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

by Noel Elias

A honors thesis submitted for the degree of Bachelor of Science in Computer Science

Department of Computer Science
The University of Texas at Austin
September 2024

Advisor: Dr. David Wu
Second Reader: Dr. Brent Waters
Honors Committee Reader: Dr. Scott Aaronson

Disclaimer. This thesis is part of a broader project, jointly conducted with Dr. David Wu and Dr. Binyi Chen. It has been submitted for publication and is intended exclusively for evaluation as part of my application through this private link. I kindly request that this work not be shared publicly or used without appropriate attribution, including its ideas and findings, as it reflects my original research. Thank you for your understanding!

Acknowledgements

First and foremost, I would like to thank God for giving me the wisdom and fortitude to complete this thesis. I know that everything that I have done so far and will do in the future is only by His grace.

Next, I would like to thank my advisor, Dr. David Wu. Dr. Wu captivated my interests in cryptography when I was a wide-eyed freshman in the Computer Science department. I was immediately drawn to his passion and depth for the subject. Throughout this project and my time in his classes, he always took time to answer my questions and provide feedback for my ideas, no matter how silly or naive they might be. Through Dr. Wu's mentorship, not only have I have been able to explore the world of theoretical cryptography but also learn the fundamentals of research. This thesis would not be possible without his guidance.

In addition, I would like to thank Dr. Brent Waters and Dr. Scott Aaronson for taking the time out of their busy schedules to let me defend my thesis in the middle of the semester! Their feedback and presence for my defense not only helped me improve my work, but also served as an inspiration for a young researcher like myself. I also want to thank Dr. Binyi Chen who is a joint-collaborator in this work.

Last but not least, I would like to thank my family, especially my parents, for all their support.

Abstract

Batch arguments (BARGs) provide a method for a prover to convince a verifier of ℓ NP statements more efficiently than verifying each NP instance separately. At present, the state-of-the-art pairing-based non-interactive batch argument introduced by Waters and Wu (Crypto 2022) requires a common reference string (CRS) that is quadratic with respect to ℓ, the number of instances. This poses an efficiency bottleneck when implementing such protocols with a large number of instances.

To solve this problem, we introduce a construction for a BARG with a linear-sized CRS utilizing q-type pairing-based assumptions for composite-order groups. Specifically, we propose utilizing a polynomial-based technique to compress the CRS size while preserving the necessary BARG properties. Lastly, we provide security analysis for the newly introduced decisional assumptions under the generic group model. The resulting scheme is not only linear in the CRS but also contains succinct proof sizes, polynomial prover time, and well-studied verification techniques.

Contents

1 Introduction

Imagine a prover has a collection of NP statements $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\ell$. The prover seeks to convince a verifier of all these statements; more precisely, the prover wants to show that $\mathbf{x}_i \in \mathcal{L}$ for all $i \in [\ell]$ (where $\mathcal{L}$ is the NP language in question). A trivial solution is to have the prover send the witness $\mathbf{x}$ for each NP statement $\mathbf{x}$ for i $\in [\ell]$. The verifier can then check each of these witnesses and statement pairs $(\mathbf{x}_i, \mathbf{w}_i)$ independently. Now, suppose we want to do this more efficiently by having the size of the proof grow sub-linear to ℓ, the number of batched instances. More concretely, how can we convince the verifier that $\mathbf{x}_i \in \mathcal{L}$ for all $i \in [\ell]$ with a proof size that is $O(\ell)$?

Batch Arguments A non-interactive *batch argument* (BARG) can be utilized to amortize the cost of NP verification across many instances. Particularly, a BARG for NP allows for a prover to generate a proof π that verifies that all statements $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\ell \in \mathcal{L}$. Such a proof π scales sub-linear with respect to the number of instances ℓ. We focus on BARGs for NP using the common reference string (CRS) model, where a trusted setup algorithm samples the common reference string crs. This crs is utilized for proof construction as well as proof verification. In addition, we study the settings where the BARG proof is *publicly-verifiable*, *non-interactive*, and *sound*. Specifically, we work with the soundness setting where no computationally bounded adversary can convince the verifier of a set of batch NP instances where for some index $i^* \in [\ell]$, $x_{i^*} \notin \mathcal{L}$. In other words, we work with batch arguments.

1.1 Literature Review

In recent times, there have been many constructions for BARGs [KVZ21, CJJ21b, CJJ21a, HJKS22, WW22, DGKV22, GSWW22, CGJ+23, KLVW23, BBK+23, GLWW24]. These constructions are built from a variety of standard falsifiable assumptions, including but not limited to LWE, DLIN, and sub-exponential DDH. In particular, the introduction of the [WW22] construction using DLIN marked the first BARG for NP using standard assumptions over bilinear groups that avoids correlation-intractability [CGH04]. However, one of the shortcomings of this seminal work of [WW22] and the subsequent work of [GLWW24] is that the size of the CRS for both constructions is not linear with respect to ℓ, the number of instances. In fact, [WW22] utilized black-box notions of cryptography to dramatically reduce the CRS size, and [GLWW24] utilizes an extremely complex q-type assumption for security analysis utilizing progression-free sets [ET36, Beh46, Elk10].

The novelty of this work lies in the proposed schemes of BARGs for NP from bilinear maps. These schemes only require a *linear*-sized CRS with respect to ℓ, the number of instances. To do this, we provide a direct approach utilizing polynomials [GGPR13, Gro16] that not only reduce the CRS size and provide efficient verification but also do not need black-box use of cryptography. In addition, we demonstrate that our newly introduced falsifiable assumption holds under the generic group model for composite-order bilinear maps.

1.2 Our Contributions

In this work, we derive simple and more efficient schemes for constructions for constructing BARGs using bilinear maps. Our main contribution is a BARG with a linear-sized CRS with respect to ℓ, the number of instances. We can capture this informally in the theorem statement before:

Scheme	CRS Size	Assumption Size
[WW22]	$O(\ell^2)$	Constant
[GLWW24]	$O(\ell^{1+o(1)})$	q-type
This work	$O(\ell)$	q-type

Table 1: Summary of pairing-based BARG constructions. For each construction, we analyze the number of group elements in the common reference string (crs) with regards to ℓ the number of batched instances and the assumption size.

Theorem 1.1 (Informal)**.** *Under the Assumption 3.3 for composite-order pairing groups, there exists a publicly-verifiable non-interactive BARG for Boolean circuit satisfiability with proof size* $\mathsf{poly}(\lambda, |C|)$*, verification complexity* $\mathsf{poly}(\lambda, h_{in}) + \ell \log^2(\ell)$*, and CRS size of* $2\ell \cdot \mathsf{poly}(\lambda)$ *where* λ *is the security parameter,* $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$ *is a boolean circuit,* h_{in} *is the statement size, and* ℓ *is the number of instances. The BARG satisfies somewhere extractability (Definition 2.4).*

Batch Verification Similar to the direct and low-tech approach introduced by [WW22], we utilize a "commit-and-prove" strategy used for pairing based non-interactive proofs [GOS06]. To quickly summarize this process, we can recall that the prover first provides succinct (sub-linear with respect to ℓ) commitments of the ℓ bits of the circuit's wire values across the ℓ batched instances. Then, at each gate, the prover provides a proof that the committed wire values are consistent with the expected gate operation, utilizing the succinct commitment for each wire. We provide a technical overview of this process in Section 1.3.

1.3 Technical Overview

In this work, we focus on developing BARGs for the Boolean circuit satisfiability language. Take any $\lambda, \ell, n \in \mathbb{N}$ and consider a Boolean circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$ of at most size n. We say that $(C, \mathbf{x}_1, \ldots, \mathbf{x}_\ell)$ is "true" if for all $i \in [\ell]$, there exists a witness $\mathbf{w}_i$ such that $C(\mathbf{x}_i, \mathbf{w}_i)$ evaluates to 1.

High Level Idea To construct a BARG proof pf for the tuple $(C, \mathbf{x}_1, \ldots, x_\ell)$ as well as the witnesses $(\mathbf{w}_1, \ldots, \mathbf{w}_\ell)$, we can follow the "commit-and-prove" paradigm described below:

- **Statement validity:** For every $j \in [h_{in}]$, let $u_{1,j}, u_{2,j}, \ldots, u_{\ell,j} \in \{0,1\}$ be the j-th bit of the statement across ℓ circuit assignments. The prover constructs a vector commitment, σ_j of each tuple $(u_{1,j}, u_{2,j}, \ldots, u_{\ell,j})$ for all $j \in [m]$. This commitment is created utilizing a deterministic function of the inputted vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$. As a result, the verifier can later check that these commitments are in fact commitments to the inputted bits $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$.

- **Wire validity:** Let $m \in \mathbb{N}$ denote the number of wires. For every $j \in [m]$, the prover computes a proof π_j that σ_j is in fact a commitment to a $0/1$ vector. More formally, it shows that $u_{i,j} \in 0, 1$ for all $i \in [\ell]$ and $j \in [m]$. The verifier utilizes this proof π_j to clarify that σ_j is in fact a binary vector.

- **Gate validity:** We describe Boolean circuits in terms of NAND gates. Let $n \in \mathbb{N}$ denote the number of gates per circuit C. Let every $k \in [n]$, let $G_k = (j_1, j_2, j_3)$ be the k-th NAND gate of C where j_1, j_2 are the indices of the input wires and j_3 is the index of the output wire of the gate. The prover computes a proof Σ_k for every $k \in [n]$ that demonstrates that the values of the commitments $\sigma_{j_1}, \sigma_{j_2}, \sigma_{j_3}$ are consistent with the NAND gate operation. The verifier uses Σ_k to check that for every $k \in [n]$ such that the gate $G_k = (j_1, j_2, j_3)$ exists, σ_{j_3} is a commitment of the NAND of the tuples $(u_{1,j_1}, \ldots, u_{\ell,j_1})$ and $(u_{1,j_2}, \ldots, u_{\ell,j_2})$.

- **Output validity:** Finally, let m be the index of the output wire of the circuit C. Then, the verifier checks that the σ_m is a commitment of the constant 1 vector.

The overall proof is outputted as $\mathsf{pf} := (\{\sigma_j, \pi_j\}_{j\in[m]}, \{\Sigma_k\}_{k\in[n]})$. To ensure efficiency requirements of the BARG (proof size and verification complexity is sub-linear to ℓ), we require the following:

- The commitment and proof, σ_j, π_j, must be succinct or $\mathsf{poly}(\lambda, n)$.
- The proof Σ must be succinct: $\mathsf{poly}(\lambda, m)$.
- The verification checks must run in time $\mathsf{poly}(\lambda, n, m)$.

1.3.1 Composite-Order Pairing Groups Construction

To illustrate the core insight of our construction, we can first describe it in the symmetric composite-order group setting utilizing the nuanced Assumption 3.3 for soundness.

Composite-order paring groups A symmetric composite-order pairing group contains two cyclic groups $\mathbb{G}, \mathbb{G}_T$ of order $N = pq$ where p, q are primes numbers and an efficiently-computable, non-degenerate bilinear map or "pairing" $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Suppose g is a generator of $\mathbb{G}$, then by the Chinese Remainder Theorem, if $\mathbb{G}_p$ is a p-order subgroup generated by $g_p = g^q$ and $\mathbb{G}_q$ is a q-order subgroup generated by $g_q = g^p$, we can write $\mathbb{G} \cong \mathbb{G}_p \times \mathbb{G}_q$. Additionally, it holds that for all $a, b \in \mathbb{Z}_N$, $e(g^a, g^b) = e(g, g)^{ab}$ and $e(g_p, g_q) = 1_{\mathbb{G}_T}$ as the subgroups are orthogonal to each other.

CRS generation We start by constructing the common reference string. The CRS consists of $\ell + 1$ group elements that are computed as $A_i = g_p^{\alpha^i}$ for some $\alpha \xleftarrow{R} \mathbb{Z}_N$. In addition, a polynomial, $Z(X)$ is interpolated such that Z vanishes for all $i \in [\ell]$ or in other words $Z(i) = 0$. We commit to this polynomial by publishing $g_p^{Z(\alpha)}$ in the CRS. Note that in practice, we replace the basis $[\ell]$ with a set outputted by a interpolation set generator (Definition 3.2) rather than all $i \in [\ell]$. The CRS is thus defined as:

$$\mathsf{pp} := ((A_i)_{i=0}^{\ell}, g_p^{Z(\alpha)}) \tag{1.1}$$

Statement validity To create a commitment to a vector $(u_{1,j}, \ldots, u_{\ell,j})$ where j is the j-th bit of the statement for the Circuit C, we first interpolate a polynomial φ_j such that $\varphi_j(\alpha^i) = u_{i,j}$ for all $i \in [\ell]$. Then we set the commitment to $\sigma_j = g_p^{\varphi_j(\alpha)}$ utilizing the terms provided in the CRS.

Wire validity Next, we provide a method for the prover to show that given a vector $(u_{1,j}, \ldots, u_{\ell,j})$ of the values of wire j across ℓ instances, each $u_{i,j} \in \{0, 1\}$. To show this, we can equivalently prove that for all $i \in [\ell]$, $u_{i,j}^2 = u_{i,j}$. Note that the commitment currently encodes $\varphi_j(\alpha)$ in the exponents. Let $\psi_j(X) = \varphi_j^2(X) - \varphi_j(X)$. If $u_{i,j} \in \{0, 1\}$ for all $i \in [\ell]$, then $\varphi_j^2(i) = \varphi_j(i)$ and $\psi_j(i) = 0$. In this case, the polynomial $Z(X)$ divides $\psi_j(X)$. Let $Q_j(X) = \frac{\psi_j(X)}{Z(X)}$ be the quotient polynomial. The prover outputs the opening $\pi_j = g_p^{Q_j(\alpha)}$. The verification relation will essentially check that $\psi_j(\alpha) = Z(\alpha)Q_j(\alpha)$. If this holds, then we can conclude that these polynomials are equal by the Schwartz-Zippel Lemma [Sch80, Zip79]. This boils down to the following verification check

$$e(\sigma_j, \sigma_j) = e(D, \pi_j)e(A_0, \sigma_j) \tag{1.2}$$

Thus, we are able to demonstrate that if σ_j is a commitment of a binary vector, then the prover can provide an opening proof π_j that satisfies the verification relation.

Gate validity For gate validity, we utilize a similar approach as the wire validity check. Particularly, given the gate $G_k = (j_1, j_2, j_3)$ where $k \in [n]$, the prover wants to show that σ_{j_3} is a commitment of the NAND of the tuples $(u_{1,j_1}, \dots, u_{\ell,j_1})$ and $(u_{1,j_2}, \dots, u_{\ell,j_2})$. To do this, we can equivalently prove that for all $k \in [n]$ and the corresponding gate $G_k = (j_1, j_2, j_3)$, it must be the case that for all $i \in [\ell]$, $u_{i,j_1}u_{i,j_2} = u_{i,j_3}$. We set $P_k(X) = \frac{\varphi_{j_1}\varphi_{j_2}(X) - \varphi_{j_3}(X)}{Z(X)}$ to be the quotient polynomial. The prover outputs the opening $\Sigma_k = g_p^{P_k(\alpha)}$. The verification relation will essentially check that $\Sigma_k(\alpha) = Z(\alpha)P_k(\alpha)$. This boils down to the following verification check

$$e(\sigma_{j_1}, \sigma_{j_2})e(A_0, \sigma_{j_3})e(D, \Sigma_k) = e(A_0, A_0) \tag{1.3}$$

Output validity Finally, we can check that the last output wire (m) of the circuit across all instances $i \in [l]$, evaluates to 1 by checking that σ_m is the constant one polynomial. This can be done by checking if $\sigma_m = A_0$.

Soundness To argue soundness of our BARG, we utilize the dual-mode approach found in [WW22, CJJ21b, CJJ21a]. Particularly, we can sample the CRS normally as described above and output a trapdoor common reference string, crs$*$ based on some input index $i* \in [\ell]$. We require that in trapdoor mode, the scheme is statistically sound for instance $i*$ where for crs$*$ there does not exist any proof pf that can convince the verifier with high probability that $\mathbf{x}_{i*}$ is false. However, it may be the case that there might be some other $\mathbf{x}_i$ which is false as long as $i \neq i*$. This is known as *somewhere statistical soundness.*

To create the trapdoor mode CRS for some index $i* \in [\ell]$, we replace $A_i = g_p^{\alpha^i}$ with $A_i = g_p^{\alpha^i} g_q^{i*^i}$ for all $i \in [\ell]$. We show that the resulting trapdoor CRS is computationally indistinguishable from the normal CRS under the generic composite-order bilinear group model (Theorem 3.13). Now we can examine the verification checks:

- **Statement validity:** For every $j \in [h_{in}]$, we write σ_j as $\sigma_j = g_p^{\beta_j} g_q^{\gamma_j}$ for some $\beta_j \in \mathbb{Z}_p$ and $\gamma_j \in \mathbb{Z}_q$. If this relation satisfies the statement validity check, it indicates that we compute $\sigma_j = (g_p^{\varphi(\alpha)} g_q^{\varphi(i*)})$. If we project this relation to the order-q subgroup of $\mathbb{G}_T$, it is clear that $\gamma_j = \varphi(i*) = \mathbf{x}_{i*,j}$.

- **Wire validity:** Now consider the wire validity checks. For every $j \in [m]$, we know that Equation 1.2 holds in the target group $\mathbb{G}_T$. Now consider the projection in the order-q subgroup of $\mathbb{G}_T$. As the D term of the CRS is also modified to be $D = g_p^{Z_S(\alpha)} g_q^{Z_S(s_{i*})}$, the above equation implies that $\gamma_j^2 = 0 + \gamma_j (\bmod q)$, or equivalent, $\gamma_j \in \{0, 1\}$.

- **Gate validity:** Similar to the wire validty checks, we can consider the projection of Equation 1.3 in the order-q subgroup of $\mathbb{G}_T$. We obtain the relation $\gamma_{j_1}\gamma_{j_2} + \gamma_{j_3} + 0 = 1$, or equivalently, $\gamma_{j_3} = 1 - \gamma_{j_1}\gamma_{j_2} = \mathsf{NAND}(\gamma_{j_1}, \gamma_{j_2})$.

- **Output validity** It is clear that $\sigma_m = A_0 = g_p g_q$, we have that $\gamma_m = 1$.

The above argument shows that if all the validity checks pass, then we can extract a witness $\mathbf{w}_{i*}$ for instance $i*$ and the statistical soundness of $\mathbf{x}_{i*}$ holds. We provide the detailed construction and security analysis in Section 3.

2 Preliminaries

2.1 Notation

In this section, we recall the notion of a batch argument for NP. Our description is taken almost exactly verbatim from Section 2.1 of [WW22] and Section 7.1 of [GLWW24].

For a positive integer n, we write n to denote the set $1, \dots, n$. For a positive integer p, we denote $\mathbb{Z}_p$ as ring of integers modulo p. We utilize bold-face letter to denote vectors and matrices and non-bold-face letter to denote their components: $\mathbf{x} = (x_1, x_2, \dots, x_9)$. Let λ denote the security parameters, then we write $\mathsf{poly}(\lambda)$ as the function $O(\lambda^c)$ for some $c \in \mathbb{N}$ and $\mathsf{negl}(\lambda)$ as the function $o(\lambda^{-c})$. An event occurs with overwhelming probability if its complement occurs with negligible probability. An algorithm is considered efficient if it runs in probabilistic polynomial time with respect to its input length. Lastly, we say that two distribution are computationally indistinguishable if no efficient algorithm can differentiate them with non-negligible probability.

In this work, we focus on developing BARGs for the NP-complete Boolean circuit satisfiability language. Take any Boolean circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$. We let C be a circuit with m wires, h_{in} bits of the input statement, and n NAND gates. Lastly, we associate the bits of the witness of the circuit C with wires $h_{\mathsf{in}} + 1, \dots, h_{\mathsf{in}} + h_{\mathsf{w}}$ and let t be the last output wire. Then, it is clear by construction that $t \leq h_{\mathsf{in}} + h_{\mathsf{w}} + n$. We now define the (batch) circuit satisfiability language considered in this work:

2.2 Construction Properties

Definition 2.1 (Circuit Satisfiability)**.** We define $\mathcal{L}_{CSAT} = \{(C, \mathbf{x}) | \exists \mathbf{w} \in \{0,1\}^{h_{\mathsf{in}}} : C(\mathbf{x}, \mathbf{w}) = 1\}$ to be the language of Boolean circuit satisfiability, where $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$ is a Boolean circuit and $\mathbf{x} \in \{0,1\}^{h_{\mathsf{in}}}$ is a statement. For a positive integer $\ell \in \mathbb{N}$, we define the *batch circuit satisfiability* language $\mathcal{L}_{CSAT,\ell}$ as follows:

$$\mathcal{L}_{CSAT,\ell} = \{(C, \mathbf{x}_1, \dots, \mathbf{x}_\ell) | \forall \mathbf{w}_i \in \{0,1\}^{h_{\mathsf{in}}} : C(\mathbf{x}_i, \mathbf{w}_i) = 1\},$$

where $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$ is a Boolean circuit and $\mathbf{x}_1, \dots, \mathbf{x}_\ell \in \{0,1\}^{h_{\mathsf{in}}}$.

Definition 2.2 (Batch Argument for Circuit Satisfiability)**.** A non-interactive batch argument (BARG) for circuit satisfiability is a tuple of three efficient algorithms $\prod_{BARG} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following properties:

- $\mathsf{Setup}(1^\lambda, 1^\ell, 1^n) \to \mathsf{pp}$: On input the security parameter $\lambda \in \mathbb{N}$, the number of instance $\ell \in \mathbb{N}$, and a bound on the circuit size $n \in \mathbb{N}$, the setup algorithms outputs a common reference string pp.

- $\mathsf{Prove}(\mathsf{pp}, C, (\mathbf{x}_1, \dots, \mathbf{x}_\ell), (\mathbf{w}_1, \dots, \mathbf{w}_\ell)) \to \mathsf{pf}$: On inputs the common reference string pp, a Boolean circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$, statements $\mathbf{x}_1, \dots, \mathbf{x}_\ell \in \{0,1\}^{h_{\mathsf{in}}}$, and witnesses $\mathbf{w}_1, \dots, \mathbf{w}_\ell \in \{0,1\}^{h_{\mathsf{w}}}$, the prove algorithms outputs a proof pf.

- $\mathsf{Verify}(\mathsf{pp}, C, (\mathbf{x}_1, \dots, \mathbf{x}_\ell), \mathsf{pf}) \to b$: On input the common reference string pp, the Boolean circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$, statements $\mathbf{x}_1, \dots, \mathbf{x}_\ell \in \{0,1\}^{h_{\mathsf{in}}}$, and a proof pf, the verification algorithm outputs a bit $b \in \{0,1\}$.

Definition 2.3 (Completeness)**.** A BARG $\prod_{BARG}$ = (Setup, Prove, Verify) is complete if for all $\lambda, \ell, n \in \mathbb{N}$, all Boolean circuits $C : \{0,1\}^{h_{\text{in}}} \times \{0,1\}^{h_{\text{w}}} \longrightarrow \{0,1\}$ of size at most n, and all witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^{h_{\text{w}}}$ where $C(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all $i \in [\ell]$.

$$Pr\left[\mathsf{Verify}(\mathsf{pp}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \mathsf{pf}) = 1 : \mathsf{pp} \longleftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^m); \mathsf{pf} \longleftarrow \mathsf{Prove}(\mathsf{pp}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), (\mathbf{w}_1, \ldots, \mathbf{w}_\ell)) :\right] = 1.$$

2.3 Security Definitions

Definition 2.4 (Somewhere Argument of Knowledge [CJJ21a])**.** A BARG $\prod_{BARG}$ = (Setup, Prove, Verify) is a somewhere argument of knowledge if there exists a pair of efficient algorithms (TrapSetup, Extract) with the following properties:

- $\mathsf{TrapSetup}(1^\lambda, 1^\ell, 1^n, \mathsf{idx}) \longrightarrow (\mathsf{pp}*, \mathsf{td})$: On input the security parameter $\lambda \in \mathbb{N}$, the number of instance $\ell \in \mathbb{N}$, and a bound on the circuit size $n \in \mathbb{N}$, and an index $\mathsf{idx} \in [\ell]$, the trapdoor setup algorithm outputs a common reference string pp∗ and an extraction trapdoor td.
- $\mathsf{Extract}(\mathsf{td}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell, \mathsf{pf}) \longrightarrow \mathbf{u}*$: On input the trapdoor td, statements $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^{h_{\text{in}}}$, and a proof pf, the extraction algorithms outputs a witness $\mathbf{u}* \in \{0,1\}^{witlen}$. This algorithm is deterministic.

We require (TrapSetup, Extract) to satisfy the following two properties:

- **CRS indistinguishability:** For a bit $b \in \{0,1\}$, and an adversary **A**, we define the CRS indistinguishability game as follows:
  1. On input the security parameter 1^λ, algorithm **A** outputs the number of statements 1^ℓ, the size of the circuit 1^n, and an index $\mathsf{idx} \in [\ell]$.
  2. If $b = 0$, the challenger gives $\mathsf{pp} \longleftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^m)$ to $\mathcal{A}$. If $b = 1$, the challenger gives $\mathsf{pp}* \longleftarrow \mathsf{TrapSetup}(1^\lambda, 1^\ell, 1^n, \mathsf{idx})$ to $\mathcal{A}$.
  3. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the outputs of the experiment.

  Then, $\prod_{BARG}$ satisfies CRS indistinguishability if for every efficient adversary $\mathcal{A}$, 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 above CRS indistinguishability game.

- **Somewhere extractable in trapdoor mode:** For an adversary $\mathcal{A}$, we define the somewhere extractable security game as follows:
  - On input the security parameter 1^λ, algorithm **A** starts by outputting the number of statements 1^ℓ, the size of the circuit 1^n, and an index $\mathsf{idx} \in [\ell]$.
  - The challenger samples $(\mathsf{pp}*, \mathsf{td}) \longleftarrow \mathsf{TrapSetup}(1^\lambda, 1^\ell, 1^n, \mathsf{idx})$ and gives pp∗ to $\mathcal{A}$.
  - Algorithm $\mathcal{A}$ outputs a Boolean circuit $C : \{0,1\}^{h_{\text{in}}} \times \{0,1\}^{h_{\text{w}}} \longrightarrow \{0,1\}$ of size at most n, statements $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^{h_{\text{in}}}$, and a proof pf. Let $\mathbf{u}* \longleftarrow \mathsf{Extract}(\mathsf{td}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell, \mathsf{pf})$
  - The output of the game is $b = 1$ if $\mathsf{Verify}(\mathsf{pp}*, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell, \mathsf{pf}) = 1$ and $C(\mathbf{x}_{\mathsf{idx}}, uv*) \neq 1$. Otherwise, the outputs is $b = 0$.

  The $\prod_{BARG}$ is somewhere extractable in trapdoor mode if for every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $Pr[b = 1] = \mathsf{negl}(\lambda)$ in the above somewhere extractable game.

2.4 Efficiency Definitions

Definition 2.5 (Succinctness)**.** A BARG $\prod_{BARG}$ = (Setup, Prove, Verify) is succinct if there exists a fixed poly-nomial $\text{poly}(\cdot, \cdot, \cdot)$ such that for all $\lambda, \ell, n \in \mathbb{N}$, all pp is the support of $\text{Setup}(1^\lambda, 1^\ell, 1^m)$, and all Boolean circuits $C : \{0,1\}^{h_{\text{in}}} \times \{0,1\}^{h_{\text{w}}} \longrightarrow \{0,1\}$ of size at most n, the following properties hold:

- **Succinct proofs:** The proof pf outputted by $\text{Prove}(\text{pp}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), (\mathbf{w}_1, \ldots, \mathbf{w}_\ell))$ satisfied $|\text{pf}| \leq \text{poly}(\lambda, log(\ell), n)$.
- **Succinct CRS:** $|\text{pp}| \leq \text{poly}(\lambda, \ell, h_{\text{in}}) + \text{poly}(\lambda, log(\ell), n)$.
- **Succinct verification:** The verification algorithms runs in time $\text{poly}(\lambda, \ell, h_{\text{in}}) + \text{poly}(\lambda, log(\ell), n)$.

2.5 Generic Group Model

Definition 2.6 (Independence of Polynomials)**.** Adapted from Definition D.2 [GLWW24]. Let N 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 polynomials over $\mathbb{Z}_N$. By the Chinese Remainder Theorem, we can view each polynomial P_i as defining a tuple of m polynomials $P_{i,1} \in \mathbb{Z}_{p_1}[X_1, \ldots, X_n], \ldots, P_{i,m} \in \mathbb{Z}_{p_i}[X_1, \ldots, X_n]$, 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 $f_1, \ldots, f_m$ where $f_j = f \bmod p_i$) is dependent on $\mathcal{P}$ if there exists coefficients $\alpha_i \in \mathbb{Z}_n$ such that

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

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

3 BARG for NP with a Linear-Size CRS from Composite Order Groups

3.1 Construction

Definition 3.1 (Composite-Order Group Generation)**.** CompGen is an efficient algorithm that takes as input a security parameter $\lambda \in \mathbb{N}$ and a number $\ell \in \mathbb{N}$, and outputs $\text{gp} := (p, q, \mathbb{G}, \mathbb{G}_T, e, g_p, g_q)$, where p, q are distinct primes, $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of order $N = pq$, $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ is an efficient bilinear map, and $g_p, g_q \in \mathbb{G}$ are *uniformly random generators* of order p and q, respectively.

Definition 3.2 (Interpolation set generator)**.** An interpolation set generator algorithm $\text{SGen}(1^\lambda, 1^\ell, N)$ takes as input a security parameter $\lambda \in \mathbb{N}$, and $\ell, N \in \mathbb{N}$, outputs a set $S = \{s_i\}_{i=1}^{\ell} \subseteq \mathbb{Z}_N$ such that for all $i, j \in [\ell]$, $i \neq j$, the value $s_i - s_j$ is invertible over $\mathbb{Z}_N$. We note that for any $a_1, a_2, \ldots, a_\ell \in \mathbb{Z}_N$, there exists a unique polynomial $f \in \mathbb{Z}_N[X]$ with degree less than ℓ, such that $f(s_i) = a_i$ for all $i \in [\ell]$. At a high level, we can accomplish this by sampling random elements in $\mathbb{Z}_N^\times$ and ensuring that no two elements are congruent modulo N.

Definition 3.3 (q-type Decision Assumption in Composite Order Groups)**.** Let $\ell = \ell(\lambda) \in \mathbb{N}$. The decision assumption holds for CompGen and SGen if for every $i \in [\ell]$, every PPT adversary $\mathcal{A}$, and every $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{A}(N, \mathbb{G}, \mathbb{G}_T, e, S, \text{srs}_0] - \Pr[\mathcal{A}(N, \mathbb{G}, \mathbb{G}_T, e, S, \text{srs}_1] | \leq \text{negl}(\lambda),$$

where $(p, q, \mathbb{G}, \mathbb{G}_T, e, g_p, g_q) \leftarrow \mathsf{CompGen}(1^\lambda, 1^\ell)$, $N = pq$, $S = \{s_i\}_{i=1}^{\ell} \leftarrow \mathsf{SGen}(1^\lambda, 1^\ell, N)$, $\alpha \xleftarrow{R} \mathbb{Z}_N$ and

$$\mathsf{srs}_0 := \left(g_p, g_p^{\alpha}, g_p^{\alpha^2}, \ldots, g_p^{\alpha^\ell}\right), \qquad \mathsf{srs}_1 := \left(g_p g_q, g_p^{\alpha} g_q^{s_i}, g_p^{\alpha^2} g_q^{s_i^2}, \ldots, g_p^{\alpha^\ell} g_q^{s_i^\ell}\right).$$

Construction 3.4. Let $\ell \in \mathbb{N}$ denote the number of instances and $n \in \mathbb{N}$ denote the number of gates per circuit. We construct a BARG for circuit satisfiability as follows.

- $\mathsf{Setup}(1^\lambda, 1^\ell, 1^n)$: The setup algorithm proceeds as follows:
  1. Run $(p, q, \mathbb{G}, \mathbb{G}_T, e, g_p, g_q) \leftarrow \mathsf{CompGen}(1^\lambda, 1^\ell)$. Set $N = pq$, set $S = \{s_i\}_{i=1}^{\ell} \leftarrow \mathsf{SGen}(1^\lambda, 1^\ell, N)$.
  2. Sample $\alpha \xleftarrow{R} \mathbb{Z}_N$. For every $i \in [0, \ell]$, set $A_i := g_p^{\alpha^i}$. Set $D := g_p^{Z_S(\alpha)}$ where the vanishing polynomial $Z_S \in \mathbb{Z}_N[X]$ is $Z_S(X) := \prod_{a \in S}(X - a)$. We note that D can be computed given $(A_i)_i$ since $\deg(Z_S) \leq \ell$.
  3. Output $\mathsf{pp} := (N, \mathbb{G}, \mathbb{G}_T, e, S, (A_i)_{i=0}^{\ell}, D)$.

- $\mathsf{Prove}(\mathsf{pp}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), (\mathbf{w}_1, \ldots, \mathbf{w}_\ell))$: On input the parameters $\mathsf{pp} := (N, \mathbb{G}, \mathbb{G}_T, e, S, (A_i)_{i=0}^{\ell}, D)$ where $S = \{s_i\}_{i=1}^{\ell} \subseteq \mathbb{Z}_N$ and $A_0 = g_p$, the circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \longrightarrow \{0,1\}$ with m wires and n gates, instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^{h_{\mathsf{in}}}$, witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^{h_{\mathsf{w}}}$, the prover does follows:
  - **Wire encoding:** For every $j \in [m]$, let $u_{1,j}, u_{2,j}, \ldots, u_{\ell,j} \in \{0,1\}$ be the j-th wire values of the ℓ circuit assignments. Interpolate the polynomial $\varphi_j \in \mathbb{Z}_N[X]$ where $\varphi_j(s_i) \bmod N = u_{i,j}$ for all $i \in [\ell]$. Compute commitment $\sigma_j := g_p^{\varphi_j(\alpha)}$ given the CRS $(A_i = g_p^{\alpha^i})_{i=0}^{\ell}$.
  - **Wire validity proof:** For every $j \in [m]$, let φ_j be the polynomial defined above. Define polynomial
  $$Q_j(X) := \frac{\varphi_j^2(X) - \varphi_j(X)}{Z_S(X)}.$$
  Compute the proof $\pi_j := g_p^{Q_j(\alpha)}$ given the CRS $(A_i = g_p^{\alpha^i})_{i=0}^{\ell}$.
  - **Gate validity proof:** For every $k \in [n]$, let $G_k = (j_1, j_2, j_3)$ be the k-th NAND gate of C. Let $\varphi_{j_1}, \varphi_{j_2}, \varphi_{j_3}$ be the polynomials defined above. Define polynomial
  $$P_k(X) := \frac{1 - \varphi_{j_3}(X) - \varphi_{j_1}(X)\varphi_{j_2}(X)}{Z_S(X)}.$$
  Compute the proof $\Sigma_k := g_p^{P_k(\alpha)}$ given $(A_i = g_p^{\alpha^i})_{i=0}^{\ell}$.

  The prover outputs the proof $\mathsf{pf} := (\{\sigma_j, \pi_j\}_{j \in [m]}, \{\Sigma_k\}_{k \in [n]})$.

- $\mathsf{Verify}(\mathsf{pp}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \mathsf{pf})$: Given parameters $\mathsf{pp} := (N, \mathbb{G}, \mathbb{G}_T, e, S, (A_i)_{i=0}^{\ell}, D)$ where $S = \{s_i\}_{i=1}^{\ell}$ and $A_0 = g_p$, circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \longrightarrow \{0,1\}$ with m wires and n gates, instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^{h_{\mathsf{in}}}$, proof $\mathsf{pf} = (\{\sigma_j, \pi_j\}_{j \in [m]}, \{\Sigma_k\}_{k \in [n]})$, the verifier does follows.
  - **Statement check:** For every $j \in [h_{\mathsf{in}}]$, let $\phi_j \in \mathbb{Z}_N[X]$ be the polynomial where $\phi_j(s_i) \bmod N = \mathbf{x}_{i,j}$ for all $i \in [\ell]$. Denote by $\phi_j = \sum_{i=0}^{\ell} c_i X^i$. Check that
  $$\sigma_j = \prod_{i=0}^{\ell} A_i^{c_i}.$$
  We note that this check can be done in the offline phase after receiving instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$.

– **Wire validity check:** For every $j \in [m]$, check

$$e(\sigma_j, \sigma_j) = e(D, \pi_j) e(A_0, \sigma_j) .$$

– **Gate validity check:** For every $k \in [n]$, let $G_k = (j_1, j_2, j_3)$ be the k-th NAND gate of C. Check

$$e(\sigma_{j_1}, \sigma_{j_2}) e(A_0, \sigma_{j_3}) e(D, \Sigma_k) = e(A_0, A_0) .$$

– **Output check:** Check that the output encoding $\sigma_m = A_0$.

The verifier outputs 1 if the previous checks pass, and outputs 0 otherwise.

Theorem 3.5 (Completeness)**.** *Construction 3.4 is complete.*

Proof. Take any circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \longrightarrow \{0,1\}$ with m wires and n gates, instances $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \{0,1\}^{h_{\mathsf{in}}}$, witnesses $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in \{0,1\}^{h_{\mathsf{w}}}$ such that $C(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all $i \in [\ell]$. Let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^n)$ and $\mathsf{pf} \leftarrow \mathsf{Prove}(\mathsf{pp}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), (\mathbf{w}_1, \ldots, \mathbf{w}_\ell))$. We show that $\mathsf{Verify}(\mathsf{pp}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \mathsf{pf})$ outputs 1. Consider each of the following verification relations:

- **Statement check:** By definition, $\sigma_j = g_p^{\varphi_j(\alpha)} = (g_p^{\alpha^0})^{\varphi_j(\alpha)} = A_0^{\varphi_j(\alpha)}$. By construction, the polynomials ϕ_j and φ_j have the property that $\phi_j(s_i) = \mathbf{x}_{i,j} = \varphi_j(s_i)$ for all $i \in [\ell]$. Thus, $A_0^{\varphi_j(\alpha)} = A_0^{\phi_j(\alpha)} = \sigma_j$.

- **Wire validity check:** Take any $j \in [m]$. Then, $\sigma_j = A_0^{\varphi_j(\alpha)} = g_p^{\varphi_j(\alpha)}$. By construction, the polynomial $Z_S(X)$ vanishes at all points $s_i \in S$. Also recall the fact that for all $i \in [\ell]$, $u_{i,j} \in \{0,1\}$ so $u_{i,j}^2 = u_{i,j}$. So, the polynomial $Z_S(X)$ divides $\varphi_j^2(\alpha) - \varphi_j(\alpha)$ which we can write as

$$Q_j(X) Z_S(X) = \varphi_j^2(X) - \varphi_j(X).$$

By construction, $D = g_p^{Z_S(\alpha)}$ and $\pi_j = g_p^{Q_j(\alpha)}$. Thus using these relations,

$$\begin{aligned} e(\sigma_j, \sigma_j) &= e(g_p^{\varphi_j(\alpha)}, g_p^{\varphi_j(\alpha)}) \\ &= e(g_p, g_p)^{\varphi_j^2(\alpha)} \\ &= e(g_p, g_p)^{Q_j(\alpha) Z_S(\alpha) + \varphi_j(\alpha)} \\ &= e(g_p, g_p)^{Q_j(\alpha) Z_S(\alpha)} e(g_p, g_p)^{\varphi_j(\alpha)} \\ &= e(g_p^{Z_S(\alpha)}, g_p^{Q_j(\alpha)}) e(g_p, g_p^{\varphi_j(\alpha)}) \\ &= e(D, \pi_j) e(A_0, \sigma_j) \end{aligned}$$

- **Gate validity check:** Take any gate $G_k = (j_1, j_2, j_3)$ for $k \in [n]$. By definition, σ_{j_1} and σ_{j_2} is a commitment of the input wires of the gate k while σ_{j_3} is an encoding of the output wire of this gate. By definition, $A_0 = g_p$, $D = g_p^{Z_S(\alpha)}$, and $\Sigma_k = g_p^{P_k(\alpha)}$ where the polynomial $P_k(X)$ is defined as

$$P_k(X) := \frac{1 - \varphi_{j_3}(X) - \varphi_{j_1}(X)\varphi_{j_2}(X)}{Z_S(X)} .$$

Thus using these relations,

$$\begin{aligned}
e(\sigma_{j_1}, \sigma_{j_2})e(A_0, \sigma_{j_3})e(D, \Sigma_k) &= e(g_p^{\varphi_{j_1}(\alpha)}, g_p^{\varphi_{j_2}(\alpha)})e(g_p, g_p^{\varphi_{j_3}(\alpha)})e(g_p^{Z_S(\alpha)}, g_p^{P_k(\alpha)}) \\
&= e(g_p, g_p)^{\varphi_{j_1}(\alpha)\varphi_{j_2}(\alpha)} e(g_p, g_p)^{\varphi_{j_3}(\alpha)} e(g_p, g_p)^{Z_S(\alpha)P_k(\alpha)} \\
&= e(g_p, g_p)^{\varphi_{j_1}(\alpha)\varphi_{j_2}(\alpha)+\varphi_{j_3}(\alpha)} e(g_p, g_p)^{1-\varphi_{j_3}(\alpha)-\varphi_{j_1}(\alpha)\varphi_{j_2}(\alpha)} \\
&= e(g_p, g_p) \\
&= e(A_0, A_0)
\end{aligned}$$

- **Output check:** Since $C(\mathbf{x}_i, \mathbf{w}_i) = 1$, it follows that $u_{i,m} = 1$ for all $i \in [\ell]$. By definition then, $\sigma_m = g_p^{\varphi_m(\alpha)}$ where φ_m is the polynomial where $\varphi_m(s_i) = 1$ for all $i \in [\ell]$. Thus, φ_m should be the constant polynomial $\varphi_m = 1$ such that $\sigma_m = g_p^{\varphi_m(\alpha)} = g_p = A_0$.

□

3.2 Security Analysis

Theorem 3.6 (Somewhere Argument of Knowledge)**.** *Construction 3.4 is a somewhere argument of knowledge if the assumption in Definition 3.3 holds for* CompGen.

Proof. We first define the trapdoor setup and extraction algorithms.

- $\mathsf{TrapSetup}(1^\lambda, 1^\ell, 1^n, \mathsf{idx})$: The trapdoor algorithm proceeds as follows: (We highlight the differences from the Setup algorithm in blue.)

  1. Run $(p, q, \mathbb{G}, \mathbb{G}_T, e, g_p, g_q, S = \{s_i\}_{i\in[\ell]}) \leftarrow \mathsf{CompGen}(1^\lambda, 1^\ell)$. Set $N = pq$, set $S = \{s_i\}_{i=1}^{\ell} \leftarrow \mathsf{SGen}(1^\lambda, 1^\ell, N)$.
  2. Sample $\alpha \xleftarrow{\mathrm{R}} \mathbb{Z}_N$. For every $i \in [0, \ell]$, set $A_i := g_p^{\alpha^i} g_q^{s_{\mathsf{idx}}^i}$.
  3. Set $D := g_p^{Z_S(\alpha)} g_q^{Z_S(s_{\mathsf{idx}})} = g_p^{Z_S(\alpha)}$ where $Z_S \in \mathbb{Z}_N[X]$ is the vanishing polynomial $Z_S(X) := \prod_{a\in S}(X - a)$.
  4. Outputs the parameter $\mathsf{pp} := (N, \mathbb{G}, \mathbb{G}_T, e, S, (A_i)_{i=0}^{\ell}, D)$ and trapdoor $\mathsf{td} = g_q$.

- $\mathsf{Extract}(\mathsf{td}, C, (\mathbf{x}_1, \dots, \mathbf{x}_\ell, \mathsf{pf})$: Given the trapdoor td, the circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$, instances $\mathbf{x}_1, \dots, \mathbf{x}_\ell \in \{0,1\}^{h_{\mathsf{in}}}$, and the proof $\mathsf{pf} = (\{\sigma_j, \pi_j\}_{j\in[m]}, \{\Sigma_k\}_{k\in[n]})$, the extraction algorithm proceeds as follows:

  1. For every $j \in [h_{\mathsf{in}} + 1, h_{\mathsf{in}} + h_{\mathsf{w}}]$, set wire value $u_j^* := 0$ if $e(g_q, \sigma_j) = 1$ and $u_j^* := 1$ otherwise.
  2. Outputs $\mathbf{u}^* := (u_{h_{\mathsf{in}}+1}^*, \dots, u_{h_{\mathsf{in}}+h_{\mathsf{w}}}^*)$.

Next, we show the CRS indistinguishability and the somewhere extractability in the trapdoor mode.

Lemma 3.7 (CRS indistinguishability)**.** *If the assumption in Definition 3.3 holds for* CompGen*, then Construction 3.4 satisfies CRS indistinguishability.*

Proof. Let $\ell = \ell(\lambda)$, $n = n(\lambda)$. Fix any index $\mathsf{idx} \in [\ell]$. Let Hyb_0 denote adversary $\mathcal{A}$'s output bit given the parameter $\mathsf{pp} = (N, \mathbb{G}, \mathbb{G}_T, e, S, (A_i)_{i=0}^{\ell}, D) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^n)$. Let Hyb_1 denote $\mathcal{A}$'s output bit given the parameter $\mathsf{TrapSetup}(1^\lambda, 1^\ell, 1^n, \mathsf{idx})$. CRS indistinguishability follows from the Claim below.

Claim 3.8. *If the assumption in Definition 3.3 holds for* CompGen *and* SGen*, then for any PPT adversaries* $\mathcal{A}$ *and for all* $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda)$.

Proof. Fix any $\mathsf{idx} \in [\ell]$. Assume for contradiction that $|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]| = \epsilon(\lambda)$ for some PPT adversary $\mathcal{A}$ and some non-negligible ϵ. We construct an adversary $\mathcal{B}$ for the Decision Problem in Definition 3.3:

1. $\mathcal{B}$ receives the group description and the structure reference string from the challenger
$$\mathcal{G} = (N, \mathbb{G}, \mathbb{G}_T, e, S, \mathsf{srs} = (A_i)_{i=0}^{\ell}) .$$
2. Let $Z_S(X) \coloneqq \prod_{a \in S}(X - a) = \sum_{i=0}^{\ell} c_i X^i \in \mathbb{Z}_N[X]$. The adversary $\mathcal{B}$ computes $D \coloneqq \prod_{i=0}^{\ell} A_i^{c_i}$.
3. $\mathcal{B}$ sends $\mathsf{pp} = (\mathcal{G}, D)$ to $\mathcal{A}$ and returns what $\mathcal{A}$ outputs.

We note that the algorithm $\mathcal{B}$ perfectly simulates the distribution of Hyb_0 if
$$\mathsf{srs} = \left(g_p, g_p^{\alpha}, g_p^{\alpha^2}, \ldots, g_p^{\alpha^{\ell}}\right)$$
and $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1]$.

Similarly, $\mathcal{B}$ perfectly simulates the distribution of Hyb_1 if
$$\mathsf{srs} = \left(g_p g_q, g_p^{\alpha} g_q^{s_{\mathsf{idx}}}, g_p^{\alpha^2} g_q^{s_{\mathsf{idx}}^2}, \ldots, g_p^{\alpha^{\ell}} g_q^{s_{\mathsf{idx}}^{\ell}}\right)$$
and $\mathcal{B}$ outputs 1 with probability $\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]$.

Therefore, the advantage of $\mathcal{B}$ in the decision problem of Definition 3.3 is $|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| = \epsilon$, contradiction. □

□

Lemma 3.9 (Somewhere Extractability). *Construction 3.4 is somewhere extractable in trapdoor mode.*

Proof. Let $\ell = \ell(\lambda)$, $n = n(\lambda)$. Let $\mathsf{idx}^* \leftarrow \mathcal{A}(1^{\lambda}, 1^{\ell}, 1^{n})$ and let $(\mathsf{pp}^*, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^{\lambda}, 1^{\ell}, 1^{n}, \mathsf{idx}^*)$ where
$$\mathsf{pp}^* = \left(N = pq, \mathbb{G}, \mathbb{G}_T, e, S = (s_i)_{i \in [\ell]}, \left\{A_i = g_p^{\alpha^i} g_q^{s_{\mathsf{idx}^*}^i}\right\}_{i=0}^{\ell}, D = g_p^{Z_S(\alpha)}\right), \qquad \mathsf{td} = g_q .$$
Let $\mathbb{G}_p \coloneqq \langle g_p \rangle$ with order p and $\mathbb{G}_q \coloneqq \langle g_q \rangle$ with order q. We note that $\mathbb{G} \cong \mathbb{G}_p \times \mathbb{G}_q$.

Given circuit $C : \{0,1\}^{h_{\mathsf{in}}} \times \{0,1\}^{h_{\mathsf{w}}} \to \{0,1\}$ and instances $\mathbf{x}_1, \ldots, \mathbf{x}_{\ell} \in \{0,1\}^{h_{\mathsf{in}}}$, let $\mathsf{pf} = (\{\sigma_j, \pi_j\}_{j \in [m]}, \{\Sigma_k\}_{k \in [n]})$ be the adversary's output proof. Suppose that pf passes the verification. For every $j \in [m]$, we write σ_j as $\sigma_j = g_p^{\beta_j} g_q^{\gamma_j}$ for some $\beta_j \in \mathbb{Z}_p$ and $\gamma_j \in \mathbb{Z}_q$. The following properties hold:

- For every $j \in [m]$, by the wire validity check, we have
$$e(\sigma_j, \sigma_j) = e(D, \pi_j) e(A_0, \sigma_j) .$$
Consider the projection in the order-q subgroup of $\mathbb{G}_T$. Recall that $A_0 = g_p g_q$, $D = g_p^{Z_S(\alpha)} g_q^{Z_S(s_{\mathsf{idx}^*})}$. Moreover, $Z_S(s_{\mathsf{idx}^*}) = 0 \bmod N$, which implies that $Z_S(s_{\mathsf{idx}^*}) = 0 \bmod q$. Thus, the above equation implies that $\gamma_j^2 = 0 + \gamma_j (\bmod q)$, or equivalent, $\gamma_j \in \{0, 1\}$.

- For every $k \in [n]$, let (j_1, j_2, j_3) be the wires of the k-th NAND gate. By the gate validity check, we have that
$$e(\sigma_{j_1}, \sigma_{j_2}) e(A_0, \sigma_{j_3}) e(D, \Sigma_k) = e(A_0, A_0)\,.$$
Consider the projection in the order-q subgroup of $\mathbb{G}_T$. Similarly, since $A_0 = g_p g_q$, $D = g_p^{Z_S(\alpha)} g_q^{Z_S(s_{\mathsf{idx}^*})}$ and $Z_S(s_{\mathsf{idx}^*}) = 0 \bmod q$, the above equation implies that $\gamma_{j_1}\gamma_{j_2} + \gamma_{j_3} + 0 = 1$, or equivalently, $\gamma_{j_3} = 1 - \gamma_{j_1}\gamma_{j_2} = \mathsf{NAND}(\gamma_{j_1}, \gamma_{j_2})$.

- For every $j \in [h_{\mathsf{in}}]$, let $\phi_j \in \mathbb{Z}_N[X]$ be the polynomial where $\phi_j(s_i) \bmod N = \mathbf{x}_{i,j}$ for all $i \in [\ell]$. Note that $\phi_j(s_i) \bmod q = \mathbf{x}_{i,j}$ as well given that $N = pq$. Denote by $\phi_j = \sum_{i=0}^{\ell} c_i X^i$. By the statement check, we have
$$\sigma_j = \prod_{i=0}^{\ell} A_i^{c_i} = \prod_{i=0}^{\ell} (g_p^{\alpha^i} g_q^{s^i_{\mathsf{idx}^*}})^{c_i}$$
Consider the projection in the order-q subgroup of $\mathbb{G}_T$, this implies that $\gamma_j = \phi_j(s_{\mathsf{idx}^*}) \bmod q = \mathbf{x}_{\mathsf{idx}^*,j}$.

- By the output check $\sigma_m = A_0 = g_p g_q$, we have that $\gamma_m = 1$.

In summary, the above properties implies that $(\gamma_1, \ldots, \gamma_m)$ is a valid assignment to the wires of C on input $\mathbf{x}_{\mathsf{idx}^*}$ and witness $\vec{\gamma} = (\gamma_{h_{\mathsf{in}}+1}, \ldots, \gamma_{h_{\mathsf{in}}+h_{\mathsf{w}}})$.

Finally, let $\mathbf{u}^* \leftarrow \mathsf{Extract}(\mathsf{td}, C, (\mathbf{x}_1, \ldots, \mathbf{x}_\ell), \mathsf{pf})$, we claim that $\mathbf{u}^* = \vec{\gamma}$. Recall that in the extraction algorithm, for $j \in [h_{\mathsf{in}} + 1, h_{\mathsf{in}} + h_{\mathsf{w}}]$, we set $u_j^* = 0$ if $e(g_q, \sigma_j) = 1$, which is the case when $\gamma_j = 0$ (i.e., $\sigma_j \in \langle g_p \rangle$). Alternatively, we set $u_j^* = 1$ if $e(g_q, \sigma_j) \neq 1$, which is the case when $\gamma_j \neq 0$. Also note that we've proved $\gamma_j \in \{0, 1\}$ for all $j \in [m]$, thus $u_j^* = \gamma_j$. Therefore, with probability $1 - \mathsf{negl}(\lambda)$, either the proof pf fails the verification or $C(\mathbf{x}_{\mathsf{idx}^*}, \mathbf{u}^*) = 1$. □

□

3.3 Efficiency Analysis

Theorem 3.10 (Succinctness)**.** *Construction 3.4 is succinct.*

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

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

- **CRS Size:** The common reference string crs consists of the group description $\mathbb{G}$, and $\ell + \ell + 2$ elements in $\mathbb{G}$. Thus, $|\mathsf{crs}| = 2\ell \cdot \mathsf{poly}(\lambda)$.

- **Verification Size:** The size of the verification components outputted by Verify consists of h_{in} group elements or more precisely $h_{in} \cdot \mathsf{poly}(\lambda)$.

- **Verification components generation time:** The verification algorithm performs polynomial interpolation h_{in} times for polynomials of degree ℓ. This takes time at most $h_{in}\ell \log^2(\ell)$ or more formally $O(\ell \log^2(\ell))$ time. However, when S is the set of roots of unity this could be optimized to $O(\ell \log(\ell))$ time.

- **Prover time:** The running time of the prover is:

$$\underbrace{m \cdot \ell \log^2(\ell)}_{\text{wire encoding}} + \underbrace{m\ell}_{\text{wire validity}} + \underbrace{n\ell}_{\text{gate validity}} = \text{poly}(n) \cdot \ell \log^2(\ell)$$

since $m = \text{poly}(n)$.

- **Online verification time:** The running time of the online verification algorithm is:

$$\underbrace{h_{in} \cdot \text{poly}(\lambda)}_{\text{statement check}} + \underbrace{m \cdot \text{poly}(\lambda)}_{\text{wire validity check}} + \underbrace{n \cdot \text{poly}(\lambda)}_{\text{gate validity check}} + \underbrace{\text{poly}(\lambda)}_{\text{output check}} = \text{poly}(\lambda, n)$$

since $h_{in}, m = \text{poly}(n)$.

□

3.4 Generic Group Model Analysis

Definition 3.11 (q-type Simpler Decision Assumption in Composite Order Groups)**.** Let $\ell = \ell(\lambda) \in \mathbb{N}$. The decision assumption holds for CompGen and SGen if for every $i \in [\ell]$, every PPT adversary $\mathcal{A}$, and every $\lambda \in \mathbb{N}$,

$$|\Pr\left[\mathcal{A}(N, \mathbb{G}, \mathbb{G}_T, e, B, g_p\right] - \Pr\left[\mathcal{A}(N, \mathbb{G}, \mathbb{G}_T, e, B, g_p g_q\right] | \leq \text{negl}(\lambda),$$

where $(p, q, \mathbb{G}, \mathbb{G}_T, e, g_p, g_q) \leftarrow \text{CompGen}(1^\lambda, 1^\ell)$, $N = pq$, $g_p, g_q \in \mathbb{G}$ are uniformly random generators of order p and q, respectively, and

$$B := \left(g_p^\alpha, g_p^{\alpha^2}, \ldots, g_p^{\alpha^\ell}\right).$$

Theorem 3.12 (Hardness of Assumption 3.3)**.** *Assumption 3.11 implies Assumption 3.3.*

Proof. Assume we have a solver A for Assumption 3.3 with non-negligible advantage ϵ. In other words, the solver A is able to differentiate between $g_p, g_p^\alpha, g_p^{\alpha^2}, \ldots, g_p^{\alpha^\ell}$ and $g_p g_q, g_p^\alpha g_q^{s_i}, g_p^{\alpha^2} g_q^{s_i^2}, \ldots, g_p^{\alpha^\ell} g_q^{s_i^\ell}$ where g_p, g_q are random generators, s_i is some element in S, and $\alpha \xleftarrow{R} \mathbb{Z}_N$. Now let adversary B receive the challenge for the security game of Assumption 3.11. Particularly, B receives $(g_p^\alpha, g_p^{\alpha^2}, \ldots, g_p^{\alpha^\ell})$ and either $C = g_p$ or $C = g_p g_q$ depending on the experiments. Adversary B then chooses some index $s_i \in S$ and tries to compute $a_\ell = g_p^{(\alpha+s_i)^\ell - s_i^\ell} C^{s_i^\ell}$ using the values it has available, the knowledge of the exponent s_i, and C. Specifically, we can compute $a_1 = g_p^\alpha \cdot C^{s_i}$, $a_2 = g_p^{\alpha^2} \cdot (g_p^\alpha)^{2s_i} \cdot C^{s_i^2}$, and so on. Adversary B then sends $(C, a_1, a_2, \ldots, a_\ell)$ to the solver A and outputs the resulting output bit b from A. In experiment 0, adversary B receives $C = g_p$ and thus sends $(g_p, g_p^{\alpha+s_i}, g_p^{(\alpha+s_i)^2}, \ldots, g_p^{(\alpha+s_i)^\ell})$. In experiment 1, adversary B receives $C = g_p g_q$ and ends up computing and sending $(g_p g_q, g_p^{\alpha+s_i} g_q^{s_i}, g_p^{(\alpha+s_i)^2} g_q^{s_i^2}, \ldots, g_p^{(\alpha+s_i)^\ell} g_q^{s_i^\ell})$. Note that this exactly mimics the view of the solver A as we can treat $\alpha + s_i$ as the hidden exponent that A samples at the beginning. So solver A which distinguishes between both experiments with non-negligible ϵ is basically differentiating between g_p and $g_p g_q$ with advantage ϵ as well. Since Adversary B outputs the same bit as A, B is able to break Assumption 3.11 with advantage ϵ as well. □

Lemma 3.13 (Generic Hardness of Assumption 3.11)**.** *If factoring a product of two primes (each of size 2^λ) is computationally hard, then Assumption 3.11 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 3.11 can be expressed as polynomials over the formal variable α where for all $i \in [\ell] \setminus \{0\}$, $P_i = [\alpha^i, 0]$. The challenge polynomials are constructed as

$$T_0(\alpha) = [1, 0] \text{ and } T_1(\alpha) = [1, 1]$$

We now consider the conditions of Theorem D.4 in [GLWW24].

1. First, it is easy to see that T_0 and T_1 are independent of the other monomials given out in the challenge ($\mathcal{P}$) as T_0 and T_1 do not depend on the formal variable α by Definition 2.6.

2. By construction, T_0 and T_1 are identical in the $\mathbb{G}_p$ subgroup and only differ in the $\mathbb{G}_q$ subgroup. However, there are no polynomials P given in the assumption that have a non-zero $\mathbb{G}_q$ component such that $T_0P \neq T_1P$. Thus, $T_0P_i = T_1P_i$ for all assumption polynomials and T_bP_i is independent of $\mathcal{S}_i^{(b)}$.

3. We can see that $T_b^2 = T_b$ for both challenges. Note that $T_bP_i = P_i$ and $\mathcal{P}^2$ contains polynomials that contain α. Thus, there is no polynomial in the set $\mathcal{S}^{(b)} = \mathcal{P}^2 \cup T_bP_i$ that contains either 1 in $\mathbb{G}_p$ or a component of $\mathbb{G}_q$. So, T_b^2 is independent of $\mathcal{S}^{(b)}$

4. Lastly, the maximum degree of the polynomials appearing in the challenge and the assumption are at most ℓ. Since $\ell = \text{poly}(\lambda)$ and the number of terms is ℓ, the claim follows from Theorem D.4 in [GLWW24].

□

References

[BBK+23] Zvika Brakerski, Maya Farber Brodsky, Yael Tauman Kalai, Alex Lombardi, and Omer Paneth. Snargs for monotone policy batch NP. In *CRYPTO*, pages 252–283, 2023.

[Beh46] Felix A Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences*, 32(12):331–332, 1946.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 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*, pages 635–668, 2023.

[CJJ21a] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In *CRYPTO*, pages 394–423, 2021.

[CJJ21b] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Snargs for $\mathcal{P}$ from LWE. In *FOCS*, pages 68–79, 2021.

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *FOCS*, pages 1057–1068, 2022.

[Elk10] Michael Elkin. An improved construction of progression-free sets. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 886–905. SIAM, 2010.

[ET36] Paul Erdös and Paul Turán. On some sequences of integers. *Journal of the London Mathematical Society*, 1(4):261–264, 1936.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*, pages 626–645. Springer, 2013.

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

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, pages 339–358, 2006.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326. Springer, 2016.

[GSWW22] Rachit Garg, Kristin Sheridan, Brent Waters, and David J. Wu. Fully succinct batch arguments for NP from indistinguishability obfuscation. In *TCC*, pages 526–555, 2022.

[HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. Snargs for P from sub-exponential DDH and QR. In *EUROCRYPT*, pages 520–549, 2022.

[KLVW23] Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and RAM delegation. In *STOC*, pages 1545–1552, 2023.

[KVZ21] Yael Tauman Kalai, Vinod Vaikuntanathan, and Rachel Yun Zhang. Somewhere statistical soundness, post-quantum security, and snargs. In *TCC*, pages 330–368, 2021.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[WW22] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In *CRYPTO*, pages 433–463, 2022.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979.