mathbf{w}) = 1$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, C, \mathbf{x}, \pi) = 1 : \begin{array}{c}\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda);\\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, C, \mathbf{x}, \mathbf{w})\end{array}\right] = 1.$$

- **Computational soundness:** For all efficient adversaries $\mathcal{A}$,

$$\Pr\left[\mathbf{x} \notin \mathcal{L}_C \wedge \mathsf{Verify}(\mathsf{crs}, C, \mathbf{x}, \pi) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (C, \mathbf{x}, \pi) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda),$$

  where for a circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, we define $\mathcal{L}_C$ to be the language of Boolean circuit satisfiability: $\mathcal{L}_C := \{\mathbf{x} \in \{0,1\}^n : \exists \mathbf{w} \in \{0,1\}^h, C(\mathbf{x}, \mathbf{w}) = 1\}$.

- **Computational zero-knowledge:** For every efficient adversary $\mathcal{A}$, there exists an efficient simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\widetilde{\mathsf{crs}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$, we have that

$$\left|\Pr\left[\mathcal{A}^{O_0(\mathsf{crs},\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{crs}) = 1\right] - \Pr\left[\mathcal{A}^{O_1(\mathsf{st}_{\mathcal{S}},\cdot,\cdot,\cdot)}(1^\lambda, \widetilde{\mathsf{crs}}) = 1\right]\right| = \mathsf{negl}(\lambda),$$

  where the oracles O_0 and O_1 are defined as follows:

  - $O_0(\mathsf{crs}, C, \mathbf{x}, \mathbf{w})$: On input crs, a circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $\mathbf{x} \in \{0,1\}^n$, and a witness $\mathbf{w} \in \{0,1\}^h$, the oracle outputs $\perp$ if $C(\mathbf{x}, \mathbf{w}) = 0$. If $C(\mathbf{x}, \mathbf{w}) = 1$, it outputs $\mathsf{Prove}(\mathsf{crs}, C, \mathbf{x}, \mathbf{w})$.
  - $O_1(\mathsf{st}_{\mathcal{S}}, C, \mathbf{x}, \mathbf{w})$: On input the simulator state $\mathsf{st}_{\mathcal{S}}$, a circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $\mathbf{x} \in \{0,1\}^n$, and a witness $\mathbf{w} \in \{0,1\}^h$, the oracle outputs $\perp$ if $C(\mathbf{x}, \mathbf{w}) = 0$. If $C(\mathbf{x}, \mathbf{w}) = 1$, it outputs $\mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, C, \mathbf{x})$.

2.2 Non-Interactive Batch Arguments for NP

The main cryptographic primitive we consider in this work is a non-interactive batch argument for NP. As before, we consider the NP-complete language of Boolean circuit satisfiability. We now recall the definition of a non-interactive batch argument for NP from [KPY19, CJJ21a]. Our construction relies on the notion of semi-adaptive soundness used in [CJJ21b, WW22, DGKV22, KLVW23, CGJ+22].

Definition 2.9 (Batch Argument for NP [CJJ21b, adapted])**.** A non-interactive batch argument (BARG) for Boolean circuit satisfiability is a tuple of three efficient algorithms $\Pi_{\mathsf{BARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^T, 1^s) \to \mathsf{crs}$: On input the security parameter $\lambda \in \mathbb{N}$, the number of instances $T \in \mathbb{N}$, and a bound on the circuit size $s \in \mathbb{N}$, the setup algorithm outputs a common reference string crs.
- $\mathsf{Prove}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_T), (\mathbf{w}_1, \ldots, \mathbf{w}_T)) \to \pi$: On input the common reference string crs, a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, statements $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \{0,1\}^n$, and witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_T \in \{0,1\}^h$, the prove algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_T), \pi) \to b$: On input the common reference string crs, the Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, statements $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \{0,1\}^n$ and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.

Moreover, Π_{BARG} should satisfy the following properties:

- **Completeness:** For all $\lambda, T, s \in \mathbb{N}$, all circuits $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size at most s, all statements $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \{0,1\}^n$, and all witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_T \in \{0,1\}^h$ where $C(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all $i \in [T]$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, C, \mathbf{x}, \pi) = 1 : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^T, 1^s); \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, C, \mathbf{x}, \mathbf{w}) \end{array}\right] = 1,$$

  where $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ and $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_T)$.

- **Succinct proof size:**[5] There exists a polynomial $\mathsf{poly}(\cdot,\cdot,\cdot)$ such that for all $\lambda, T, s \in \mathbb{N}$, all crs in the support of $\mathsf{Setup}(1^\lambda, 1^T, 1^s)$, and all Boolean circuits $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size at most s, the size of the proof π output by $\mathsf{Prove}(\mathsf{crs}, C, \cdot, \cdot)$ satisfies $|\pi| \le \mathsf{poly}(\lambda, \log T, s)$.

[5] Previous works [KPY19, CJJ21a, CJJ21b, WW22, DGKV22, CGJ+22] also impose requirements on the size of the CRS and the running time of the verifier. These additional properties are not needed in our work.

- **Semi-adaptive soundness:** For a security parameter λ, we define the semi-adaptive soundness game between an adversary $\mathcal{A}$ and a challenger as follows:

  1. Algorithm $\mathcal{A}$ starts by outputting the number of instances 1^T, the bound on the circuit size 1^s, and an index $i \in [T]$.
  2. The challenger samples a common reference string $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^T, 1^s)$ and gives crs to $\mathcal{A}$.
  3. Algorithm $\mathcal{A}$ outputs a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of size at most s, statements $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ where each $\mathbf{x}_i \in \{0,1\}^n$, and a proof π.
  4. The output of the experiment is $b = 1$ if $\mathsf{Verify}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_T), \pi) = 1$ and for all $\mathbf{w}_i \in \{0,1\}^h$, $C(\mathbf{x}_i, \mathbf{w}_i) = 0$. Otherwise, the output is $b = 0$.

  Then Π_{BARG} satisfies semi-adaptive soundness if for all efficient adversaries $\mathcal{A}$, $\Pr[b = 1] = \mathsf{negl}(\lambda)$ in the semi-adaptive soundness game.

Constructions of batch arguments for NP. Batch arguments for NP have recently been realized from a broad range of standard assumptions including lattice-based assumptions [CJJ21b, DGKV22] as well as assumptions over pairing groups [KVZ21, WW22] and pairing-free groups [CGJ+22].

2.3 Hidden-Bits Generator

We recall the notion of a hidden-bits generator with subset-dependent proofs from [KMY20]. For a bitstring $\mathbf{r} \in \{0,1\}^n$ and a set of indices $I \subseteq [n]$, we write $\mathbf{r}_I \in \{0,1\}^{|I|}$ to denote the substring corresponding to the bits of $\mathbf{r}$ indexed by I.

Definition 2.10 (Hidden-Bits Generator [KMY20, Definition 11])**.** A hidden-bits generator with subset-dependent proofs is a tuple of efficient algorithms $\Pi_{\mathsf{HBG}} = (\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^m) \to \mathsf{crs}$: On input the security parameter λ, and the output length m, the setup algorithm outputs a common reference string crs.
- $\mathsf{GenBits}(\mathsf{crs}) \to (\mathbf{r}, \mathsf{st})$: On input the the common reference string crs, the generator algorithm outputs a string $\mathbf{r} \in \{0,1\}^m$ and a state st.
- $\mathsf{Prove}(\mathsf{st}, I) \to \pi$: On input the state st and a subset $I \subseteq [m]$, the prove algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}, I, \mathbf{r}_I, \pi) \to b$: On input a common reference string crs, a subset $I \subseteq [m]$, a string $\mathbf{r}_I \in \{0,1\}^{|I|}$, and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.

We require Π_{HBG} to satisfy the following properties:

- **Correctness:** For all $m, \lambda \in \mathbb{N}$ and all subsets $I \subseteq [m]$, we have

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, I, \mathbf{r}_I, \pi) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^m); \\ (\mathbf{r}, \mathsf{st}) \leftarrow \mathsf{GenBits}(\mathsf{crs}); \\ \pi \leftarrow \mathsf{Prove}(\mathsf{st}, I) \end{array}\right] = 1.$$

- **Somewhat computational binding:** For every crs in the support of the algorithm $\mathsf{Setup}(1^\lambda, 1^m)$, there exists a set $\mathcal{V}^{\mathsf{crs}}$ with the following properties:

  (i) **Output sparsity.** There exists a universal constant $\gamma < 1$ and a fixed polynomial $p(\cdot)$ such that for every polynomial $m = m(\lambda)$, and every crs in the support of $\mathsf{Setup}(1^\lambda, 1^m)$, $|\mathcal{V}^{\mathsf{crs}}| \le 2^{m^\gamma \cdot p(\lambda)}$

  (ii) **Computational binding.** For every efficient and stateful adversary $\mathcal{A}$,

$$\Pr\left[\mathbf{r}_I \notin \mathcal{V}_I^{\mathsf{crs}} \wedge \mathsf{Verify}(\mathsf{crs}, I, \mathbf{r}_I, \pi) = 1 : \begin{array}{c} 1^m \leftarrow \mathcal{A}(1^\lambda); \\ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^m); \\ (I, \mathbf{r}_I, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda),$$

  where $\mathcal{V}_I^{\mathsf{crs}} := \{\mathbf{r}_I : \mathbf{r} \in \mathcal{V}^{\mathsf{crs}}\}$.

- **Computationally hiding:** For every polynomial $m = m(\lambda)$, every subset $I \subseteq [m]$, and all efficient adversaries $\mathcal{A}$, we have
$$\left|\Pr[\mathcal{A}(\mathsf{crs}, I, \mathbf{r}_I, \pi, \mathbf{r}_{\bar{I}}) = 1] - \Pr[\mathcal{A}(\mathsf{crs}, I, \mathbf{r}_I, \pi, \mathbf{r}'_{\bar{I}}) = 1]\right| = \mathsf{negl}(\lambda),$$
where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^m)$, $(\mathbf{r}, \mathsf{st}) \leftarrow \mathsf{GenBits}(\mathsf{crs})$, $\pi \leftarrow \mathsf{Prove}(\mathsf{st}, I)$, $\mathbf{r}' \xleftarrow{R} \{0,1\}^m$, and $\bar{I} = [m] \setminus I$.

Theorem 2.11 (NIZK from Hidden-Bits Generator [KMY20, Theorem 5])**.** *If there exists a hidden-bits generator with subset-dependent proofs, then there exists a computational NIZK argument for* NP.

3 Hidden-Bits Generator from Batch Arguments

In this section, we show how to construct a hidden-bits generator with subset-dependent proofs using a batch argument for NP together with a dual-mode commitment and a low-complexity PRG. Then combined with Theorem 2.11, we obtain a NIZK for NP from the same underlying set of assumptions.

Construction 3.1 (Hidden-Bits Generator from Batch Arguments)**.** Let $\lambda \in \mathbb{N}$ be a security parameter and m be an output length parameter. Let $n = n(\lambda, m)$ be a PRG seed length parameter and let $B = B(\lambda, m)$ be a block length parameter. These parameters will be determined in the security analysis. Our construction relies on the following primitives:

- Let $G_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ be a family of PRGs. Let $k = k(\lambda)$ be the locality of the PRG (i.e., each output bit of G_λ depends on at most k input bits). In the following description, we require that $\ell(n) \geq mB$.
- Let $\Pi_{\mathsf{BC}} = (\mathsf{BC.Setup}, \mathsf{BC.Commit}, \mathsf{BC.Verify})$ be a dual-mode bit commitment scheme.
- Let $\Pi_{\mathsf{BARG}} = (\mathsf{BARG.Setup}, \mathsf{BARG.Prove}, \mathsf{BARG.Verify})$ be a batch argument for NP with proof length $\ell_{\mathsf{BARG}} = \ell_{\mathsf{BARG}}(\lambda, T, s)$, where s denotes the size of the underlying NP relation and T denotes the number of instances.
- For an index $i \in [\ell]$ where $\ell = \ell(n)$, let $i_1, \ldots, i_k \in [n]$ be the indices of the k seed bits on which the i^{th} output bit of $G_n(\cdot)$ depends. Let $\mathbf{s} \in \{0,1\}^n$ be a seed for the PRG, and let $G_n^{(i)} : \{0,1\}^k \to \{0,1\}$ be the circuit that takes as input the seed bits $s_{i_1}, \ldots, s_{i_k} \in \{0,1\}$ and outputs the i^{th} bit of $G_n(\mathbf{s})$. Then, for a common reference string $\mathsf{crs}_{\mathsf{BC}}$ for the bit commitment scheme, define the NP relation $\mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$ as follows:

> **Hard-wired:** PRG seed length n, common reference string $\mathsf{crs}_{\mathsf{BC}}$
> **Statement:** circuit $G_n^{(i)} : \{0,1\}^k \to \{0,1\}$, commitments $c_1, \ldots, c_k$, output $t \in \{0,1\}$
> **Witness:** bits $s_1, \ldots, s_k \in \{0,1\}$, openings $\sigma_1, \ldots, \sigma_k$
>
> Output 1 if all of the following conditions hold:
>
> - For each $i \in [k]$, $\mathsf{BC.Verify}(\mathsf{crs}_{\mathsf{BC}}, c_i, s_i, \sigma_i) = 1$;
> - $t = G_n^{(i)}(s_1, \ldots, s_k)$.
>
> Otherwise, output 0.

Figure 1: Relation $\mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]\big((G_n^{(i)}, c_1, \ldots, c_k, t), (s_1, \ldots, s_k, \sigma_1, \ldots, \sigma_k)\big)$.

We now construct our hidden-bits generator $\Pi_{\mathsf{HBG}} = (\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Prove}, \mathsf{Verify})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^m)$: On input the security parameter λ and the output length m, the setup algorithm proceeds as follows:
  1. Sample a CRS for the dual-mode commitment scheme: $(\mathsf{crs}_{\mathsf{BC}}, \mathsf{td}) \leftarrow \mathsf{BC.Setup}(1^\lambda, \mathsf{bind})$.
  2. Let $n = n(\lambda, m)$ be the PRG seed length. Let C be the circuit that computes the NP relation $\mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$. Sample a common reference string $\mathsf{crs}_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Setup}(1^\lambda, 1^{mB}, 1^{|C|})$.

3. Let $B = B(\lambda, m)$ be the block size and sample $\mathbf{v}_1, \ldots, \mathbf{v}_m \xleftarrow{R} \{0,1\}^B$.

Output the common reference string $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$.

- $\mathsf{GenBits}(\mathsf{crs})$: On input the common reference string $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$, the generator algorithm proceeds as follows:

  1. Sample a PRG seed $\mathbf{s} \xleftarrow{R} \{0,1\}^n$, and compute $\mathbf{t} = \{0,1\}^{mB} \leftarrow G_n(\mathbf{s})$.[6] Then, for each $i \in [n]$, compute a commitment $(c_i, \sigma_i) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, s_i)$ to the bits of the seed.
  2. Split $\mathbf{t} = \mathbf{t}_1 \| \mathbf{t}_2 \| \cdots \| \mathbf{t}_m$ into blocks where each $\mathbf{t}_i \in \{0,1\}^B$ for each $i \in [m]$. Next, for each $i \in [m]$, compute $r_i \leftarrow \mathbf{v}_i^{\mathsf{T}} \mathbf{t}_i$ (where the vectors $\mathbf{v}_i$ and $\mathbf{t}_i$ are interpreted as vectors in $\mathbb{Z}_2^B$).

  The algorithm outputs the hidden-bits string $\mathbf{r} = r_1 \| r_2 \| \cdots \| r_m \in \{0,1\}^m$ together with the generator state $\mathsf{st} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{s}, c_1, \ldots, c_n, \sigma_1, \ldots, \sigma_n)$.

- $\mathsf{Prove}(\mathsf{st}, I)$: On input the state $\mathsf{st} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{s}, c_1, \ldots, c_n, \sigma_1, \ldots, \sigma_n)$ and a set of indices $I \subseteq [m]$, the prove algorithm proceeds as follows:

  1. Let $\mathbf{t} = G_n(\mathbf{s})$ and parse $\mathbf{t} = \mathbf{t}_1 \| \mathbf{t}_2 \| \cdots \| \mathbf{t}_m$ where each $\mathbf{t}_i \in \{0,1\}^B$. We will also use the notation $\mathbf{t}[i] := \mathbf{t}_i$ to refer to the i^{th} block of $\mathbf{t}$ and $G_n[i] := G_n^{(i)}$ to refer to the circuit computing the i^{th} bit of $G_n(\mathbf{s})$. In the analysis, we will often associate an index $i \in [mB]$ with a pair $(j, \beta) \in [m] \times [B]$ and vice versa (where $i = (j-1)B + \beta$).
  2. Let $I = \{i^{(1)}, \ldots, i^{(L)}\}$, where the indices $i^{(1)}, \ldots, i^{(L)} \in [m]$ are in sorted order. For each $j \in [L], \beta \in [B]$, let $\mathbf{t}[i^{(j)}, \beta] \in \{0,1\}$ denote the β^{th} bit of $\mathbf{t}[i^{(j)}]$.
  3. By construction, the value of $\mathbf{t}[i^{(j)}, \beta]$ depends on at most k bits of $\mathbf{s}$. We define $G_n[i^{(j)}, \beta]$ to denote the circuit that reads up to k bits of $\mathbf{s}$ and outputs $\mathbf{t}[i^{(j)}, \beta]$. Next, we define the function $\mathsf{idx}\colon [L] \times [B] \times [k] \to [n]$ where $\mathsf{idx}(i^{(j)}, \beta, \gamma)$ outputs the γ^{th} input bit of $\mathbf{s}$ on which the output bit $\mathbf{t}[i^{(j)}, \beta]$ depends. In particular, the inputs to the circuit $G_n[i^{(j)}, \beta]$ consist of bits $\mathsf{idx}(i^{(j)}, \beta, 1), \ldots, \mathsf{idx}(i^{(j)}, \beta, k)$ of $\mathbf{s}$.
  4. For each $j \in [L]$ and $\beta \in [B]$, define the statement $x_{j,\beta}$ and associated witness $w_{j,\beta}$ as follows:

$$x_{j,\beta} = \left(G_n[i^{(j)}, \beta], c_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, c_{\mathsf{idx}(i^{(j)},\beta,k)}, \mathbf{t}[i^{(j)}, \beta]\right) \tag{3.1}$$

$$w_{j,\beta} = \left(s_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, s_{\mathsf{idx}(i^{(j)},\beta,k)}, \sigma_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, \sigma_{\mathsf{idx}(i^{(j)},\beta,k)}\right). \tag{3.2}$$

     Let C be the circuit that computes the NP relation in Fig. 1. Then, compute the proof

$$\pi_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, C, (x_{1,1}, \ldots, x_{1,B}, \ldots, x_{L,1}, \ldots, x_{L,B}), (w_{1,1}, \ldots, w_{1,B}, \ldots, w_{L,1}, \ldots, w_{L,B})).$$

  5. Output $\pi = \left(\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}})\right)$.

- $\mathsf{Verify}(\mathsf{crs}, I, \mathbf{r}_I, \pi)$: On input $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$, a set of indices $I = \{i^{(1)}, \ldots, i^{(L)}\} \subseteq [m]$ (in sorted order), a string $\mathbf{r}_I \in \{0,1\}^L$, and a proof $\pi = \left(\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}})\right)$, the verification algorithm proceeds as follows:

  1. For each $j \in [L]$, let $r_{i^{(j)}} \in \{0,1\}$ be the bit of $\mathbf{r}_I$ associated with index $i^{(j)}$. Then, for each $j \in [L]$, check that $r_{i^{(j)}} = \mathbf{v}_{i^{(j)}}^{\mathsf{T}} \mathbf{t}_{i^{(j)}}$. Output 0 if any check fails.
  2. Using the commitments $c_1, \ldots, c_n$ and the bits of $\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}}$, construct the statements $x_{j,\beta}$ for each $j \in [L]$ and $\beta \in [B]$ according to Eq. (3.1). Let C be the circuit that computes the NP relation $\mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$ in Fig. 1.
  3. Output $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C, (x_{1,1}, \ldots, x_{1,B}, \ldots, x_{L,1}, \ldots, x_{L,B}), \pi_{\mathsf{BARG}})$.

[6] As noted above, we require that $\ell(n) \geq mB$. If $\ell(n) > mB$, we truncate the output of G_n to output a string of length exactly mB.

Theorem 3.2 (Correctness). *If Π_{BARG} is complete and Π_{BC} is correct, then Construction 3.1 is correct.*

Proof. Take any security parameter λ, output length m, and set of indices $I \subseteq [m]$. Let $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^m)$ where $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$. Let $(\mathbf{r}, \mathsf{st}) \leftarrow \mathsf{GenBits}(\mathsf{crs})$ and $\pi \leftarrow \mathsf{Prove}(\mathsf{st}, I)$. Consider the output of $\mathsf{Verify}(\mathsf{crs}, I, \mathbf{r}_I, \pi)$.

- By construction of GenBits, $r_i = \mathbf{v}_i^\intercal \mathbf{t}_i$ for all $i \in [L]$. Thus, the first set of checks in Verify pass.
- Next, for each $j \in [L]$ and $\beta \in [B]$, let $x_{j,\beta}$ and $w_{j,\beta}$ be the statement and witness defined as in Eqs. (3.1) and (3.2). By correctness of Π_{BC}, it follows that $(x_{j,\beta}, w_{j,\beta}) \in \mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$.
- Let $\pi = \left(\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}})\right)$. Since $(x_{j,\beta}, w_{j,\beta}) \in \mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$ for all $j \in [L]$ and $\beta \in [B]$, completeness of Π_{BARG} implies that

$$\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C, (x_{1,1}, \ldots, x_{1,B}, \ldots, x_{L,1}, \ldots, x_{L,B}), \pi_{\mathsf{BARG}}) = 1.$$

  Correspondingly, $\mathsf{Verify}(\mathsf{crs}, I, \mathbf{r}_I, \pi) = 1$, as required. □

Theorem 3.3 (Somewhat Computational Binding). *Let λ be a security parameter. Suppose there exists a universal constant $\delta < 1$ and a fixed polynomial $p(\cdot)$ such that for every polynomial $m = m(\lambda)$, it follows that $n = n(\lambda, m) \leq m^\delta \cdot p(\lambda)$. Suppose also that Π_{BARG} satisfies semi-adaptive soundness, Π_{BC} is extractable in binding mode, and that $B = B(\lambda, m)$ is polynomially bounded. Then, Construction 3.1 satisfies somewhat computational binding.*

Proof. Let $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$ be a common reference string in the support of $\mathsf{Setup}(1^\lambda, 1^m)$. We define the set $\mathcal{V}^{\mathsf{crs}} \subset \{0,1\}^m$ as follows:

$$\mathcal{V}^{\mathsf{crs}} := \left\{(\mathbf{v}_1^\intercal \mathbf{t}_1, \ldots, \mathbf{v}_m^\intercal \mathbf{t}_m) \mid \exists \mathbf{s} \in \{0,1\}^n : \mathbf{t}_1 \| \cdots \| \mathbf{t}_m = G_n(\mathbf{s})\right\}.$$

We now show that each of the requirements in Definition 2.10 is satisfied:

Output sparsity. This is immediate from the construction: $|\mathcal{V}^{\mathsf{crs}}| \leq 2^n \leq 2^{m^\delta \cdot p(\lambda)}$.

Computational binding. To argue computational binding, we appeal to the fact that Π_{BC} is extractable in binding mode and to semi-adaptive soundness of Π_{BARG}. Formally, suppose there is an efficient adversary $\mathcal{A}$ that breaks computational binding of Construction 3.1 with non-negligible advantage ε. We construct an adversary $\mathcal{B}$ that breaks semi-adaptive soundness of the BARG as follows:

1. Algorithm $\mathcal{B}$ starts running $\mathcal{A}$ on input the security parameter 1^λ. Algorithm $\mathcal{A}$ chooses the output length 1^m.
2. Algorithm $\mathcal{B}$ then samples $(\mathsf{crs}_{\mathsf{BC}}, \mathsf{td}) \leftarrow \mathsf{BC.Setup}(1^\lambda, \mathsf{bind})$ as well as an index $i^* \xleftarrow{\mathrm{R}} [mB]$. It outputs 1^{mB} as the number of instances, 1^s as the size of the circuit (for computing the relation $\mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$ in Fig. 1), and the chosen index i^*.
3. Algorithm $\mathcal{B}$ receives $\mathsf{crs}_{\mathsf{BARG}}$ from its challenger. Then, it samples the strings $\mathbf{v}_1, \ldots, \mathbf{v}_m \xleftarrow{\mathrm{R}} \{0,1\}^B$. It gives $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$ to $\mathcal{A}$.
4. Algorithm $\mathcal{A}$ outputs an opening $(I, \mathbf{r}_I, \pi)$.
5. Algorithm $\mathcal{B}$ parses $I = \left\{i^{(1)}, \ldots, i^{(L)}\right\}$ and $\pi = (\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}}))$. It constructs the statement $\mathbf{x} = (x_{1,1}, \ldots, x_{1,B}, \ldots, x_{L,1}, \ldots, x_{L,B})$ from $c_1, \ldots, c_m$ and $\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}}$ according to Eq. (3.1) and defines C to be the circuit that computes the NP relation $\mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$ from Fig. 1.
6. Now, for each $i \in [n]$, algorithm $\mathcal{B}$ runs $s_i \leftarrow \mathsf{BC.Extract}(\mathsf{td}, c_i)$. Let $\mathbf{s} = s_1 \| \cdots \| s_n \in \{0,1\}^n$ be the extracted seed. Algorithm $\mathcal{B}$ now outputs $(C, \mathbf{x}, \pi_{\mathsf{BARG}})$ if the index i^* satisfies $i^* \in I$ and $t_{i^*} \neq t'_{i^*}$, where $\mathbf{t}' = G_n(\mathbf{s})$. Otherwise algorithm $\mathcal{B}$ outputs $\perp$.

First, we argue that algorithm $\mathcal{B}$ is admissible.

- Suppose that $t_{i^*} \neq t'_{i^*}$ where $\mathbf{t}' = G_n(\mathbf{s})$. Write $i^* = (i^{(j)}, \beta) \in [m] \times [B]$. Then,

$$\mathbf{t}[i^{(j)}, \beta] = t_{i^*} \neq G_n^{(i^*)}(\mathbf{s}) = G_n[i^{(j)}, \beta]\left(s_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, s_{\mathsf{idx}(i^{(j)},\beta,k)}\right).$$

- Consider the instance

$$x_{i^{(j)},\beta} = \left(G_n[i^{(j)}, \beta], c_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, c_{\mathsf{idx}(i^{(j)},\beta,k)}, \mathbf{t}[i^{(j)}, \beta]\right),$$

  and any candidate witness

$$w_{i^{(j)},\beta} = \left(s'_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, s'_{\mathsf{idx}(i^{(j)},\beta,k)}, \sigma_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, \sigma_{\mathsf{idx}(i^{(j)},\beta,k)}\right).$$

  We consider two possibilities:

  - Suppose there exists $\gamma \in [k]$ where $s'_{\mathsf{idx}(i^{(j)},\beta,\gamma)} \neq s_{\mathsf{idx}(i^{(j)},\beta,\gamma)}$. By extractability of Π_{BC}, with overwhelming probability over the choice of $\mathsf{crs}_{\mathsf{BC}}$,

$$\mathsf{BC.Verify}\left(\mathsf{crs}_{\mathsf{BC}}, c_{\mathsf{idx}(i^{(j)},\beta,\gamma)}, s'_{\mathsf{idx}(i^{(j)},\beta,\gamma)}, \sigma_{\mathsf{idx}(i^{(j)},\beta,\gamma)}\right) = 0.$$

    Correspondingly, $\mathcal{R}[\mathsf{crs}_{\mathsf{BC}}](x_{i^{(j)},\beta}, w_{i^{(j)},\beta}) = 0$.
  - Suppose that for all $\gamma \in [k]$, $s'_{\mathsf{idx}(i^{(j)},\beta,\gamma)} = s_{\mathsf{idx}(i^{(j)},\beta,\gamma)}$. In this case,

$$G_n[i^{(j)}, \beta]\left(s'_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, s'_{\mathsf{idx}(i^{(j)},\beta,k)}\right) = G_n[i^{(j)}, \beta]\left(s_{\mathsf{idx}(i^{(j)},\beta,1)}, \ldots, s_{\mathsf{idx}(i^{(j)},\beta,k)}\right) \neq \mathbf{t}[i^{(j)}, \beta].$$

    Once again, $\mathcal{R}[\mathsf{crs}_{\mathsf{BC}}](x_{i^{(j)},\beta}, w_{i^{(j)},\beta}) = 0$.

Thus, we conclude that if $t_{i^*} \neq t'_{i^*}$, then instance $x_{i^{(j)},\beta} = x_{i^*}$ is false with all but negligible probability over the choice of $\mathsf{crs}_{\mathsf{BC}}$. Algorithm $\mathcal{B}$ only produces an output when $t_{i^*} \neq t'_{i^*}$ (i.e., when x_{i^*} is false), so algorithm $\mathcal{B}$ is admissible for the semi-adaptive soundness game. To conclude the proof, we compute the advantage of $\mathcal{B}$. In the semi-adaptive soundness game, the challenger constructs $\mathsf{crs}_{\mathsf{BARG}}$ using $\mathsf{BARG.Setup}(1^\lambda, 1^{mB}, 1^{|C|})$, which is identical to the distribution in computational binding game. Thus, algorithm $\mathcal{B}$ perfectly simulates an execution of the binding game for $\mathcal{A}$. This means that with probability ε, algorithm $\mathcal{A}$ outputs (I, r_I, π) where $\pi = (\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}}))$ with the following two properties:

- Let $\mathbf{x} = (x_{1,1}, \ldots, x_{1,B}, \ldots, x_{L,1}, \ldots, x_{L,B})$ be the statement constructed from $c_1, \ldots, c_n$ and $\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}}$ according to Eq. (3.1). Then, we have $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C, \mathbf{x}, \pi_{\mathsf{BARG}}) = 1$, where C is the circuit computing the NP relation $\mathcal{R}[\mathsf{crs}_{\mathsf{BC}}]$ from Fig. 1.
- The bits $\mathbf{r}_I$ satisfy $\mathbf{r}_I \notin \mathcal{V}_I^{\mathsf{crs}}$. This means that for every seed $\mathbf{s} \in \{0,1\}^n$, there must exist some output index $i \in [mB]$ such that $t_i \neq G_n^{(i)}(\mathbf{s})$.

Thus, with probability ε, both of the above conditions hold. In particular, this means that $\mathcal{A}$ outputs $(I, \mathbf{r}_I, \pi)$ such that there exists some index $\hat{i} \in [mB]$ where $t_{\hat{i}} \neq t'_{\hat{i}} = G_n^{(\hat{i})}(\mathbf{s})$. Now, algorithm $\mathcal{B}$ samples $i^* \xleftarrow{\mathrm{R}} [mB]$ and moreover i^* is independent of $\mathcal{A}$'s view. Thus, $\hat{i} = i^*$ with probability at least $1/mB$, in which case, algorithm $\mathcal{B}$ outputs the instance $(C, \mathbf{x})$ with the proof π_{BARG}. Again from the above conditions, $\mathsf{BARG.Verify}(\mathsf{crs}_{\mathsf{BARG}}, C, \mathbf{x}, \pi_{\mathsf{BARG}}) = 1$, and algorithm $\mathcal{B}$ succeeds in breaking semi-adaptive soundness of Π_{BARG}. We conclude that algorithm $\mathcal{B}$ breaks semi-adaptive soundness of Π_{BARG} with advantage $\varepsilon/mB - \mathsf{negl}(\lambda)$, and the claim follows. □

Theorem 3.4 (Computational Hiding)**.** *Suppose the following conditions hold:*

- *The PRG G_λ is sub-exponentially secure (i.e., there exists a constant $\alpha > 0$ and a negligible function $\varepsilon_{\mathsf{PRG}} = \mathsf{negl}(\lambda)$ such that G_λ is $(2^{\lambda^\alpha}, \varepsilon_{\mathsf{PRG}})$-secure).*
- *The bit commitment scheme Π_{BC} satisfies mode indistinguishability and is statistically hiding in hiding mode.*

- *The length of the BARG proof* $\ell_{\mathsf{BARG}} = \mathsf{poly}(\lambda, m)$ *is polynomially-bounded.*
- *The length of the PRG seed satisfies* $n = n(\lambda, m) \geq \max(\lambda, \ell_{\mathsf{BARG}}^c)$ *for some constant* $c > 1/\alpha$*, and the block size satisfies* $B = B(\lambda, m) \geq \omega(\log \lambda) + \ell_{\mathsf{BARG}}$.

Then, for all polynomially-bounded $m = m(\lambda)$*, Construction 3.1 is computationally hiding.*

Proof. Let $I \subseteq [m]$ be an arbitrary subset. We start by defining two distributions $\mathcal{D}_{\mathsf{real}}$ and $\mathcal{D}_{\mathsf{ideal}}$ that will be helpful for our analysis:

- $\mathcal{D}_{\mathsf{real}}(1^\lambda)$: On input the security parameter $\lambda \in \mathbb{N}$, the real distribution constructs the output as follows:
  - Sample $(\mathsf{crs}_{\mathsf{BC}}, \mathsf{td}) \leftarrow \mathsf{BC.Setup}(1^\lambda, \mathsf{hide})$.
  - Sample $\mathsf{crs}_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Setup}(1^\lambda, 1^{mB}, 1^{|C|})$.
  - Let $n = n(\lambda, m)$ and sample $\mathbf{s} \xleftarrow{R} \{0,1\}^n$ and compute $(c_i, \sigma_i) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, s_i)$ for each $i \in [n]$.
  - Compute $\mathbf{t} \leftarrow G_n(\mathbf{s})$ and output $(I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t})$.
- $\mathcal{D}_{\mathsf{ideal}}(1^\lambda)$: Same as $\mathcal{D}_{\mathsf{real}}(1^\lambda)$ except we replace each c_i with a commitment to 0 and $\mathbf{t}$ with a uniformly random string: $(c_i, \sigma_i) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, 0)$ for each $i \in [n]$ and $\mathbf{t} \xleftarrow{R} \{0,1\}^{mB}$.

We now show that if G_n is sub-exponentially secure (and the commitment scheme is statistically hiding), then $\mathcal{D}_{\mathsf{real}}$ and $\mathcal{D}_{\mathsf{ideal}}$ are also indistinguishable to a sub-exponential time algorithm.

Lemma 3.5. *Suppose* G_λ *is* $\left(2^{\lambda^\alpha}, \varepsilon_{\mathsf{PRG}}\right)$*-secure for some constant* $\alpha > 0$ *and negligible function* $\varepsilon_{\mathsf{PRG}} = \mathsf{negl}(\lambda)$ *and that* Π_{BC} *is statistically hiding in hiding mode. Suppose also that* $\ell_{\mathsf{BARG}} = \mathsf{poly}(\lambda, m)$*, and* $B \geq \omega(\log \lambda) + \ell_{\mathsf{BARG}}$*,* $n \geq \max(\lambda, \ell_{\mathsf{BARG}}^c)$ *for some constant* $c > 1/\alpha$*. Then, there exists a negligible function* $\varepsilon_{\mathsf{ideal}} = \mathsf{negl}(\lambda)$ *such that for all subsets* $I \subseteq [m]$*,* $\mathcal{D}_{\mathsf{real}}$ *and* $\mathcal{D}_{\mathsf{ideal}}$ *are* $\left(2^{n^\alpha}, \varepsilon_{\mathsf{ideal}}\right)$*-indistinguishable*

Proof. We start by defining a sequence of hybrid experiments:

- Hyb_0: This is the real distribution $\mathcal{D}_{\mathsf{real}}$.
- Hyb_i: Same as Hyb_0 except for all $j \leq i$, we now sample commitments $(c_j, \sigma_j) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, 0)$. The commitments for $j > i$ are sampled as in Hyb_0.
- Hyb_{n+1}: Same as Hyb_n except $\mathbf{t} \xleftarrow{R} \{0,1\}^{mB}$. This is the ideal distribution $\mathcal{D}_{\mathsf{ideal}}$.

We now show that each adjacent pair of experiments are indistinguishable.

Claim 3.6. *Suppose* Π_{BC} *is statistically hiding in hiding mode. Then, there exists a negligible function* $\varepsilon_0 = \mathsf{negl}(\lambda)$ *such that for all (possibly super-polynomial) functions* $s_0 = s_0(\lambda)$ *and all* $i \in [n]$*, the distributions* Hyb_{i-1} *and* Hyb_i *are* (s_0, ε_0)*-indistinguishable.*

Proof. Suppose there exists an adversary $\mathcal{A}$ of size s_0 that can distinguish Hyb_{i-1} and Hyb_i with non-negligible advantage δ. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks hiding of Π_{BC} as follows:

1. Algorithm $\mathcal{B}$ receives $\mathsf{crs}_{\mathsf{BC}}$ from its challenger. It samples $\mathsf{crs}_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Setup}(1^\lambda, 1^{mB}, 1^{|C|})$, $\mathbf{s} \xleftarrow{R} \{0,1\}^n$, and computes $\mathbf{t} \leftarrow G_n(\mathbf{s})$.
2. Then, for $j < i$, algorithm $\mathcal{B}$ computes $(c_j, \sigma_j) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, 0)$ and for $j > i$, it computes $(c_j, \sigma_j) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, s_j)$. Algorithm $\mathcal{B}$ submits $(s_i, 0)$ as its challenge and sets c_i to be the challenger's response.
3. Algorithm $\mathcal{B}$ gives $(I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t})$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

If c_i is a commitment to s_i, then algorithm $\mathcal{B}$ perfectly simulates distribution Hyb_{i-1} and if c_i is a commitment to 0, then algorithm $\mathcal{B}$ perfectly simulates distribution Hyb_i. Thus, algorithm $\mathcal{B}$ also succeeds with advantage δ, and the claim follows. □

Claim 3.7. *Suppose G_λ is $(2^{\lambda^\alpha}, \varepsilon_{\mathsf{PRG}})$-secure for some constant $\alpha > 0$ and negligible function $\varepsilon_{\mathsf{PRG}} = \varepsilon_{\mathsf{PRG}}(\lambda) = \mathsf{negl}(\lambda)$. Then, Hyb_n and Hyb_{n+1} are $(2^{n^\alpha}, \varepsilon'_{\mathsf{PRG}})$-indistinguishable, where $\varepsilon'_{\mathsf{PRG}} = \varepsilon_{\mathsf{PRG}}(n)$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ of size $s_{\mathcal{A}} \le 2^{n^\alpha}$ that can distinguish $\mathsf{Hyb}_n(1^\lambda)$ and $\mathsf{Hyb}_{n+1}(1^\lambda)$ with advantage $\delta > \varepsilon'_{\mathsf{PRG}}$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks PRG security with seed length n:

1. At the beginning of the experiment, algorithm $\mathcal{B}$ receives a challenge $\mathbf{t} \in \{0,1\}^{mB}$.
2. $\mathcal{B}$ samples $\mathsf{crs}_{\mathsf{BC}} \leftarrow \mathsf{BC.Setup}(1^\lambda, \mathsf{hide})$, $\mathsf{crs}_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Setup}(1^\lambda, 1^{mB}, 1^{|C|})$. For each $i \in [n]$, it computes $(c_i, \sigma_i) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, 0)$.
3. Algorithm $\mathcal{B}$ gives $(I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t})$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

By construction, the size of algorithm $\mathcal{B}$ is $s_{\mathcal{A}} + \mathsf{poly}(\lambda, m, B, |C|) \le s_{\mathcal{A}} \cdot \mathsf{poly}(n)$, where the inequality holds since $m, B, |C|$ are all polynomially-bounded (in both λ and n). If $\mathbf{t} = G_n(\mathbf{s})$ for some $\mathbf{s} \xleftarrow{R} \{0,1\}^n$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_n(1^\lambda)$ for $\mathcal{A}$. Otherwise, if $\mathbf{t} \xleftarrow{R} \{0,1\}^{mB}$, then $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_{n+1}(1^\lambda)$ for $\mathcal{A}$. Thus, algorithm $\mathcal{B}$ breaks security of G_n with the same advantage $\delta > \varepsilon'_{\mathsf{PRG}} = \varepsilon_{\mathsf{PRG}}(n)$. □

By Claims 3.6 and 3.7, we can set $\varepsilon_{\mathsf{ideal}} = n \cdot \varepsilon_0 + \varepsilon_{\mathsf{PRG}}(n(\lambda, m)) = \mathsf{negl}(\lambda)$. The latter equality follows since $n(\lambda, m) \ge \lambda$. The lemma now follows by a hybrid argument. □

To complete the proof, we start by appealing to the Gentry-Wichs leakage simulation lemma (Lemma 2.2). Take any subset $I \subseteq [m]$. We start by defining the augmented distribution $\mathcal{D}^*_{\mathsf{real}} = \mathcal{D}^*_{\mathsf{real}}(1^\lambda)$:

- Sample $(I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}) \leftarrow \mathcal{D}_{\mathsf{real}}(1^\lambda)$ according to the real distribution. Each commitment c_i is computed as $(c_i, \sigma_i) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, s_i)$ where $\mathbf{s} \xleftarrow{R} \{0,1\}^n$ and $\mathbf{t} = G_n(\mathbf{s})$.
- Let $\mathsf{st} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{s}, c_1, \ldots, c_n, \sigma_1, \ldots, \sigma_n)$, and compute $\pi = (\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i(1)}, \ldots, \mathbf{t}_{i(L)})) \leftarrow \mathsf{Prove}(\mathsf{st}, I)$.
- Output $(I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t})$ and the auxiliary information $\mathsf{aux} = \pi_{\mathsf{BARG}}$. By definition, $|\mathsf{aux}| = |\pi_{\mathsf{BARG}}| = \ell_{\mathsf{BARG}}$.

By Lemma 3.5, the distributions $\mathcal{D}_{\mathsf{real}}$ and $\mathcal{D}_{\mathsf{ideal}}$ are $(s_{\mathsf{ideal}}, \varepsilon_{\mathsf{ideal}})$-indistinguishable for $s_{\mathsf{ideal}} = 2^{n^\alpha}$ and a negligible function $\varepsilon_{\mathsf{ideal}} = \mathsf{negl}(\lambda)$. Without loss of generality, we can assume that $\varepsilon_{\mathsf{ideal}} \ge 2^{-n^{\alpha/2}}$ (e.g., we can set $\varepsilon_{\mathsf{ideal}} = \max(\varepsilon'_{\mathsf{ideal}}, 2^{-n^{\alpha/2}})$, where $\varepsilon'_{\mathsf{ideal}}$ is the negligible function from Lemma 3.5). By Lemma 2.2, there exists an augmented distribution $\mathcal{D}^*_{\mathsf{ideal}} = \mathcal{D}^*_{\mathsf{ideal}}(1^\lambda)$ over tuples $((I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}), \mathsf{aux}')$ where $(I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}) \leftarrow \mathcal{D}_{\mathsf{ideal}}(1^\lambda)$ and $\mathsf{aux}' \in \{0,1\}^{\ell_{\mathsf{BARG}}}$. Moreover, the distributions $\mathcal{D}^*_{\mathsf{real}}$ and $\mathcal{D}^*_{\mathsf{ideal}}$ are $(s_{\mathsf{aug}}, \varepsilon_{\mathsf{aug}})$-indistinguishable where

$$s_{\mathsf{aug}} = s_{\mathsf{ideal}} \cdot \mathsf{poly}(\varepsilon_{\mathsf{ideal}}/2^{\ell_{\mathsf{BARG}}}) = 2^{n^\alpha} \cdot \mathsf{poly}(\varepsilon_{\mathsf{ideal}}/2^{\ell_{\mathsf{BARG}}})$$

and $\varepsilon_{\mathsf{aug}} = 2 \cdot \varepsilon_{\mathsf{ideal}} = \mathsf{negl}(\lambda)$. Since $\varepsilon_{\mathsf{ideal}} \ge 2^{-n^{\alpha/2}}$ and $n \ge \ell^c_{\mathsf{BARG}}$ for some constant $c > 1/\alpha$, this means that $s_{\mathsf{aug}} = 2^{\Omega(n^\alpha)}$. We summarize this in the following claim:

Claim 3.8. *Under the same conditions as in the statement of Lemma 3.5, the distributions $\mathcal{D}^*_{\mathsf{real}}$ and $\mathcal{D}^*_{\mathsf{ideal}}$ are $(s_{\mathsf{aug}}, \varepsilon_{\mathsf{aug}})$-indistinguishable where $s_{\mathsf{aug}} = 2^{\Omega(n^\alpha)}$ and $\varepsilon_{\mathsf{aug}} = \mathsf{negl}(\lambda)$.*

To complete the proof, we proceed via a sequence of hybrid experiments:

- Hyb_0: This is the real distribution where the challenger samples the bits $\mathbf{r}$ and the proof π as in the real scheme:
  - The challenger first samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^m)$. In particular, $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$, where $(\mathsf{crs}_{\mathsf{BC}}, \mathsf{td}) \leftarrow \mathsf{BC.Setup}(1^\lambda, \mathsf{bind})$, $\mathsf{crs}_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Setup}(1^\lambda, 1^{mB}, 1^{|C|})$, C is the circuit that computes the NP relation $\mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$ from Fig. 1, and $\mathbf{v}_1, \ldots, \mathbf{v}_m \xleftarrow{R} \{0,1\}^B$.

– Next, the challenger samples the bits $\mathbf{r}$ by running $(\mathbf{r}, \mathsf{st}) \leftarrow \mathsf{GenBits}(\mathsf{crs})$ and a proof by computing $\pi \leftarrow \mathsf{Prove}(\mathsf{st}, I)$. In particular, the challenger first samples a seed $\mathbf{s} \xleftarrow{R} \{0,1\}^n$ and computes $\mathbf{t} \leftarrow G_n(\mathbf{s})$. It splits $\mathbf{t} = \mathbf{t}_1 \| \mathbf{t}_2 \| \cdots \| \mathbf{t}_m$ into blocks where each $\mathbf{t}_i \in \{0,1\}^B$ for each $i \in [m]$. For each $i \in [m]$, the challenger computes $r_i \leftarrow \mathbf{v}_i^\intercal \mathbf{t}_i$ and sets $\mathbf{r} = r_1 \| \cdots \| r_m \in \{0,1\}^m$.

– To construct the proof π, the challenger computes the commitments $(c_i, \sigma_i) \leftarrow \mathsf{BC.Commit}(\mathsf{crs}_{\mathsf{BC}}, s_i)$ for each $i \in [n]$. It then parses $I = \{i^{(1)}, \ldots, i^{(L)}\}$, where the indices $i^{(1)}, \ldots, i^{(L)} \in [m]$ are in sorted order. The challenger then constructs the statement $(x_{1,1}, \ldots, x_{1,B}, \ldots, x_{L,1}, \ldots, x_{L,B})$ and witness $(w_{1,1}, \ldots, w_{1,B}, \ldots, w_{L,1}, \ldots, w_{L,B})$ according to Eq. (3.1) and Eq. (3.2). It constructs the BARG proof as in Prove:

$$\pi_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Prove}(\mathsf{crs}_{\mathsf{BARG}}, C, (x_{1,1}, \ldots, x_{1,B}, \ldots, x_{L,1}, \ldots, x_{L,B}), (w_{1,1}, \ldots, w_{1,B}, \ldots, w_{L,1}, \ldots, w_{L,B})),$$

and sets $\pi = \left(\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}})\right)$.

– The challenger gives $(\mathsf{crs}, I, \mathbf{r}_I, \pi, \mathbf{r}_{\bar{I}})$ to $\mathcal{A}$. Algorithm $\mathcal{A}$ then outputs a bit $b \in \{0,1\}$ which is the output of the experiment.

- Hyb_1: Same as Hyb_0 except the challenger now samples the commitment CRS $(\mathsf{crs}_{\mathsf{BC}}, \mathsf{td}) \leftarrow \mathsf{BC.Setup}(1^\lambda, \mathsf{hide})$. In this experiment, the distribution of $\left((\mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}), \pi_{\mathsf{BARG}}\right)$ is distributed according to $\mathcal{D}^*_{\mathsf{real}}$.
- Hyb_2: Same as Hyb_1 except the challenger samples components $\left((\mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}), \pi_{\mathsf{BARG}}\right) \leftarrow \mathcal{D}^*_{\mathsf{ideal}}$. Specifically, the experiment now proceeds as follows:
  – The challenger samples $\left((\mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}), \pi_{\mathsf{BARG}}\right) \leftarrow \mathcal{D}^*_{\mathsf{ideal}}$, $\mathbf{v}_1, \ldots, \mathbf{v}_m \xleftarrow{R} \{0,1\}^B$, and sets $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$.
  – Next, the challenger splits $\mathbf{t} = \mathbf{t}_1 \| \mathbf{t}_2 \| \cdots \| \mathbf{t}_m$ into blocks where each $\mathbf{t}_i \in \{0,1\}^B$ for each $i \in [m]$. For each $i \in [m]$, the challenger computes $r_i \leftarrow \mathbf{v}_i^\intercal \mathbf{t}_i$ and sets $\mathbf{r} = r_1 \| \cdots \| r_m \in \{0,1\}^m$.
  – The challenger sets the proof $\pi = \left(\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}})\right)$ and gives $(\mathsf{crs}, I, \mathbf{r}_I, \pi, \mathbf{r}_{\bar{I}})$ to $\mathcal{A}$.
- Hyb_3: Same as Hyb_2 except the challenger samples $\mathbf{r}_{\bar{I}} \xleftarrow{R} \{0,1\}^{|\bar{I}|}$.
- Hyb_4: Same as Hyb_3 except the challenger samples $\left((\mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}), \pi_{\mathsf{BARG}}\right) \leftarrow \mathcal{D}^*_{\mathsf{real}}$.
- Hyb_5: Same as Hyb_4 except the challenger samples $\mathsf{crs}_{\mathsf{BC}} \leftarrow \mathsf{BC.Setup}(1^\lambda, \mathsf{bind})$. Note that this coincides with the ideal distribution.

Lemma 3.9. *Suppose Π_{BC} satisfies mode indistinguishability. Then, Hyb_0 and Hyb_1 are computationally indistinguishable.*

Proof. Suppose there is an adversary $\mathcal{A}$ of size $s_0 = \mathsf{poly}(\lambda)$ that distinguishes the outputs of Hyb_0 and Hyb_1 with non-negligible probability δ. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks mode indistinguishability:

1. At the beginning of the game, algorithm $\mathcal{B}$ receives the security parameter 1^λ and a common reference string $\mathsf{crs}_{\mathsf{BC}}$ from the challenger.
2. $\mathcal{B}$ samples $\mathsf{crs}_{\mathsf{BARG}} \leftarrow \mathsf{BARG.Setup}(1^\lambda, 1^{mB}, 1^{|C|})$ and $\mathbf{v}_1, \ldots, \mathbf{v}_m \xleftarrow{R} \{0,1\}^B$. It constructs the common reference string $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$.
3. Algorithm $\mathcal{B}$ computes $(\mathbf{r}, \mathsf{st}) \leftarrow \mathsf{GenBits}(\mathsf{crs})$ and $\pi \leftarrow \mathsf{Prove}(\mathsf{st}, I)$.
4. Algorithm $\mathcal{B}$ gives $(\mathsf{crs}, I, \mathbf{r}_I, \pi, \mathbf{r}_{\bar{I}})$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

By construction, algorithm $\mathcal{B}$ has size $s_0 + \mathsf{poly}(\lambda, m, B, |C|) \leq s_0 \cdot \mathsf{poly}(\lambda)$ which holds due to $m, B, |C|$ all being $\mathsf{poly}(\lambda)$. When $\mathsf{crs}_{\mathsf{BC}}$ is sampled in binding mode, then algorithm $\mathcal{B}$ perfectly simulates Hyb_0 for $\mathcal{A}$. Alternatively, if $\mathsf{crs}_{\mathsf{BC}}$ is sampled in hiding mode, then algorithm $\mathcal{B}$ perfectly simulates Hyb_1 for $\mathcal{A}$. Critically, neither the GenBits nor the Prove algorithms require knowledge of the trapdoor td for the bit commitment scheme. Thus, algorithm $\mathcal{B}$ succeeds with the same advantage δ. □

Lemma 3.10. *Under the same conditions as in the statement of Claim 3.8,* Hyb_1 *and* Hyb_2 *are* $(s_{\mathsf{aug}}, \varepsilon_{\mathsf{aug}})$*-indistinguishable for* $s_{\mathsf{aug}} = 2^{\Omega(n^\alpha)}$ *and* $\varepsilon_{\mathsf{aug}} = \mathsf{negl}(\lambda)$.

Proof. Suppose there is an adversary $\mathcal{A}$ with size s_{aug} that distinguishes Hyb_1 and Hyb_2 with advantage $\delta > \varepsilon_{\mathsf{aug}}$. We construct algorithm $\mathcal{B}$ that distinguishes the distributions $\mathcal{D}^*_{\mathsf{real}}(1^\lambda)$ and $\mathcal{D}^*_{\mathsf{ideal}}(1^\lambda)$ as follows:

1. Algorithm $\mathcal{B}$ receives $(I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}, \pi_{\mathsf{BARG}})$ from the challenger. It parses $\mathbf{t} = \mathbf{t}_1 \| \cdots \| \mathbf{t}_m \in \{0,1\}^{mB}$ where each $\mathbf{t}_i \in \{0,1\}^B$. In addition, algorithm $\mathcal{B}$ samples $\mathbf{v}_1, \ldots, \mathbf{v}_m \xleftarrow{\mathrm{R}} \{0,1\}^B$.
2. Algorithm $\mathcal{B}$ computes $n = n(\lambda, m)$ and sets $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$. For each $i \in [m]$, it computes $r_i \leftarrow \mathbf{v}_i^\mathsf{T} \mathbf{t}_i$ and sets $\mathbf{r} = r_1 \| \cdots \| r_m$. Finally, it sets $\pi = (\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}}))$, where $I = \{i^{(1)}, \ldots, i^{(L)}\}$.
3. Algorithm $\mathcal{B}$ gives $(\mathsf{crs}, I, \mathbf{r}_I, \pi, \mathbf{r}_{\bar{I}})$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

Since algorithm $\mathcal{A}$ has size s_{aug}, the size of algorithm $\mathcal{B}$ is bounded by $s_{\mathsf{aug}} + \mathsf{poly}(\lambda, m, B) \le s_{\mathsf{aug}} \cdot \mathsf{poly}(\lambda)$ since m and B are both polynomially-bounded. By construction, when the challenge is sampled from the real distribution $\mathcal{D}^*_{\mathsf{real}}$, algorithm $\mathcal{B}$ perfectly simulates the distribution in Hyb_1. Alternatively, if the challenge is sampled from the ideal distribution $\mathcal{D}^*_{\mathsf{ideal}}$, algorithm $\mathcal{B}$ perfectly simulates the distribution in Hyb_2. Correspondingly, algorithm $\mathcal{B}$ is able to distinguish $\mathcal{D}^*_{\mathsf{real}}(1^\lambda)$ and $\mathcal{D}^*_{\mathsf{ideal}}(1^\lambda)$ with advantage $\delta > \varepsilon_{\mathsf{aug}}$ which contradicts Claim 3.8. □

Lemma 3.11. *Suppose* $B \ge \omega(\log \lambda) + \ell_{\mathsf{BARG}}$. *Then,* Hyb_2 *and* Hyb_3 *are statistically indistinguishable.*

Proof. Let $\bar{I} = \{i^{(1)}, i^{(2)}, \ldots, i^{(m-L)}\} \subseteq [m]$. We define a sequence of intermediate experiments $\mathsf{Hyb}_{2,j}$ for each $j \in \{0, \ldots, m-L\}$ as follows:

- $\mathsf{Hyb}_{2,0}$: Same as Hyb_2. In particular, the challenger samples $\left((I, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, c_1, \ldots, c_n, \mathbf{t}), \pi_{\mathsf{BARG}}\right) \leftarrow \mathcal{D}^*_{\mathsf{ideal}}$, $\mathbf{v}_1, \ldots, \mathbf{v}_m \xleftarrow{\mathrm{R}} \{0,1\}^B$ and sets $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m)$. It parses $\mathbf{t} = \mathbf{t}_1 \| \cdots \| \mathbf{t}_m$ where $\mathbf{t}_i \in \{0,1\}^B$ and computes $\mathbf{r} \leftarrow (\mathbf{v}_1^\mathsf{T} \mathbf{t}_1 \| \cdots \| \mathbf{v}_m^\mathsf{T} \mathbf{t}_m)$. Finally, it sets $\pi = \left(\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), (\mathbf{t}_{i^{(1)}}, \ldots, \mathbf{t}_{i^{(L)}})\right)$ and gives $(\mathsf{crs}, I, \mathbf{r}_I, \pi, \mathbf{r}_{\bar{I}})$ to the adversary.
- $\mathsf{Hyb}_{2,j}$: Same as $\mathsf{Hyb}_{2,j-1}$ except $r_{i^{(j)}} \xleftarrow{\mathrm{R}} \{0,1\}$. Note that $\mathsf{Hyb}_{2,m-L}$ is identical to Hyb_3.

We now appeal to the leftover hash lemma to show that for all $j \in [m-L]$, the statistical distance between $\mathsf{Hyb}_{2,j-1}(1^\lambda)$ and $\mathsf{Hyb}_{2,j}(1^\lambda)$ is negligible.

Claim 3.12. *Suppose* $B \ge \omega(\log \lambda) + \ell_{\mathsf{BARG}}$. *Then, for all* $j \in [m-L]$, *the statistical distance between* $\mathsf{Hyb}_{2,j-1}(1^\lambda)$ *and* $\mathsf{Hyb}_{2,j}(1^\lambda)$ *is negligible.*

Proof. The only difference between the two distributions is that in $\mathsf{Hyb}_{2,j-1}$, the challenger samples $r_{i^{(j)}} \leftarrow \mathbf{v}_{i^{(j)}}^\mathsf{T} \mathbf{t}_{i^{(j)}}$, whereas in $\mathsf{Hyb}_{2,j}$, the challenger samples $r_{i^{(j)}} \xleftarrow{\mathrm{R}} \{0,1\}$. First, define the random variable Z to be

$$Z = \left(n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \{\mathbf{v}_i\}_{i \ne i^{(j)}}, I, \mathbf{r}_I, \left(\pi_{\mathsf{BARG}}, (c_1, \ldots, c_n), \{\mathbf{t}_i\}_{i \in I}\right), \mathbf{r}_{\bar{I} \setminus \{i^{(j)}\}}\right).$$

Observe that the adversary's view in the two experiments then consists of the tuple $\left(r_{i^{(j)}}, \mathbf{v}_{i^{(j)}}, Z\right)$. In both $\mathsf{Hyb}_{2,j-1}$ and $\mathsf{Hyb}_{2,j}$, the challenger samples $\mathbf{t} \xleftarrow{\mathrm{R}} \{0,1\}^{mB}$. By construction, $\mathbf{t}_{i^{(j)}}$ is independent of all of the components in Z other than π_{BARG}. In conjunction with Lemma 2.1, we can now write

$$\mathbf{H}_\infty\left(\mathbf{t}_{i^{(j)}} \mid Z\right) = \mathbf{H}_\infty\left(\mathbf{t}_{i^{(j)}} \mid \pi_{\mathsf{BARG}}\right) \ge \mathbf{H}_\infty(\mathbf{t}_{i^{(j)}}) - |\pi_{\mathsf{BARG}}| = B - \ell_{\mathsf{BARG}} \ge \omega(\log \lambda),$$

since $B \ge \omega(\log \lambda) + \ell_{\mathsf{BARG}}$. Then, by the (generalized) leftover hash lemma (Corollary 2.4), we can conclude that the statistical distance between the distributions

$$\left(\mathbf{v}_{i^{(j)}}^\mathsf{T} \mathbf{t}_{i^{(j)}}, \mathbf{v}_{i^{(j)}}, Z\right) \quad \text{and} \quad \left(r_{i^{(j)}}, \mathbf{v}_{i^{(j)}}, Z\right),$$

where $\mathbf{v}_{i^{(j)}} \xleftarrow{\mathrm{R}} \{0,1\}^B$ and $r_{i^{(j)}} \xleftarrow{\mathrm{R}} \{0,1\}$ is at most $2^{-(\omega(\log \lambda)-1)/2} = \mathsf{negl}(\lambda)$. Since the statistical distance between the two experiments is negligible, the claim holds. □

The lemma now follows from Claim 3.12 and a standard hybrid argument (since $m = \mathsf{poly}(\lambda)$). □

Lemma 3.13. *Under the same conditions as in the statement of Claim 3.8,* Hyb_3 *and* Hyb_4 *are* $(s_{\mathsf{aug}}, \varepsilon_{\mathsf{aug}})$*-indistinguishable for* $s_{\mathsf{aug}} = 2^{\Omega(n^\alpha)}$ *and* $\varepsilon_{\mathsf{aug}} = \mathsf{negl}(\lambda)$.

Proof. Follows by an analogous argument as the proof of Lemma 3.10. □

Lemma 3.14. *Suppose* Π_{BC} *satisfies mode indistinguishability. Then,* Hyb_4 *and* Hyb_5 *are computationally indistinguishable.*

Proof. Follows by an analogous argument as the proof of Lemma 3.9. □

Combining Lemmas 3.9 to 3.11, 3.13 and 3.14 yields the theorem. □

Parameter selection. We now describe one candidate approach for instantiating the parameters in Construction 3.1:

Corollary 3.15 (Hidden-Bits Generator from Batch Arguments)**.** *Let* $k = k(\lambda)$ *be a locality parameter and suppose that* $G_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ *is a* k*-local PRG. Suppose* Π_{BARG} *is a non-interactive batch argument satisfying semi-adaptive soundness,* Π_{BC} *is a dual-mode commitment scheme, and that these underlying schemes satisfy the following conditions:*

- **PRG parameters:** *Suppose there exists a constant* $\alpha \in (0,1)$ *and a negligible function* $\varepsilon_{\mathsf{PRG}} = \mathsf{negl}(\lambda)$ *such that* G_λ *is* $\left(2^{\lambda^\alpha}, \varepsilon_{\mathsf{PRG}}\right)$ *secure. Moreover, suppose there exists a constant* $\delta_1 \in (0,1)$ *such that* $k \le \delta_1 \log \lambda$ *and a constant* $s > 1$ *such that* $\ell(\lambda) \ge \lambda^s$*. In words, we assume that* G_λ *has super-linear stretch, logarithmic locality, and sub-exponential security.*
- **BARG succinctness:** *Suppose there exists constants* $d > 0$, $\delta_2 \in (0,1)$ *and a polynomial* $q = q(\lambda)$ *such that the proof length* $\ell_{\mathsf{BARG}} = \ell_{\mathsf{BARG}}(\lambda, T, s)$ *for* Π_{BARG} *satisfies*

$$\ell_{\mathsf{BARG}}(\lambda, T, s) \le s^d \cdot T^{\delta_2} \cdot q(\lambda),$$

  where T *denotes the number of instances and* s *denotes a bound on the size of the circuit.*
- **Block size:** *Suppose* $B = \lambda + \ell_{\mathsf{BARG}}$.

Let $\delta_1' = (d \cdot \delta_1 + \varepsilon)/(1 - \delta_2)$ *for an arbitrarily small constant* $\varepsilon > 0$, $\delta_2' = \delta_2/(1-\delta_2)$, *and* $q'(\lambda) = q(\lambda)^{1/(1-\delta_2)}$*. Suppose moreover that the parameters satisfy the following properties:*

- **Hardness parameter:** $\alpha > \delta_1' + \delta_2'$.
- **Seed length:** $n = \max(\lambda, (m^{\delta_2'} \cdot q'(\lambda) \cdot O(\lambda^{\delta_2'}))^{1/(\alpha - \delta_1' - \varepsilon')})$ *for an arbitrary constant* $0 < \varepsilon' < \alpha - \delta_1' - \delta_2'$.
- **Stretch:** $s \ge (\alpha - \delta_1' - \varepsilon')(1 + \delta_2')/\delta_2' + \delta_1'$.

Then Construction 3.1 is a hidden-bits generator with subset-dependent proofs.

Proof. Take any input length m. Let $\mathsf{crs} = (n, \mathsf{crs}_{\mathsf{BARG}}, \mathsf{crs}_{\mathsf{BC}}, \mathbf{v}_1, \ldots, \mathbf{v}_m) \leftarrow \mathsf{Setup}(1^\lambda, 1^m)$. We first bound the size of the circuit C that computes the relation $\mathcal{R}[n, \mathsf{crs}_{\mathsf{BC}}]$:

- By construction, $|\mathsf{crs}_{\mathsf{BC}}| = \mathsf{poly}(\lambda)$. Correspondingly, the size of the circuit computing BC.Verify is $\mathsf{poly}(\lambda)$.
- Next, $G_n^{(i)}$ is a function on k-bit inputs, so it can be computed by a circuit of size $2^k \cdot \mathsf{poly}(k)$. Since $k \le \delta_1 \log n$, we can bound

$$|C| \le n^{\delta_1} \cdot \mathsf{poly}(\log n) = O(n^{\delta_1 + \varepsilon/d}).$$

For this choice of parameters, the length ℓ_{BARG} of the BARG proof satisfies

$$\ell_{\mathsf{BARG}} = \ell_{\mathsf{BARG}}(\lambda, mB, |C|) \leq |C|^d \cdot (mB)^{\delta_2} \cdot q(\lambda) = n^{d\delta_1+\varepsilon} \cdot m^{\delta_2} \cdot \ell_{\mathsf{BARG}}^{\delta_2} \cdot q(\lambda) \cdot O(\lambda^{\delta_2}).$$

Equivalently, this means

$$\ell_{\mathsf{BARG}} \leq (n^{d\delta_1+\varepsilon} \cdot m^{\delta_2})^{1/(1-\delta_2)} \cdot q(\lambda)^{1/(1-\delta_2)} \cdot O(\lambda^{\delta_2/(1-\delta_2)}) = n^{\delta_1'} \cdot m^{\delta_2'} \cdot q'(\lambda) \cdot O(\lambda^{\delta_2'}),$$

We now consider the requirements of Theorem 3.3, Theorem 3.4 and the requirement on the PRG stretch:

- Theorem 3.4 requires that $n \geq \max(\lambda, \ell_{\mathsf{BARG}}^c)$ for some constant $c > 1/\alpha$. Let $c = 1/(\alpha - \varepsilon') > 1/\alpha$. By assumption, we now have
$$n^{\alpha-\delta_1'-\varepsilon'} \geq m^{\delta_2'} \cdot q'(\lambda) \cdot O(\lambda^{\delta_2'}).$$
In particular, this means that
$$n^{\alpha-\varepsilon'} \geq n^{\delta_1'} \cdot m^{\delta_2'} \cdot q'(\lambda) \cdot O(\lambda^{\delta_2'}) \geq \ell_{\mathsf{BARG}}.$$
Correspondingly, we have $(n^{\alpha-\varepsilon'})^c = n \geq \ell_{\mathsf{BARG}}^c$, as required.

- Theorem 3.3 requires that $n \leq m^{\delta} \cdot \mathsf{poly}(\lambda)$ for some (universal) constant $\delta \in (0,1)$. Since $q = \mathsf{poly}(\lambda)$ and $\alpha, \delta_1', \varepsilon'$ are all constants, we currently have that $n \leq m^{\delta_2'/(\alpha-\delta_1'-\varepsilon')} \cdot \mathsf{poly}(\lambda)$. By construction, we have that $0 < \alpha - \delta_1' - \delta_2' - \varepsilon'$, so $\delta_2' < \alpha - \delta_1' - \varepsilon$. Thus, setting $\delta = \delta_2'/(\alpha - \delta_1' - \varepsilon') < 1$ satisfies the requirement.

- Finally, we require that $\ell(n) \geq mB$, or equivalently, $n^s \geq mB$. By construction,
$$\begin{aligned} n^s = n^{\delta_1'} n^{s-\delta_1'} &\geq n^{\delta_1'} \cdot \left(m^{\delta_2'} \cdot q'(\lambda) \cdot O(\lambda^{\delta_2'})\right)^{(s-\delta_1')/(\alpha-\delta_1'-\varepsilon')} \\ &\geq n^{\delta_1'} \cdot \left(m^{\delta_2'} \cdot q'(\lambda) \cdot O(\lambda^{\delta_2'})\right)^{(1+\delta_2')/\delta_2'} \\ &\geq n^{\delta_1'} \cdot m^{1+\delta_2'} \cdot (q'(\lambda))^{(1+\delta_2')/\delta_2'} \cdot O(\lambda^{1+\delta_2'}). \end{aligned}$$
Finally, we have
$$mB = m\lambda + m\ell_{\mathsf{BARG}} \leq m\lambda + n^{\delta_1'} \cdot m^{1+\delta_2'} \cdot q'(\lambda) \cdot O(\lambda^{\delta_2'}) \leq n^s,$$
as required. □

Candidate instantiations. For illustrative purposes, we now describe some instantiations of Corollary 3.15.

- Suppose we instantiate Construction 3.1 and Corollary 3.15 with a batch argument where the proof size scales *polylogarithmically* with the number of instances:
$$\ell_{\mathsf{BARG}}(\lambda, T, s) \leq s^d \cdot \mathsf{polylog}(T) \cdot q(\lambda)$$
for some constant $d > 0$. This is satisfied by most existing BARG constructions [CJJ21b, WW22, DGKV22, KLVW23, CGJ+22]. In this case, the constant δ_2 in Corollary 3.15 can be made *arbitrarily* small. Then we can instantiate Corollary 3.15 with any k-local PRG that is secure against 2^{λ^α}-size adversaries with locality $k \leq \delta_1 \log \lambda$ and stretch $s > 1 + d\delta_1$, provided that $\alpha/\delta_1 > d$. For example, we can rely on sub-exponential hardness of Goldreich's local PRG [Gol00] with *logarithmic* locality.

- We can also instantiate Construction 3.1 and Corollary 3.15 with a "mildly-succinct" batch argument where the BARG proof size scales polynomially with the number of instances:[7]
$$\ell_{\mathsf{BARG}}(\lambda, T, s) \leq s^d \cdot T^{\delta_2} \cdot q(\lambda)$$
for constants $\delta_2 \in (0, 1/2)$ and $d > 0$. In this case, we can instantiate Corollary 3.15 with a k-local PRG that is secure against 2^{λ^α}-size adversaries with locality $k \leq \delta_1 \log \lambda$ and stretch $s > 1 + \delta_1' + \delta_2'$, as long as $\delta_1' < \alpha - \delta_2'$ (for δ_1', δ_2' as in Corollary 3.15). In particular, we can still rely on sub-exponential hardness of Goldreich's PRG with logarithmic locality, but the sub-exponential hardness parameter α increases as δ_2 increases.

[7] We note here that additionally assuming a rate-1 string oblivious transfer protocol [DGI+19], such a BARG can be transformed into a BARG where the proof size scales polylogarithmically with the number of instances [KLVW23]. In this case, we would be able to appeal to our previous instantiation.

NIZK from batch arguments. Combining Theorem 2.11 and Corollary 3.15, we now obtain a NIZK for NP from a batch argument for NP:

Corollary 3.16 (NIZK from Batch Arguments)**.** *Suppose there exists a semi-adaptively-sound BARG, a dual-mode commitment scheme, and a sub-exponentially secure PRG with super-linear stretch and locality at most $k = c \log n$ with $c < 1$ and n-bit inputs. Then there exists a computational NIZK argument for* NP*.*

Remark 3.17 (Using Non-Local PRGs)**.** We note that a local PRG is not strictly necessary for Construction 3.1. It is sufficient to construct a PRG where each output bit of the PRG can be *verified* by a circuit of size n^{δ} where n is the seed length and $\delta < 1$ is a constant. Any PRG with this local verification property suffices for our main transformation.

Acknowledgments

We thank Brent Waters for helpful feedback on this work. D. J. Wu is supported by NSF CNS-2151131, CNS-2140975, a Microsoft Research Faculty Fellowship, and a Google Research Scholar award.

References

[ABR12] Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. In *TCC*, 2012.

[ACL+22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable. In *CRYPTO*, 2022.

[AFMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *ASIACRYPT*, 2020.

[AK19] Benny Applebaum and Eliran Kachlon. Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In *FOCS*, 2019.

[AL16] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In *STOC*, 2016.

[App12] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In *STOC*, 2012.

[App15] Benny Applebaum. The cryptographic hardness of random local functions - survey. *IACR Cryptol. ePrint Arch.*, 2015.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, 2018, 2018.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2), 1988.

[BCC+17] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *J. Cryptol.*, 30(4), 2017.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, 2012.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *STOC*, 2014.

[BDK+11] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In *CRYPTO*, 2011.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, 1988.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, 2009.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In *EUROCRYPT*, 2017.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In *EUROCRYPT*, 2018.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In *CRYPTO*, 2020.

[BKP+23a] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. *IACR Cryptol. ePrint Arch.*, 2023. https://eprint.iacr.org/archive/2023/754/20230525:044715.

[BKP+23b] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. *IACR Cryptol. ePrint Arch.*, 2023. https://eprint.iacr.org/archive/2023/754/20230725:080608.

[BQ09] Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. In *APPROX-RANDOM*, 2009.

[BS23] Ward Beullens and Gregor Seiler. Labrador: Compact proofs for R1CS from module-sis. In *EUROCRYPT*, 2023.

[BY92] Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In *CRYPTO*, 1992.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In *EUROCRYPT*, 2023.

[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 *STOC*, 2019.

[CDM+18] Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the concrete security of goldreich's pseudorandom generator. In *ASIACRYPT*, 2018.

[CEMT09] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In *TCC*, 2009.

[CGJ+22] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and SNARGs from sub-exponential DDH. *IACR Cryptol. ePrint Arch.*, 2022.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, 2003.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT*, 2020.

[CJJ21a] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In *CRYPTO*, 2021.

[CJJ21b] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for P from LWE. In *FOCS*, 2021.

[CL18] Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. In *TCC*, 2018.

[CM01] Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC. In *FOCS*, 2001.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *EUROCRYPT*, 2020.

[CPW23] Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs. A map of witness maps: New definitions and connections. In *PKC*, 2023.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *TCC*, 2012.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO*, pages 3–32, 2019.

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *FOCS*, 2022.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO*, 2002.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, 2004.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, 1990.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, 2013.

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

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. *IACR Cryptol. ePrint Arch.*, 2000.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, 2006.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for non-interactive zero-knowledge. *J. ACM*, 59(3), 2012.

[GR13] Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *J. Cryptol.*, 26(3), 2013.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, 2016.

[GSWW22] Rachit Garg, Kristin Sheridan, Brent Waters, and David J. Wu. Fully succinct batch arguments for np from indistinguishability obfuscation. In *TCC*, 2022.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, 2011.

[HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. Snargs for P from sub-exponential DDH and QR. In *EUROCRYPT*, 2022.

[HLOV11] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *ASIACRYPT*, 2011.

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. In *EUROCRYPT*, 2021.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In *EUROCRYPT*, 2022.

[KLVW23] Yael Tauman Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and RAM delegation. In *STOC*, 2023.

[KMY20] Fuyuki Kitagawa, Takahiro Matsuda, and Takashi Yamakawa. NIZK from SNARG. In *TCC*, 2020.

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In *STOC*, 2019.

[KVZ21] Yael Tauman Kalai, Vinod Vaikuntanathan, and Rachel Yun Zhang. Somewhere statistical soundness, post-quantum security, and SNARGs. In *TCC*, 2021.

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *CRYPTO*, 2017.

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In *ASIACRYPT*, 2013.

[LPWW20] Benoît Libert, Alain Passelègue, Hoeteck Wee, and David J. Wu. New constructions of statistical NIZKs: Dual-mode DV-NIZKs and more. In *EUROCRYPT*, 2020.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In *CRYPTO*, 2017.

[Mic95] Silvio Micali. Computationally-sound proofs. In *Proceedings of the Annual European Summer Meeting of the Association of Symbolic Logic*, 1995.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On ϵ-biased generators in NC^0. In *FOCS*, 2003.

[OW14] Ryan O'Donnell and David Witmer. Goldreich's PRG: evidence for near-optimal polynomial stretch. In *CCC*, 2014.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, 2013.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, 2019.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, 2008.

[QRW19] Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier NIZKs for all NP from CDH. In *EUROCRYPT*, 2019.

[Set20] Srinath T. V. Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *CRYPTO*, 2020.

[Üna23] Akin Ünal. Worst-case subexponential attacks on PRGs of constant degree or constant locality. In *EUROCRYPT*, 2023.

[WW22] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In *CRYPTO*, 2022.