Pairing-Based Batch Arguments for NP with a Linear-Size CRS

Binyi Chen
Stanford University
binyi@cs.stanford.edu

Noel Elias
UT Austin
nelias@utexas.edu

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

Abstract

Non-interactive batch arguments (BARGs) for NP allow a prover to prove ℓ NP statements with a proof whose size scales sublinearly with ℓ. In this work, we construct a pairing-based BARG where the size of the common reference string (CRS) scales linearly with the number of instances and the prover's computational overhead is quasi-linear in the number of instances. Our construction is fully black box in the use of the group. Security relies on a q-type assumption in composite-order pairing groups.

The best black-box pairing-based BARG prior to this work has a nearly-linear size CRS (i.e., a CRS of size $\ell^{1+o(1)}$) and the prover overhead is quadratic in the number of instances. All previous pairing-based BARGs with a sublinear-size CRS relied on some type of recursive composition and correspondingly, non-black-box use of the group. The main technical insight underlying our construction is to substitute the vector commitment in previous pairing-based BARGs with a polynomial commitment. This yields a scheme that does *not* rely on cross terms in the common reference string. In previous black-box pairing-based schemes, the super-linear-size CRS and quadratic prover complexity was due to the need for cross terms.

1 Introduction

In a non-interactive batch argument (BARG) for an NP relation $\mathcal{R}$ [BHK17, KPY19], a prover has a batch of NP statements $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ together with their associated witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_\ell$, and its goal is to produce a short proof π such that $\mathcal{R}(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all $i \in [\ell]$. The size of the proof π should be $\mathsf{poly}(\lambda, |\mathcal{R}|, \log \ell)$, where λ is the security parameter and $|\mathcal{R}|$ is the size of the Boolean circuit that computes the NP relation $\mathcal{R}$. Beyond their immediate application to amortizing the communication cost of NP verification, batch arguments (and their generalizations) have proven to be useful for building numerous cryptographic primitives including aggregate signatures [WW22, DGKV22, BCJP24, NWW24, NWW25], RAM delegation [KVZ21, CJJ21b, KLVW23, GSW23, ACG+24b], incremental verifiable computation [DGKV22, PP22], non-interactive zero-knowledge proofs [CW23, BKP+24, BWW24, BDS25], homomorphic signatures [ABF24, ACG24a], and more.

Driven by their numerous applications, batch arguments have been extensively studied in recent years, and we currently have constructions from many number-theoretic assumptions. These include learning with errors (LWE) [CJJ21b], the k-Lin assumption over a pairing group [WW22], the sub-exponential decisional Diffie-Hellman (DDH) assumption in a pairing-free group [CGJ+23], or a combination of quadratic residuosity (QR) and either LWE or sub-exponential DDH [CJJ21a]. Among these constructions, the pairing-based construction of Waters and Wu [WW22] has two appealing properties:

- **Does not require "heavy machinery:"** The [WW22] batch argument gives a direct construction from pairing groups and does not require any heavyweight tools or non-black-box use of cryptography. In contrast, the alternative approaches for constructing batch arguments rely on the correlation-intractability framework [CGH04, CCH+19], and in many cases, the PCP theorem (c.f., [CJJ21b,

CGJ+23]). Consequently, the [WW22] scheme (or the variant from [GLWW24]) is still the most concretely-efficient BARG based on standard cryptographic assumptions.

- **Somewhere *statistical* soundness:** Another appealing property of the [WW22] construction is that it satisfies somewhere *statistical* soundness (or extractability). This means that the common reference string of the BARG can be programmed to be *statistically* sound on a hidden index i. When the CRS is programmed to be statistically sound for index i, there does *not* exist any proofs for any batch of statements $(\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$ where $\mathbf{x}_i$ is false. In contrast, other constructions based on correlation-intractable hash functions only provide somewhere *computational* soundness (or extractability), which only asserts that such proofs are computationally hard to find. This stronger notion of somewhere statistical soundness is relevant for applications that combine BARGs with witness encryption or indistinguishability obfuscation (e.g., [DJWW25]).

A limitation: CRS size and prover complexity. A major limitation of the [WW22] BARG is that it relies on a structured reference string whose size scales *quadratically* with the number of instances. Namely, to support a proof of over ℓ statements, their scheme requires a CRS with $O(\ell^2)$ group elements. Correspondingly, the prover complexity in [WW22] is also quadratic in the number of instances: namely, $|C| \cdot O(\ell^2)$ group operations. A recent work of Garg, Lu, Waters, and Wu [GLWW24] showed how to reduce the size of the CRS to be *nearly* linear (i.e., $\ell^{1+o(1)}$) via a combinatoric approach of progression-free sets.[1] The prover overhead is still quadratic in the number of instances even with progression-free sets. In both constructions, the size of the CRS is super-linear and the prover complexity is quadratic because the scheme relies on "cross terms" (see Section 1.1) that are essential for correctness.

A natural question is whether we can construct a pairing-based BARG with a strictly linear-size CRS (and sub-quadratic prover overhead). With recursive composition and *non-black-box* use of the group, it is known how to obtain a BARG with CRS size ℓ^ε for any constant $\varepsilon > 0$ [WW22], or even $\mathrm{polylog}(\ell)$ [KLVW23]. Thus, we ask whether we can reduce the size of the CRS (and prover complexity) without needing non-black-box use of the group. Such a BARG would be very appealing from a concrete efficiency standpoint.

This work. In this work, we show how to adapt the [WW22] framework to obtain a BARG with a linear-size CRS using composite-order pairing groups. Moreover, constructing a proof requires $|C| \cdot \tilde{O}(\ell)$ group operations, where $\tilde{O}(\cdot)$ suppresses polylogarithmic factors. In other words, the total prover cost is quasi-linear in the number of instances (and linear in the size of the circuit). Like [WW22, GLWW24], our construction is fully black-box in the use of the group and satisfies somewhere statistical soundness (or extractability). Security relies on a new q-type assumption over composite-order pairing groups (which can be viewed as a combination of the bilinear Diffie-Hellman exponent assumption [BBG05] with the classic subgroup decision assumption [BGN05]). A notable feature of our construction is that it obviates the need to give out "cross terms" in the CRS, which is the reason for the super-linear-size CRS and quadratic prover overhead in previous pairing-based constructions. We summarize our construction in the following theorem:

Theorem 1.1 (Informal)**.** *Let λ be a security parameter and ℓ be the number of instances. Under the ℓ-subgroup decision exponent assumption (Assumption 3.2) over composite-order pairing groups, there exists a publicly-*

[1] The work of [GLWW24] notes that the $\ell^{1+o(1)}$ construction is interesting mostly in an asymptotic sense. For many practical values of ℓ (e.g., $\ell \leq 10^5$; see [GK24, Appendix VII]), the most concretely-efficient progression-free set construction is the construction based on ternary encodings from [ET36], which gives a scheme with a CRS of size $\ell^{\log_2 3}$. For $\ell = 10^5$, the CRS from [GLWW24] would contain roughly 10^8 group elements, which is already colossal. The asymptotic savings of more sophisticated progression-free set constructions only kick in beyond this point. On the flip side, the CRS in our construction contains exactly $\ell + 1$ group elements, where ℓ is the number of instances.

verifiable non-interactive BARG for any NP *relation* $\mathcal{R}$ *(on statements of size* n*) with proof size* $\mathsf{poly}(\lambda, |\mathcal{R}|)$*, prover complexity* $\tilde{O}(\ell)\cdot|\mathcal{R}|\cdot\mathsf{poly}(\lambda)$*, verification complexity* $\mathsf{poly}(\lambda, n, \ell)+\mathsf{poly}(\lambda, |\mathcal{R}|)$*, and CRS size* $\ell\cdot\mathsf{poly}(\lambda)$*.*

1.1 Technical Overview

The starting point of our construction is the Waters-Wu BARG [WW22]. We start by sketching their approach over composite-order pairing groups. A (symmetric) composite-order pairing group consists of two cyclic groups $\mathbb{G}, \mathbb{G}_T$ of order $N = pq$, where p, q are distinct primes, along with an efficiently-computable non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. In the following, we let $g \in \mathbb{G}$ be a generator for $\mathbb{G}$, and write $g_p := g^q$ and $g_q := g^p$ to denote generators of the order-p and order-q subgroups of $\mathbb{G}$, respectively. We will often write $\mathbb{G}_p$ and $\mathbb{G}_q$ to denote the order-p and order-q subgroups of $\mathbb{G}$, respectively. By bilinearity, we have

$$\forall u, v, x, y \in \mathbb{Z}_N : e\left(g_p^u g_q^v,\ g_p^x g_q^y\right) = e(g_p, g_p)^{ux} \cdot e(g_q, g_q)^{vy}.$$

Suppose we want to construct a BARG for ℓ instances. The CRS in the [WW22] scheme consists of ℓ random group elements $A_1, \ldots, A_\ell$ where $A_i = g_p^{\alpha_i}$ and $\alpha_i \xleftarrow{R} \mathbb{Z}_p$ together with a collection of cross terms $B_{i,i'} = g_p^{\alpha_i \alpha_{i'}}$ for all $i \neq i'$. The idea in [WW22] construction is then as follows:

- **Setup.** Consider the language of Boolean circuit satisfiability. Let $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit with s gates and t wires. Let $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$ be a list of statements, and $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^h$ be the associated witnesses. For each $i \in [\ell]$ and $j \in [t]$, let $w_{i,j} \in \{0,1\}$ be the value of the j^{th} wire of $C(\mathbf{x}_i, \mathbf{w}_i)$.
- **Wire commitments.** The [WW22] prover starts by committing to all of the wires. In particular, for each $j \in [t]$, the prover constructs a "vector commitment" to the values of wire j across the ℓ instances. Concretely, the prover computes

$$U_j = \prod_{i \in [\ell]} A_i^{w_{i,j}} = g_p^{\sum_{i \in [\ell]} \alpha_i w_{i,j}}. \tag{1.1}$$

  The prover includes the wire commitments $U_1, \ldots, U_t$ in the proof.
- **Proving validity of the commitments.** Next, the [WW22] prover constructs proofs that the commitments $U_1, \ldots, U_t$ are properly constructed. There are two types of claims:
  - **Wire validity:** First, the prover needs to convince the verifier that each U_j is a commitment to a 0/1 vector. The idea is to exploit the fact that $w_{i,j} \in \{0,1\}$ if and only if $w_{i,j}^2 = w_{i,j}$. In particular, this means

$$\begin{aligned}\left(\sum_{i \in [\ell]} \alpha_i w_{i,j}\right)\left(\sum_{i' \in [\ell]} \alpha_{i'} w_{i',j}\right) &= \sum_{i \in [\ell]} \alpha_i^2 w_{i,j}^2 + \sum_{i \in [\ell]} \sum_{i' \neq i} \alpha_i \alpha_{i'} w_{i,j} w_{i',j} \\ &= \sum_{i \in [\ell]} \alpha_i^2 w_{i,j} + \sum_{i \in [\ell]} \sum_{i' \neq i} \alpha_i \alpha_{i'} w_{i,j} w_{i',j} \\ &= \left(\sum_{i \in [\ell]} \alpha_i\right)\left(\sum_{i' \in [\ell]} \alpha_{i'} w_{i',j}\right) + \sum_{i \in [\ell]} \sum_{i' \neq i} \alpha_i \alpha_{i'} (w_{i,j} w_{i',j} - w_{i',j}).\end{aligned} \tag{1.2}$$

Using the cross terms, the prover can compute

$$V_j = \prod_{i\in[\ell]} \prod_{i'\neq i} B_{i,i'}^{w_{i,j}w_{i',j}-w_{i',j}} = g_p^{\sum_{i\in[\ell]}\sum_{i'\neq i}\alpha_i\alpha_{i'}(w_{i,j}w_{i',j}-w_{i',j})}.$$

If $w_{i,j}^2 = w_{i,j}$ for all $i \in [\ell]$, then Eq. (1.2) implies

$$e(U_j, U_j) = e(A, U_j) \cdot e(g_p, V_j), \tag{1.3}$$

where $A = \prod_{i\in[\ell]} A_i = g_p^{\sum_{i\in[\ell]}\alpha_i}$. The prover includes $V_1, \ldots, V_t$ as part of the proof.

– **Gate validity:** For each gate in the circuit, the prover includes an analogous proof that the wire commitments associated with the gate are consistent with the wire commitments associated with the inputs to the gate. Concretely, consider a NAND gate $G_k = (k_1, k_2, k_3)$ where $k_1, k_2 \in [t]$ are the indices of the input wire and k_3 is the index of the output wire. If $w_{i,k_3} = \mathsf{NAND}(w_{i,k_1}, w_{i,k_2})$, then it holds that $w_{i,k_3} = 1 - w_{i,k_1}w_{i,k_2}$. This is again a *quadratic* relation of the wire values, which can be handled in an analogous manner as the wire validity checks. Specifically, for each gate $G_k = (k_1, k_2, k_3)$ in the circuit, the prover constructs the group element $W_k \in \mathbb{G}$ corresponding to the "cross term" associated with the verification relation and the verifier checks that

$$e(A, U_{k_3}) = \frac{e(A, A)}{e(U_{k_1}, U_{k_2})} \cdot e(g_p, W_k). \tag{1.4}$$

The prover includes $W_1, \ldots, W_s$ as part of the proof.

- **Verification:** To check the proof $\pi = (U_1, \ldots, U_t, V_1, \ldots, V_t, W_1, \ldots, W_s)$, the verifier checks that the commitments to the statement wires are correctly computed (it can compute these itself), that Eq. (1.3) holds for all $j \in [t]$, that Eq. (1.4) holds for all $k \in [s]$, and that the output commitment U_t is a commitment to the all-ones vector (i.e., $U_t = A$).

The CRS in the [WW22] construction has size that scales quadratically with the number of instances ℓ. This is due to the cross terms $B_{i,k}$, which the prover uses to construct the wire validity and gate validity proofs. Similarly, constructing the wire validity proofs V_j and gate validity proofs W_k requires $|C| \cdot O(\ell^2)$ group operations. The recent work of [GLWW24] show how to use progression-free sets [ET36, Beh46, Elk10] to reduce the number of cross terms that need to be given out to be nearly linear in the number of instances (i.e., $\ell^{1+o(1)}$). This work essentially exploits the fact that different pairs of indices (i, j) and (i', j') could "share" a single cross term $B_{i,j}$. This reduce the number of cross terms in the CRS. However, in light of lower bounds on the minimum size of progression-free sets [Rot53, HB87, Sze90, BS20, KM23], this approach cannot give a construction with a strictly-linear-size CRS. Moreover, even though multiple pairs of indices may share a cross term, the prover computation still requires iterating over all distinct pairs of indices $i \neq i'$.

Encoding wire values using polynomials. The key insight in this work is to use a different mechanism to commit to the wire values. Whereas previous approaches [WW22, GLWW24] used a Pedersen-style vector commitment (see Eq. (1.1)) to commit to the wire values, we instead commit to a *polynomial* that represents the wire values. Specifically, for each wire $j \in [t]$, let $(w_{1,j}, \ldots, w_{\ell,j})$ be the vector of wire values across the ℓ instances. Define the polynomial $\Phi_j \in \mathbb{Z}_N[x]$ to be the (unique) polynomial of degree at most $\ell - 1$ where $\Phi_j(i) = w_{i,j}$. Suppose we want to argue that $w_{i,j} \in \{0, 1\}$ for all $i \in [\ell]$. As above, this amounts to checking that $\Phi_j(i) = \Phi_j^2(i)$ for all $i \in [\ell]$. Equivalently, this means $\Phi_j^2(i) - \Phi_j(i) = 0$ on all $i \in [\ell]$. Let

$\zeta(x) := \prod_{i\in[\ell]}(x-i)$ be the vanishing polynomial that is zero on all inputs $i \in [\ell]$. Then, $\zeta(x)$ divides $\Phi_j^2(x) - \Phi_j(x)$ if and only if $\Phi_j(i) = \Phi_j^2(i)$ for all $i \in [\ell]$, or equivalently, if and only if $w_{i,j} = \Phi_j(i) \in \{0,1\}$. Thus, checking whether the vector of wire values $(w_{1,j}, \ldots, w_{\ell,j}) \in \{0,1\}^\ell$ is binary or not boils down to checking whether the polynomial $\zeta(x)$ divides the polynomial $\Phi_j^2(x) - \Phi_j(x)$. The latter is equivalent to showing that there exists a quotient polynomial $Q_j \in \mathbb{Z}_N[x]$ such that

$$Q_j(x) \cdot \zeta(x) = \Phi_j^2(x) - \Phi_j(x). \tag{1.5}$$

We now proceed as follows:

- For each wire $j \in [t]$, the prover commits to the polynomial $\Phi_j(x)$ that interpolates the values associated with wire j. Let $U_1, \ldots, U_t$ be the commitments to $\Phi_1, \ldots, \Phi_t$.
- For each wire $j \in [t]$, the prover also commits to the quotient polynomial $Q_j(x)$ that satisfies Eq. (1.5). This check ensures that Φ_j is a commitment to a valid set of wire assignments. Let $V_1, \ldots, V_t$ be the commitments to $Q_1, \ldots, Q_t$.
- For each NAND gate $G_k = (k_1, k_2, k_3)$, the prover commits to a quotient polynomial $R_k(x)$ where

  $$R_k(x) \cdot \zeta(x) = 1 - \Phi_{k_3}(x) - \Phi_{k_1}(x)\Phi_{k_2}(x). \tag{1.6}$$

  By construction, if $\Phi_{k_3}(i) = \mathsf{NAND}(\Phi_{k_1}(i), \Phi_{k_2}(i))$ for all $i \in [\ell]$, then $\zeta(x)$ divides the polynomial $(1 - \Phi_{k_3}(x) - \Phi_{k_1}(x)\Phi_{k_2}(x))$, and correspondingly, the polynomial R_k exists. Let $W_1, \ldots, W_s$ be the commitments to $R_1, \ldots, R_s$.

Similar to [WW22], the proof now consists of the commitments $(U_1, \ldots, U_t, V_1, \ldots, V_t, W_1, \ldots, W_s)$, and the verifier checks the same set of relations as in [WW22]. The question now is how to construct the commitments to the polynomials and how the verifier checks Eq. (1.5) and Eq. (1.6).

- **Constructing the commitments.** The structure of our polynomial commitment is the same as the classic pairing-based construction from [KZG10], except we work over composite-order groups. Working over composite-order groups will enable us to argue somewhere extractability of our overall BARG. Specifically, the commitment to a polynomial $f \in \mathbb{Z}_N[x]$ is an encoding of $f(\alpha)$ where $\alpha \in \mathbb{Z}_N$ is a random point. To support this, the common reference string contains the group elements $A_i = g_p^{\alpha^i}$ for $i \in [0, \ell-1]$. In addition to $A_0, \ldots, A_{\ell-1}$, the CRS also contains a commitment $Z = g_p^{\zeta(\alpha)}$ to the polynomial $\zeta(x)$. Concretely, the CRS in our construction has the following form:

  $$\mathsf{crs} = ((\mathbb{G}, \mathbb{G}_T, e, g, g_p), A_0, \ldots, A_{\ell-1}, Z) \quad \text{where} \quad A_i = g_p^{\alpha^i} \text{ and } Z = g_p^{\zeta(\alpha)}. \tag{1.7}$$

  Now, to commit to a polynomial $f(x) := \sum_{i\in[0,d]} f_i x^i$ of degree at most $d \le \ell - 1$, the prover can compute

  $$\prod_{i\in[0,d]} A_i^{f_i} = g_p^{\sum_{i\in[0,d]} f_i \alpha^i} = g_p^{f(\alpha)}.$$

- **Wire validity checks.** Suppose U_j is a commitment to $\Phi_j(x)$ and V_j is a commitment to the quotient polynomial $Q_j(x)$ satisfying Eq. (1.5). This means $U_j = g_p^{\Phi_j(\alpha)}$ and $V_j = g_p^{Q_j(\alpha)}$. To check that Eq. (1.5) holds, the verifier can simply check that

  $$e(V_j, Z) = \frac{e(U_j, U_j)}{e(A_0, U_j)}. \tag{1.8}$$

This relations holds only if

$$Q_j(\alpha) \cdot \zeta(\alpha) = \Phi_j^2(\alpha) - \Phi_j(\alpha) \bmod p.$$

Intuitively, the verification test in Eq. (1.8) is checking polynomial equality by evaluating the polynomials in Eq. (1.5) at the random point α. Since a (univariate) polynomial (over $\mathbb{Z}_p$) of degree $2(\ell-1)$ has at most $2(\ell-1)$ roots, this test will only pass if Eq. (1.5) holds (except with negligible probability over the choice of α). The wrinkle, of course, is that the CRS includes encodings of the powers of α, and as such, we cannot simply argue that α is hidden from the view of the prover. The actual soundness analysis is more delicate, but this captures the basic intuition.

- **Gate validity checks.** The gate validity checks are implemented in an analogous manner as the wire validity checks. Namely, the prover commits to the quotient polynomial $R_k(x)$ for each gate (see Eq. (1.6)). Let $W_k = g_p^{R_k(\alpha)}$ be the commitment. The verifier then checks Eq. (1.6) by checking

$$e(W_k, Z) = \frac{e(A_0, A_0)}{e(A_0, U_{k_3}) \cdot e(U_{k_1}, U_{k_2})}. \tag{1.9}$$

  Observe that this essentially corresponds to checking that Eq. (1.6) holds at the random point α.

The advantage of encoding the wire values as polynomials is that we no longer need to give out cross terms in the CRS. The pairing relations just correspond to checking whether the polynomial evaluations at a *single* point match or not. For this reason, the resulting scheme only needs a CRS whose size is linear in the number of instances. In fact, the CRS in the scheme contains exactly $\ell + 1$ group elements. Moreover, the most expensive component of the prover computation is interpolating the polynomials $\Phi_j(x)$ for each $j \in [t]$. This can be done in *quasi-linear* time $\tilde{O}(\ell)$, so the overall prover complexity now consists of $|C| \cdot \tilde{O}(\ell)$ group operations[2] as opposed to $|C| \cdot O(\ell^2)$ in previous pairing-based constructions [WW22, GLWW24].

Somewhere extractability. The security requirement on the BARG is somewhere extractability [CJJ21b]. Specifically, there should be a trapdoor algorithm that takes as input an index $i^* \in [\ell]$ and outputs a CRS and an extraction trapdoor td. The "trapdoor CRS" should be computationally indistinguishable from the normal CRS (Eq. (1.7)). Moreover, there should be an efficient algorithm that takes as input the trapdoor td, a batch of statements $(\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$, and an accepting proof π, and outputs a witness $\mathbf{w}_{i^*}$ where $C(\mathbf{x}_{i^*}, \mathbf{w}_{i^*}) = 1$. In particular, this means the BARG is *statistically* sound on index i when the CRS is sampled to be extractable on index i.

We achieve this property using a similar idea as in [WW22, GLWW24]. Specifically, we lift the elements in the CRS from the $\mathbb{G}_p$ subgroup to the full group instead. We set things up so the $\mathbb{G}_q$ component of the wire commitments $U_1, \ldots, U_t$ exactly encodes the wire value associated with instance i^*; moreover, the wire validity and gate validity checks enforce that the wire values in the $\mathbb{G}_q$ subgroup satisfy the respective requirements. If all of the checks pass, then we can extract the wires associated with instance i, and specifically, the inputs $\mathbf{x}_{i^*}$ and $\mathbf{w}_{i^*}$ such that $C(\mathbf{x}_{i^*}, \mathbf{w}_{i^*}) = 1$. This suffices to argue somewhere extractability.

In more detail, recall in the above construction that for each wire $j \in [t]$, the value of the polynomial Φ_j at i is exactly the wire value $w_{i,j} \in \{0, 1\}$. To support extracting the wire values for instance i^* from a commitment, we define the CRS components A_i (see Eq. (1.7)) as

$$A_i = g_p^{\alpha^i} \rightarrow A_i = g_p^{\alpha^i} g_q^{(i^*)^i}.$$

We make a few observations:

[2] More precisely, our construction requires $\tilde{O}(\ell)$ operations over $\mathbb{Z}_N$ (for polynomial interpolation) followed by $O(\ell)$ group operations.

- Consider an honest commitment U_j to Φ_j. In this case, the user would compute

$$U_j = g_p^{\Phi_j(\alpha)} g_q^{\Phi_j(i^*)} = g_p^{\Phi_j(\alpha)} g_q^{w_{i^*,j}}.$$

  Observe that the $\mathbb{G}_q$-component of U_j is precisely the value $w_{i^*,j}$ of the j^{th} wire in instance i^*.

- Suppose the prover chooses wire commitments $U_1, \ldots, U_t$. By the Chinese Remainder Theorem, we can express each of these elements as $U_j = g_p^{\gamma_{j,p}} g_q^{\gamma_{j,q}}$ for some $\gamma_{j,p}, \gamma_{j,q} \in \mathbb{Z}_N$. Consider now the wire validity check from Eq. (1.8), and in particular, consider the check in the order-q subgroup. By definition, the element $Z = g_p^{\zeta(\alpha)}$ is in the order-p subgroup[3], so $e(V_j, Z)$ vanishes in the order-q subgroup. Since $A_0 = g_p g_q$, Eq. (1.8) holds if and only if $\gamma_{j,q}^2 = \gamma_{j,q} \bmod q$. Namely, $\gamma_{j,q} \in \{0, 1\}$.

- Next, consider the gate validity check. If we again consider the check in the order-q subgroup and use the fact that Z vanishes in the order-q subgroup, Eq. (1.9) holds if and only if

$$\gamma_{k_3,q} + \gamma_{k_1,q}\gamma_{k_2,q} = 1 \bmod q,$$

  or equivalently, if $\gamma_{k_3,q} = 1 - \gamma_{k_1,q}\gamma_{k_2,q} = \mathsf{NAND}(\gamma_{k_1,q}, \gamma_{k_2,q})$.

This shows that the wire validity checks and gate validity checks enforce that the $\mathbb{G}_q$-components of the wire commitments are binary-valued and satisfy the gate constraints. Coupled with the fact that the statements are correctly computed and that the output is a commitment to 1, this means the $\mathbb{G}_q$-components of $U_1, \ldots, U_t$ correspond to the wire values of $C(\mathbf{x}_{i^*}, \mathbf{w}_{i^*})$ for some $\mathbf{w}_{i^*}$ where $C(\mathbf{x}_{i^*}, \mathbf{w}_{i^*}) = 1$. Projecting into the $\mathbb{G}_q$ subgroup allows us to *extract* a witness $\mathbf{w}_{i^*}$ for $\mathbf{x}_{i^*}$, thus proving somewhere extractability.

CRS indistinguishability. The final requirement is that the trapdoor CRS (which has the hidden index i) is computationally indistinguishable from the normal CRS in Eq. (1.7). To argue this, it suffices to show that the subgroup decision assumption holds [BGN05] (i.e., a random element of the subgroup $\mathbb{G}_p$ is computationally indistinguishable from a random element of the full group $\mathbb{G}$) even given powers $(g_p^{\alpha}, \ldots, g_p^{\alpha^{\ell-1}})$. Specifically, we require the following two distributions be computationally indistinguishable:

$$(g_p, g_p^{\alpha}, g_p^{\alpha^2}, \ldots, g_p^{\alpha^{\ell-1}}) \quad \text{and} \quad (g_p g_q, g_p^{\alpha}, g_p^{\alpha^2}, \ldots, g_p^{\alpha^{\ell-1}}),$$

where g_p, g_q are *random* generators of $\mathbb{G}_p$ and $\mathbb{G}_q$, respectively. In Appendix A, we show that this assumption holds (assuming hardness of factoring) in the generic composite-order bilinear group model.

A retrospective: pairing-based SNARGs for NP. Beginning with the seminal work of Groth [Gro10], a long sequence of works (c.f., [Lip12, PHGR13, GGPR13, BCI+13, DFGK14, Gro16]) has showed how to reduce the CRS size and prover complexity of pairing-based succinct non-interactive arguments (SNARGs) for NP. Similar to the case with pairing-based BARGs, the initial constructions either had a quadratic-size CRS [Gro10] or a nearly-linear size CRS [Lip12] (using progression-free sets). The [Gro10, Lip12] constructions (implicitly) rely on a "linear PCP" [IKO07, GGPR13, BCI+13] based on the Hadamard encoding, and the CRS for the SNARG essentially consists of encodings of pairwise products (i.e., the analog of cross terms in the BARG setting). In fact, the Hadamard encoding of a statement-witness pair $(\mathbf{x}, \mathbf{w})$ is precisely a random linear combination of the wire values of $C(\mathbf{x}, \mathbf{w})$. This is conceptually similar to how

[3] Technically, the reduction constructs Z by evaluating the polynomial ζ using the components $A_0 = g_p g_q, \ldots, A_\ell = g_p^{\alpha^\ell} g_q^{(i^*)^\ell}$ from the CRS components. Since $\zeta(i^*) = 0$, this ensures $Z = g_p^{\zeta(\alpha)} \in \mathbb{G}_p$.

the [WW22, GLWW24] BARG encodes the wire labels for different instances by taking a random linear combination of the wire values associated with each instance.

The breakthrough work of Gennaro, Gentry, Parno, and Raykova [GGPR13] showed how to eliminate the cross-terms with a new linear PCP based on quadratic arithmetic programs. Essentially, these constructions operate by encoding the instances as evaluations of a polynomial (and more generally, as the codewords of a multiplication code [BBC+19, BHI+24]). This yielded the first pairing-based SNARGs for NP with a linear-size CRS and the core approach has subsequently enabled many efficient pairing-based constructions [PHGR13, BCG+13, BCI+13, BCTV14, DFGK14, Gro16].

We can view the progress on BARG constructions as following a similar trajectory. The [WW22, GLWW24] constructions can be viewed as analogs of the early SNARGs with a quadratic or nearly-linear CRS. These construction essentially rely on a Hadamard-like encoding of the instances, and correctness requires publishing a collection of cross terms in the CRS. In this work, we represent the instances as polynomial evaluations, which is similar to the pairing-based SNARGs with a linear-size CRS (e.g., [GGPR13, BCI+13, DFGK14, Gro16]). In the case of SNARGs, the prover constructs a polynomial that interpolates all of the wires in the circuit whereas in our application to BARGs, the prover constructs a polynomial for each wire that interpolates to the wire values across the different instances. The key difference between these two lines of work is the fact that with BARGs, we demand security from *falsifiable* assumptions, whereas the aforementioned SNARGs for NP have all relied on stronger *knowledge* assumptions.

2 Preliminaries

Throughout this work, we write λ to denote the security parameter. For an integer $n \in \mathbb{N}$, we write $[n] \coloneqq \{1, \ldots, n\}$. We use bold lowercase letters (e.g., $\mathbf{x}$) to denote vectors. We use non-boldface letters to refer to their components (e.g., $\mathbf{x} = [x_1, \ldots, x_n]$). We write $\mathsf{poly}(\lambda)$ to denote a fixed function that is bounded by some polynomial in λ and $\mathsf{negl}(\lambda)$ to denote a function that is negligible in λ (i.e., $f(\lambda) = \mathsf{negl}(\lambda)$ if $f = o(\lambda^{-c})$ for all constants $c \in \mathbb{N}$). We say an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We say that two ensembles of distributions $\mathcal{D}_0 = \{\mathcal{D}_{0,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{A}(1^\lambda, x) = 1 : x \leftarrow \mathcal{D}_{0,\lambda}] - \Pr[\mathcal{A}(1^\lambda, x) = 1 : x \leftarrow \mathcal{D}_{1,\lambda}]| = \mathsf{negl}(\lambda).$$

We say the two distributions are statistically indistinguishable if the statistical distance between $\mathcal{D}_{0,\lambda}$ and $\mathcal{D}_{1,\lambda}$ is $\mathsf{negl}(\lambda)$. We say an event occurs with overwhelming probability if its complement occurs with negligible probability.

Boolean circuits. Like [WW22], we focus exclusively on the NP-complete language of Boolean circuit satisfiability and assume without loss of generality that the circuit consists exclusively of NAND gates. For a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ with t wires and s NAND gates, we associate wires $1, \ldots, n$ with the bits of the statement, wires $n+1, \ldots, n+h$ with the wires of the witness, and wire t with the output wire. We model each NAND gate as a triple $G = (k_1, k_2, k_3)$ where k_1, k_2 correspond to the indices of the input wire, and k_3 corresponds to the index of the output wire.

Batch argument for NP. We now define the notion of a somewhere-extractable batch argument for circuit satisfiability [CJJ21b]:

Definition 2.1 (Batch Argument for Circuit Satisfiability)**.** A non-interactive batch argument (BARG) for circuit satisfiability is a tuple of three efficient algorithms $\Pi_{\mathsf{BARG}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following properties:

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

Moreover, we require Π_{BARG} to satisfy the following properties:

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

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^s) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), (\mathbf{w}_1, \ldots, \mathbf{w}_\ell)) \end{array}\right] = 1.$$

- **Somewhere extractable:** There exists a pair of efficient algorithms $(\mathsf{TrapSetup}, \mathsf{Extract})$ with the following syntax:
  - $\mathsf{TrapSetup}(1^\lambda, 1^\ell, 1^s, i^*) \to (\mathsf{crs}, \mathsf{td})$: On input the security parameter $\lambda \in \mathbb{N}$, the number of instances $\ell \in \mathbb{N}$, a bound on the circuit size $s \in \mathbb{N}$, and a special index $i^* \in [n]$, the trapdoor setup algorithm outputs a common reference string crs and an extraction trapdoor td.
  - $\mathsf{Extract}(\mathsf{td}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi) \to \mathbf{w}^*$: On input the trapdoor td, a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a tuple of statements $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$, and a proof π, the extraction algorithm outputs a witness $\mathbf{w}^* \in \{0,1\}^h$.

  Moreover, these algorithms satisfy the following properties:

  - **CRS indistinguishability:** For integers $\ell, s \in \mathbb{N}$, an adversary $\mathcal{A}$, and a bit $b \in \{0,1\}$, we define the CRS indistinguishability experiment as follows:
    1. On input $(1^\lambda, 1^\ell, 1^s)$, algorithm $\mathcal{A}$ outputs an index $i^* \in [\ell]$.
    2. If $b = 0$, the challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^s)$. If $b = 1$, the challenger samples $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^\lambda, 1^\ell, 1^s, i^*)$. The challenger gives crs to $\mathcal{A}$.
    3. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

    We say that Π_{BARG} satisfies CRS indistinguishability if for all polynomials $\ell = \ell(\lambda)$, $s = s(\lambda)$, and all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

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

    in the CRS indistinguishability experiment.

– **Somewhere extractable in trapdoor mode:** For integers $\ell, s \in \mathbb{N}$ and an adversary $\mathcal{A}$, we define the somewhere extractability experiment as follows:

1. On input $(1^\lambda, 1^\ell, 1^s)$, algorithm $\mathcal{A}$ outputs an index $i^* \in [\ell]$.
2. The challenger samples $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^\lambda, 1^\ell, 1^s, i^*)$ and gives crs to $\mathcal{A}$.
3. Algorithm $\mathcal{A}$ outputs a circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a collection of statements $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$, and a proof π.
4. The output of the experiment is $b' = 1$ if $\mathsf{Verify}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi) = 1$ and $C(\mathbf{x}_{i^*}, \mathbf{w}^*) = 0$, where $\mathbf{w}^* \leftarrow \mathsf{Extract}(\mathsf{td}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi)$.

We say that Π_{BARG} is somewhere extractable in trapdoor mode if for all polynomials $\ell = \ell(\lambda)$, $s = s(\lambda)$, and all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr[b' = 1] = \mathsf{negl}(\lambda)$$

in the somewhere extractability game. If this holds for all (possibly unbounded) adversaries $\mathcal{A}$, then we say Π_{BARG} is statistically somewhere extractable in trapdoor mode.

3 BARG with Linear-Size CRS from Composite-Order Bilinear Groups

In this section, we show how to construct a BARG for NP with a linear-size CRS using composite-order pairing groups. We begin by recalling the notion of a composite-order pairing group.

Definition 3.1 (Composite-Order Bilinear Group [BGN05])**.** A (symmetric) composite-order bilinear group generator is an efficient algorithm CompGroupGen that takes as input the security parameter $\lambda \in \mathbb{N}$ and outputs the description $(\mathbb{G}, \mathbb{G}_T, p, q, g, e)$ of a bilinear group where $p, q > 2^\lambda$ are distinct primes, $\mathbb{G}$, $\mathbb{G}_T$ are cyclic groups of order $N = pq$, and $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate bilinear map. Moreover, the group operation in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the pairing e are efficiently-computable.

Subgroup decision exponent assumption. Security of our scheme will rely on a variant of the subgroup decision assumption that combines features of the standard subgroup decision assumption [BGN05] and the bilinear Diffie-Hellman exponent assumption [BBG05]. The standard subgroup decision assumption in a pairing group $(\mathbb{G}, \mathbb{G}_T, N, g, e)$ of composite order $N = pq$ asserts that a random element of the subgroup $\mathbb{G}_p \subset \mathbb{G}$ of order p is computationally indistinguishable from a random element of the full group $\mathbb{G}$. In this work, we assume subgroup decision holds even given $g_p^\alpha, g_p^{\alpha^2}, \ldots, g_p^{\alpha^\ell}$ for any polynomially-bounded $\ell = \ell(\lambda)$ and where $\alpha \xleftarrow{R} \mathbb{Z}_N$, and g_p is a random generator of the subgroup of order p. We give the formal statement below, and show that the assumption holds in the generic composite-order bilinear group model (assuming hardness of factoring) in Appendix A.

Assumption 3.2 (Subgroup Decision Exponent)**.** Let $\ell \in \mathbb{N}$. We say the ℓ-subgroup decision exponent assumption holds with respect to a composite-order group generator CompGroupGen if the distributions $\mathcal{D}_0 = \{\mathcal{D}_{0,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable. Both distributions $\mathcal{D}_{0,\lambda}$ and $\mathcal{D}_{1,\lambda}$ start by sampling $(\mathbb{G}, \mathbb{G}_T, p, q, g, e) \leftarrow \mathsf{CompGroupGen}(1^\lambda)$. Set $N = pq$, sample a random $r \xleftarrow{R} \mathbb{Z}_N$, and let $g_p = g^{qr}$, $g_q = g^{pr}$ be random generators of the order-p and order-q subgroups of $\mathbb{G}$, respectively. Let $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, N, g, e)$. In addition, sample $\alpha \xleftarrow{R} \mathbb{Z}_N$. The output of each distribution is then

- $\mathcal{D}_{0,\lambda}$: Output $(\mathcal{G}, (g_p^\alpha, \ldots, g_p^{\alpha^\ell}), g_p)$.
- $\mathcal{D}_{1,\lambda}$: Output $(\mathcal{G}, (g_p^\alpha, \ldots, g_p^{\alpha^\ell}), g_p g_q)$.

BARG from composite-order pairing groups. We now describe our BARG from composite-order pairing groups.

Construction 3.3 (BARG for NP from Composite-Order Pairing Groups)**.** Let CompGroupGen be a composite-order group generator. We construct a BARG for the language of circuit satisfiability as follows.

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: On input the security parameter λ and the number of instances $\ell \leq 2^\lambda$, the setup algorithm proceeds as follows:
  - Run $(\mathbb{G}, \mathbb{G}_T, p, q, e, g) \leftarrow \mathsf{CompGroupGen}(1^\lambda)$. Let $N = pq$, sample $r \xleftarrow{R} \mathbb{Z}_N$, and set $g_p = g^{qr}$. Let $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, N, g, e)$.
  - Sample $\alpha \xleftarrow{R} \mathbb{Z}_N$. For each $i \in [0, \ell - 1]$, let $A_i = g_p^{\alpha^i}$.
  - Define the polynomial $\zeta(x) := \prod_{i \in [\ell]} (x - i) \in \mathbb{Z}_N[x]$ and let $Z = g_p^{\zeta(\alpha)}$.
  - Output the common reference string $\mathsf{crs} = (\mathcal{G}, A_0, \ldots, A_{\ell-1}, Z)$.
- $\mathsf{Prove}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), (\mathbf{w}_1, \ldots, \mathbf{w}_\ell))$: On input the reference string $\mathsf{crs} = (\mathcal{G}, A_0, \ldots, A_{\ell-1}, Z)$, a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$, and witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^h$, let t be the number of wires in C and s be the number of gates in C. For each $i \in [\ell]$ and $j \in [t]$, let $w_{i,j}$ denote the value of the j^{th} wire in $C(\mathbf{x}_i, \mathbf{w}_i)$. The prover constructs the proof as follows:
  - **Interpolate wire values:** For each $j \in [t]$, let $\Phi_j \in \mathbb{Z}_N[x]$ be the (unique) polynomial of degree at most $\ell - 1$ where $\Phi_j(i) = w_{i,j}$ for all $i \in [\ell]$. Compute Φ_j using Lagrange interpolation modulo N.[4]
  - **Encoding wire values:** For each $j \in [t]$, write $\Phi_j(x) = \sum_{i \in [0,\ell-1]} \varphi_{j,i} x^i$. Compute $U_j = \prod_{i \in [0,\ell-1]} A_i^{\varphi_{j,i}}$.
  - **Validity of wire assignments:** For each $j \in [t]$, define the polynomial
$$Q_j(x) := \frac{\Phi_j^2(x) - \Phi_j(x)}{\zeta(x)}.$$
    Write $Q_j(x) = \sum_{i \in [0,\ell-1]} q_{j,i} x^i$. If the polynomial Q_j does not exist or cannot be written in this form, then abort with output $\perp$. Compute $V_j = \prod_{i \in [0,\ell-1]} A_i^{q_{j,i}}$.
  - **Validity of gate computation:** For each NAND gate $G_k = (k_1, k_2, k_3) \in [t]^3$ where $k \in [s]$, compute the polynomial
$$R_k(x) := \frac{1 - \Phi_{k_3}(x) - \Phi_{k_1}(x)\Phi_{k_2}(x)}{\zeta(x)}.$$
    Write $R_k(x) = \sum_{i \in [0,\ell-1]} r_{k,i} x^i$. If the polynomial R_k does not exist or cannot be written in this form, then abort with output $\perp$. Compute $W_k = \prod_{i \in [0,\ell-1]} A_i^{r_{k,i}}$.

  Finally, output the proof $\pi = (\{(j, U_j, V_j)\}_{j \in [t]}, \{(k, W_k)\}_{k \in [s]})$.

[4] In particular, when $\ell \leq 2^\lambda < p, q$, where p, q are the prime factors of N, the interpolation base $\{1, \ldots, \ell\}$ consists of unique elements over $\mathbb{Z}_p$ and $\mathbb{Z}_q$. Correctness of Lagrange interpolation and uniqueness of the interpolating polynomial now follow via the Chinese Remainder theorem.

- Verify$(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi)$: On input the common reference string $\mathsf{crs} = (\mathcal{G}, A_0, \ldots, A_{\ell-1}, Z)$, a Boolean circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$, and a proof $\pi = (\{(j, U_j, V_j)\}_{j\in[t]}, \{(k, W_k)\}_{k\in[s]})$, the verification algorithm checks the following properties:

  - **Statement check:** For each $j \in [n]$, let $\Phi_j \in \mathbb{Z}_N[x]$ be the (unique) polynomial of degree at most $\ell - 1$ where $\Phi_j(i) = x_{i,j}$. Compute Φ_j using Lagrange interpolation modulo N. Write $\Phi_j(x) = \sum_{i\in[0,\ell-1]} \varphi_{j,i} x^i$ and check that

$$U_j = \prod_{i\in[0,\ell-1]} A_i^{\varphi_{j,i}}. \tag{3.1}$$

  - **Validity of wire assignments:** For each $j \in [t]$, check that

$$e(U_j, U_j) = e(A_0, U_j) \cdot e(Z, V_j).$$

  - **Validity of gate computation:** For each NAND gate $G_k = (k_1, k_2, k_3) \in [t]^3$ where $k \in [s]$, check that

$$e(U_{k_1}, U_{k_2}) \cdot e(A_0, U_{k_3}) \cdot e(Z, W_k) = e(A_0, A_0).$$

  - **Output satisfiability:** Check that $U_t = A_0$.

  Output 1 if all checks pass and 0 otherwise.

Theorem 3.4 (Completeness)**.** *Construction 3.3 is complete.*

Proof. Take any circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$, and witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^h$ such that $C(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all $i \in [\ell]$. Let

$$\begin{aligned} \mathsf{crs} &\leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ \pi &\leftarrow \mathsf{Prove}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), (\mathbf{w}_1, \ldots, \mathbf{w}_\ell)). \end{aligned}$$

We show that $\mathsf{Verify}(\mathsf{crs}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi)$ outputs 1. First, write $\pi = (\{(j, U_j, V_j)\}_{j\in[t]}, \{(k, W_k)\}_{k\in[s]})$, where t is the number of wires in C and s is the number of gates. Then, the elements in π satisfy the following properties:

- For each $i \in [\ell]$ and $j \in [t]$, let $w_{i,j}$ denote the value assigned to the j^{th} wire in $C(\mathbf{x}_i, \mathbf{w}_i)$.
- For all $j \in [t]$, let $\Phi_j(x)$ be the unique polynomial of degree at most $\ell - 1$ where $\Phi_j(i) = w_{i,j}$. Write $\Phi_j(x) = \sum_{i\in[0,\ell-1]} \varphi_{j,i} x^i$. Next, $A_i = g_p^{\alpha^i}$ so we can write

$$U_j = \prod_{i\in[0,\ell-1]} A_i^{\varphi_{j,i}} = g_p^{\sum_{i\in[0,\ell-1]} \varphi_{j,i}\alpha^i} = g_p^{\Phi_j(\alpha)}. \tag{3.2}$$

- Since C is a Boolean circuit, it holds that $w_{i,j} \in \{0,1\}$ for all $i \in [\ell]$ and $j \in [t]$. Correspondingly, this means that $w_{i,j}^2 = w_{i,j}$. Since $\Phi_j(i) = w_{i,j}$ this means that for all $j \in [t]$:

$$\forall i \in [\ell] : \Phi_j^2(i) = w_{i,j}^2 = w_{i,j} = \Phi_j(i).$$

In particular, this means that for all $i \in [\ell]$, $\Phi_j^2(i) - \Phi_j(i) = 0$. This means the polynomial $\Phi_j^2(x) - \Phi_j(x)$ has roots at $x \in \{1, \ldots, \ell\}$, and thus, there exists a polynomial $Q_j \in \mathbb{Z}_N[x]$ such that

$$\Phi_j^2(x) - \Phi_j(x) = Q_j(x) \cdot \prod_{i \in [\ell]} (x - i) = Q_j(x) \cdot \zeta(x). \tag{3.3}$$

Since $\Phi_j^2 - \Phi_j$ has maximum degree $2(\ell - 1)$ and ζ has degree ℓ, we conclude that the polynomial Q_j has degree at most $\ell - 2$, and thus $Q_j(x)$ can always be written as $Q_j(x) = \sum_{i \in [0, \ell-1]} q_{j,i} x^i$. Finally, the prove algorithm computes

$$V_j = \prod_{i \in [0, \ell-1]} A_i^{q_{j,i}} = g_p^{\sum_{i \in [0,\ell-1]} q_{j,i} \alpha^i} = g_p^{Q_j(\alpha)}. \tag{3.4}$$

- Finally, take any NAND gate $G_k = (k_1, k_2, k_3) \in [t]^3$ in C (where $k \in [s]$). By construction, we have for all $i \in [\ell]$ that $w_{i,k_3} = \mathsf{NAND}(w_{i,k_1}, w_{i,k_2}) = 1 - w_{i,k_1} w_{i,k_2}$. This means for all $i \in [\ell]$,

$$0 = 1 - w_{i,k_1} w_{i,k_2} - w_{i,k_3} = 1 - \Phi_{k_1}(i)\Phi_{k_2}(i) - \Phi_{k_3}(i).$$

This means the polynomial $1 - \Phi_{k_1}(x)\Phi_{k_2}(x) - \Phi_{k_3}(x)$ has roots at $x \in \{1, \ldots, \ell\}$, and thus, there exists a polynomial $R_k \in \mathbb{Z}_N[x]$ such that

$$1 - \Phi_{k_1}(x)\Phi_{k_2}(x) - \Phi_{k_3}(x) = R_k(x) \cdot \prod_{i \in [\ell]} (x - i) = R_k(x) \cdot \zeta(x). \tag{3.5}$$

Since $1 - \Phi_{k_1}(x)\Phi_{k_2}(x) - \Phi_{k_3}(x)$ has maximum degree $2(\ell - 1)$ and ζ has degree ℓ, the polynomial R_k has degree at most $\ell - 2$, and thus R_k can always be written as $R_k(x) = \sum_{i \in [0,\ell-1]} r_{k,i} x^i$. Finally, the prove algorithm computes

$$W_k = \prod_{i \in [0, \ell-1]} A_i^{r_{k,i}} = g_p^{\sum_{i \in [0,\ell-1]} r_{k,i} \alpha^i} = g_p^{R_k(\alpha)}. \tag{3.6}$$

Consider now the verification checks that Verify performs:

- **Statement check:** For each $j \in [n]$, the Verify algorithm interpolates the polynomial $\Phi_j(x)$ where $\Phi_j(i) = x_{i,j} = w_{i,j}$. Thus, the Verify algorithm constructs the same polynomials $\Phi_1, \ldots, \Phi_n$ as the Prove algorithm, and correspondingly, the same elements $U_1, \ldots, U_n$. Thus, this check passes by construction.
- **Validity of wire assignments:** From Eqs. (3.2) to (3.4), we can write

$$\begin{aligned} e(U_j, U_j) = e\big(g_p^{\Phi_j(\alpha)}, g_p^{\Phi_j(\alpha)}\big) = e(g_p, g_p)^{\Phi_j^2(\alpha)} &= e(g_p, g_p)^{\Phi_j(\alpha)} \cdot e(g_p, g_p)^{Q_j(\alpha)\zeta(\alpha)} \\ &= e(A_0, U_j) \cdot e(Z, V_j), \end{aligned}$$

since $A_0 = g_p^{\alpha^0} = g_p$, $Z = g_p^{\zeta(\alpha)}$, and $V_j = g_p^{Q_j(\alpha)}$. Thus, the wire validity check passes.

- **Validity of gate computation:** From Eqs. (3.2), (3.5) and (3.6), we can write

$$\begin{aligned} e(A_0, A_0) = e(g_p, g_p) &= e(g_p, g_p)^{\Phi_{k_1}(\alpha)\Phi_{k_2}(\alpha) + \Phi_{k_3}(\alpha) + R_k(\alpha)\zeta(\alpha)} \\ &= e\big(g_p^{\Phi_{k_1}(\alpha)}, g_p^{\Phi_{k_2}(\alpha)}\big) e\big(g_p, g_p^{\Phi_{k_3}(\alpha)}\big) e\big(g_p^{\zeta(\alpha)}, g_p^{R_k(\alpha)}\big) \\ &= e(U_{k_1}, U_{k_2}) e(A_0, U_{k_3}) e(Z, R_k), \end{aligned}$$

so the gate validity check passes.

- **Output satisfiability:** From Eq. (3.2), the prover computes $U_t = g_p^{\Phi_t(\alpha)}$, where Φ_t is the unique polynomial of degree at most $\ell - 1$ that satisfies $\Phi_t(i) = w_{i,t} = 1$ for all $i \in [\ell]$. This means $\Phi_t(x) \equiv 1$ is the *constant* polynomial equal to 1 everywhere. In this case $U_t = g_p^{\Phi_t(\alpha)} = g_p = A_0$, as required.

Since all of the verification checks pass, we conclude that Verify outputs 1. □

Theorem 3.5 (Somewhere Extractability)**.** *Suppose the ℓ-subgroup decision exponent assumption (Assumption 3.2) holds with respect to* CompGroupGen*. Then Construction 3.3 satisfies somewhere extractability.*

Proof. We start by defining the trapdoor setup and extraction algorithms:

- $\mathsf{TrapSetup}(1^\lambda, 1^\ell, i^*)$: On input the security parameter λ, the number of instances ℓ, and the target index $i^* \in [\ell]$, the trapdoor setup algorithm works as follows:
  - Run $(\mathbb{G}, \mathbb{G}_T, p, q, e, g) \leftarrow \mathsf{CompGroupGen}(1^\lambda)$. Let $N = pq$, sample $r \xleftarrow{R} \mathbb{Z}_N$, and set $g_p = g^{qr}$, $g_q = g^{pr}$. Let $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, N, g, e)$.
  - Sample $\alpha \xleftarrow{R} \mathbb{Z}_N$. For each $i \in [0, \ell - 1]$, let $A_i = g_p^{\alpha^i} g_q^{(i^*)^i}$.
  - Define the polynomial $\zeta(x) := \prod_{i \in [\ell]} (x - i) \in \mathbb{Z}_N[x]$ and let $Z = g_p^{\zeta(\alpha)}$.
  - Output the common reference string $\mathsf{crs} = (\mathcal{G}, A_0, \ldots, A_{\ell-1}, Z)$ and the extraction trapdoor $\mathsf{td} = g_q$.
- $\mathsf{Extract}(\mathsf{td}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi)$: On input the trapdoor $\mathsf{td} = g_q$, the circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$, and a proof $\pi = (\{(k, U_j, V_j)\}_{j \in [t]}, \{(k, W_k)\}_{k \in [s]})$, the extraction algorithm sets wire value $w_j^* = 0$ if $e(g_q, U_j) = 1$ and $w_j^* = 1$ otherwise for all $j \in [n+1, n+h]$. Then, it outputs the extracted witness $\mathbf{w}^* = (w_{n+1}^*, \ldots, w_{n+h}^*)$.

We now show the CRS indistinguishability and the somewhere extractability in trapdoor mode properties.

Lemma 3.6 (CRS Indistinguishability)**.** *Suppose the ℓ-subgroup decision exponent assumption (Assumption 3.2) holds with respect to* CompGroupGen*. Then Construction 3.3 satisfies CRS indistinguishability.*

Proof. Take any $\ell = \ell(\lambda)$ where $\ell \le 2^\lambda$. Suppose there exists an efficient adversary $\mathcal{A}$ that can break CRS indistinguishability. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the ℓ-subgroup decision exponent assumption:

1. At the beginning of the game, algorithm $\mathcal{B}$ receives a challenge $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, N, g, e)$, a list of group elements $Y_1, \ldots, Y_\ell \in \mathbb{G}$, and a challenge element $T \in \mathbb{G}$.
2. Algorithm $\mathcal{B}$ starts running algorithm $\mathcal{A}$ which starts by outputting an index $i^* \in [\ell]$. Algorithm $\mathcal{B}$ now constructs the crs as follows:
   - For all $i \in [0, \ell]$, compute
   $$A_i = T^{(i^*)^i} \prod_{j \in [i]} Y_j^{\binom{i}{j} (i^*)^{i-j}}.$$
   - Next, it defines the polynomial $\zeta(x) = \prod_{i \in [\ell]} (x - i)$, writes $\zeta(x) = \sum_{i \in [0,\ell]} \zeta_i x^i$, and computes $Z = \prod_{i \in [0,\ell]} A_i^{\zeta_i}$.

   Algorithm $\mathcal{B}$ gives $\mathsf{crs} = (\mathcal{G}, A_0, \ldots, A_{\ell-1}, Z)$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs. Note that A_ℓ is used in computing Z but not published in crs.

By construction, the challenger in the ℓ-subgroup decision exponent assumption samples $(\mathbb{G}, \mathbb{G}_T, p, q, g, e) \leftarrow \mathsf{CompGroupGen}(1^\lambda)$ and then sets $N = pq$, $g_p = g^{qr}$, $g_q = g^{pr}$, where $r \xleftarrow{R} \mathbb{Z}_N$, and $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, N, g_p, e)$. In addition, it also samples $\alpha \xleftarrow{R} \mathbb{Z}_N$ and sets $Y_i = g_p^{\alpha^i}$. Consider now the distribution of each A_i and Z:

- If $T = g_p$, then
$$A_i = T^{(i^*)^i} \prod_{j \in [i]} Y_j^{\binom{i}{j}(i^*)^{i-j}} = g_p^{(i^*)^i + \sum_{j \in [i]} \binom{i}{j} \alpha^j (i^*)^{i-j}} = g_p^{(\alpha + i^*)^i}.$$
Moreover,
$$Z = \prod_{i \in [0,\ell]} A_i^{\zeta_i} = \prod_{i \in [0,\ell]} g_p^{\zeta_i (\alpha + i^*)^i} = g_p^{\zeta(\alpha + i^*)}.$$
Since the distribution of $\alpha \xleftarrow{R} \mathbb{Z}_N$ is independent of i^*, the distribution of $(\alpha + i^*)$ is uniform over $\mathbb{Z}_N$. Thus, the crs outputs by this process is distributed according to $\mathsf{Setup}(1^\lambda, 1^\ell)$.

- If $T = g_p g_q$, then
$$A_i = T^{(i^*)^i} \prod_{j \in [i]} Y_j^{\binom{i}{j}(i^*)^{i-j}} = g_p^{(i^*)^i + \sum_{j \in [i]} \binom{i}{j} \alpha^j (i^*)^{i-j}} g_q^{(i^*)^i} = g_p^{(\alpha + i^*)^i} g_q^{(i^*)^i}.$$
Moreover,
$$Z = \prod_{i \in [0,\ell]} A_i^{\zeta_i} = \prod_{i \in [0,\ell]} g_p^{\zeta_i (\alpha + i^*)^i} g_q^{\zeta_i (i^*)^i} = g_p^{\zeta(\alpha + i^*)} g_q^{\zeta(i^*)} = g_p^{\zeta(\alpha + i^*)},$$
since $\zeta(x)$ vanishes at $i^* \in [\ell]$. In this case the distribution of crs is precisely that output by $\mathsf{TrapSetup}(1^\lambda, 1^\ell, i^*)$.

We conclude that $\mathcal{B}$ breaks the ℓ-subgroup decision exponent assumption with the same advantage ε. □

Lemma 3.7 (Somewhere Extractable in Trapdoor Mode)**.** *Construction 3.3 is statistically somewhere extractable in trapdoor mode.*

Proof. Take any polynomial $\ell = \ell(\lambda)$ and let $i^* \leftarrow \mathcal{A}(1^\lambda, 1^\ell)$. Then, sample $(\mathsf{crs}^*, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^\lambda, 1^\ell, i^*)$. Then, we can write
$$\mathsf{crs}^* = (\mathcal{G}, A_0, \ldots, A_{\ell-1}, Z),$$
where $\mathcal{G} = (\mathbb{G}, \mathbb{G}_T, N, g, e)$, $A_i = g_p^{\alpha^i} g_q^{(i^*)^i}$, $Z = g_p^{\zeta(\alpha)}$, where $\alpha, r \xleftarrow{R} \mathbb{Z}_N$ and $g_p = g^{qr}$, $g_q = g^{pr}$. Let $\mathbb{G}_p \subset \mathbb{G}$ be the subgroup of $\mathbb{G}$ of order p and $\mathbb{G}_q \subset \mathbb{G}$ be the subgroup of $\mathbb{G}$ of order q. With overwhelming probability over the choice of r, g_p is a generator of $\mathbb{G}_p$ and g_q is a generator of $\mathbb{G}_q$. Then, by the Chinese Remainder Theorem, we can decompose every element $h \in \mathbb{G}$ as $g_p^{\gamma_p} g_q^{\gamma_q}$ for some choice of $\gamma_p \in \mathbb{Z}_p, \gamma_q \in \mathbb{Z}_q$. Take any circuit $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$ and a proof $\pi = (\{(k, U_j, V_j)\}_{j \in [t]}, \{(k, W_k)\}_{k \in [s]})$ where $\mathsf{Verify}(\mathsf{crs}^*, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi) = 1$. We now consider each of the verification checks:

- Write $U_j = g_p^{\gamma_{j,p}} g_q^{\gamma_{j,q}}$ for some choice of $\gamma_{j,p} \in \mathbb{Z}_p, \gamma_{j,q} \in \mathbb{Z}_q$. We first claim that $\gamma_{j,q} \in \{0,1\}$ for all $j \in [t]$. Consider the wire validity check. Since π is a valid proof, this means $e(U_j, U_j) = e(A_0, U_j) \cdot e(Z, V_j)$. Consider this relation in the mod-q subgroup. By construction $Z \in \mathbb{G}_p$, so if we restrict our attention to only the mod-q components, then this verification relation implies that $\gamma_{j,q}^2 = \gamma_{j,q}$, which means $\gamma_{j,q} \in \{0,1\}$, as required.

- Next, for every NAND gate $G_k = (k_1, k_2, k_3) \in [t]^3$, we claim that $\mathsf{NAND}(\gamma_{k_1,q}, \gamma_{k_2,q}) = \gamma_{k_3,q}$. We use the gate validity check. Again, because π is a valid proof, this means $e(U_{k_1}, U_{k_2}) \cdot e(A_0, U_{k_3}) \cdot e(Z, R_k) = e(A_0, A_0)$. Again, if we just consider the relation in the mod-q subgroup and using again the fact that $Z \in \mathbb{G}_p$, we obtain the relation $\gamma_{k_1,q}\gamma_{k_2,q} + \gamma_{k_3,q} = 1$. This is equivalent to the statement $\mathsf{NAND}(\gamma_{k_1,q}, \gamma_{k_2,q}) = \gamma_{k_3,q}$.

- Finally, we argue that $\gamma_{j,q} = x_{i^*,j}$ for all $j \in [n]$. This follows from the input validity check. Since the verification algorithm passes, we have $U_j = \prod_{i \in [0,\ell-1]} A_j^{\varphi_{j,i}} = g_p^{\Phi_j(\alpha)} g_q^{\Phi_j(i^*)}$, where $\Phi_j(x) = \sum_{i \in [0,\ell-1]} \varphi_{j,i} x^i$ satisfies $\Phi_j(i) = x_{j,i}$ for all $i \in [\ell]$. If we consider again the verification relation in the mod-q subgroup, then we conclude that $\gamma_{j,q} = \Phi_j(i^*) = x_{i^*,j}$.

- Finally, the output validity check requires that $U_t = A_0 = g_p g_q$. This means $\gamma_{t,q} = 1$.

Taken together, the above properties show that $\gamma_{1,q}, \ldots, \gamma_{t,q} \in \{0,1\}$ are a valid labeling of the wires of C where the first n inputs coincide with $\mathbf{x}_{i^*}$ and the output bit is $\gamma_{t,q} = 1$. To complete the proof, we analyze the output of $\mathbf{w}^* = \mathsf{Extract}(\mathsf{td}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \pi)$. By definition, $w_i^* = 0$ if and only if $e(g_q, U_j) = 1$, or equivalently, if $\gamma_{j,q} = 0$. Alternatively, $w_i^* = 1$ if and only if $e(g_q, U_j) \neq 1$, or equivalently, if $\gamma_{j,q} \neq 0$. We conclude then that $\mathbf{w}^* = (\gamma_{n+1,q}, \ldots, \gamma_{n+h,q})$. The above analysis shows that $C(\mathbf{x}, \mathbf{w}^*) = 1$, so extraction succeeds (with overwhelming probability over the choice of g_p and g_q). □

Somewhere extractability now follows by combining Lemmas 3.6 and 3.7. □

Theorem 3.8 (Succinctness)**.** *Construction 3.3 is succinct.*

Proof. Take any $\lambda, \ell, s \in \mathbb{N}$ and consider a Boolean circuit $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ of at most size s. Let $t = \mathsf{poly}(s)$ be the number of wires in circuit C. We check each property:

- **Proof size:** A proof π consists of $2t + s$ elements in $\mathbb{G}$, each of which can be represented by $\mathsf{poly}(\lambda)$ bits. Thus the proof size satisfies $|\pi| = (2t + s) \cdot \mathsf{poly}(\lambda) = \mathsf{poly}(\lambda, s)$.

- **CRS size:** The common reference string crs consists of the group description $\mathbb{G}$, and $\ell + 1$ elements in $\mathbb{G}$. Thus, $|\mathsf{crs}| = \ell \cdot \mathsf{poly}(\lambda)$.

- **Verification complexity:** The running time of the verification algorithm is then

$$\underbrace{n \cdot \mathsf{poly}(\lambda, \ell)}_{\text{statement check}} + \underbrace{t \cdot \mathsf{poly}(\lambda)}_{\text{wire validity check}} + \underbrace{s \cdot \mathsf{poly}(\lambda)}_{\text{gate validity check}} + \underbrace{\mathsf{poly}(\lambda)}_{\text{output check}} = \mathsf{poly}(\lambda, n, \ell) + \mathsf{poly}(\lambda, s)$$

  since $n, t = \mathsf{poly}(s)$. □

Remark 3.9 (Faster Verification with Preprocessing)**.** Like the [WW22] construction, if the statements $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^n$ are known in advance, we can precompute the "correct" statement encodings $U_1, \ldots, U_n$ in an "offline" phase (via Eq. (3.1)). This requires time $\mathsf{poly}(\lambda, n, \ell)$. In the online phase, the verifier can use the pre-computed encodings and implement the statement check in $n \cdot \mathsf{poly}(\lambda)$ time (i.e., independent of the number of instances ℓ). Correspondingly, the online verification complexity is simply $\mathsf{poly}(\lambda, s)$, independent of ℓ.

Acknowledgments

We thank Brent Waters for multiple helpful discussions on this work. Binyi Chen is supported by NSF, DARPA, the Simons Foundation, and UBRI. David J. Wu is supported by NSF CNS-2140975, CNS-2318701, a Sloan Fellowship, a Microsoft Research Faculty Fellowship, and a Google Research Scholar award. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

[ABF24] Gaspard Anthoine, David Balbás, and Dario Fiore. Fully-succinct multi-key homomorphic signatures from standard assumptions. In *CRYPTO*, 2024.

[ACG24a] Abtin Afshar, Jiaqi Cheng, and Rishab Goyal. Multi-hop multi-key homomorphic signatures with context hiding from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2024.

[ACG+24b] Abtin Afshar, Jiaqi Cheng, Rishab Goyal, Aayush Yadav, and Saikumar Yadugiri. Encrypted RAM delegation: Applications to rate-1 extractable arguments, homomorphic NIZKs, MPC, and more. *IACR Cryptol. ePrint Arch.*, 2024.

[BBC+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In *CRYPTO*, 2019.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, 2005.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO*, 2013.

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

[BCJP24] Maya Farber Brodsky, Arka Rai Choudhuri, Abhishek Jain, and Omer Paneth. Monotone-policy aggregate signatures. In *EUROCRYPT*, 2024.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von Neumann architecture. In *USENIX Security Symposium*, 2014.

[BDS25] Pedro Branco, Nico Döttling, and Akshayaram Srinivasan. Rate-1 statistical non-interactive zero-knowledge. In *CRYPTO*, 2025.

[Beh46] F. Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences*, 32(12), 1946.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC*, 2005.

[BHI+24] Nir Bitansky, Prahladh Harsha, Yuval Ishai, Ron D. Rothblum, and David J. Wu. Dot-product proofs and their applications. In *FOCS*, 2024.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *STOC*, 2017.

[BKP+24] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron D. Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. In *STOC*, 2024.

[BS20] Thomas F Bloom and Olof Sisask. Breaking the logarithmic barrier in Roth's theorem on arithmetic progressions. *arXiv preprint arXiv:2007.03528*, 2020.

[BWW24] Eli Bradley, Brent Waters, and David J. Wu. Batch arguments to NIZKs from one-way functions. In *TCC*, 2024.

[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.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4), 2004.

[CGJ+23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and SNARGs from sub-exponential DDH. In *CRYPTO*, 2023.

[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 $\mathcal{P}$ from LWE. In *FOCS*, 2021.

[CW23] Jeffrey Champion and David J. Wu. Non-interactive zero-knowledge from non-interactive batch arguments. In *CRYPTO*, 2023.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT*, 2014.

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

[DJWW25] Lalita Devadas, Abhishek Jain, Brent Waters, and David J. Wu. Succinct witness encryption for batch languages and applications. *IACR Cryptol. ePrint Arch.*, 2025.

[Elk10] Michael Elkin. An improved construction of progression-free sets. In *SODA*, 2010.

[ET36] Paul Erdös and Paul Turán. On some sequences of integers. *Journal of the London Mathematical Society*, 1(4), 1936.

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

[GK24] William Gasarch and Clyde Kruskal. Finding large sets without arithmetic progressions of length three: An empirical view and survey II, 2024. Available at https://www.cs.umd.edu/~gasarch/BLOGPAPERS/3apsurvey.pdf.

[GLWW24] Rachit Garg, George Lu, Brent Waters, and David J. Wu. Reducing the CRS size in registered ABE systems. In *CRYPTO*, 2024.

[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.

[GSW23] Riddhi Ghosal, Amit Sahai, and Brent Waters. Non-interactive publicly-verifiable delegation of committed programs. In *PKC*, 2023.

[HB87] David Rodney Heath-Brown. Integer sets containing no arithmetic progressions. *Journal of the London Mathematical Society*, 2(3), 1987.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *CCC*, 2007.

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

[KM23] Zander Kelley and Raghu Meka. Strong bounds for 3-progressions. In *FOCS*, 2023.

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

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

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

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT*, 2010.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, 2012.

[NWW24] Shafik Nassar, Brent Waters, and David J. Wu. Monotone policy BARGs from BARGs and additively homomorphic encryption. In *TCC*, 2024.

[NWW25] Shafik Nassar, Brent Waters, and David J. Wu. Monotone-policy BARGs and more from BARGs and quadratic residuosity. In *PKC*, 2025.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE S&P*, 2013.

[PP22] Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. In *FOCS*, 2022.

[Rot53] Klaus F Roth. On certain sets of integers. *Journal of the London Mathematical Society*, 1(1), 1953.

[Sze90] Endre Szemerédi. Integer sets containing no arithmetic progressions. *Acta Mathematica Hungarica*, 56(1), 1990.

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

A Generic Hardness of Assumption 3.2

In this section, we show that the subgroup decision exponent assumption (Assumption 3.2) we rely on holds in the generic bilinear group model of composite order. To do so, we rely on the "master theorems" from [KSW08, Appendix A]; specifically, we use the corrected version from [GLWW24, Theorem D.4]. We start by recalling the generic bilinear group model and [GLWW24, Theorem D.4]. Our definitions are taken nearly verbatim from [GLWW24, Appendix D].

Definition A.1 (Generic Bilinear Group Model)**.** For a positive integer $N \in \mathbb{Z}$, let $\mathcal{L} \subseteq \{0,1\}^*$ be a set of strings of cardinality at least N. The generic (symmetric) bilinear group model is initialized with two *random* injective mappings $\sigma, \sigma_T : \mathbb{Z}_N \to \mathcal{L}$ (which map a discrete log over $\mathbb{Z}_N$ to an associated label in $\mathcal{L}$). Here, σ is the labeling function associated with the base group and σ_T is the labeling function associated with the target group. In the generic group model, the parties have oracle access to a generic bilinear group oracle which supports the following operations:

- **Base group encoding:** On input $x \in \mathbb{Z}_N$, the oracle responds with $\sigma(x)$.
- **Base group operation:** On input labels $\ell_1, \ell_2 \in \mathcal{L}$, if both ℓ_1, ℓ_2 are in the image of σ, then the oracle replies with $\sigma(\sigma^{-1}(\ell_1) + \sigma^{-1}(\ell_2))$. Otherwise, if either ℓ_1 or ℓ_2 is not in the image of σ, then the oracle replies with $\perp$.
- **Target group encoding:** On input $x \in \mathbb{Z}_N$, the oracle responds with receives $\sigma_T(x)$.
- **Target group operation:** On input labels $\ell_1, \ell_2 \in \mathcal{L}$, if both ℓ_1, ℓ_2 are in the image of σ_T, then the oracle replies with $\sigma_T(\sigma_T^{-1}(\ell_1) + \sigma_T^{-1}(\ell_2))$. If either ℓ_1 or ℓ_2 is not in the image of σ_T, the oracle replies with $\perp$.
- **Pairing:** On input labels $\ell_1, \ell_2 \in \mathcal{L}$, if both ℓ_1, ℓ_2 are in the image of σ, then the oracle replies with $\sigma_T(\sigma^{-1}(\ell_1) \cdot \sigma^{-1}(\ell_2))$. If either ℓ_1 or ℓ_2 is not in the image of σ, then the oracle replies with $\perp$.

Notation. We will write g to denote the label for $\sigma(1)$ and g^x to denote $\sigma(x)$. Similarly, we write $e(g,g)$ to denote $\sigma_T(1)$ and $e(g,g)^x$ to denote $\sigma_T(x)$. We write $\mathbb{G}$ and $\mathbb{G}_T$ to denote the groups induced by the labeling functions σ and σ_T, respectively (i.e., $\mathbb{G} = \{\sigma(x) : x \in \mathbb{Z}_N\}$ and $\mathbb{G}_T = \{\sigma_T(x) : x \in \mathbb{Z}_N\}$). To analyze our assumptions, we follow the methodology from [BBG05, KSW08, GLWW24] by enumerating a set of sufficient conditions for security to hold unconditionally in the generic bilinear group model. We begin with a notion of independence and then recall [GLWW24, Theorem D.4] which we use for our analysis.

Definition A.2 (Independence of Polynomials)**.** Let $N = \prod_{i \in [m]} p_i$ be a positive integer that is a product of $m \geq 1$ distinct primes p_i. Let $\mathcal{P} = \{P_i\}_{i \in [k]}$ be a collections of polynomials where each $P_i \in \mathbb{Z}_N[X_1, \ldots, X_n]$ is an n-variate polynomial over $\mathbb{Z}_N$. By the Chinese Remainder Theorem, we can view each polynomial P_i as a tuple of m polynomials $P_{i,1} \in \mathbb{Z}_{p_1}[X_1, \ldots, X_n], \ldots, P_{i,m} \in \mathbb{Z}_{p_m}[X_1, \ldots, X_n]$ and where

$P_{i,j}(x_1, \ldots, x_n) = P_i(x_1, \ldots, x_n) \bmod p_j$ for all $j \in [m]$. We say that a polynomial $f \in \mathbb{Z}_N[X_1, \ldots, X_n]$ (with associated components $f_1, \ldots, f_m$) is dependent on $\mathcal{P}$ if there exists coefficients $\alpha_i \in \mathbb{Z}_N$ such that

$$\forall j \in [m] : f_j(X_1, \ldots, X_n) = \sum_{i \in [k]} \alpha_i P_{i,j}(X_1, \ldots, X_n) \bmod p_j.$$

We say f is independent on $\mathcal{P}$ if f is not dependent on $\mathcal{P}$.

Theorem A.3 (Generic Hardness in Composite-Order Groups [GLWW24, Theorem D.4])**.** *Let $N = \prod_{j \in [m]} p_j$ be a product of distinct primes where each $p_j \geq 2^\lambda$. Let $\mathcal{P} = \{P_i\}_{i \in [k]}$ and $\mathcal{Q} = \{Q_i\}_{i \in [\ell]}$ be collections of linearly independent polynomials where each $P_i, Q_i \in \mathbb{Z}_N[X_1, \ldots, X_n]$ is an n-variate polynomial over $\mathbb{Z}_N$. We assume that $P_1 = Q_1 = 1$. As in Definition A.2, we write $P_{i,j}$ and $Q_{i,j}$ to denote the action of the polynomial P_i and Q_i, respectively, modulo p_j. Let $T_0, T_1 \in \mathbb{Z}_N[X_1, \ldots, X_n]$ be two challenge polynomials. Then, for a bit $b \in \{0, 1\}$ and an adversary $\mathcal{A}$, define the following experiment in the generic bilinear group model of order N:*

- *At the beginning of the game, the challenger samples $x_1, \ldots, x_n \xleftarrow{R} \mathbb{Z}_N$. For each $i \in [k]$ and $j \in [\ell]$, it computes*

$$\begin{aligned} \ell_i &= \sigma(P_i(x_1, \ldots, x_n)) \\ \ell'_j &= \sigma_T(Q_j(x_1, \ldots, x_n)) \\ \tau_0 &= \sigma(T_0(x_1, \ldots, x_n)) \\ \tau_1 &= \sigma(T_1(x_1, \ldots, x_n)). \end{aligned}$$

- *The challenger gives $\left(N, \{\ell_i\}_{i \in [k]}, \{\ell'_j\}_{j \in [\ell]}, \tau_b\right)$ to the adversary.*
- *The adversary outputs a bit $b' \in \{0, 1\}$, which is the output of the experiment.*

For an adversary $\mathcal{A}$, define its advantage $\delta_\mathcal{A}$ to be

$$\delta_\mathcal{A} := |\Pr[b' = 1 : b = 0] - \Pr[b' = 1 : b = 1]|$$

in the above distinguishing experiment. Let $\mathcal{P}^2 := \{P_i P_j : i, j \in [k]\}$. For a bit $b \in \{0, 1\}$, let

$$\mathcal{S}^{(b)} := \mathcal{P}^2 \cup \mathcal{Q} \cup \{T_b P_i : i \in [k]\}.$$

For an index $i \in [k]$, define $\mathcal{S}_i^{(b)} := \mathcal{S}^{(b)} \setminus \{T_b P_i\}$. Suppose now the following properties hold:

- *The total degree of P_i, Q_j, T_0, T_1 is at most d.*
- *For all $i \in [k]$ and $b \in \{0, 1\}$, the polynomial T_b is independent of $\mathcal{P}$.*
- *For all $i \in [k]$, if $T_0 P_i \neq T_1 P_i$, then for all $b \in \{0, 1\}$, the polynomial $T_b P_i$ is independent of $\mathcal{S}_i^{(b)}$.*
- *For all $b \in \{0, 1\}$, the polynomial T_b^2 is independent of $\mathcal{S}^{(b)}$.*

Then, for all adversaries $\mathcal{A}$ making at most q queries to the generic group oracle, if $\mathcal{A}$ has distinguishing advantage $\delta_\mathcal{A}$ in the above distinguishing experiment, then there is an algorithm that runs in time polynomial in λ and the running time of $\mathcal{A}$ that outputs a non-trivial factor of N with success probability at least $\delta_\mathcal{A} - O((q + k + \ell)^2 d / 2^\lambda)$.

Theorem A.4 (Generic Hardness of Assumption 3.2)**.** *If factoring a product of two primes (each of size 2^λ) is computationally hard, then for all polynomials $\ell = \ell(\lambda)$, the ℓ-subgroup decision exponent assumption (Assumption 3.2) holds in the generic group model of order N where N is a product of two primes (each of size 2^λ).*

Proof. The components given out in Assumption 3.2 can be expressed as polynomials over the formal variables $\alpha, r \in \mathbb{Z}_N$. The components in the assumption correspond to the polynomials $P_i(\alpha, r) := [P_{i,1}(\alpha, r), P_{i,2}(\alpha, r)] = [\alpha^i r, 0]$ for all $i \in [\ell]$. Let $\mathcal{P} = \{P_i\}_{i\in[\ell]}$. The challenge polynomials are then

$$T_0(\alpha, r) := [r, 0] \text{ and } T_1(\alpha, r) := [r, r]$$

We now consider the conditions in Theorem A.3:

- The total degree of the polynomials appearing in the challenge and the assumption is at most $\ell + 1$.
- By definition, T_0 and T_1 are monomials in r only whereas each polynomial P_i is a non-zero monomial in both α and r. Thus, any non-zero linear combination of $\{P_i\}_{i\in[\ell]}$ would yield either a constant polynomial or one that is non-zero in α. Thus T_0 and T_1 are independent on the set of polynomials $\{P_i\}_{i\in[\ell]}$.
- By construction, T_0 and T_1 are distributed identically in the $\mathbb{G}_p$ subgroup and only differ in the $\mathbb{G}_q$ subgroup. By construction of P_i, this means $T_0 P_i = T_1 P_i$ for all $i \in [\ell]$.
- For the final condition, let $\mathcal{S}^{(b)} = \mathcal{P}^2 \cup \{T_b P_i : i \in [\ell]\}$. First, $T_0^2(\alpha, r) = [r^2, 0]$ and $T_1^2(\alpha, r) = [r^2, r^2]$. Notably, T_0 and T_1 are monomials in r only. For all $i, j \in [\ell]$, the polynomial $P_i P_j$ is a monomial in *both* α and r; similarly, $T_b P_i$ is a monomial in α and r. So, any non-zero linear combination of the polynomials in $\mathcal{S}^{(b)}$ would lead to a polynomial that is either a constant polynomial or one that is non-zero in α. Thus, T_b^2 is independent of $\mathcal{S}^{(b)}$.

Since $\ell = \text{poly}(\lambda)$ and the number of terms given out in the assumption is also ℓ, the claim now follows from Theorem A.3. □