Multi-party Setup Ceremony for Generating Tokamak zk-SNARK Parameters

Muhammed Ali Bingol

Tokamak Network

muhammed@tokamak.network

Abstract

This document provides a specification guide for the Multi-party Computation (MPC) setup ceremony for the Tokamak zk-SNARK scheme [1]. It begins by revisiting the MMORPG protocol proposed in BGM17 [2] for Groth16 setup generation, which leverages a random beacon to ensure public randomness. Additionally, it explores the alternative design approach presented in the "Snarky Ceremonies" paper KMSV21 [3], which removes the need for a random beacon. The document includes detailed pseudocode and workflow for each stage of parameter generation in the Tokamak zk-SNARK protocol.

Tokamak zk-SNARK employs a universal setup through sub-circuits, which allows for CRS reuse across multiple circuits. This approach reduces the need for repeated trusted setups and emphasizes efficiency in verifier preprocessing. The document also introduces pseudocodes for various types of parameter generation during the MPC setup. This includes the generation of parameters like Powers of τ, circuit-specific parameters, and different types of mappings across both the random beacon and non-random beacon approaches. These pseudocodes ensure clarity in the protocol's step-by-step process, from the computation of shared parameters to the verification of correctness.

Finally, the document presents a sketch security analysis of both protocols, relying on the Algebraic Group Model (AGM) and the Random Oracle Model (ROM) to prove knowledge soundness and security of the generated CRS. The analysis considers potential attacks and demonstrates that, even without a random beacon, the setup remains secure under the assumptions of these models.

Keywords: Multi-party computation, zk-SNARKs, setup ceremony, cryptographic protocol

1 Introduction

Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) have become a cornerstone in modern cryptographic systems due to their ability to efficiently and succinctly verify the correctness of computations without revealing any additional information beyond the fact that the computation was performed correctly. zk-SNARKs are highly valued for their conciseness, efficiency, and public verifiability, making them indispensable in fields such as privacy-preserving protocols, decentralized systems, and particularly, blockchain technologies. Recent advancements in zk-SNARKs, particularly Groth16 [4], have further driven their adoption in practical applications due to their small proof size and fast verification times.

However, the deployment of zk-SNARKs requires a crucial setup phase known as the generation of a Common Reference String (CRS), or public parameters, which are essential for both the proof construction and verification processes. This setup phase introduces a significant vulnerability: the entity responsible for generating the CRS gains access to secret trapdoor information, commonly referred to as "toxic waste." This toxic waste can be exploited to forge fraudulent proofs, an issue that poses a critical security risk in sensitive applications such as cryptocurrencies, where such an attack could result in undetected financial losses worth billions of dollars.

Groth16, introduced at EUROCRYPT 2016 [4], represents the current state-of-the-art zk-SNARK and is widely deployed in practice. Despite its efficiency, Groth16 still requires a trusted setup process, in which both proving and verification keys are generated. The security of this setup is of paramount importance, as the trusted party must delete the toxic waste securely. Failure to do so could enable the creation of fraudulent proofs that would be indistinguishable from valid ones by any verifier.

To mitigate this risk, Multi-Party Computation (MPC) protocols are often employed during the setup phase to distribute the responsibility of generating the CRS among multiple participants. If at least one

participant in the setup ceremony is honest and securely deletes their portion of the toxic waste, the integrity of the system is guaranteed. A prominent example of this approach is the MMORPG (Multi-Party Mathematical Operations with Random Public Generation) protocol, proposed by Bowe, Gabizon, and Miers in [2]. MMORPG operates in random beacon mode, providing a decentralized and publicly verifiable setup, which has been widely adopted in zk-SNARK setups like Zcash, Semaphore, and others.

The use of a random beacon in the MMORPG protocol helps ensure the randomness and integrity of the setup. However, obtaining a secure random beacon is a non-trivial problem, as it introduces its own set of challenges. Alternatives, such as using blockchain block headers or verifiable delay functions (VDFs), have been explored, but these come with their own limitations, such as susceptibility to bias or the need for specialized hardware.

This document builds upon these works by presenting a secure and scalable MPC protocol for Groth16's parameter generation, specifically designed for practical implementation in real-world applications. The proposed protocol operates in both random beacon and non-random beacon modes, enabling greater flexibility and reducing the dependency on secure randomness sources. By distributing the setup process across multiple participants, the protocol ensures that the toxic waste cannot be reconstructed unless all participants collude, thus enhancing the overall security of the zk-SNARK system.

Furthermore, we explore recent improvements in ceremony protocols, such as the Snarky Ceremonies framework [3], which removes the dependency on a random beacon and introduces optimizations for handling distributed setups. The framework also simplifies security proofs by extending the Algebraic Group Model (AGM) and the Random Oracle Model (ROM) to handle zk-SNARK setups. By adopting these improvements, we aim to provide a robust and adaptable MPC setup protocol that can be applied to a wide range of zk-SNARK systems, including the Groth16 and Tokamak zk-SNARK schemes.

In this document, we present a comprehensive protocol design for conducting a secure, decentralized, and scalable MPC setup scheme, both with and without the use of a random beacon. We first revisit the Groth16 setup ceremonies based on BGM17 [2] and the "Snarky Ceremonies" framework [3]. We then introduce our own MPC setup ceremony design for the recent Tokamak Network protocol [1]. Detailed pseudocode is provided for the MPC setup ceremony for the Tokamak zk-SNARK, outlining the key steps in the parameter generation and verification phases for both the random beacon and beaconless cases. We present the pseudocodes for each different type of parameter. After that, we give a detailed step-by-step parameter generation flow and map the algorithm types in each parameter generation. Additionally, we present the security analysis, focusing on its resilience within the Algebraic Group Model (AGM) and Random Oracle Model (ROM), and the steps necessary to ensure soundness and zero-knowledge properties under various adversarial scenarios. Through this, we aim to contribute to the ongoing efforts to make Tokamak zk-SNARK setups secure and scalable for real-world applications.

Organization of the document. The rest of the document is organised as follows: Section 2 covers the cryptographic preliminaries, notation, and definitions necessary for understanding the protocols, including a high-level overview of the Groth16 [4] setup phase and its key parameters. Section 3 introduces the proposed MPC scheme for Groth16, detailing both random beacon and beaconless designs, and discusses their adaptation to the Tokamak zk-SNARK protocol[1] with a focus on its universal setup and CRS reuse. Section 4 presents the non-random beacon design from the Snarky Ceremonies [3] framework, with optimizations and a discussion on its impact on soundness and security in the Algebraic Group Model (AGM). Section 5 provides pseudocode for generating and verifying parameters in both Groth16 and Tokamak ceremonies to ensure secure contributions by participants. Section 6 offers a security analysis of the protocols, emphasizing assumptions in the AGM and Random Oracle Model (ROM) and ensuring knowledge soundness under adversarial conditions. Section 7 includes detailed pseudocode for the MPC setup ceremony in the Tokamak zk-SNARK, outlining key steps in parameter generation and verification. Section 8 concludes with a sketch security analysis of the update proofs, focusing on soundness and zero-knowledge properties under the AGM and ROM models, and preventing adversarial manipulation.

2 Definitions and notation

Our operations will be conducted within bilinear groups $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$ each of prime order p, together with respective generators G_1, G_2 and G_T. These groups are equipped with a non-degenerate bilinear pairing $e : G_1 \times G_2 \to G_T$ with $e(G_1, G_2) = G_T$. We write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively, and $\mathbb{G}_T$ multiplicatively. For $k \in \mathbb{F}_p$, we denote $[K]_1 := k \cdot G_1$, $[K]_2 := k \cdot G_2$. We use the notation $\mathbb{G} : \mathbb{G}_1 \times \mathbb{G}_2$. Given an element $h \in \mathbb{G}$, we denote by $h_1(h_2)$ the $G_1(G_2)$ element of h. We denote $\mathbb{G}_1^*$, $\mathbb{G}_2^*$ the non-zero elements of $\mathbb{G}_1$, $\mathbb{G}_2$ and denote $\mathbb{G}^* : G_1^* \times G_2^*$.

2.1 Preview of the setup phase in the Groth16 scheme

In this section, we provide an overview of the parameters used in the setup phase of the Groth16 protocol without going into detail. We present a pairing-based non-interactive zero-knowledge (NIZK) argument for quadratic arithmetic programs

$$\sum_{i=0}^{m} a_i u_i(X) \cdot \sum_{i=0}^{m} a_i v_i(X) = \sum_{i=0}^{m} a_i w_i(X) + h(X)t(X),$$

for some degree $n-2$ quotient polynomial $h(X)$.

Let $\{u_i,\, v_i,\, w_i\}_{i\in[0\ldots m]}$ and $\{t\}$ be the polynomials of a degree n QAP over $\mathbb{F}_p$, where t is the degree n target polynomial of the QAP and the other polynomials have degree smaller than n. Suppose that are the indices of the public input.

2.2 Groth's Setup Phase:

Choose random $\alpha, \beta, \gamma, \delta, x \longleftarrow \mathbb{Z}_p^*$. $\tau = (\alpha, \beta, \gamma, \delta, x)$ and compute $([\sigma_1]_1, [\sigma_2]_2)$, $[\sigma_1]_1 = \sigma_1 \cdot G_1$ and $[\sigma_1]_2 = \sigma_1 \cdot G_2$ where

$$\sigma_1 = \left(\begin{array}{c} \alpha,\, \beta,\, \delta,\, \left\{x^i\right\}_{i=0}^{n-1},\, \left\{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\gamma}\right\}_{i=0}^{\ell} \\ \left\{\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)+w_i(x)}{\delta}\right\}_{i=l+1}^{m},\, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n} \end{array} \right)_{i=0} , \sigma_2 = \left(\beta, \gamma, \delta, \left\{x^i\right\}_{i=0}^{n-1}\right).$$

3 Multi-party Computation for Parameters Generation of Groth16 Scheme

In this section, we present the application of BGM17's [2] multiparty computation (MPC) scheme, called MMORP, on Groth16 [4], which is a well-known zk-SNARK scheme.

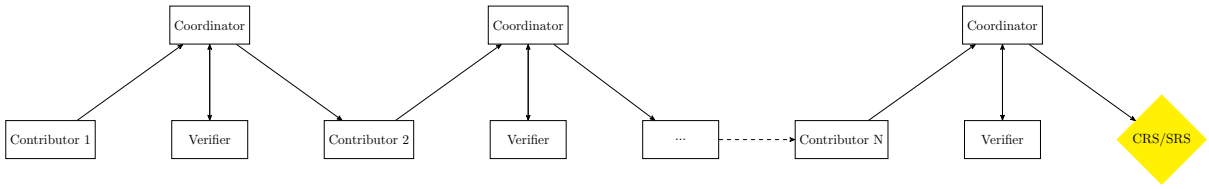


Figure 1: Multi-party Setup Ceremony Flow for Generating zk-SNARK Parameters

According the MPC protocol, the CRS is generated through two stages. The initial phase, known as "Powers of Tau", generates universal setup parameters applicable to all circuits within the scheme, up to a specified size. The "Powers of Tau" ceremony offers several improvements compared to previous schemes. Firstly, participants are not required to be pre-selected; instead, the protocol utilizes a random beacon that generates public, random values at regular intervals, enabling a continuous ceremony. This means that participants do not need to always be present and connected on-line. The use of the random beacon also guarantees the coordinator's public verifiability. Consequently, the protocol can theoretically accommodate hundreds or even thousands of participants. The subsequent phase transforms the results from the Powers of Tau phase into a CRS specific to the NP-relation.

In this protocol, a central *protocol coordinator* facilitates the communication of messages between participants. Instead, the protocol employs a random beacon that generates public random values at predetermined intervals to sustain an ongoing ceremony. However, there is no requirement to trust the *coordinator*, as any individual can subsequently confirm the accuracy of the protocol coordinator's messages within the protocol transcript. Specifically, the protocol verifier's responsibilities will encompass, beyond the explicitly outlined procedures, independently computing the protocol coordinator's messages and verifying their correctness.

This setup allows participants not always to be required to be online and available. The random beacon also guarantees the public verifiability of the coordinator. Consequently, the protocol can theoretically accommodate hundreds or even thousands of participants.

3.1 The MPC Protocol Description

We present the MMORG MPC protocol designed to compute the common reference string (CRS) elements of Groth16.

The resulting output will be structured as follows:

$$\left\{\left[x^i\right]\right\}_{i\in[0..n-1]}, \left\{\left[x^i\right]_1\right\}_{i\in[n..2n-2]}, \left\{\left[\alpha x^i\right]_1\right\}_{i\in[0..n-1]},$$
$$[\beta], \left\{\left[\beta x^i\right]_1\right\}_{i\in[1...n-1]}, \left\{\left[x^i \cdot t(x)/\delta\right]_1\right\}_{i\in[0...n-2]},$$
$$\left\{\left[\frac{\beta u_i(x)+\alpha v_i(x)+w_i(x)}{\delta}\right]_1\right\}_{i\in[\ell+1..m]}$$

We execute the protocol in two rounds, and each round we compute M_1 and M_2 set of parameters, respectively. M is an output in $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbf{G}$. We denote $[M]^j$ the "partial M" after N participants $P_1, P_2, ..., P_N$ have contributed their shares, where $j \in [N]$. $[M]^0$ will be initialized to a predetermined value as outlined in the protocol description. It is assumed that $\boldsymbol{g}$ is publicly known.

3.1.1 First Round: Powers of τ

We will compute the M_1:

$$M_1 = \left\{ \begin{array}{l} \{[x^i]\}_{i\in[0...n-1]}, \{[x^i]_1\}_{i\in[n\cdot 2n-2]}, \\ \{[\alpha x^i]_1\}_{i\in[0..n-1]}, [\beta], [\delta]\{[\delta x^i]_1\}_{i\in[1...n-1]} \end{array} \right\}$$

Initialization:

We first initialize the parameters with the following public values.

1. $[x^i]^0 := \mathbf{g}$, $i \in [1...n-1]$.
2. $[x^i]^0 := G_1$, $i \in [n...2n-2]$.
3. $[\alpha x^i]^0 := G_1$, $i \in [1...n-1]$.
4. $[\beta]^0 := \mathbf{g}$.
5. $\left[\beta x^i\right]^0 := G_1,\ i \in [1...n-1]$.

Computations:

Then, the participants $(j \in [N], P_j)$ perform the below computations:

1. $[\alpha_j]_1, [\beta_j]_1, [x_j]_1$
2. $y_{\alpha,j} := \mathbf{POK}(\alpha_j, \text{transcript}_{1,j-1})$
3. $y_{\beta,j} := \mathbf{POK}(\beta_j, \text{transcript}_{1,j-1})$
4. $y_{x,j} := \mathbf{POK}(x_j, \text{transcript}_{1,j-1})$
5. For each $i \in [1..2n-2]$, $[x^i]^j := x_j^i \cdot [x^i]^{j-1}$
6. For each $j \in \left[0..m-1\right]$, $\left[\alpha x^i\right]^{\mathrm{j}} := \alpha_j x_j^i \cdot \left[\alpha x^i\right]^{\mathbf{j-1}}$
7. For each $j \in \left[0..m-1\right]$, $\left[\beta x^i\right]^{\mathrm{j}} := \beta_j x_j^i \cdot \left[\beta x^i\right]^{\mathbf{j-1}}$

Let $J-1$ be the time-slot where P_N sends their message. Let $(x', \alpha', \beta') := \text{RandomBeacon}(J, 3)$

1. $\left[x^i\right] := x'^i \cdot \left[x^i\right]^{\mathrm{N}},\ i \in [1..2n-2]$.
2. $\left[\alpha x^i\right] := \alpha' x'^i \cdot \left[\alpha x^i\right]^{\mathrm{N}},\ i \in [0...n-1]$.
3. $\left[\beta x^i\right] := \beta' x'^i \cdot \left[\beta x^i\right]^{\mathrm{N}},\ i \in [0...n-1]$.

Verifications:

The protocol verifier computes for each $j \in [N]$,

$$\begin{aligned} r_{\alpha,j} &:= \mathcal{R}([\alpha_j]_1, \text{transcript}_{1,j-1}), \\ r_{\beta,j} &:= \mathcal{R}([\beta_j]_1, \text{transcript}_{1,j-1}), \\ r_{x,j} &:= \mathcal{R}([x_j]_1, \text{transcript}_{1,j-1}), \end{aligned}$$

and checks $j \in [N]$ that

1. $\mathbf{CheckPOK}([\alpha_{\mathrm{j}}]_1, \text{transcriptt}_{1,j-1}, y_{\alpha,j})$,
2. $\mathbf{CheckPOK}([\beta_{\mathrm{j}}]_1, \text{transcriptt}_{1,j-1}, y_{\beta,j})$,
3. $\mathbf{CheckPOK}([\mathrm{x_j}]_1, \text{transcriptt}_{1,j-1}, y_{x,j})$,
4. $\mathbf{Consistent}([\alpha]^{j-1} - [\alpha]^j; (r_{\alpha,j}, y_{\alpha,j}))$,
5. $\mathbf{Consistent}([\beta]^{j-1} - [\beta]^j; (r_{\beta,j}, y_{\beta,j}))$,
6. $\mathbf{Consistent}([x]^{j-1} - [x]^j; (r_{x,j}, y_{x,j}))$,
7. For each $i \in [1..2n-2]$, $\mathbf{Consistent}\left([x^{i-1}]^{\mathbf{j}} - [x^i]^{\mathbf{j}}; t[x]^{\mathbf{j}}\right)$,
8. For each $i \in [1...n-1]$, $\mathbf{Consistent}\left([x^i]_1^{\mathrm{j}} - [\alpha x^i]^{\mathrm{j}}; [\alpha]^{\mathrm{j}}\right)$,
9. For each $i \in [1...n-1]$, $\mathbf{Consistent}\left([x^i]_1^{\mathrm{j}} - [\beta x^i]^{\mathrm{j}}; [\beta]^{\mathrm{j}}\right)$,

3.1.2 Second Round

During this subsequent phase of the protocol, parameters specific to the circuit will be generated.

We will compute the M_2:

$$\mathrm{M}_2 = \left\{[\delta], \left\{[K_i]_1\right\}_{i\in[\ell+1...m]}, \left\{[H_i]_1\right\}_{i\in[0...n-2]}\right\}, \text{ where}$$

$$K_i := \tfrac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}, \text{ where } i \in [\ell+1...m], \text{ and}$$
$$H_i := \tfrac{t(x)x^i}{\delta}, \text{ where } i \in [\ell+0...n-2].$$

Initialization:

1. $[K_i]^0 := K_i',\ i \in [\ell+1..m]$.
2. $[H_i]^0 := H_i',\ i \in [\ell+1..m]$.
3. $[\delta]^0 := \mathbf{g}$.

Computations:

For $j \in [N], P_j$ outputs:

1. $[\delta_j]_1$.
2. $y_{\delta,j} := \mathbf{POK}(\delta_j, \text{transcript}_{2,j-1})$.
3. $[\delta]^{\mathbf{j}} := [\delta]^{\mathbf{j-1}}/\delta_{\mathbf{j}}$.
4. For each $i \in [\ell+1...m]$, $[K_i]^{\mathbf{j}} := ([K_i]^{\mathbf{j-1}})/\delta_j$.
5. For each $i \in [\ell+0...n-2]$, $[H_i]^{\mathbf{j}} := ([H_i]^{\mathbf{j-1}})/\delta_j$.

Finally, $J-1$ be the time-slot where P_N sends their message.
Let $\delta' := \text{RandomBeacon}(J, 1)$

1. $[\delta] := [\delta]^{\mathbf{N}}/\delta'$.
2. $[K_i]_1 := [K_i]^{\mathbf{N}}/\delta'$.
3. $[H_i]_1 := [H_i]^{\mathbf{N}}/\delta'$.

Verifications:

The protocol verifier computes for each $j \in [N]$,

$$r_{\delta,j} := \mathcal{R}([\delta_j]_1\,, \text{transcript}_{2,j-1}),$$

and for each $j \in [N]$ checks that

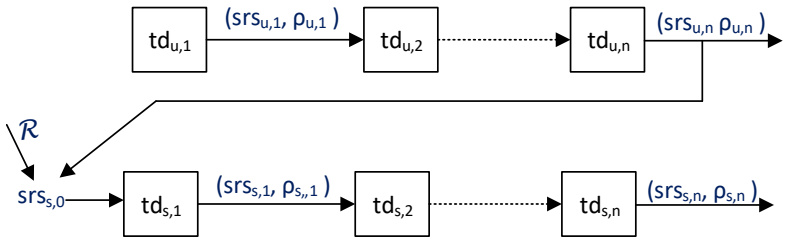
1. $\mathbf{CheckPOK}([\delta_j]_1\,, \text{transcriptt}_{2,j-1}, y_{\delta,j})$.
2. For $j \in [N]$, $\mathbf{Consistent}\big([\delta]^{\mathbf{j-1}} - [\delta]^{\mathbf{j}}; (r_{\delta,j}, y_{\delta,j})\,\big)$,
3. For each $i \in [\ell+1..m]$, $j \in [N]$, $\mathbf{Consistent}([K_i]^j - [K_i]^{j-1}; [\delta_j])$.
4. For each $i \in [\ell+0..n-2]$, $j \in [N]$, $\mathbf{Consistent}([H_i]^j - [H_i]^{j-1}; [\delta_j])$.

4 Second version: Removing the Random Beacon requirement

In this section, we present the second design which is based on Kohlweiss, Maller, Siim and Volkhov's [3] MPC scheme which is also known as the "Snarky Ceremonies" (KMSV21).

They re-examine the ceremony protocol of Groth's SNARK [2]. Their analysis reveals that the original construction can be both simplified and optimized, and we subsequently establish its security within their new framework. In particular, their construction eliminates the need for the random beacon model employed in the original work. Their work simplifies the widely used scheme of MMORPG [2] and establishes it on a more robust security foundation.

They separate the SRS into a universal component srs_u, which is independent of the specific relation being proven, and a specialized component srs_s, which is dependent on a specific relation R. Both srs_u and srs_s are updatable; however, the initial srs_s must be derived from srs_u and the relation $\mathcal{R}$. Consequently, parties must first update srs_u, and only after sufficient updates can they proceed to update srs_s. The universal srs_u has the potential to be reused for other relations.



Their protocol diverges from [3] in several key aspects related to both performance and security. In addition to the *RO* switch to $\mathbb{G}_1$ and the optional inclusion of $\top_{\prod}$ in the evaluation of *RO*. They eliminate the update with the random beacon at the end of each phase. Although this may introduce a slight bias in the *SRS*, they demonstrate that this bias is insufficient to compromise the argument's security. They consider this to be the most significant contribution of their work, as obtaining random beacons poses substantial challenges in both theory and practice. Their approach circumvents this issue entirely by proving the protocol without relying on the random beacon model.

$$\mathcal{R}_{QAP} = \left\{ (\phi, w)\,| \begin{array}{c} \phi = (a_0 = 1, a_1, \ldots, a_\ell) \in \mathbb{Z}_p^{1+\ell}, \\ w = (a_{\ell+1}, \cdots, a_m) \in \mathbb{Z}_p^{m-\ell}, \\ \exists h(X) \in \mathbb{Z}_p[X] \text{ of degree} \leq n-2 \text{ such that} \\ (\sum_{i=0}^m a_i u_i(X))\,(\sum_{i=0}^m a_i v_i(X)) = \sum_{i=0}^m a_i w_i(X) + h(X)t(X) \end{array} \right\}$$

4.1 Groth's zk-SNARK setup description

We present the Groth's zk-SNARK setup description in term of [3]'s notation. $\underline{\mathbf{Setup}}(\mathcal{R}_{\text{QAP}})$: Sample $\tau = (\alpha, \beta, \delta, x) \longleftarrow (\mathbb{Z}_p^*)^4$ and returns $(\ srs \longrightarrow (\ srs_u, srs_s)\,, \tau)$ s.t.

$$\text{srs}_u \leftarrow \left(\{G^{x^i}, H^{x^i}\}_{i=0}^{2n-2}, \{G^{\alpha x^i}, G^{\beta x^i}, H^{\alpha x^i}, H^{\beta x^i}\}_{i=0}^{n-1}\right),$$
$$\text{srs}_s \leftarrow \left(G^{\delta}, H^{\delta}, \{G^{\frac{\beta u_i(x)+\alpha v_i(x)+\omega_i(x)}{\delta}}\}_{i=\ell+1}^{m}, \{G^{\frac{x^i\, t(x)}{\delta}}\}_{i=0}^{n-2}\right)$$

4.2 KMSV21's setup description

They present default SRS and update algorithm for Groth's SNARK as below:
if $\varphi = 1$:

1. Parse $\text{srs}_u = \left(\{G_{x:i}, H_{x:i}\}_{i=0}^{2n-2}, \{G_{\alpha x:i}, G_{\beta x:i}, H_{\alpha x:i}, H_{\beta x:i}\}_{i=0}^{n-1}\right)$;
2. sample $\alpha^{'}, \beta^{'}, x^{'}, \longleftarrow \mathbb{Z}_p^*$;
3. For $(\iota \in |\alpha, \beta, x\}; \pi_{\iota'} \longleftarrow \mathsf{Prove}_{dl}^{\mathsf{RO}(\cdot)}(\mathsf{G}^{\iota'}, H^{\iota'}, \iota')$;
4. $\rho_{\alpha'} \longleftarrow (G_{\alpha x:0}^{\alpha'}, G^{\alpha'}, H^{\alpha'}, \pi_{\alpha'})$;
5. $\rho_{\beta'} \longleftarrow (G_{\beta x:0}^{\beta'}, G^{\beta'}, H^{\beta'}, \pi_{\beta'})$;
6. $\rho_{x'} \longleftarrow (G_{\alpha x:0}^{x'}, G^{x'}, H^{x'}, \pi_{x'})$;
7. $\rho \leftarrow (\rho_{\alpha'}, \rho_{\beta'}, \rho_{x'})$;
8. $\text{srs}'_u \leftarrow \left(\{G_{x;i}^{(x')^i}, H_{x;i}^{(x')^i}\}_{i=0}^{2n-2}, \{G_{\alpha x;i}^{\alpha'(x')^t}, G_{\beta x;i}^{\beta'(x')^i}, H_{\delta x;i}^{\alpha'(x')^i}\}_{i=0}^{n-1}\right)$;
9. $\text{srs}'_s \leftarrow \text{Specialize}(\text{QAP}, \text{srs}'_u)$;
10. **return** $((srs'_u, srs'_s), \rho)$;

if $\varphi = 2$:

11. Parse $\text{srs}_s = \left(G_\delta, H_\delta, \{G_{sum:i}\}_{i=\ell+1}^{m}, \{G_{t(x):i}\}_{i=0}^{n-2}\right)$
12. $\delta' \longleftarrow \mathbb{Z}_p^*$;
13. $\pi_{\delta'} \longleftarrow \mathsf{Prove}_{dl}^{\mathsf{RO}(\cdot)}(\mathsf{G}^{\delta'}, H^{\delta'}, \delta')$
14. $\rho \leftarrow (G_\delta^{\delta'}, G^{\delta'}, H^{\delta'}, \pi_{\delta'})$
15. $\text{srs}'_s = \left(G_\delta^{\delta'}, H_\delta^{\delta'}, \left\{G_{sum:i}^{1/\delta'}\right\}_{i=\ell+1}^{m}, \left\{G_{t(x):i}^{1/\delta'}\right\}_{i=0}^{n-2}\right)$
16. **return** $((srs_u, srs'_s), \rho)$;

Specialize$((\mathcal{R}_{\text{QAP}}, \text{srs}_u))$: //computes srs_s with $\delta = 1$

17. Parse $\text{srs}_u = \left(\{G_{x:i}, H_{x:i}\}_{i=0}^{2n-2}, \{G_{\alpha x:i}, G_{\beta x:i}, H_{\alpha x:i}, H_{\beta x:i}\}_{i=0}^{n-1}\right)$;
18. $\mathsf{srs}_s \leftarrow \left(G, H, \left\{\prod_{j=0}^{n-1} G_{\beta x:j}^{u_{ij}^{*j}} \cdot G_{\alpha x:j}^{v_{ij}} \cdot G_{x:j}^{w_{ij}}\right\}_{i=\ell+1}^{m}, \left\{\prod_{j=0}^{n} G_{x:(i+j)}^{\ell_j^{*}\cdots\prime}\right\}_{i=0}^{n-2}\right)$
19. **return** srs_s;

Their security proof for update knowledge-soundness employs a combination of the algebraic group model (AGM) [5] and the random oracle (RO) model. Already the original AGM paper [5] proved knowledge soundness of the Groth16 SNARK. Also, Fuchsbauer et al. [5] show how to integrate the AGM with the random oracle (RO) model. In the AGM model (assuming a trusted SRS), they proved it under the q-discrete logarithm assumption, which involves a discrete logarithm challenge of the form $(G^z, G^{z^2}, \ldots, G^{z^q})$. The main idea behind the reduction is that G^z can be embedded in the SRS of the SNARK.

Bowe et al. [2] demonstrated that the proof system is secure under a Knowledge-of-Exponent assumption. However, their analysis does not account for the possibility that an attacker might leverage additional knowledge obtained from the ceremony to compromise the update proof. Kohlweiss et al.'s analysis is more comprehensive and considers this additional knowledge. As a result, they cannot rely solely on the Knowledge-of-Exponent assumption. Instead, they utilize the algebraic group model (AGM), which is currently the weakest idealized model in which Groth16 has provable security. Therefore, they do not consider this reliance to be a theoretical drawback.

5 The Security Analysis of The Update Proofs

In this section, we focus on the security of the update proofs in BGM17 [2] and subsequently examine the security of Groth's ceremonial protocol.

The authors of BGM17 establish the security of the proof system based on the Knowledge-of-Exponent assumption. However, their analysis does not consider the potential for an attacker to leverage additional knowledge acquired during the ceremony to undermine the update proof. For this reason, we think that [3]'s security approach is more appropriate. Their analysis is more comprehensive and considers this additional knowledge, requiring us to move beyond the simple Knowledge-of-Exponent assumption. Consequently, they claimed that they utilize the algebraic group model (AGM), which, to date, is the most lenient idealized model in which Groth's protocol can be proven secure. Therefore, they do not regard this as a theoretical limitation. The proof of knowledge pertains to the discrete logarithm relation

$\mathcal{R}_{dl} = \{(\phi = (m, G^{y_1}, H^{y_2}), w) \mid y_1 = y_2 = w\}$, where m is an auxiliary input that was used in the original [2] proof of knowledge. The auxiliary input is redundant but we retain it to maintain consistency with the original protocol. We also aim to validate the security of ceremony protocols that are already in use.

We adopt their [3] following security definitions and formal description of the BKM17 protocol.

Definition 5.1. *An argument Ω for $\mathcal{R}$ is perfectly complete if for any adversary $\boldsymbol{A}$, it has the following properties:*
Update completeness:

$$\Pr\left[\begin{array}{c}(\varphi, \text{srs}, \{\rho_i\}_i) \leftarrow A(\mathbf{1}^\lambda), (\text{srs}', \rho') \leftarrow \text{Update}(\varphi, \text{srs}, \{\rho_i\}_i) : \\ \text{VerifySRS}(\text{srs}, \{\rho_i\}_i) = 1 \wedge \text{VerifySRS}(\text{srs}', \{\rho_i\}_i \cup \{\rho'\}) = 0\end{array}\right] = 0.$$

Prover completeness:

$$\Pr\left[\begin{array}{c}(\text{srs}, \{\rho_i\}_i, \phi, w) \leftarrow A(\mathbf{1}^\lambda), (\pi) \leftarrow \text{Prove}(srs, \phi, w) : \\ \text{VerifySRS}(\text{srs}, \{\rho_i\}_i) = 1 \wedge (\phi, w) \in R \wedge \text{Verify}(\text{srs}, \phi, \pi) \neq 1\end{array}\right] = 0.$$

Definition 5.2. *An argument Ω for $\mathcal{R}$ is perfectly complete in the random oracle model, if for any adversary $\boldsymbol{A}$,*

$$\Pr\left[(\phi, w) \leftarrow A^{\text{RO}(\cdot)}, \pi \leftarrow \text{Prove}^{\text{RO}(\cdot)}(\phi, w) : (\phi, w) \in \mathcal{R} \wedge \text{Verify}^{\text{RO}(\cdot)}(\phi, \pi) \neq 1\right] = 0.$$

Definition 5.3. *An argument Ω for $\mathcal{R}$ is perfectly zero-knowledge in the random oracle model if for all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$, $\varepsilon_0 = \varepsilon_1$, where $\varepsilon_b := \Pr\left[\mathcal{A}^{\mathcal{O}_b(\cdot), \text{RO}^{(\cdot)}}(1^\lambda) = 1\right]$. $\mathcal{O}_b$ is a proof oracle that takes as an input (ϕ, w) and only proceeds if $(\phi, w) \in \mathcal{R}$. If $b = 0$, $\mathcal{O}_b$ returns an honest proof $\mathsf{Prove}^{\text{RO}(\cdot)}(\phi, w)$ and when $b = 1$, it returns a simulated proof $\text{Sim}^{\text{RO}_1(\cdot)}(\phi)$.*

Sim is allowed to have access to RO discrete logarithms. They mention the following formal description of the BKM17 protocol in [3].

Definition 5.4. *Formal description of the BKM17 scheme:*

- $\mathsf{Prove}_{dl}^{\text{RO}(\cdot)}(\phi, w)$ *outputs G^{rw}, while generating $G^r \leftarrow \text{RO}(\phi)$.*
- $\text{Verify}_{dl}^{\text{RO}(\cdot)}(\phi = (\cdot, G^{y_1}, H^{y_2}), \pi)$ *checks that $\hat{e}(G^{y_1}, H) = (G, H^{y_2}) \wedge\ \hat{e}(\pi, H) = \hat{e}(G^r, H^{y_2})$, while using $G^r \leftarrow \text{RO}(\phi)$.*
- $\text{Sim}_{dl}^{\text{RO}_1(\cdot)}(\phi = (\cdot, G^{y_1}, H^{y_2}))$ *outputs $\pi \leftarrow (G^{y_1})^{r_\phi}$, by asserting $\hat{e}(G^{y_1}, H) = (G, H^{y_2})$ and using $r_\phi \leftarrow \text{RO}_1(\phi)$.*

Theorem 5.1. $\prod_{dl} = \left(\text{Prove}_{dl}^{\text{RO}(\cdot)}, \text{Verify}_{dl}^{\text{RO}(\cdot)}, \text{Sim}_{dl}^{\text{RO}_1(\cdot)}\right)$ *is complete, perfect zero-knowledge argument in random oracle model.*

Proof. Completeness and perfect zero-knowledge are directly derived from the construction of the prover, verifier, and simulator algorithms. Completeness are provided straightforwardly. Moreover, it is easy to see that $\prod_{dl}$ is perfect zero-knowledge with respect to *Simulator*. When the simulator obtains an input $\phi = (m, G^w, H^w)$, it queries r for $G^r = \text{RO}(\phi)$ using random oracle, and has G^{wr}. The adversary cannot distinguish between honest and simulated proofs because of its equality.

□

Theorem 5.2. *Groth's SNARK has perfect completeness, i.e. it has update completeness and prover completeness.*

Proof. Let's first note that if a bitstring $s = (srs, \{\rho_i\}_i)$ satisfies $\text{VerifySRS}(s) = 1$, then there are unique values $\alpha, \beta, x, \delta \in \mathbb{Z}_p^*$ that define a well-formed srs.

If the SRS passes the VerifySRS check, it constitutes a valid Groth's SNARK SRS.

We prove the statement following VerifySRS:

1. $G_{x:1} \neq [0]_1$, $G_{\alpha x:0} \neq [0]_1$, $G_{\beta x:0} \neq [0]_1$. Assume that their values are are x, α, and β respectively (Line 4 confirms)
2. $G_{x:i}$ has the exponent as $H_{x:i}$, it is x too, and that exponent of $G_{x:i}$ is exponent of $G_{x:i-1}$ multiplied by x. Hence, $G_{x:i} = \left[x^i\right]_1$, and $H_{x:i} = \left[x^i\right]_2$ (Line 5 confirms)
3. Likewise, line 6 guarantees that $G_{\iota x:i}$ has the exponent as $H_{\iota x:i}$, and that exponent of $G_{\iota x:i}$ is ιx^i. Thus, $H_{\iota x:i}$ is ιx^i too.

4. $G_\delta \neq [0]$ and that exponent of $H_\delta]$ is the same (Line 9 confirms).
5. $G_{sum:i}$ is the ith x-power of $\sum_0^{n-1}(\beta u(x) + \alpha v(x) + w(x))/\delta$ (Line 10 confirms).
6. Likewise, line 11 guarantees that $G_{t(x):i} = t(x)x^i/\delta$.

Thus, the SRS is in the exact same format as in the *Setup*.
Update completeness: Once more, we are examining *Update* in conjunction with *VerifySRS*.

- for $\varphi = 1$:
  First, we will verify that the new SRS is well-formed. Line 8 begins by multiplying each G^{x^i} and H^{x^i} by x'^i replacing x with xx'. Next it updates each ιx^i to $\iota\iota'(xx')^i$ in $G^{\iota x^i}$ and $H^{\iota x^i}$ for $\iota \in \alpha, \beta$. Specialize simply recomputes srs_s from srs_u and its correctness is easy to verify. Hence, the new *srs* is well-formed. Additionally, the update proof is correct because for each ι: First, the proof of knowledge created on line 3 will be correct because it is applied to the same instance; Secondly, for $i > 1$, assuming the previous update was correct, the verification equation will verify that the exponent of $G_\iota^{(i)}$ is equal to exponent of $G_\iota^{(i-1)}$ (ι) multiplied by the exponent of $H_{i'}^{(i)}$ (ι').
- for $\varphi = 2$: Likewisely, the SRS itself updates δ to $\delta\delta'$, and proofs are verified exactly in the same manner, but for δ instead of α, β, and x.

Prover completeness: Suppose that the Adversary **Adv** outputs $(\text{srs}, \{\rho_i\}_i, \phi,\ w)$ s.t. $(\phi, w) \in \mathcal{R}_{\text{QAP}}$, and $\text{VerifySRS}(\text{srs}, \{\rho_i\},) = 1$. It follows that the SRS is well-formed for Groth's SNARK. Consequently, the prover completeness is derived from the completeness proof in [4]. □

Update Proofs of Knowledge

A key component of the setup ceremony is the proof of update knowledge, designed to ensure that the adversary knows the values used to update the SRS. In this section, we have examined the proof of knowledge proposed by [2]. Their proof was shown to be secure only against adversaries that can make random oracle queries. However, this definition is inadequate for ensuring security, as adversaries could potentially manipulate other users' proofs or update elements to cheat. Therefore, we define a much stronger property that is sufficient for proving the security of our update ceremony

Security is established against algebraic adversaries [5] within the random oracle model. Knowledge soundness and subversion zero-knowledge are ensured by requiring at least one honest party in each phase of the protocol. Unlike [2], dependence on a random beacon is avoided; instead, potential insecurity of a random beacon is addressed by treating it as an additional malicious entity. This approach offers robust security validation for real-world protocols used in cryptocurrencies. Moreover, in [2], a novel discrete logarithm argument was used to prove knowledge of update contributions, with knowledge soundness established under the knowledge of exponent assumption in the random oracle model. However, proving the security of the ceremony protocol requires even stronger security properties: the argument must be zero-knowledge and straight-line simulation extractable, ensuring knowledge soundness even with simulated proofs. Additionally, simulation-extractability must hold even when the adversary receives group elements as auxiliary input without knowing their discrete logarithms. To achieve these stronger properties, the original argument is slightly modified and shown to be secure within the algebraic group model with random oracles.

Definition 5.5 (Update Knowledge Soundness)**.** *An argument Ψ for $\mathcal{R}$ is update knowledge-sound if for all PPT adversaries $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_\mathcal{A}$ such that* $\Pr\left[Game_{uks}^{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda) = 1\right]$ *is negligible in λ, where*

$$Game_{uks}^{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda) := \left[\begin{array}{l} (\phi, \pi) \leftarrow \mathcal{A}^{\mathcal{O}_{srs}(\cdot)}(1^\lambda); \\ \textit{get } (srs, \phi) \textit{ from } \mathcal{O}_{srs}; w \leftarrow \mathcal{E}_\mathcal{A}(view_\mathcal{A}); \\ \textit{return } \textit{Verify}(srs, \phi, \pi) = 1 \wedge (\phi, w) \notin \mathcal{R} \wedge \phi > \phi_{\max} \end{array}\right].$$

SRS update oracle $\mathcal{O}_{\text{srs}}$ is described in Algorithm 1. In this definition of update knowledge soundness, it is required that an honest verifier cannot be convinced by any adversary of a statement unless either (1) a valid witness is known; (2) the SRS fails the setup ceremony verification VerifySRS; or (3) none of the phases included any honest updates. It should be noted that completeness and zero-knowledge are maintained for any SRS that passes the setup ceremony verification, even in the absence of any honest updates. Furthermore, if $\varphi_{\max} = 1$, the standard concept of update knowledge soundness is achieved. For the remainder of this analysis, we focus solely on the case where $\varphi_{\max} = 2$. Specifically, in the first phase, a universal SRS $\text{srs}_\text{u} = \text{srs}_1$ is generated, which is independent of the relation, and in the second phase.

$\underline{\mathcal{E}_\mathcal{A}(\mathbf{view}_\mathcal{A})}$

1. Extract the set of algebraic coefficients $T_\pi \leftarrow \mathcal{E}_{\mathcal{A}}^{\text{agm}}(\text{view}_{\mathcal{A}})$ and obtain $\{C_{i:x:j}\}_{i,j=(1,1),(1,l+1)}^{m_1,m}$ from it, corresponding to the elements $\{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta\}$ in the second phase, where m_1 is the number of update queries made in the first phase, and m is the QAP parameter.

2. From $\text{view}_{\mathcal{A}}$, deduce $i_{\text{crit}_2} \to \mathcal{O}_{\text{srs}}$ query index that corresponds to the last honest update in the final SRS.

3. Return coefficients $w = \{C_{i_{\text{crit}_2}:x:j}\}_{j=l+1}^{m}$.

Algorithm 1 SRS update oracle $\mathcal{O}_{\text{srs}}$ given to the adversary in Definition 5.

Input: $(\text{intent}, \text{srs}^*, \mathcal{Q}^*)$ ▷ Initially $\mathcal{Q}_1 = \cdots = \mathcal{Q}_{\varphi_{\max}} = \emptyset$; $\varphi = 1$
if $\varphi > \varphi_{\max}$ **then return** $\perp$ ▷ SRS already finalized for all phases
end if
$\text{srs}_{\text{new}} \leftarrow (\text{srs}_1, \ldots, \text{srs}_{\varphi-1}, \text{srs}^*, \ldots, \text{srs}_{\varphi_{\max}})$
if $\text{VerifySRS}(\text{srs}_{\text{new}}, \mathcal{Q}^*) = 0$ **then return** $\perp$ ▷ Invalid SRS
end if
if intent = UPDATE **then**
    $(\text{srs}', \rho') \leftarrow \text{Update}(\varphi, \text{srs}_{\text{new}}, \mathcal{Q}^*)$; $\mathcal{Q}_\varphi \leftarrow \mathcal{Q}_\varphi \cup \{\rho'\}$
    return (srs', ρ')
end if
if intent = FINALIZE $\wedge\ \mathcal{Q}_\varphi \cap \mathcal{Q}^* \neq \emptyset$ **then**
    Assign $\text{srs}_\varphi \leftarrow \text{srs}^*$; $\varphi \leftarrow \varphi + 1$
end if

SRS update oracle $\mathcal{O}_{\text{srs}}$ given to the adversary in Definition 5.5. UPDATE returns $\mathcal{A}$ an honest update for φ, and FINALIZE finalizes the current phase. Current phase φ and current SRS srs are shared with the KS challenger. $\{\mathcal{Q}_{\varphi_i}\}_i$ is a local set of proofs for honest updates, one for each phase.

Theorem 5.3. *Let us assume the $(2n-1, 2n-2)$-edlog assumption holds. Then Groth's SNARK has update knowledge soundness with respect to all PPT algebraic adversaries in the random oracle model.*

Proof. Let $\mathcal{A}$ be an algebraic adversary against update knowledge soundness and let us denote the update knowledge soundness game Game_{uks} by Game_0. We construct an explicit white-box extractor $\mathcal{E}_{\mathcal{A}}$ and prove it to succeed with an overwhelming probability. The theorem statement is thus $\text{Adv}_{\mathcal{A},\mathcal{E}_{\mathcal{A}}}^{\text{Game}_0}(\lambda) = \text{negl}(\lambda)$. We assume that $\mathcal{A}$ makes at most q_1 update queries in phase 1 and at most q_2 in phase 2. Often we will use ι to denote any of the elements x, α, β or δ. □

6 The MPC Setup Ceremony for the Tokamak zkSNARK

The Multi-Party Computation (MPC) setup ceremony is a crucial process for establishing the trust and security of the Tokamak zkSNARK protocol. This ceremony involves multiple independent participants who collaboratively generate the public parameters required for the zkSNARK protocol without any single party being able to compromise the setup.

In this section, we explore the structure and significance of the MPC setup ceremony within the Tokamak zkSNARK framework. We will cover the roles of participants, the steps involved in the generation of parameters, and the security guarantees provided by the ceremony. The goal of the MPC setup is to ensure that the final parameters are free from bias or manipulation, making them secure for use in zero-knowledge proofs on the Tokamak platform.

6.1 The differences of TOKAMAK zk-SNARK from Groth16 related to the setup procedure

Recently the Tokamak team's zk-SNARK paper "An Efficient SNARK for Field-Programmable and RAM Circuits" has been published in [1]. In this section, we first present a brief summary of the paper and then we highlight the difference in the paper's setup from Groth's.

Groth's scheme [4] relies on a circuit-specific setup using a Common Reference String (CRS), which must be generated for each new circuit. This setup cannot be updated or reused for different circuits. One (Groth16) is a non-universal and relies on a one-time non-updatable setup for specific circuits.

Tokamak scheme [1] utilizes a universal setup, allowing a single CRS to be used for multiple circuits. This reduces the dependence on trusted setups for every new computation. In this design, the authors

present a SNARK that aims to efficiently manage the dependency on verifier preprocessing, which is often a critical aspect in SNARK systems. To achieve this, they utilize a method called field-programmable circuit derivation. This approach begins with a universal circuit that is composed of multiple subcircuits. From this universal circuit, a program-specific circuit can be generated by duplicating these subcircuits and establishing connections between them. One key point of their design is that, while it does not support updatable CRS, the field-programmable circuit derivation significantly reduces the amount of data that the verifier needs to preprocess. Instead of dealing with the entire circuit, the preprocessing mainly focuses on the wiring and connections of the subcircuits, thus reducing the dimensionality of the data involved. The authors suggest that this reduction in preprocessing dependency could help alleviate the high communication complexity typically encountered in scenarios involving verifiable RAM computations, particularly in distributed computing networks where nodes may not be trusted. Additionally, they propose that the verifier preprocessing step can be completely eliminated without altering the core SNARK structure. This can be achieved by increasing the complexity of the subcircuit designs to handle unrolling instructions, which refers to expanding repetitive processes into a single, more complex structure. This approach streamlines the verification process and enhances efficiency without compromising the underlying SNARK protocol.

In this paper, the authors introduce a SNARK that efficiently manages verifier preprocessing through field-programmable circuit derivation, which starts with a universal circuit composed of subcircuits and derives program-specific circuits by replicating and connecting these subcircuits. While the setup does not support updatability, it significantly reduces the data dimensionality for verifier preprocessing, addressing high communication complexity in verifiable RAM computation within distributed networks of untrusted nodes. By integrating a permutation argument, they transform a SNARK with a common reference string, like Groth16, into one with a universal setup, separating circuit configuration into two algorithms—setup and verifier—allowing adjustable security dependencies. their SNARK achieves state-of-the-art communication and computation efficiency with verifier preprocessing and surpasses others when preprocessing is eliminated. It combines R1CS and Plonkish circuit representations, using a permutation map for wiring, which reduces data dimensionality compared to PlonK or Marlin but sacrifices setup updatability. This SNARK is highly efficient, suitable for verifiable machine computation, and particularly effective in distributed networks like blockchains, where it reduces the burden of verifier preprocessing data on network resources.

The paper defines a probabilistic algorithm $Setup(pp_\lambda, \mathcal{L}) \mapsto (\boldsymbol{\tau}, \boldsymbol{\sigma})$ to generate an encoded reference string $\boldsymbol{\sigma}$ of the library subcircuit polynomials in $\mathcal{L}$.

$Setup(pp_\lambda, \mathcal{L})$ takes as input the bilinear pairing group $pp_\lambda = (\mathbb{H}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ and the subcircuit library $\mathcal{L} = \{u_j(X), v_j(X), w_j(X)\}_{j=0}^{m_D-1}$, picks uniform random parameters

$$\boldsymbol{\tau} := (x, y, z, \alpha, \beta, \gamma, \delta, \eta_0, \eta_1, \mu, \nu, \psi_0, \psi_1, \psi_2, \psi_3, \kappa) \xrightarrow{\S} (\mathbb{F}^*)^{16},$$

and returns $\boldsymbol{\sigma} = ([\boldsymbol{\sigma}_{A,I}]_1, [\boldsymbol{\sigma}_C]_1, [\boldsymbol{\sigma}_{zk}]_1, [\boldsymbol{\sigma}_\nu]_2)$, where

$$o_j(X) := \beta u_j(X) + \alpha \nu_j(X) + w_j(X),$$

$$M_j(X, Z) := \sum_{k=l, k\neq j}^{l_D-1} \frac{o_k(X)}{l_D - l} \left(\frac{\omega_Z^k K_{j-l}(Z) - \omega_Z^j K_{k-l}(Z)}{\omega_Z^j - \omega_Z^k} \right),$$

The three vectors $\boldsymbol{\sigma}_{AI}$, $\boldsymbol{\sigma}_C$, and $\boldsymbol{\sigma}_{zk}$ will be utilized for distinct purposes: $\boldsymbol{\sigma}_{AI}$ will address arithmetic constraints and inner-product arguments, $\boldsymbol{\sigma}_C$ will be used for copy constraint arguments, and $\boldsymbol{\sigma}_{zk}$ will incorporate zero-knowledge elements.

$$\boldsymbol{\sigma}_{A,I} := \left\{ \begin{array}{l} \alpha, \left(x^h y^i\right)_{h=0,i=0}^{n-1, s_{\max}-1}, \\ \left(\gamma^{-1} L_0(y) o_j(x)\right)_{j=0}^{l_{in}-1}, \left(\gamma^{-1} L_{-1}(y) o_j(x)\right)_{j=l_{in}}^{l-1}, \left(\eta_1^{-1} L_i(y) o_j(x)\right)_{i=0,j=l}^{s_{\max}-1, l_D-1}, \left(\delta^{-1} L_i(y) o_j(x)\right)_{i=0,j=l_D}^{s_{\max}-1, m_D-1}, \\ \left(\eta_0^{-1} L_i(y) o_j(x) \left(K_{j-1}^2(z) - 1\right)\right)_{i=0,j=l}^{s_{\max}-1, l_D-1}, \\ \left(\delta^{-1} x^h y^i t_{\mathcal{X}}(x)\right)_{h=0,i=0}^{n-2, s_{\max}-1}, \left(\delta^{-1} x^h y^i t_{\mathcal{Y}}(y)\right)_{h=0,i=0}^{2n-2, s_{\max}-2}, \left(\eta_0^{-1} L_i(y) M_j(x, z) t_{\mathcal{Z}}(z)\right)_{i=0,j=l}^{s_{\max}-1, l_D-1} \end{array} \right\}$$

$$\boldsymbol{\sigma}_C := \left\{ \begin{array}{l} \left(\mu^{-1} L_i(y) K_j(z)\right)_{i=0,j=0}^{s_{\max}-1, l_D-l-1}, \\ \left(v^{-1} y^i z^j t_{\mathcal{Y}}(y)\right)_{i=0,j=0}^{s_{\max}-2, 2l_D-2l-2}, \left(v^{-1} y^i z^j t_{\mathcal{Z}}(z)\right)_{i=0,j=2(l_D-l)-3}^{2s_{\max}-2, j=0}, \\ \left(\psi_0^{-1} \kappa^h y^i z^j\right)_{h=0,i=0,j=0}^{1, 2s_{\max}-3, 3(l_D-1)-1}, \left(\psi_1^{-1} z^j\right)_{j=0}^{3(l_D-1)-4}, \left(\psi_2^{-1} \kappa^2 y^i z^j\right)_{i=0,j=0}^{s_{\max}-2, l_D-l-1}, \left(\psi_3^{-1} \kappa^h z^j\right)_{h=1,j=0}^{2, l_D-l-2} \end{array} \right\}$$

$$\boldsymbol{\sigma}_{zk} := \left\{ \begin{array}{l} \beta, \delta, \eta_1, \left(\mu^{-1} y^i t_{\mathcal{Y}}(y)\right)_{i=0}^{l}, \eta_0^{-1} t_{\mathcal{Y}}(y) \sum_{j=1}^{l_D-1} M_j(x,z) t_{\mathcal{Z}}(z), \\ \eta_1^{-1} t_{\mathcal{Y}}(y) \sum_{j=l}^{l_D-1} o_j(x), \eta_0^{-1} t_{\mathcal{Y}}(y) \sum_{j=0}^{l_D-1} o_j(x) \left(K_{j-l}^2(z) - 1\right), \\ \left(v^{-1} y^i z^j t_{\mathcal{Y}}(y)\right)_{i=s_{\max}-1, j=0}^{s_{\max}+1, 2l_D-2l-2}, \left(\psi_0^{-1} \kappa^h y^i z^j\right)_{h=0, i=2s_{\max}-2, j=0}^{1, 2s_{\max}, 3(l_D-l-1)}, \left(\psi_2^{-1} \kappa^2 y^i z^j\right)_{i=s_{\max}-1, j=0}^{s_{\max}-1, l_D-l-1} \end{array} \right\}$$

$$\boldsymbol{\sigma}_V := \left\{ \begin{array}{l} \beta, \gamma, \delta, \eta_1, \mu\eta_0, \mu\eta_1, \left(x^h y^i\right)_{h=0, i=0}^{n-1, s_{\max}-1}, \mu^2 \sum_{j=l}^{l_D-1} o_j(x) K_{j-l}(z)), \\ \mu^3 v, \left(\mu^4 \kappa^h\right)_{h=0}^{2}, \left(\mu^3 \psi_h y^i z^j\right)_{h=0, i=0, j=0}^{3, 1, 1} \end{array} \right\}$$

7 Pseudocode for Tokamak Scheme MPC Setup

In this section, we will present the pseudocode for Tokamak scheme MPC setup ceremony. We assume that all parties can access to the same oracle $\mathcal{R}$ during the ceremony protocol. The oracle $\mathcal{R}$ takes as input strings of arbitrary length and output is a uniform independent elements of $\mathbb{G}_2^*$.

Note that we use the notation $\mathbb{G} : \mathbb{G}_1 \times \mathbb{G}_2$ and $\mathbf{G} := (G_1, G_2)$.

We denote by $\text{transcript}_{\ell,i}$ the transcript of the protocol up to the point where player i sent his message in phase ℓ. $[\alpha]^{\mathbf{j}} \in \mathbb{G}_1$ or $[\alpha]^{\mathbf{j}} \in \mathbb{G}$, If $[\alpha]^{\mathbf{j}} \in \mathbb{G}_1$; then $[\alpha]^{\mathbf{j}} := \alpha_1 \cdot \alpha_2 \ldots \alpha_{j-1} \cdot \alpha_j \cdot G_1$. $[\alpha x^i]^{\mathbf{j}} := (\alpha_0 x_0^i)(\alpha_1 x_1^i)...(\alpha_j x_j^i) G_1$.

We define the following key algorithms (Algorithm-2, Algorithm-3, Algorithm-4, Algorithm-5 [2]) to utilize computation and verification algorithms in the form of types, from Algorithms 7.1.1 to Algorithms 7.11.2. In Algorithm-2, a proof of knowledge for α is constructed, and the verification of the proof is performed using Algorithm-3.

Algorithm 2 Construct a proof of knowledge of α

function POK(α, v) ▷ Where α is the input, v is a string
  $y \leftarrow \mathcal{R}([\alpha]_1, v) \in \mathbb{G}_2^*$ ▷ $[\alpha]_1 := \alpha \cdot G_1$
  return $(\alpha \cdot y)$
end function

Algorithm 3 Verify a proof of knowledge of α

function CheckPOK(A, v, B) ▷ Where $A \in \mathbb{G}_1^*$, $B \in \mathbb{G}_2^*$
  $y \leftarrow \mathcal{R}(A, v) \in \mathbb{G}_2^*$
  return SameRatio$((G_1, A), (y, B))$
end function

Moreover, the same ratio between two sets of points can be verified using Algorithm-4, while the consistency of the ratio between two different sets of points is controlled by Algorithm-5, which also utilizes Algorithm-4.

Algorithm 4 Determine if $x \in \mathbb{F}_p^*$ exists such that $B = x \cdot A$, and $D = x \cdot C$.

function SameRatio$((A, B), (C, D))$ ▷ Where $A, B \in \mathbb{G}_1$ and $C, D \in \mathbb{G}_2$
  if $e(A, D) = e(B, C)$ **then** ▷ A, B, C, D are not the identity elements.
    return $True$
  else
    return $False$
  end if
end function

Using the algorithms described above, we will construct different types of algorithms for generating the parameters in the setup ceremony for the Tokamak scheme. We define below types of algorithms, each consisting of two sub-algorithms: *compute* and *verify*. Each participant P_j computes and can verify all parameters by using the algorithms according to their respective types. We present the structure of each algorithm set using a generic parameter representation, including forms such as α, $\alpha\beta$, x^i, αx^i, $x^i y^k$, $\alpha f(y) o_i(x)$, $x^i y^k$, and others. Table 1 presents the parameter list for the Tokamak setup scheme in relation to the types of algorithms. A summary of all types and their corresponding parameters is provided in Table 2.

Algorithm 5 Check whether the ratio between A and B is the $s \in \mathbb{F}_p^*$ that is encoded in C.

function CONSISTENT($(A - B; C)$) ▷ Where $A, B \in \mathbb{G}_1^2$ or $A, B \in \mathbb{G}^2$. And $C \in \mathbb{G}_2^*$ or $C \in (\mathbb{G}_2^*)^2$
  if $C \in (\mathbb{G}_2^*)^2$ **then**
    $r \leftarrow$ **SameRatio**$((A_1, B_1), (C_1, C_2))$
  else
    $r \leftarrow$ **SameRatio**$((A_1, B_1), (G_1, C_2))$
  end if
  if $A, B \in \mathbb{G}_1$ **then**
    return r
  else
    return r AND **SameRatio**$((A_1, B_1), (A_2, B_2))$
  end if
end function

For instance, the first type (Type-1: α) category is consist of computation and verification algorithms. Accordingly, Algorithm 7.1.1 and Algorithm 7.2.1 are executed sequentially to compute and verify the setup parameters ($\boldsymbol{\sigma}_{AI}$, $\boldsymbol{\sigma}_C$, $\boldsymbol{\sigma}_{zk}$, and $\boldsymbol{\sigma}_V$; see 6.1) in the form of "α."

7.1 Type-1: α

Let $[\alpha]^{\mathbf{0}} = G_1$ or $[\alpha]^{\mathbf{0}} = \mathbf{G}$.

Algorithm 7.1.1 The participant P_j computes all parameters for Type-1 parameter ($[\alpha]_1 := \alpha \cdot G_1$)

function COMPUTE1($[\alpha]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$) ▷ $\mathrm{v}_{rd,j-1} :=$ transcript$_{1,j-1}$, where $rd \in \{1, 2\}$
  $\alpha_j \in_R \mathbb{F}_p^*$ ▷ pick α random number
  $y = \mathbf{POK}(\alpha_j, \mathrm{v}_{rd,j-1})$
  $[\alpha]^{\mathbf{j}} = \alpha_j \cdot [\alpha]^{\mathbf{j-1}}$
  return $([\alpha]^{\mathbf{j}}, [\alpha_j]_1, y)$
end function

Algorithm 7.1.2 The protocol verifier verifies for each $j \in [N]$ for a single parameter ($[\alpha]_1 := \alpha \cdot G_1$))

function VERIFY1($[\alpha]^{\mathbf{j-1}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, \mathrm{v}_{rd,j-1}, y_{\alpha,j}$) ▷ $\mathrm{v}_{rd,j-1} :=$ transcript$_{1,j-1}$, where $rd \in \{1, 2\}$
  $r_{\alpha,j} = \mathcal{R}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  if (**CheckPOK**$([\alpha_j]_1, \mathrm{v}_{rd,j-1}, y_{\alpha,j})$) **then**
    return **Consistent**$\big([\alpha]^{\mathbf{j-1}} - [\alpha]^{\mathbf{j}}; (r_{\alpha,j}, y_{\alpha,j})\big)$
  else
    return $False$
  end if
end function

7.2 Type-2: $\alpha\beta$

Let $[\alpha\beta]^{\mathbf{0}} = G_1$ or $[\alpha\beta]^{\mathbf{0}} = \mathbf{G}$

Algorithm 7.2.1 The P_j computes parameters for (Type-2: $\alpha\beta$)

function COMPUTE2($[\alpha\beta]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$) ▷ $\mathrm{v}_{rd,j-1} :=$ transcript$_{1,j-1}$, where $rd \in \{1, 2\}$
  $\alpha_j, \beta_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
  $y_\alpha = \mathbf{POK}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  $y_\beta = \mathbf{POK}([\beta_j]_1, \mathrm{v}_{rd,j-1})$
  $[\alpha\beta]^{\mathbf{j}} = \alpha_j\beta_j \cdot [\alpha\beta]^{\mathbf{j-1}}$
  return $([\alpha\beta]^{\mathbf{j}}, [\alpha_j]_1, [\beta_j]_1, [\alpha_j\beta_j]_2, y_\alpha, y_\beta)$
end function

7.3 Type-3: x^i

Let $[x^i]^{\mathbf{0}} = G_1$ or $[x^i]^{\mathbf{0}} = \mathbf{G}$

Algorithm 7.2.2 Verification for each $j \in [N]$ for a power of parameter (Type-2: $\alpha\beta$)

function VERIFY2($([\alpha\beta]^{\mathbf{j-1}}, [\alpha\beta]^{\mathbf{j}}, [\alpha_j]_1, [\beta_j]_1, [\alpha_j\beta_j]_2, y_\alpha, y_\beta)$)
  $r_\alpha = \mathcal{R}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  if ((**CheckPOK**$([\alpha_\mathrm{j}]_1, \mathrm{v}_{rd,j-1}, y_\alpha)$ AND (**CheckPOK**$([\beta_\mathrm{j}]_1, \mathrm{v}_{rd,j-1}, y_\beta)$) **then**
    if $e([\alpha_j]_1, y_\beta) == e(r_\alpha G_1, [\alpha_j\beta_j]_2)$ **then**
      return Consistent$\big([\alpha\beta]^{\mathbf{j-1}} - [\alpha\beta]^{\mathbf{j}}; (G_2, [\alpha_j\beta_j]_2)$
    else
      return False
    end if
  end if
end function

Algorithm 7.3.1 The P_j computes parameters for (Type-3: $[x^i]_1 := x^i G_1$, where $i \in [1, ..., n]$)

function COMPUTE2($[x]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$) ▷ $\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$
  $x_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
  $y_x = \mathbf{POK}(x_j, \mathrm{v}_{rd,j-1})$
  $[x^i]^{\mathbf{j}} = x_j^i \cdot [x^i]^{\mathbf{j-1}}$, for each $i \in [1, ..., n]$
  return $([x^i]^{\mathbf{j}}, [x^i{}_j]_1, y_x)$, for each $i \in [1, ..., n]$
end function

Algorithm 7.3.2 Verification for each $j \in [N]$ for a power of parameter
(Type-3: $[x^i]_1 := x^i G_1$, where $i \in [1, ..., n]$)

function VERIFY2($([x^i]^{\mathbf{j}}, [x^i]^{\mathbf{j-1}}, y_x)$)
  $r_x = \mathcal{R}([x_j]_1, \mathrm{v}_{rd,j-1})$
  if (**CheckPOK**$([\mathrm{x_j}]_1, \mathrm{v}_{rd,j-1}, y_x)$ **then**
    if **Consistent**$\big([x^i]^{\mathbf{j-1}} - [x^i]^{\mathbf{j}}; (r_x, y_x)\big)$ **then** ▷ for each $i \in [1, ..., n]$
      return Consistent$\big([x^{i-1}]^{\mathbf{j}} - [x^i]^{\mathbf{j}}; [x]^{\mathbf{j}}\big)$
    else
      return False
    end if
  end if
end function

7.4 Type-4: αx^i

Let $[x^i]^{\mathbf{0}} = G_1$ or $[x^i]^{\mathbf{0}} = \mathbf{G}$
Let $[\alpha x^i]^{\mathbf{0}} = G_1$ or $[\alpha x^i]^{\mathbf{0}} = \mathbf{G}$

Algorithm 7.4.1 The P_j computes Type-4 parameter s.t. $([\alpha x^i]_1 := \alpha x^i G_1$, where $i \in [1, ..., n])$

function COMPUTE4($[\alpha x^i]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$) ▷ $\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$
  $\alpha_j, x_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
  $y_\alpha = \mathbf{POK}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  $[x]^{\mathbf{j}} = x_j \cdot [x]^{\mathbf{j-1}}$,
  $[\alpha x^i]^{\mathbf{j}} = \alpha_j x_j^i \cdot [\alpha x^i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$
  return $([\alpha x^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, y_\alpha)$, $i \in [1, ..., n]$
end function

7.5 Type-5: $x^i y^k$

Let $[x^i]^{\mathbf{0}} = G_1$ or $[x^i]^{\mathbf{0}} = \mathbf{G}$
Let $[x^i y^k]^{\mathbf{0}} = G_1$ or $[x^i y^k]^{\mathbf{0}} = \mathbf{G}$

Algorithm 7.4.2 Verification for each $j \in [N]$ for a power of parameter Type-4 s.t. ($[\alpha x^i]_1 := \alpha x^i G_1$, where $i \in [1, ..., n]$))

function VERIFY4($[\alpha x^i]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [x]^{\mathbf{j}}, [x^{i-1}]^{\mathbf{j}}, \mathrm{v}_{rd,j-1}, y_\alpha$)
  $r_\alpha = \mathcal{R}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  if CheckPOK$([\alpha_\mathrm{j}]_1, \mathrm{v}_{rd,j-1}, y_\alpha)$ **then**
    if Consistent$\big([x^{i-1}]^{\mathbf{j}} - [x^i]^{\mathbf{j}}; [x]^{\mathbf{j}}\big)$ **then** ▷ for each $i \in [1, ..., n]$
      return Consistent$\big([x^i]^{\mathbf{j}} - [\alpha x^i]^{\mathbf{j}}; [\alpha]^{\mathbf{j}}\big)$
    end if
  end if
end function

Algorithm 7.5.1 The P_j computes Type-5 parameter s.t. ($x^i y^k$, where $i \in [1, ..., n]$, $k \in [1, ..., m]$)

function COMPUTE5($[y^k x^i]^{\mathbf{j-1}}, [x^i]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$) ▷ $\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$
  $y_j, x_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
  $y_{kj} = \mathbf{POK}([y_j^k]_1, \mathrm{v}_{rd,j-1})$, $k \in [1, ..., m]$
  $[x]^{\mathbf{j}} = x_j \cdot [x]^{\mathbf{j-1}}$
  $[y^k x^i]^{\mathbf{j}} = y_j^k x_j^i \cdot [y^k x^i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$, $k \in [1, ..., m]$
  return $([y^k x^i]^{\mathbf{j}}, [y^k]^{\mathbf{j}}, [x]^{\mathbf{j}}, y_{kj})$, $i \in [1, ..., n]$, $k \in [1, ..., m]$
end function

Algorithm 7.5.2 Verification for each $j \in [N]$ for a power of parameter Type-5 s.t. ($x^i y^k$, where $i \in [1, ..., n]$, $k \in [1, ..., m]$))

function VERIFY5($[y^k x^i]^{\mathbf{j}}, [y^k]^{\mathbf{j}}, [x]^{\mathbf{j}}, [x^i]^{\mathbf{j}}, \mathrm{v}_{rd,j-1}, y_{kj}$)
  $r_{kj} = \mathcal{R}([y_j^k]_1, \mathrm{v}_{rd,j-1})$, $k \in [1, ..., m]$
  if CheckPOK$([y_j^k]_1, \mathrm{v}_{rd,j-1}, y_{kj})$ **then** ▷ for each $k \in [1, ..., m]$
    if Consistent$\big([x^{i-1}]^{\mathbf{j}} - [x^i]^{\mathbf{j}}; [x]^{\mathbf{j}}\big)$ **then** ▷ for each $i \in [1, ..., n]$
      return Consistent$\big([x^i]^{\mathbf{j}} - [y^k x^i]^{\mathbf{j}}; [y^k]^{\mathbf{j}}\big)$
    end if
  end if
end function

7.6 Type-6: $\alpha f(y) o_i(x)$

Let $[\alpha]^{\mathbf{0}} = G_1$ or $[\alpha]^{\mathbf{0}} = \mathbf{G}$
Let $[H_i]^{\mathbf{0}} := H_i'$, where $H_i' := [f(y)o_i(x)]_1$, $H_i = \alpha f(y) o_i(x)$.

Algorithm 7.6.1 The P_j computes Type-6 parameter s.t. ($\alpha f(y) o_i(x)$, where $i \in [1, ..., n]$)

function COMPUTE6($[H_i]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$) ▷ $\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$
  $\alpha_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
  $y_\alpha = \mathbf{POK}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  $[\alpha]^{\mathbf{j}} = \alpha_j \cdot [\alpha]^{\mathbf{j-1}}$,
  $[H_i]^{\mathbf{j}} = \alpha_j \cdot [H_i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$
  return $([H_i]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, y_\alpha)$, $i \in [1, ..., n]$
end function

Algorithm 7.6.2 Verification for each $j \in [N]$ for a power of parameter Type-6 s.t. ($\alpha f(y) o_i(x)$, where $i \in [1, ..., n]$))

function VERIFY6($[H_i]^{\mathbf{j-1}}, [H_i]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, \mathrm{v}_{rd,j-1}, y_\alpha$) ▷ $\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$
  if (**CheckPOK**$([\alpha_\mathrm{j}]_1, \mathrm{v}_{rd,j-1}, y_\alpha)$ **then**
    return Consistent$\big([H_i]^{\mathbf{j-1}} - [H_i]^{\mathbf{j}}; [\alpha]^{\mathbf{j}}\big)$, $i \in [1, ..., n]$
  else
    return *False*
  end if
end function

7.7 Type-7: $\alpha x^i y^k$

Let $[\alpha]^{\mathbf{0}} = G_1$ or $[\alpha]^{\mathbf{0}} = \mathbf{G}$, $[x]^{\mathbf{0}} = G_1$ or $[x]^{\mathbf{0}} = \mathbf{G}$, $[x^i]^{\mathbf{0}} = G_1$ or $[x^i]^{\mathbf{0}} = \mathbf{G}$
Let $[y^k x^i]^{\mathbf{0}} = G_1$ or $[y^k x^i]^{\mathbf{0}} = \mathbf{G}$
Let $[\alpha y^k x^i]^{\mathbf{0}} = G_1$ or $[\alpha y^k x^i]^{\mathbf{0}} = \mathbf{G}$

Algorithm 7.7.1 The P_j computes Type-7 parameter s.t. ($\alpha x^i y^k$, where $i \in [1, ..., n]$, $k \in [1, ..., m]$)
Let $out := [\alpha y^k x^i]^{\mathbf{j}}, [y^k x^i]^{\mathbf{j}}, [y^k]^{\mathbf{j}}, [x^{i-1}]^{\mathbf{j}}, [x^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, y_{\alpha j}, y_{kj}, \mathrm{v}_{rd,j-1}$, $i \in [1, ..., n]$, $k \in [1, ..., m]$
$\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$

function COMPUTE7($[\alpha y^k x^i]^{\mathbf{j-1}}, [y^k x^i]^{\mathbf{j-1}}, [x^i]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$)
  $\alpha_j, y_j, x_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
  $y_{\alpha j} = \mathbf{POK}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  $y_{kj} = \mathbf{POK}([y_j^k]_1, \mathrm{v}_{rd,j-1})$, $k \in [1, ..., m]$
  $[\alpha]^{\mathbf{j}} = \alpha_j \cdot [\alpha]^{\mathbf{j-1}}$
  $[x]^{\mathbf{j}} = x_j \cdot [x]^{\mathbf{j-1}}$
  $[x^i]^{\mathbf{j}} = x_j^i \cdot [x^i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$
  $[y^k x^i]^{\mathbf{j}} = y_j^k x_j^i \cdot [y^k x^i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$, $k \in [1, ..., m]$
  $[\alpha y^k x^i]^{\mathbf{j}} = \alpha_j y_j^k x_j^i \cdot [\alpha y^k x^i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$, $k \in [1, ..., m]$
  return (out)
end function

Algorithm 7.7.2 Verification for each $j \in [N]$ for a power of parameter Type-7 s.t.
($\alpha x^i y^k$, where $i \in [1, ..., n]$, $k \in [1, ..., m]$))
Let $in := [\alpha y^k x^i]^{\mathbf{j}}, [y^k x^i]^{\mathbf{j}}, [y^k]^{\mathbf{j}}, [x^{i-1}]^{\mathbf{j}}, [x^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, y_{\alpha j}, y_{kj}, \mathrm{v}_{rd,j-1}$, $i \in [1, ..., n]$, $k \in [1, ..., m]$

function VERIFY7(in)
  if not CheckPOK$([\alpha_j]_1, \mathrm{v}_{rd,j-1}, y_{\alpha j})$ **then**
    return False
  end if
  if not CheckPOK$([y_j^k]_1, \mathrm{v}_{rd,j-1}, y_{kj})$ **then** ▷ for each $k \in [1, ..., m]$
    return False
  end if
  if Consistent$([x^{i-1}]^{\mathbf{j}} - [x^i]^{\mathbf{j}}; [x]^{\mathbf{j}})$ **then** ▷ for each $i \in [1, ..., n]$
    if Consistent$([x^i]^{\mathbf{j}} - [y^k x^i]^{\mathbf{j}}; [y^k]^{\mathbf{j}})$ **then** ▷ for each $i \in [1, ..., n], k \in [1, ..., m]$
      return Consistent$([y^k x^i]^{\mathbf{j}} - [\alpha y^k x^i]^{\mathbf{j}}; [\alpha]^{\mathbf{j}})$ ▷ for each $i \in [1, ..., n], k \in [1, ..., m]$
    end if
  else
    return False
  end if
end function

7.8 Type-8: $\alpha x^i y^k f(x)$

Let $[\alpha]^{\mathbf{0}} = G_1$ or $[\alpha]^{\mathbf{0}} = \mathbf{G}$
Let $[y^k x^i]^{\mathbf{0}} = G_1$ or $[y^k x^i]^{\mathbf{0}} = \mathbf{G}$
Let $[\alpha y^k x^i]^{\mathbf{0}} = [f(x)]_1$.

Algorithm 7.8.1 The P_j computes Type-8 parameter s.t. ($\alpha x^i y^k f(x)$, where $i \in [1, ..., n]$, $k \in [1, ..., m]$)

function COMPUTE8($[\alpha y^k x^i]^{\mathbf{j-1}}, [y^k x^i]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$) ▷ $\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$
  $\alpha_j, y_j, x_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
  $y_{\alpha j} = \mathbf{POK}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  $y_{kj} = \mathbf{POK}([y_j^k]_1, \mathrm{v}_{rd,j-1})$, $k \in [1, ..., m]$
  $[\alpha]^{\mathbf{j}} = \alpha_j \cdot [\alpha]^{\mathbf{j-1}}$
  $[x]^{\mathbf{j}} = x_j \cdot [x]^{\mathbf{j-1}}$
  $[y^k x^i]^{\mathbf{j}} = y_j^k x_j^i \cdot [y^k x^i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$, $k \in [1, ..., m]$
  $[\alpha y^k x^i]^{\mathbf{j}} = \alpha_j y_j^k x_j^i \cdot [\alpha y^k x^i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$, $k \in [1, ..., m]$
  return $([\alpha y^k x^i]^{\mathbf{j}}, [y^k x^i]^{\mathbf{j}} [y^k]^{\mathbf{j}}, [x^{i-1}]^{\mathbf{j}}, [x^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, y_{\alpha j}, y_{kj})$, $i \in [1, ..., n]$, $k \in [1, ..., m]$
end function

Algorithm 7.8.2 Verification for each $j \in [N]$ for a power of parameter Type-8 s.t. ($\alpha x^i y^k f(x)$, where $i \in [1, ..., n]$, $k \in [1, ..., m]$))

function VERIFY8($[\alpha y^k x^i]^{\mathbf{j}}, [y^k x^i]^{\mathbf{j}} [y^k]^{\mathbf{j}}, [x^{i-1}]^{\mathbf{j}}, [x^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, y_{\alpha j}, y_{kj}, \mathrm{v}_{rd,j-1}$)
  $r_{\alpha j} = \mathcal{R}([y_j^{\alpha}]_1, \mathrm{v}_{rd,j-1})$
  $r_{kj} = \mathcal{R}([y_j^k]_1, \mathrm{v}_{rd,j-1})$, $k \in [1, ..., m]$
  if CheckPOK$([\alpha_j]_1, \mathrm{v}_{rd,j-1}, y_{\alpha j})$ AND **CheckPOK**$([y_j^k]_1, \mathrm{v}_{rd,j-1}, y_{kj})$ **then**
    if Consistent$([x^{i-1}]^{\mathbf{j}} - [x^i]^{\mathbf{j}}; [x]^{\mathbf{j}})$ **then** ▷ for each $i \in [1, ..., n], k \in [1, ..., m]$
      if Consistent$([x^i]^{\mathbf{j}} - [y^k x^i]^{\mathbf{j}}; [y^k]^{\mathbf{j}})$ **then**
        return Consistent$([y^k x^i]^{\mathbf{j}} - [\alpha y^k x^i]^{\mathbf{j}}; [\alpha]^{\mathbf{j}})$
      end if
    end if
  end if
end function

7.9 Type-9: $\alpha z^h x^i y^k$

Let $[\alpha]^{\mathbf{0}} = G_1$ or $[\alpha]^{\mathbf{0}} = \mathbf{G}$, $[x]^{\mathbf{0}} = G_1$ or $[x]^{\mathbf{0}} = \mathbf{G}$, $[x^i]^{\mathbf{0}} = G_1$ or $[x^i]^{\mathbf{0}} = \mathbf{G}$
Let $[y^k x^i]^{\mathbf{0}} = G_1$ or $[y^k x^i]^{\mathbf{0}} = \mathbf{G}$
Let $[y^k x^i]^{\mathbf{0}} = G_1$ or $[y^k x^i]^{\mathbf{0}} = \mathbf{G}$
Let $[z^h x^i y^k]^{\mathbf{0}} = G_1$ or $[z^h x^i y^k]^{\mathbf{0}} = \mathbf{G}$
Let $[\alpha z^h x^i y^k]^{\mathbf{0}} = G_1$ or $[\alpha z^h x^i y^k]^{\mathbf{0}} = \mathbf{G}$

Algorithm 7.9.1 The P_j computes Type-9 parameter s.t. ($\alpha z^h x^i y^k$, where $i \in [1,...,n]$, $k \in [1,...,m]$)
Let $out := ([\alpha z^h y^k x^i]^{\mathbf{j}}, [z^h y^k x^i]^{\mathbf{j}}, [y^k x^i]^{\mathbf{j}}, [y^k]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [x^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, [z_j^h]_1, [y_j^k]_1, y_{\alpha j}, y_{kj}, \mathrm{v}_{rd,j-1})$, $i \in [1,...,n]$, $h \in [1,...,s]$, $k \in [1,...,m]$
$\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1,2\}$

function COMPUTE9($[\alpha y^k x^i]^{\mathbf{j-1}}, [y^k x^i]^{\mathbf{j-1}}, [x^i]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$)
  $\alpha_j, z_j, y_j, x_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
  $y_{\alpha j} = \mathbf{POK}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
  $y_{\kappa j} = \mathbf{POK}([z_j^h]_1, \mathrm{v}_{rd,j-1})$, $h \in [1,...,s]$
  $y_{kj} = \mathbf{POK}([y_j^k]_1, \mathrm{v}_{rd,j-1})$, $k \in [1,...,m]$
  $[\alpha]^{\mathbf{j}} = \alpha_j \cdot [\alpha]^{\mathbf{j-1}}$
  $[x]^{\mathbf{j}} = x_j \cdot [x]^{\mathbf{j-1}}$
  $[x^i]^{\mathbf{j}} = x_j^i \cdot [x^i]^{\mathbf{j-1}}$, $i \in [1,...,n]$
  $[y^k x^i]^{\mathbf{j}} = y_j^k x_j^i \cdot [y^k x^i]^{\mathbf{j-1}}$, $i \in [1,...,n]$, $k \in [1,...,m]$
  $[z^h y^k x^i]^{\mathbf{j}} = z_j^h y_j^k x_j^i \cdot [z^h y^k x^i]^{\mathbf{j-1}}$, $i \in [1,...,n]$, $k \in [1,...,m]$, $h \in [1,...,s]$
  $[\alpha y^k x^i]^{\mathbf{j}} = \alpha_j y_j^k x_j^i \cdot [\alpha y^k x^i]^{\mathbf{j-1}}$, $i \in [1,...,n]$, $k \in [1,...,m]$, $h \in [1,...,s]$
  return (out)
end function

Algorithm 7.9.2 Verification for each $j \in [N]$ for a power of parameter Type-9 s.t.
($\alpha z^h x^i y^k$, where $h \in [1,...,s]$, $i \in [1,...,n]$, $k \in [1,...,m]$))
Let $in := ([\alpha z^h y^k x^i]^{\mathbf{j}}, [z^h y^k x^i]^{\mathbf{j}}, [y^k x^i]^{\mathbf{j}}, [y^k]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [x^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, [z_j^h]_1, [y_j^k]_1, y_{\alpha j}, y_{kj}, \mathrm{v}_{rd,j-1})$, $h \in [1,...,s]$, $i \in [1,...,n]$, $k \in [1,...,m]$

function VERIFY9(in)
  if not CheckPOK$([\alpha_j]_1, \mathrm{v}_{rd,j-1}, y_{\alpha j})$ **then**
    return False
  end if
  if not CheckPOK$([z_j^h]_1, \mathrm{v}_{rd,j-1}, y_{zj})$ **then** ▷ for each $h \in [1,...,s]$
    return False
  end if
  if not CheckPOK$([y_j^k]_1, \mathrm{v}_{rd,j-1}, y_{kj})$ **then** ▷ for each $k \in [1,...,m]$
    return False
  end if
  if Consistent$([x^{i-1}]^{\mathbf{j}} - [x^i]^{\mathbf{j}}; [x]^{\mathbf{j}})$ **then** ▷ for each $i \in [1,...,n]$
    if Consistent$([x^i]^{\mathbf{j}} - [y^k x^i]^{\mathbf{j}}; [y^k]^{\mathbf{j}})$ **then** ▷ for each $i \in [1,...,n], k \in [1,...,m]$
      if Consistent$([y^k x^i]^{\mathbf{j}} - [z^h y^k x^i]^{\mathbf{j}}; [z^h]^{\mathbf{j}})$ **then** ▷ $h \in [1,...,s]$, $i \in [1,...,n]$, $k \in [1,...,m]$
        return Consistent$([z^h y^k x^i]^{\mathbf{j}} - [\alpha z^h y^k x^i]^{\mathbf{j}}; [\alpha]^{\mathbf{j}})$
      end if
    end if
  else
    return False
  end if
end function

7.10 Type-10: $\alpha f_i(x) g_k(y) p_h(z)$

Let $[\alpha]^{\mathbf{0}} = G_1$ or $[\alpha]^{\mathbf{0}} = \mathbf{G}$
Let $H_i' := [f_i(x) g_k(y) p_h(z)]_1$, $H_i := \alpha f_i(x) g_k(y) p_h(z)$, $i \in [1,...,n]$, $h \in [1,...,s]$, $k \in [1,...,m]$.
Let $[H_i]^{\mathbf{0}} := H_i'$.

7.11 Type-11: $\alpha f_i(x,z) g_k(y) p_h(z)$

Let $[\alpha]^{\mathbf{0}} = G_1$ or $[\alpha]^{\mathbf{0}} = \mathbf{G}$
Let $H_i' := [f_i(x,z)) g_k(y) p_h(z)]_1$, $H_i := \alpha f_i(x,z) g_k(y) p_h(z)$, $i \in [1,...,n]$, $h \in [1,...,s]$, $k \in [1,...,m]$.
Let $[H_i]^{\mathbf{0}} := H_i'$.

The Table 1 provides a comprehensive overview of the parameters that need to be generated as part of the setup in a MPC ceremony for the Tokamak scheme. Each parameter listed in the table is associated with a specific type and formula represantation, and for each type, there are corresponding computation

Algorithm 7.10.1 The P_j computes Type-10 parameter s.t. $(\alpha f_i(x)g_k(y)p_h(z)$, where $i \in [1, ..., n]$, $h \in [1, ..., s]$, $k \in [1, ..., m])$
$\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$

1: **function** COMPUTE10($[H_i]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$)
2: $\alpha_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
3: $y_\alpha = \mathbf{POK}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
4: $[\alpha]^{\mathbf{j}} = \alpha_j \cdot [\alpha]^{\mathbf{j-1}}$,
5: $[H_i]^{\mathbf{j}} = \alpha_j \cdot [H_i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$
6: **return** $([H_i]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, y_\alpha)$, $i \in [1, ..., n]$, $h \in [1, ..., s]$, $k \in [1, ..., m]$
7: **end function**

Algorithm 7.10.2 Verification for each $j \in [N]$ for a power of parameter Type-10 s.t. $(\alpha f_i(x)g_k(y)p_h(z)$, where $i \in [1, ..., n]$, $h \in [1, ..., s]$, $k \in [1, ..., m]$,
$\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$

1: **function** VERIFY10($[H_i]^{\mathbf{j-1}}, [H_i]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, \mathrm{v}_{rd,j-1}, y_\alpha$)
2: **if** (**CheckPOK**$([\alpha_\mathrm{j}]_1, \mathrm{v}_{rd,j-1}, y_\alpha)$ **then**
3: **return Consistent**$\big([H_i]^{\mathbf{j-1}} - [H_i]^{\mathbf{j}}; [\alpha]^{\mathbf{j}}\big)$, $i \in [1, ..., n]$, $h \in [1, ..., s]$, $k \in [1, ..., m]$
4: **else**
5: **return** $False$
6: **end if**
7: **end function**

Algorithm 7.11.1 The P_j computes Type-11 parameter s.t. $(\alpha f_i(x, z)g_k(y)p_h(z)$, where $i \in [1, ..., n]$, $h \in [1, ..., s]$, $k \in [1, ..., m])$
$\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$

1: **function** COMPUTE11($[H_i]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}$)
2: $\alpha_j \in_R \mathbb{F}_p^*$ ▷ pick random numbers
3: $y_\alpha = \mathbf{POK}([\alpha_j]_1, \mathrm{v}_{rd,j-1})$
4: $[\alpha]^{\mathbf{j}} = \alpha_j \cdot [\alpha]^{\mathbf{j-1}}$,
5: $[H_i]^{\mathbf{j}} = \alpha_j \cdot [H_i]^{\mathbf{j-1}}$, $i \in [1, ..., n]$
6: **return** $([H_i]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, y_\alpha)$, $i \in [1, ..., n]$, $h \in [1, ..., s]$, $k \in [1, ..., m]$
7: **end function**

Algorithm 7.11.2 Verification for each $j \in [N]$ for a power of parameter Type-11 s.t. $(\alpha f_i(x, z)g_k(y)p_h(z)$, where $i \in [1, ..., n]$, $h \in [1, ..., s]$, $k \in [1, ..., m]$,
$\mathrm{v}_{rd,j-1} := \mathrm{transcript}_{1,j-1}$, where $rd \in \{1, 2\}$

1: **function** VERIFY11($[H_i]^{\mathbf{j-1}}, [H_i]^{\mathbf{j}}, [\alpha]^{\mathbf{j}}, [\alpha_j]_1, \mathrm{v}_{rd,j-1}, y_\alpha$)
2: **if** (**CheckPOK**$([\alpha_\mathrm{j}]_1, \mathrm{v}_{rd,j-1}, y_\alpha)$ **then**
3: **return Consistent**$\big([H_i]^{\mathbf{j-1}} - [H_i]^{\mathbf{j}}; [\alpha]^{\mathbf{j}}\big)$, $i \in [1, ..., n]$, $h \in [1, ..., s]$, $k \in [1, ..., m]$
4: **else**
5: **return** $False$
6: **end if**
7: **end function**

and verification algorithms. The table categorizes these parameters into different types, where each type defines the scope and role of the parameters in the setup. The algorithms listed alongside each type are used to either compute or verify the parameters, ensuring that the setup is performed accurately and securely. This clear mapping of parameters to types and algorithms allows for an organized and systematic approach to the multi-party generation process.

The parameters are represented using a variety of forms, such as α, $\alpha\beta$, x^i, and combinations like $x^i y^k$ or $\alpha f(y)o_i(x)$, which are key to constructing the proof system. Each participant in the setup ceremony follows the computation and verification steps for their assigned type, ensuring that all parameters are computed and verified across multiple parties. This design ensures a secure, decentralized generation of the Tokamak scheme parameters, avoiding any single point of trust. The table serves as a reference for participants to know which algorithms they must execute and which parameters they are responsible for during the setup, enabling a transparent and auditable process. The sigmas column shows the group of the parameters sets of $\boldsymbol{\sigma}_{AI}$, $\boldsymbol{\sigma}_C$, $\boldsymbol{\sigma}_{zk}$, and $\boldsymbol{\sigma}_V$ (see 6.1). Param column shows the each parameter of the

Tokamak scheme. The following two columns (Types and Formula) show the type and and formula of each parameter, respectively.

Table 1: The parameter list for the Tokamak setup scheme in relation to the types of algorithms.

Sigmas	**Param No**	**Param**	**Types**	**Formula**
$\boldsymbol{\sigma}_{A,I}$	**param 1**	α	Type-1	α
$\boldsymbol{\sigma}_{A,I}$	**param 2**	$\left(x^h y^i\right)_{h=0,i=0}^{n-1,s_{\max}-1}$	Type-5	$x^i y^k$
$\boldsymbol{\sigma}_{A,I}$	**param 3**	$\left(\gamma^{-1} L_0(y) o_j(x)\right)_{j=0}^{l_{in}-1}$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_{A,I}$	**param 4**	$\left(\gamma^{-1} L_{-1}(y) o_j(x)\right)_{j=l_{in}}^{l-1}$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_{A,I}$	**param 5**	$\left(\eta_1^{-1} L_i(y) o_j(x)\right)_{i=0,j=l}^{s_{\max}-1,l_D-1}$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_{A,I}$	**param 6**	$\left(\delta^{-1} L_i(y) o_j(x)\right)_{i=0,j=l_D}^{s_{\max}-1,m_D-1}$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_{A,I}$	**param 7**	$\left(\eta_0^{-1} L_i(y) o_j(x) \left(K_{j-1}^2(z) - 1\right)\right)_{i=0,j=l}^{s_{\max}-1,l_D-1}$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_{A,I}$	**param 8**	$\left(\delta^{-1} x^h y^i t_{\mathcal{X}}(x)\right)_{h=0,i=0}^{n-2,s_{\max}-1}$	Type-8	$\alpha x^i y^k f(x)$
$\boldsymbol{\sigma}_{A,I}$	**param 9**	$\left(\delta^{-1} x^h y^i t_{\mathcal{Y}}(y)\right)_{h=0,i=0}^{2n-2,s_{\max}-2}$	Type-8	$\alpha x^i y^k f(x)$
$\boldsymbol{\sigma}_{A,I}$	**param 10**	$\left(\eta_0^{-1} L_i(y) M_j(x,z) t_{\mathcal{Z}}(z)\right)_{i=0,j=l}^{s_{\max}-1,l_D-1}$	Type-11	$\alpha f_i(x,z) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_C$	**param 11**	$\left(\mu^{-1} L_i(y) K_j(z)\right)_{i=0,j=0}^{s_{\max}-1,l_D-l-1}$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_C$	**param 12**	$\left(v^{-1} y^i z^j t_{\mathcal{Y}}(y)\right)_{i=0,j=0}^{s_{\max}-2,2l_D-2l-2}$	Type-8	$\alpha x^i y^k f(x)$
$\boldsymbol{\sigma}_C$	**param 13**	$\left(v^{-1} y^i z^j t_{\mathcal{Z}}(z)\right)_{i=0,j=2(l_D-l)-3}^{2s_{\max}-2,j=0}$	Type-8	$\alpha x^i y^k f(x)$
$\boldsymbol{\sigma}_C$	**param 14**	$\left(\psi_0^{-1} \kappa^h y^i z^j\right)_{h=0,i=0,j=0}^{1,2s_{\max}-3,3(l_D-1)-1}$	Type-9	$\alpha z^h x^i y^k$
$\boldsymbol{\sigma}_C$	**param 15**	$\left(\psi_1^{-1} z^j\right)_{j=0}^{3(l_D-1)-4}$	Type-4	αx^i
$\boldsymbol{\sigma}_C$	**param 16**	$\left(\psi_2^{-1} \kappa^2 y^i z^j\right)_{i=0,j=0}^{s_{\max}-2,l_D-l-1}$	Type-9	$\alpha z^h x^i y^k$
$\boldsymbol{\sigma}_C$	**param 17**	$\left(\psi_3^{-1} \kappa^h z^j\right)_{h=1,j=0}^{2,l_D-l-2}$	Type-7	$\alpha x^i y^k$
$\boldsymbol{\sigma}_{zk}$	**param 18**	β	Type-1	α
$\boldsymbol{\sigma}_{zk}$	**param 19**	δ	Type-1	α
$\boldsymbol{\sigma}_{zk}$	**param 20**	η_1	Type-1	α
$\boldsymbol{\sigma}_{zk}$	**param 21**	$\left(\mu^{-1} y^i t_{\mathcal{Y}}(y)\right)_{i=0}^{l}$	Type-8	$\alpha x^i y^k f(x)$
$\boldsymbol{\sigma}_{zk}$	**param 22**	$\eta_0^{-1} t_{\mathcal{Y}}(y) \sum_{j=1}^{l_D-1} M_j(x,z) t_{\mathcal{Z}}(z)$	Type-11	$\alpha f_i(x,z) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_{zk}$	**param 23**	$\eta_1^{-1} t_{\mathcal{Y}}(y) \sum_{j=l}^{l_D-1} o_j(x)$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_{zk}$	**param 24**	$\eta_0^{-1} t_{\mathcal{Y}}(y) \sum_{j=0}^{l_D-1} o_j(x) \left(K_{j-l}^2(z) - 1\right)$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_{zk}$	**param 25**	$\left(v^{-1} y^i z^j t_{\mathcal{Y}}(y)\right)_{i=s_{\max}-1,j=0}^{s_{\max}+1,2l_D-2l-2}$	Type-8	$\alpha x^i y^k f(x)$
$\boldsymbol{\sigma}_{zk}$	**param 26**	$\left(\psi_0^{-1} \kappa^h y^i z^j\right)_{h=0,i=2s_{\max}-2,j=0}^{1,2s_{\max},3(l_D-l-1)}$	Type-9	$\alpha z^h x^i y^k$
$\boldsymbol{\sigma}_{zk}$	**param 27**	$\left(\psi_2^{-1} \kappa^2 y^i z^j\right)_{i=s_{\max}-1,j=0}^{s_{\max},l_D-l-1}$	Type-9	$\alpha z^h x^i y^k$
$\boldsymbol{\sigma}_V$	**param 28**	β	Type-1	α
$\boldsymbol{\sigma}_V$	**param 29**	γ	Type-1	α
$\boldsymbol{\sigma}_V$	**param 30**	δ	Type-1	α
$\boldsymbol{\sigma}_V$	**param 31**	η_1	Type-1	α
$\boldsymbol{\sigma}_V$	**param 32**	$\mu\eta_0$	Type-2	$\alpha\beta$
$\boldsymbol{\sigma}_V$	**param 33**	$\mu\eta_1$	Type-2	$\alpha\beta$
$\boldsymbol{\sigma}_V$	**param 34**	$\left(x^h y^i\right)_{h=0,i=0}^{n-1,s_{\max}-1}$	Type-5	$x^i y^k$
$\boldsymbol{\sigma}_V$	**param 35**	$\mu^2 \sum_{j=l}^{l_D-1} o_j(x) K_{j-l}(z)$	Type-10	$\alpha f_i(x) g_k(y) p_h(z)$
$\boldsymbol{\sigma}_V$	**param 36**	$\mu^3 v$	Type-2	$\alpha\beta$
$\boldsymbol{\sigma}_V$	**param 37**	$\left(\mu^4 \kappa^h\right)_{h=0}^{2}$	Type-4	αx^i
$\boldsymbol{\sigma}_V$	**param 38**	$\left(\mu^3 \psi_h y^i z^j\right)_{h=0,i=0,j=0}^{3,1,1}$	Type-9	$\alpha z^h x^i y^k$

Furthermore, the relationship between the types, parameters, and the corresponding computation and verification algorithms is summarized in Table 2. In addition to Table 1, Table 2 specifies which parameters are generated and verified according to each type. For example, parameters numbered $[1, 18, 19, 20, 28, 29, 30, 31]$ belong to Type-1, and their computation and verification are handled using the *Compute-1* and *Verify-1* algorithms, respectively.

Table 2: Summary of Types, Parameters, Computation, and Verification Algorithms

Type	Parameter No	Computation Algorithm	Verification Algorithm
Type-1	$[1, 18, 19, 20, 28, 29, 30, 31]$	Compute1	Verify1
Type-2	$[32, 33, 36]$	Compute2	Verify2
Type-3	$[---]$	Compute3	Verify3
Type-4	$[15, 37]$	Compute4	Verify4
Type-5	$[2, 34]$	Compute5	Verify5
Type-6	$[---]$	Compute6	Verify6
Type-7	$[17]$	Compute7	Verify7
Type-8	$[8, 9, 12, 13, 21, 25]$	Compute8	Verify8
Type-9	$[14, 16, 26, 27, 38]$	Compute9	Verify9
Type-10	$[3, 4, 5, 6, 7, 11, 23, 24, 35]$	Compute10	Verify10
Type-11	$[10, 22]$	Compute11	Verify11

7.12 First Version: MPC Setup Ceremony flow with Random Beacon

In this section, we introduce the Multi-Party Computation (MPC) protocol tailored for the Tokamak zkSNARK scheme [1] using Random Beacon based on the paper [2]. We present the MPC protocol for the Tokamak zkSNARK in two phases: the initial round, known as "Powers of τ", which generates universal setup parameters applicable to all circuits within the scheme, and the subsequent phase that generates circuit-specific parameters. This protocol ensures the secure and decentralized generation of the common reference string (CRS) elements required for the Tokamak zkSNARK, thereby enhancing the overall security and integrity of the setup process.

7.12.1 First Round: Powers of τ

Initialization:

1. $[\alpha]^0 := G_1$, $[\gamma]^0 := G_2$ (Type-1)
2. $[\beta]^0 := \mathbf{G}$, $[\delta]^0 := \mathbf{G}$, $[\eta_1]^0 := \mathbf{G}$ (Type-1)
3. $[\mu\eta_0]^0 := G_2$, $[\mu\eta_1]^0 := G_2$ (Type-2)
4. $\left[\mu^3 v\right]^0 := G_2$ (Type-2)
5. $\left[\mu^4 \kappa^h\right]^0 := G_2$, $h \in [0...2]$ (Type-4)
6. $[x^h y^i]^0 := \mathbf{G}$, $h \in [0...n-1], i \in [0...s_{max}-1]$ (Type-5)
7. $\left[\psi_1^{-1} z^k\right]^0 := G_1$, $k \in [0, 3(l_D-1)-4]$ (Type-4)
8. $\left[\psi_3^{-1} \kappa^h z^i\right]^0 := G_1$, $h \in [1, 2]$, $i \in [0, l_D - l - 2]$ (Type-7)
9. $\left[\psi_0^{-1} \kappa^h z^k y^i\right]^0 := G_1$, $h \in [0, 1]$, $k \in [0, 3(l_D-1)-1]$, $i \in [0, 2s_{max}-3]$ (Type-9)
10. $\left[\psi_0^{-1} \kappa^h z^k y^i\right]^0 := G_1$, $h \in [0, 1]$, $k \in [0, 3(l_D-l-1)]$, $i \in [2s_{max}-2, 2s_{max}]$ (Type-9)
11. $\left[\psi_2^{-1} \kappa^2 z^h y^i\right]^0 := G_1$, $h \in [0, l_D - l - 1]$, $i \in [0, s_{\max}-2]$ (Type-9)
12. $\left[\psi_2^{-1} \kappa^2 z^h y^i\right]^0 := G_1$, $h \in [0, l_D - l - 1]$, $i \in [s_{\max}-1, s_{\max}]$ (Type-9)
13. $\left[\mu^3 \psi_h z^k y^i\right]^0 := G_2$, $h \in [0, 3]$, $k \in [0, 1]$, $i \in [0, 1]$ (Type-9)

Computations:

The participant P_j, performs the below computations, where $j \in [N]$:

1. **Compute1**$([\alpha]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\alpha]^{\mathbf{j}}, [\alpha_j]_1, y_{\alpha j}\right)$
2. **Compute1**$([\gamma]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\gamma]^{\mathbf{j}}, [\gamma_j]_1, y_{\gamma j}\right)$
3. **Compute1**$([\beta]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\beta]^{\mathbf{j}}, [\beta j]_1, y_{\beta j}\right)$

4. **Compute1**$([\delta]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\delta]^{\mathbf{j}}, [\delta_j]_1, y_{\delta j}\right)$
5. **Compute1**$([\eta_1]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\eta_1]^{\mathbf{j}}, [\eta_{1j}]_1, y_{\eta_1 j}\right)$
6. **Compute2**$([\mu\eta_0]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\mu\eta_0]^{\mathbf{j}}, [\mu_j]_1, [\eta_{0j}]_1, [\mu_j\eta_{0j}]_2, y_{\mu j}, y_{\eta_0 j}\right)$
7. **Compute2**$([\mu\eta_1]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\mu\eta_0]^{\mathbf{j}}, [\mu_j]_1, [\eta_{1j}]_1, [\mu_j\eta_{1j}]_2, y_{\mu j}, y_{\eta_1 j}\right)$
8. **Compute2**$([\mu^3 v]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\mu^3 v]^{\mathbf{j}}, [\mu_j^3]_1, [v_{1j}]_1, [\mu_j^3 v_j]_2, y_{\mu^3 j}, y_{v_1 j}\right)$
9. **Compute4**$([\mu^4\kappa^h]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\mu^4\kappa^h]^{\mathbf{j}}, [\kappa]^{\mathbf{j}}, [\mu^4]^{\mathbf{j}}, y_{\mu^4 j}\right),\ h \in [0...2]$
10. **Compute5**$([y^i x^h]^{\mathbf{j-1}}, [x^h]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([y^i x^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, y_{hj}\right),\ h \in [0...n-1], i \in [0...s_{max}-1]$
11. **Compute4**$([\psi_1^{-1} z^k]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\psi_1^{-1} z^k]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\psi_1^{-1}]^{\mathbf{j}}, y_{\psi_1^{-1} j}\right),\ k \in [0, 3(l_D-1)-4]$
12. **Compute7**$([\psi_3^{-1} y^k x^i]^{\mathbf{j-1}}, [y^k x^i]^{\mathbf{j-1}}, [x^i]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}) \rightarrow$
$\left([\psi_3^{-1}\kappa^h z^i]^{\mathbf{j}}, [\kappa^h z^i]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [z^{i-1}]^{\mathbf{j}}, [z^i]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\psi_3^{-1}]^{\mathbf{j}}, y_{\psi_3^{-1} j}, y_{hj}, \mathrm{v}_{1,j-1}\right),\ h \in [1,2],\ i \in [0, l_D-l-2]$
13. **Compute9**$([\psi_0^{-1} z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_0^{-1}\kappa^h z^k y^i]^{\mathbf{j}}, [\kappa^h z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [\psi_{0_j}^{-1}]_1, [\kappa_j^h]_1, [z_j^k]_1, y_{\psi_0^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right),$
$h \in [0,1],\ k \in [0, 3(l_D-1)-1],\ i \in [2s_{max}-2, 2s_{max}]$
14. **Compute9**$([\psi_0^{-1} z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_0^{-1}\kappa^h z^k y^i]^{\mathbf{j}}, [\kappa^h z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [\psi_{0_j}^{-1}]_1, [\kappa_j^h]_1, [z_j^k]_1, y_{\psi_0^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right),$
$h \in [0,1],\ k \in [0, 3(l_D-1)-1],\ i \in [0, 2s_{max}-3]$
15. **Compute9**$([\psi_2^{-1} z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_2^{-1}\kappa^2 z^k y^i]^{\mathbf{j}}, [\kappa^2 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^2]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_2^{-1}]^{\mathbf{j}}, [\psi_{2_j}^{-1}]_1, [\kappa_j^2]_1, [z_j^k]_1, y_{\psi_2^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right),$
$k \in [0, l_D-l-1],\ i \in [0, s_{\max}-2]$
16. **Compute9**$([\psi_2^{-1} z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_2^{-1}\kappa^2 z^k y^i]^{\mathbf{j}}, [\kappa^2 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^2]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_2^{-1}]^{\mathbf{j}}, [\psi_{2_j}^{-1}]_1, [\kappa_j^2]_1, [z_j^k]_1, y_{\psi_2^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right),$
$k \in [0, l_D-l-1],\ i \in [s_{\max}-1, s_{\max}]$
17. **Compute9**$([\psi_h z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_h\mu^3 z^k y^i]^{\mathbf{j}}, [\mu^3 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\mu^3]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_h]^{\mathbf{j}}, [\psi_{hj}]_1, [\mu_j^3]_1, [z_j^k]_1, y_{\psi_h^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right),$
$k \in [0,1],\ i \in [0,1]$

Let $J-1$ be the time-slot where P_N sends their message.
Let $(\alpha', \gamma', \beta', \delta', {\eta_1}') := \text{RandomBeacon}(J, 5)$
Let $((x^h y^i)') := \text{RandomBeacon}(J, 1)$
Let $(\psi_0^{-1}\kappa^h z^k)' := \text{RandomBeacon}(J, 1)$, for each $h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$
Let $(\psi_1^{-1})' := \text{RandomBeacon}(J, 1)$
Let $(\psi_2^{-1}\kappa^2 z^h)' := \text{RandomBeacon}(J, 1)$, for each $h \in [0, l_D-l-1]$
Let $(\psi_3^{-1}\kappa^h)' := \text{RandomBeacon}(J, 1)$, for each $h \in [1,2]$
Let $(\psi_0^{-1}\kappa^h z^k)' := \text{RandomBeacon}(J, 1)$, for each $h \in [0,1]$, $k \in [0, 3(l_D-l-1)]$
Let $(\psi_2^{-1}\kappa^2 z^h)' := \text{RandomBeacon}(J, 1)$, for each $h \in [0, l_D-l-1]$
Let $(\mu^3\psi_h z^k)' := \text{RandomBeacon}(J, 1)$, for each $h \in [0,3]$, $k \in [0,1]$

1. $[\alpha] := \alpha' \cdot [\alpha]^{\mathbf{N}}$, $[\gamma] := \gamma' \cdot [\gamma]^{\mathbf{N}}$
2. $[\beta] := \beta' \cdot [\beta]^{\mathbf{N}}$, $[\delta] := \delta' \cdot [\delta]^{\mathbf{N}}$, $[\eta_1] := {\eta_1}' \cdot [\eta_1]^{\mathbf{N}}$
3. $[\mu\eta_0] := \mu{\eta_0}' \cdot [\mu\eta_0]^{\mathbf{N}}$
4. $[\mu\eta_1] := \mu{\eta_1}' \cdot [\mu\eta_1]^{\mathbf{N}}$
5. $[\mu^3 v] := (\mu^3 v)' \cdot [\mu^3 v]^{\mathbf{N}}$

6. $\left[\mu^4\kappa^h\right] := (\mu^4\kappa^h)' \cdot \left[\mu^4\kappa^h\right]^{\mathbf{N}}$, $h \in [0...2]$
7. $\left[x^h y^i\right] := (x^h y^i)' \cdot \left[x^h y^i\right]^{\mathbf{N}}$, $h \in [0...n-1], i \in [0...s_{max}-1]$
8. $\left[\psi_1^{-1} z^k\right]_1 := \left[\psi_1^{-1} z^k\right]^{\mathbf{N}} \cdot (\psi_1^{-1})'$, $k \in [0, 3(l_D-1)-4]$
9. $\left[\psi_3^{-1}\kappa^h z^i\right]_1 := \left[psi_3^{-1}\kappa^h z^i\right]^{\mathbf{N}} \cdot (\psi_3^{-1}\kappa^h)'$, $h \in [1,2]$, $i \in [0, l_D - l - 2]$
10. $\left[\psi_0^{-1}\kappa^h z^k y^i\right]_1 := \left[\psi_0^{-1}\kappa^h z^k y^i\right]^{\mathbf{N}} \cdot (\psi_0^{-1}\kappa^h z^k)'$, $h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$
11. $\left[\psi_0^{-1}\kappa^h z^k y^i\right]_1 := \left[\psi_0^{-1}\kappa^h z^k y^i\right]^{\mathbf{N}} \cdot (\psi_0^{-1}\kappa^h z^k)'$, $h \in [0,1]$, $k \in [0, 3(l_D-l-1)]$
12. $\left[\psi_2^{-1}\kappa^2 z^h y^i\right]_1 := \left[\psi_2^{-1}\kappa^2 z^h y^i\right]^{\mathbf{N}} \cdot (\psi_2^{-1}\kappa^2 z^h)'$, $h \in [0, l_D - l - 1]$, $i \in [0, s_{\max} - 2]$
13. $\left[\psi_2^{-1}\kappa^2 z^h y^i\right]_1 := \left[\psi_2^{-1}\kappa^2 z^h y^i\right]^{\mathbf{N}} \cdot (\psi_2^{-1}\kappa^2 z^h)'$, $h \in [0, l_D - l - 1]$, $i \in [s_{\max} - 2, s_{\max}]$
14. $\left[\mu^3\psi_h z^k y^i\right]_1 := \left[\mu^3\psi_h z^k y^i\right]^{\mathbf{N}} \cdot (\mu^3\psi_h z^k)'$, $h \in [0,3]$, $k \in [0,1]$, $i \in [0,1]$

Verifications:

The protocol verifier runs the following algorithms for each $j \in [N]$,

1. **Verify1** $\left([\alpha]^{\mathbf{j-1}}, [\alpha]^{\mathbf{j}}, [\alpha_{\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\alpha,\mathrm{j}}\right)$
2. **Verify1** $\left([\gamma]^{\mathbf{j-1}}, [\gamma]^{\mathbf{j}}, [\gamma_{\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\gamma,\mathrm{j}}\right)$
3. **Verify1** $\left([\beta]^{\mathbf{j-1}}, [\beta]^{\mathbf{j}}, [\beta_{\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\beta,\mathrm{j}}\right)$
4. **Verify1** $\left([\delta]^{\mathbf{j-1}}, [\delta]^{\mathbf{j}}, [\delta_{\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\delta,\mathrm{j}}\right)$
5. **Verify1** $\left([\eta_1]^{\mathbf{j-1}}, [\eta_1]^{\mathbf{j}}, [\eta_{1\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\eta_1,\mathrm{j}}\right)$
6. **Verify2** $\left([\mu\eta_0]^{\mathbf{j-1}}, [\mu\eta_0]^{\mathbf{j}}, [\mu_j]_1, [\eta_{0\,j}]_1, [\mu_j\eta_{0\,j}]_2, y_{\mu j}, y_{\eta_0 j}\right)$
7. **Verify2** $\left([\mu^3 v]^{\mathbf{j-1}}, [\mu^3 v]^{\mathbf{j}}, [\mu_j^3]_1, [v_j]_1, [\mu_j^3 v_j]_2, y_{\mu^3 j}, y_{vj}\right)$
8. **Verify4** $\left([\mu^4\kappa^h]^{\mathbf{j}}, [\mu^4]^{\mathbf{j}}, [\kappa]^{\mathbf{j}}, [\kappa^{h-1}]^{\mathbf{j}}, \mathrm{v}_{1,j-1}, y_{\mu^4 j}\right)$, $h \in [0...2]$
9. **Verify5** $\left([y^i x^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, \mathrm{v}_{1,j-1}, y_{ij}\right)$, $h \in [0...n-1], i \in [0...s_{max}-1]$
10. **Verify4** $\left([\psi_1^{-1} z^k]^{\mathbf{j}}, [\psi_1^{-1}]^{\mathbf{j}}, [z]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, \mathrm{v}_{1,j-1}, y_{\psi_1^{-1} j}\right)$, $k \in [0, 3(l_D-1)-4]$
11. **Verify7** $\Big([\psi_3^{-1}\kappa^h z^i]^{\mathbf{j}}, [\kappa^h z^i]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [z^{i-1}]^{\mathbf{j}}, [z^i]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\psi_3^{-1}]^{\mathbf{j}}, y_{\psi_3^{-1} j}, y_{hj}, \mathrm{v}_{1,j-1}\Big)$,
    $h \in [1,2]$, $i \in [0, l_D - l - 2]$
12. **Verify9** $\Big([\psi_0^{-1}\kappa^h z^k y^i]^{\mathbf{j}}, [\kappa^h z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [{\psi_0}_j^{-1}]_1, [\kappa_j^h]_1, [z_j^k]_1, y_{\psi_0^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\Big)$,
    $h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$, $i \in [0, 2s_{max}-3]$
13. **Verify9** $\Big([\psi_0^{-1}\kappa^h z^k y^i]^{\mathbf{j}}, [\kappa^h z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [{\psi_0}_j^{-1}]_1, [\kappa_j^h]_1, [z_j^k]_1, y_{\psi_0^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\Big)$,
    $h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$, $i \in [2s_{max}-2, 2s_{max}]$
14. **Verify9** $\Big([\psi_2^{-1}\kappa^2 z^k y^i]^{\mathbf{j}}, [\kappa^2 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^2]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [{\psi_2}_j^{-1}]_1, [\kappa_j^2]_1, [z_j^k]_1, y_{\psi_2^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\Big)$,
    $k \in [0, l_D - l - 1]$, $i \in [0, s_{\max} - 2]$
15. **Verify9** $\Big([\psi_2^{-1}\kappa^2 z^k y^i]^{\mathbf{j}}, [\kappa^2 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^2]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [{\psi_2}_j^{-1}]_1, [\kappa_j^2]_1, [z_j^k]_1, y_{\psi_2^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\Big)$,
    $k \in [0, l_D - l - 1]$, $i \in [s_{\max} - 1, s_{\max}]$

7.12.2 Second Round

Initialization:

1. $[\delta^{-1}]^{\mathbf{0}} := G_1$, $[v^{-1}]^{\mathbf{0}} := G_1$, $[\mu^{-1}]^{\mathbf{0}} := G_1$
2. $[y^k x^h]^{\mathbf{0}} := G_1$, $h \in [0, 2n-2]$, $i \in [0, s_{\max}-2]$
3. $[\delta^{-1} y^i x^h]^{\mathbf{0}} := [t_{\mathcal{X}}(x)]_1$, $h \in [0, n-2]$, $i \in [0, s_{\max}-2]$ (Type-8)
4. $[\delta^{-1} y^i x^h]^{\mathbf{0}} := [t_{\mathcal{Y}}(y)]_1$, $h \in [0, 2n-2]$, $i \in [0, s_{\max}-2]$ (Type-8)
5. $[v^{-1} y^i z^h]^{\mathbf{0}} := [t_{\mathcal{Y}}(y)]_1$, $h \in [0, 2l_D - 2l - 2]$, $i \in [0, s_{\max}-2]$ (Type-8)
6. $[v^{-1} y^i z^h]^{\mathbf{0}} := [t_{\mathcal{Z}}(z)]_1$, $h \in [2(l_D - l) - 3, 0]$, $i \in [0, 2s_{\max}-2]$ (Type-8)
7. $[v^{-1} y^i z^h]^{\mathbf{0}} := [t_{\mathcal{Z}}(z)]_1$, $h \in [0, 2l_D - 2l - 2]$, $i \in [s_{\max}-1, s_{\max}+1]$ (Type-8)
8. $[\mu^{-1} y^i z^h]^{\mathbf{0}} := [t_{\mathcal{Y}}(y)]_1$, $h = 0$, $i \in [0, l]$ (Type-8)
9. $[\gamma^{-1}]^{\mathbf{0}} = G_1$,
10. $[\eta_0^{-1}]^{\mathbf{0}} = G_1$, $[\eta_1^{-1}]^{\mathbf{0}} = G_1$
11. $[\mu^2]^{\mathbf{0}} = G_1$, $[\mu^{-1}]^{\mathbf{0}} = G_1$
12. $H_{1i} := \gamma^{-1} L_0(y) o_i(x)$
    $[H_{1i}]^{\mathbf{0}} := [L_0(y) o_i(x)]_1$, $i \in [0, l_{in}-1]$ (Type-10)
13. $H_{2i} := \gamma^{-1} L_{-1}(y) o_i(x)$
    $[H_{2i}]^{\mathbf{0}} := [L_{-1}(y) o_i(x)]_1$, $i \in [l_{in}, l-1]$ (Type-10)
14. $H_{3ik} := \eta_1^{-1} L_i(y) o_k(x)$
    $[H_{3ik}]^{\mathbf{0}} := [L_i(y) o_k(x)]_1$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$ (Type-10)
15. $H_{4ik} := \delta^{-1} L_i(y) o_k(x)$
    $[H_{4ik}]^{\mathbf{0}} := [L_i(y) o_k(x)]_1$, $i \in [0, s_{\max}-1]$, $k \in [l_D, m_D-1]$ (Type-10)
16. $H_{5ik} := \eta_0^{-1} L_i(y) o_k(x) \left(K_{k-1}^2(z) - 1\right)$
    $[H_{5ik}]^{\mathbf{0}} := \left[L_i(y) o_k(x) \left(K_{k-1}^2(z) - 1\right)\right]_1$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$ (Type-10)
17. $H_6 := \eta_0^{-1} t_{\mathcal{Y}}(y) \sum_{k=0}^{l_D-1} o_k(x) \left(K_{k-l}^2(z) - 1\right)$
    $[H_6]^{\mathbf{0}} := \left[t_{\mathcal{Y}}(y) \sum_{k=0}^{l_D-1} o_k(x) \left(K_{k-l}^2(z) - 1\right)\right]_1$, (Type-10)
18. $H_7 := \eta_1^{-1} t_{\mathcal{Y}}(y) \sum_{k=l}^{l_D-1} o_k(x)$
    $[H_7]^{\mathbf{0}} := [t_{\mathcal{Y}}(y) \sum_{k=l}^{l_D-1} o_k(x)]_1$, (Type-10)
19. $H_8 := \mu^2 \sum_{k=l}^{l_D-1} o_k(x) K_{k-l}(z)$
    $[H_8]^{\mathbf{0}} := \left[\sum_{k=l}^{l_D-1} o_k(x) K_{k-l}(z)\right]_2$, (Type-10)
20. $H_{9ik} := \mu^{-1} L_i(y) K_k(z)$
    $[H_{9ik}]^{\mathbf{0}} := [L_i(y) K_k(z)]_1$, $i \in [0, s_{\max}-1]$, $k \in [0, l_D-l-1]$ (Type-10)
21. $H_{10ik} := \eta_0^{-1} L_i(y) M_k(x,z) t_{\mathcal{Z}}(z)$
    $[H_{10ik}]^{\mathbf{0}} := [L_i(y) M_k(x,z) t_{\mathcal{Z}}(z)]_1$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$ (Type-10)
22. $H_{11} := \eta_0^{-1} t_{\mathcal{Y}}(y) \sum_{k=1}^{l_D-1} M_k(x,z) t_{\mathcal{Z}}(z)$
    $[H_{11}]^{\mathbf{0}} := \left[t_{\mathcal{Y}}(y) \sum_{k=1}^{l_D-1} M_k(x,z) t_{\mathcal{Z}}(z)\right]_1$ (Type-10)

Computations:

For $j \in [N]$, P_j outputs:

1. $\mathbf{Compute8}([\delta^{-1}y^i x^h]^{\mathbf{j-1}}, [y^i x^h]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([\delta^{-1}y^i x^h]^{\mathbf{j}}, [y^i x^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [x^{h-1}]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, y_{\delta^{-1}j}, y_{ij}\big)$, $h \in [0, n-2]$, $i \in [0, s_{\max}-2]$
2. $\mathbf{Compute8}([\delta^{-1}y^i x^h]^{\mathbf{j-1}}, [y^i x^h]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([\delta^{-1}y^i x^h]^{\mathbf{j}}, [y^i x^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [x^{h-1}]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, y_{\delta^{-1}j}, y_{ij}\big)$, $h \in [0, 2n-2]$, $i \in [0, s_{\max}-2]$
3. $\mathbf{Compute8}([v^{-1}y^i z^h]^{\mathbf{j-1}}, [y^i z^h]^{\mathbf{j-1}}, [z]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}\big)$, $h \in [0, 2l_D - 2l - 2]$, $i \in [0, s_{\max}-2]$
4. $\mathbf{Compute8}([v^{-1}y^i z^h]^{\mathbf{j-1}}, [y^i z^h]^{\mathbf{j-1}}, [z]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}\big)$, $h \in [2(l_D - l) - 3, 0]$, $i \in [0, 2s_{\max}-2]$
5. $\mathbf{Compute8}([v^{-1}y^i z^h]^{\mathbf{j-1}}, [y^i z^h]^{\mathbf{j-1}}, [z]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}\big)$, $h \in [0, 2l_D - 2l - 2]$, $i \in [s_{\max}-1, s_{\max}+1]$
6. $\mathbf{Compute8}([\mu^{-1}y^i z^h]^{\mathbf{j-1}}, [y^i z^h]^{\mathbf{j-1}}, [z]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([\mu^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\mu^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}\big)$, $h = 0$, $i \in [0, l]$
7. $\mathbf{Compute10}([H_{1i}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{1i}]^{\mathbf{j}}, [\gamma^{-1}]^{\mathbf{j}}, [\gamma_j^{-1}]_1, y_{\gamma^{-1}})$, $i \in [0, l_{in}-1]$
8. $\mathbf{Compute10}([H_{2i}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{2i}]^{\mathbf{j}}, [\gamma^{-1}]^{\mathbf{j}}, [\gamma_j^{-1}]_1, y_{\gamma^{-1}})$, $i \in [l_{in}, l-1]$
9. $\mathbf{Compute10}([H_{3ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{3ik}]^{\mathbf{j}}, [\eta_1^{-1}]^{\mathbf{j}}, [\eta_{1j}^{-1}]_1, y_{\eta_1^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
10. $\mathbf{Compute10}([H_{4ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{4ik}]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, [\delta_j^{-1}]_1, y_{\delta^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [l_D, m_D-1]$
11. $\mathbf{Compute10}([H_{5ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{5ik}]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, y_{\eta_0^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
12. $\mathbf{Compute10}([H_6]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_6]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, y_{\eta_0^{-1}})$,
13. $\mathbf{Compute10}([H_7]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_7]^{\mathbf{j}}, [\eta_1^{-1}]^{\mathbf{j}}, [\eta_{1j}^{-1}]_1, y_{\eta_1^{-1}})$,
14. $\mathbf{Compute10}([H_8]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_8]^{\mathbf{j}}, [\mu^2]^{\mathbf{j}}, [\mu_j^2]_1, y_{\mu^2})$,
15. $\mathbf{Compute10}([H_{9ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{9ik}]^{\mathbf{j}}, [\mu^{-1}]^{\mathbf{j}}, [\mu_j^{-1}]_1, y_{\mu^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [0, l_D-l-1]$
16. $\mathbf{Compute11}([H_{10ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{10ik}]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, y_{\eta_0^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
17. $\mathbf{Compute11}([H_{11}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{11}]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, y_{\eta_0^{-1}})$,

Finally, $J-1$ be the time-slot where P_N sends their message.

Let $(\delta^{-1}x^h y^i)' := \mathrm{RandomBeacon}(J, 1)$, for each $h \in [0, n-2]$, $i \in [0, s_{\max}-2]$
Let $(\delta^{-1}x^h y^i)' := \mathrm{RandomBeacon}(J, 1)$, for each $h \in [0, 2n-2]$, $i \in [0, s_{\max}-2]$
Let $(v^{-1}z^h y^i)' := \mathrm{RandomBeacon}(J, 1)$, for each $h \in [0, 2l_D - 2l - 2]$, $i \in [0, s_{\max}-2]$
Let $(v^{-1}z^h y^i)' := \mathrm{RandomBeacon}(J, 1)$, for each $h \in [2(l_D - l) - 3, 0]$, $i \in [0, 2s_{\max}-2]$
Let $(v^{-1}z^h y^i)' := \mathrm{RandomBeacon}(J, 1)$, for each $h \in [0, 2l_D - 2l - 2]$, $i \in [s_{\max}-1, s_{\max}+1]$
Let $(\mu^{-1}z^h y^i)' := \mathrm{RandomBeacon}(J, 1)$, for each $h = 0$, $i \in [0, l]$
Let $(H_{1i})' := \mathrm{RandomBeacon}(J, 1)$, for each $i \in [0, l_{in}-1]$
Let $(H_{2i})' := \mathrm{RandomBeacon}(J, 1)$, for each $i \in [l_{in}, l-1]$
Let $(H_{3ik})' := \mathrm{RandomBeacon}(J, 1)$, for each $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
Let $(H_{4ik})' := \mathrm{RandomBeacon}(J, 1)$, for each $i \in [0, s_{\max}-1]$, $k \in [l_D, m_D-1]$
Let $(H_{5ik})' := \mathrm{RandomBeacon}(J, 1)$, for each $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
Let $(H_6)' := \mathrm{RandomBeacon}(J, 1)$,
Let $(H_7)' := \mathrm{RandomBeacon}(J, 1)$,
Let $(H_8)' := \mathrm{RandomBeacon}(J, 1)$,
Let $(H_{9ik})' := \mathrm{RandomBeacon}(J, 1)$, for each $i \in [0, s_{\max}-1]$, $k \in [0, l_D-l-1]$
Let $(H_{10ik})' := \mathrm{RandomBeacon}(J, 1)$, for each $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
Let $(H_{11})' := \mathrm{RandomBeacon}(J, 1)$,

1. $\left[\delta^{-1}x^h y^i\right]_1 := (\delta^{-1}x^h y^i)' \cdot \left[\delta^{-1}x^h y^i\right]^{\mathbf{N}}$, $h \in [0, n-2]$, $i \in [0, s_{\max}-2]$
2. $\left[\delta^{-1}x^h y^i\right]_1 := (\delta^{-1}x^h y^i)' \cdot \left[\delta^{-1}x^h y^i\right]^{\mathbf{N}}$, $h \in [0, 2n-2]$, $i \in [0, s_{\max}-2]$
3. $\left[v^{-1}z^h y^i\right]_1 := (v^{-1}z^h y^i)' \cdot \left[v^{-1}z^h y^i\right]^{\mathbf{N}}$, $h \in [0, 2l_D-2l-2]$, $i \in [0, s_{\max}-2]$
4. $\left[v^{-1}z^h y^i\right]_1 := (v^{-1}z^h y^i)' \cdot \left[v^{-1}z^h y^i\right]^{\mathbf{N}}$, $h \in [2(l_D-l)-3, 0]$, $i \in [0, 2s_{\max}-2]$
5. $\left[v^{-1}z^h y^i\right]_1 := (v^{-1}z^h y^i)' \cdot \left[v^{-1}z^h y^i\right]^{\mathbf{N}}$, $h \in [0, 2l_D-2l-2]$, $i \in [s_{\max}-1, s_{\max}+1]$
6. $\left[\mu^{-1}z^h y^i\right]_1 := (\mu^{-1}z^h y^i)' \cdot \left[\mu^{-1}z^h y^i\right]^{\mathbf{N}}$, $h = 0$, $i \in [0, l]$
7. $[H_{1i}]_1 := (H_{1i})' \cdot [H_{1i}]^{\mathbf{N}}$, $i \in [0, l_{in}-1]$
8. $[H_{2i}]_1 := (H_{2i})' \cdot [H_{2i}]^{\mathbf{N}}$, $i \in [l_{in}, l-1]$
9. $[H_{3ik}]_1 := (H_{3ik})' \cdot [H_{3ik}]^{\mathbf{N}}$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
10. $[H_{4ik}]_1 := (H_{4ik})' \cdot [H_{4ik}]^{\mathbf{N}}$, $i \in [0, s_{\max}-1]$, $k \in [l_D, m_D-1]$
11. $[H_{5ik}]_1 := (H_{5ik})' \cdot [H_6]^{\mathbf{N}}$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
12. $[H_6]_1 := (H_6)' \cdot [H_6]^{\mathbf{N}}$
13. $[H_7]_1 := (H_7)' \cdot [H_7]^{\mathbf{N}}$
14. $[H_8]_1 := (H_8)' \cdot [H_8]^{\mathbf{N}}$
15. $[H_{9ik}]_1 := (H_{9ik})' \cdot [H_{9ik}]^{\mathbf{N}}$, $i \in [0, s_{\max}-1]$, $k \in [0, l_D-l-1]$
16. $[H_{10ik}]_1 := (H_{10ik})' \cdot [H_{10ik}]^{\mathbf{N}}$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
17. $[H_{11}]_1 := (H_{11})' \cdot [H_{11}]^{\mathbf{N}}$

,

Verifications:

The protocol verifier runs the following algorithms for each $j \in [N]$,

1. **Verify8** $\left([\delta^{-1}y^i x^h]^{\mathbf{j}}, [y^i x^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [x^{h-1}]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, y_{\delta^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\right)$, $h \in [0, n-2]$, $i \in [0, s_{\max}-2]$
2. **Verify8** $\left([\delta^{-1}y^i x^h]^{\mathbf{j}}, [y^i x^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [x^{h-1}]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, y_{\delta^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\right)$, $h \in [0, 2n-2]$, $i \in [0, s_{\max}-2]$
3. **Verify8** $\left([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\right)$, $h \in [0, 2l_D-2l-2]$, $i \in [0, s_{\max}-2]$
4. **Verify8** $\left([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\right)$, $h \in [2(l_D-l)-3, 0]$, $i \in [0, 2s_{\max}-2]$
5. **Verify8** $\left([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\right)$, $h \in [0, 2l_D-2l-2]$, $i \in [s_{\max}-1, s_{\max}+1]$
6. **Verify8** $\left([\mu^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\mu^{-1}]^{\mathbf{j}}, y_{\mu^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\right)$, $h = 0$, $i \in [0, l]$
7. **Verify10** $\left([H_{1i}]^{\mathbf{j-1}}, [H_{1i})]^{\mathbf{j}}, [\gamma^{-1}]^{\mathbf{j}}, [\gamma_j^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\gamma^{-1}}\right)$, $i \in [0, l_{in}-1]$
8. **Verify10** $\left([H_{2i}]^{\mathbf{j-1}}, [H_{2i})]^{\mathbf{j}}, [\gamma^{-1}]^{\mathbf{j}}, [\gamma_j^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\gamma^{-1}}\right)$, $i \in [l_{in}, l-1]$
9. **Verify10** $\left([H_{3ik}]^{\mathbf{j-1}}, [H_{3ik})]^{\mathbf{j}}, [\eta_1^{-1}]^{\mathbf{j}}, [\eta_{1j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_1^{-1}}\right)$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
10. **Verify10** $\left([H_{4ik}]^{\mathbf{j-1}}, [H_{4ik})]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, [\delta_j^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\delta^{-1}}\right)$, $i \in [0, s_{\max}-1]$, $k \in [l_D, m_D-1]$

11. **Verify10** $\left([H_{5ik}]^{\mathbf{j-1}}, [H_{5ik})]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_0^{-1}}\right)$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
12. **Verify10** $\left([H_6]^{\mathbf{j-1}}, [H_6)]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_0^{-1}}\right)$
13. **Verify10** $\left([H_7]^{\mathbf{j-1}}, [H_7)]^{\mathbf{j}}, [\eta_1^{-1}]^{\mathbf{j}}, [\eta_{1j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_1^{-1}}\right)$
14. **Verify10** $\left([H_8]^{\mathbf{j-1}}, [H_8)]^{\mathbf{j}}, [\mu^2]^{\mathbf{j}}, [\mu_j^2]_1, \mathrm{v}_{2,j-1}, y_{\mu^2}\right)$
15. **Verify10** $\left([H_{9ik}]^{\mathbf{j-1}}, [H_{9ik})]^{\mathbf{j}}, [\mu^{-1}]^{\mathbf{j}}, [\mu_j^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\mu^{-1}}\right)$, $i \in [0, s_{\max}-1]$, $k \in [0, l_D-l-1]$
16. **Verify11** $\left([H_{10ik}]^{\mathbf{j-1}}, [H_{10ik})]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_0^{-1}}\right)$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D-1]$
17. **Verify11** $\left([H_{11}]^{\mathbf{j-1}}, [H_{11})]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_0^{-1}}\right)$

7.13 Second Version: MPC Setup Ceremony flow without Random Beacon

7.13.1 First Round: Powers of τ

Initialization:

1. $[\alpha]^0 := G_1$, $[\gamma]^0 := G_2$ (Type-1)
2. $[\beta]^0 := \mathbf{G}$, $[\delta]^0 := \mathbf{G}$, $[\eta_1]^0 := \mathbf{G}$ (Type-1)
3. $[\mu\eta_0]^0 := G_2$, $[\mu\eta_1]^0 := G_2$ (Type-2)
4. $\left[\mu^3 v\right]^0 := G_2$ (Type-2)
5. $\left[\mu^4\kappa^h\right]^0 := G_2$, $h \in [0...2]$ (Type-4)
6. $[x^h y^i]^0 := \mathbf{G}$, $h \in [0...n-1], i \in [0...s_{max}-1]$ (Type-5)
7. $\left[\psi_1^{-1} z^k\right]^0 := G_1$, $k \in [0, 3(l_D-1)-4]$ (Type-4)
8. $\left[\psi_3^{-1}\kappa^h z^i\right]^0 := G_1$, $h \in [1,2]$, $i \in [0, l_D-l-2]$ (Type-7)
9. $\left[\psi_0^{-1}\kappa^h z^k y^i\right]^0 := G_1$, $h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$, $i \in [0, 2s_{max}-3]$ (Type-9)
10. $\left[\psi_0^{-1}\kappa^h z^k y^i\right]^0 := G_1$, $h \in [0,1]$, $k \in [0, 3(l_D-l-1)]$, $i \in [2s_{max}-2, 2s_{max}]$ (Type-9)
11. $\left[\psi_2^{-1}\kappa^2 z^h y^i\right]^0 := G_1$, $h \in [0, l_D-l-1]$, $i \in [0, s_{\max}-2]$ (Type-9)
12. $\left[\psi_2^{-1}\kappa^2 z^h y^i\right]^0 := G_1$, $h \in [0, l_D-l-1]$, $i \in [s_{\max}-1, s_{\max}]$ (Type-9)
13. $\left[\mu^3\psi_h z^k y^i\right]^0 := G_2$, $h \in [0,3]$, $k \in [0,1]$, $i \in [0,1]$ (Type-9)

Computations:

The participant P_j, performs the below computations, where $j \in [N]$:

1. **Compute1**$([\alpha]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\alpha]^{\mathbf{j}}, [\alpha_j]_1, y_{\alpha j}\right)$
2. **Compute1**$([\gamma]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\gamma]^{\mathbf{j}}, [\gamma_j]_1, y_{\gamma j}\right)$
3. **Compute1**$([\beta]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\beta]^{\mathbf{j}}, [\beta j]_1, y_{\beta j}\right)$
4. **Compute1**$([\delta]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\delta]^{\mathbf{j}}, [\delta_j]_1, y_{\delta j}\right)$
5. **Compute1**$([\eta_1]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\eta_1]^{\mathbf{j}}, [\eta_{1j}]_1, y_{\eta_1 j}\right)$
6. **Compute2**$([\mu\eta_0]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\mu\eta_0]^{\mathbf{j}}, [\mu_j]_1, [\eta_{0j}]_1, [\mu_j\eta_{0j}]_2, y_{\mu j}, y_{\eta_0 j}\right)$
7. **Compute2**$([\mu\eta_1]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\mu\eta_0]^{\mathbf{j}}, [\mu_j]_1, [\eta_{1j}]_1, [\mu_j\eta_{1j}]_2, y_{\mu j}, y_{\eta_1 j}\right)$
8. **Compute2**$([\mu^3 v]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\mu^3 v]^{\mathbf{j}}, [\mu_j^3]_1, [v_{1j}]_1, [\mu_j^3 v_j]_2, y_{\mu^3 j}, y_{v_1 j}\right)$
9. **Compute4**$([\mu^4\kappa^h]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \to \left([\mu^4\kappa^h]^{\mathbf{j}}, [\kappa]^{\mathbf{j}}, [\mu^4]^{\mathbf{j}}, y_{\mu^4 j}\right)$, $h \in [0...2]$

10. **Compute5**$([y^i x^h]^{\mathbf{j-1}}, [x^h]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([y^i x^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, y_{hj}\right)$, $h \in [0...n-1], i \in [0...s_{max}-1]$
11. **Compute4**$([\psi_1^{-1} z^k]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow \left([\psi_1^{-1} z^k]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\psi_1^{-1}]^{\mathbf{j}}, y_{\psi_1^{-1} j}\right)$, $k \in [0, 3(l_D-1)-4]$
12. **Compute7**$([\psi_3^{-1} y^k x^i]^{\mathbf{j-1}}, [y^k x^i]^{\mathbf{j-1}}, [x^i]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{rd,j-1}) \rightarrow$
$\left([\psi_3^{-1} \kappa^h z^i]^{\mathbf{j}}, [\kappa^h z^i]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [z^{i-1}]^{\mathbf{j}}, [z^i]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\psi_3^{-1}]^{\mathbf{j}}, y_{\psi_3^{-1} j}, y_{hj}, \mathrm{v}_{1,j-1}\right)$, $h \in [1,2]$, $i \in [0, l_D - l - 2]$
13. **Compute9**$([\psi_0^{-1} z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_0^{-1} \kappa^h z^k y^i]^{\mathbf{j}}, [\kappa^h z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [{\psi_0}_j^{-1}]_1, [\kappa_j^h]_1, [z_j^k]_1, y_{\psi_0^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right)$,
$h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$, $i \in [2s_{max}-2, 2s_{max}]$
14. **Compute9**$([\psi_0^{-1} z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_0^{-1} \kappa^h z^k y^i]^{\mathbf{j}}, [\kappa^h z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [{\psi_0}_j^{-1}]_1, [\kappa_j^h]_1, [z_j^k]_1, y_{\psi_0^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right)$,
$h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$, $i \in [0, 2s_{max}-3]$
15. **Compute9**$([\psi_2^{-1} z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_2^{-1} \kappa^2 z^k y^i]^{\mathbf{j}}, [\kappa^2 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^2]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_2^{-1}]^{\mathbf{j}}, [{\psi_2}_j^{-1}]_1, [\kappa_j^2]_1, [z_j^k]_1, y_{\psi_2^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right)$,
$k \in [0, l_D - l - 1]$, $i \in [0, s_{\max} - 2]$
16. **Compute9**$([\psi_2^{-1} z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_2^{-1} \kappa^2 z^k y^i]^{\mathbf{j}}, [\kappa^2 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^2]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_2^{-1}]^{\mathbf{j}}, [{\psi_2}_j^{-1}]_1, [\kappa_j^2]_1, [z_j^k]_1, y_{\psi_2^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right)$,
$k \in [0, l_D - l - 1]$, $i \in [s_{\max} - 1, s_{\max}]$
17. **Compute9**$([\psi_h z^k y^i]^{\mathbf{j-1}}, [z^k y^i]^{\mathbf{j-1}}, [y^i]^{\mathbf{j-1}}, [y]^{\mathbf{j-1}}, \mathrm{v}_{1,j-1}) \rightarrow$
$\left([\psi_h \mu^3 z^k y^i]^{\mathbf{j}}, [\mu^3 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\mu^3]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_h]^{\mathbf{j}}, [\psi_{hj}]_1, [\mu_j^3]_1, [z_j^k]_1, y_{\psi_h^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right)$,
$k \in [0,1]$, $i \in [0,1]$

Verifications:

The protocol verifier runs the following algorithms for each $j \in [N]$,

1. **Verify1** $\left([\alpha]^{\mathbf{j-1}}, [\alpha]^{\mathbf{j}}, [\alpha_{\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\alpha,\mathrm{j}}\right)$
2. **Verify1** $\left([\gamma]^{\mathbf{j-1}}, [\gamma]^{\mathbf{j}}, [\gamma_{\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\gamma,\mathrm{j}}\right)$
3. **Verify1** $\left([\beta]^{\mathbf{j-1}}, [\beta]^{\mathbf{j}}, [\beta_{\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\beta,\mathrm{j}}\right)$
4. **Verify1** $\left([\delta]^{\mathbf{j-1}}, [\delta]^{\mathbf{j}}, [\delta_{\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\delta,\mathrm{j}}\right)$
5. **Verify1** $\left([\eta_1]^{\mathbf{j-1}}, [\eta_1]^{\mathbf{j}}, [\eta_{1\mathrm{j}}]_1, \mathrm{v}_{1,\mathrm{j}-1}, \mathrm{y}_{\eta_1,\mathrm{j}}\right)$
6. **Verify2** $\left([\mu\eta_0]^{\mathbf{j-1}}, [\mu\eta_0]^{\mathbf{j}}, [\mu_j]_1, [\eta_{0j}]_1, [\mu_j \eta_{0j}]_2, y_{\mu j}, y_{\eta 0 j}\right)$
7. **Verify2** $\left([\mu^3 v]^{\mathbf{j-1}}, [\mu^3 v]^{\mathbf{j}}, [\mu_j^3]_1, [v_j]_1, [\mu_j^3 v_j]_2, y_{\mu^3 j}, y_{vj}\right)$
8. **Verify4** $\left([\mu^4 \kappa^h]^{\mathbf{j}}, [\mu^4]^{\mathbf{j}}, [\kappa]^{\mathbf{j}}, [\kappa^{h-1}]^{\mathbf{j}}, \mathrm{v}_{1,j-1}, y_{\mu^4 j}\right)$, $h \in [0...2]$
9. **Verify5** $\left([y^i x^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [x]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, \mathrm{v}_{1,j-1}, y_{ij}\right)$, $h \in [0...n-1], i \in [0...s_{max}-1]$
10. **Verify4** $\left([\psi_1^{-1} z^k]^{\mathbf{j}}, [\psi_1^{-1}]^{\mathbf{j}}, [z]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, \mathrm{v}_{1,j-1}, y_{\psi_1^{-1} j}\right)$, $k \in [0, 3(l_D-1)-4]$
11. **Verify7** $\left([\psi_3^{-1} \kappa^h z^i]^{\mathbf{j}}, [\kappa^h z^i]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [z^{i-1}]^{\mathbf{j}}, [z^i]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\psi_3^{-1}]^{\mathbf{j}}, y_{\psi_3^{-1} j}, y_{hj}, \mathrm{v}_{1,j-1}\right)$,
$h \in [1,2]$, $i \in [0, l_D - l - 2]$
12. **Verify9** $\left([\psi_0^{-1} \kappa^h z^k y^i]^{\mathbf{j}}, [\kappa^h z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [{\psi_0}_j^{-1}]_1, [\kappa_j^h]_1, [z_j^k]_1, y_{\psi_0^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right)$,
$h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$, $i \in [0, 2s_{max}-3]$
13. **Verify9** $\left([\psi_0^{-1} \kappa^h z^k y^i]^{\mathbf{j}}, [\kappa^h z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^h]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [{\psi_0}_j^{-1}]_1, [\kappa_j^h]_1, [z_j^k]_1, y_{\psi_0^{-1} j}, y_{kj}, \mathrm{v}_{1,j-1}\right)$,
$h \in [0,1]$, $k \in [0, 3(l_D-1)-1]$, $i \in [2s_{max}-2, 2s_{max}]$

14. **Verify9** $\Big([\psi_2^{-1}\kappa^2 z^k y^i]^{\mathbf{j}}, [\kappa^2 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^2]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [\psi_{2_j}^{-1}]_1, [\kappa_j^2]_1, [z_j^k]_1, y_{\psi_2^{-1}j}, y_{kj}, \mathrm{v}_{1,j-1}\Big)$, $k \in [0, l_D - l - 1]$, $i \in [0, s_{\max} - 2]$

15. **Verify9** $\Big([\psi_2^{-1}\kappa^2 z^k y^i]^{\mathbf{j}}, [\kappa^2 z^k y^i]^{\mathbf{j}}, [z^k y^i]^{\mathbf{j}}, [z^k]^{\mathbf{j}}, [\kappa^2]^{\mathbf{j}}, [y^i]^{\mathbf{j}}, [y]^{\mathbf{j}}, [\psi_0^{-1}]^{\mathbf{j}}, [\psi_{2_j}^{-1}]_1, [\kappa_j^2]_1, [z_j^k]_1, y_{\psi_2^{-1}j}, y_{kj}, \mathrm{v}_{1,j-1}\Big)$, $k \in [0, l_D - l - 1]$, $i \in [s_{\max} - 1, s_{\max}]$

7.13.2 Second Round

Initialization:

1. $[\delta^{-1}]^{\mathbf{0}} := G_1$, $[v^{-1}]^{\mathbf{0}} := G_1$, $[\mu^{-1}]^{\mathbf{0}} := G_1$
2. $[y^k x^h]^{\mathbf{0}} := G_1$, $h \in [0, 2n-2]$, $i \in [0, s_{\max} - 2]$
3. $[\delta^{-1} y^i x^h]^{\mathbf{0}} := [t_{\mathcal{X}}(x)]_1$, $h \in [0, n-2]$, $i \in [0, s_{\max} - 2]$ (Type-8)
4. $[\delta^{-1} y^i x^h]^{\mathbf{0}} := [t_{\mathcal{Y}}(y)]_1$, $h \in [0, 2n-2]$, $i \in [0, s_{\max} - 2]$ (Type-8)
5. $[v^{-1} y^i z^h]^{\mathbf{0}} := [t_{\mathcal{Y}}(y)]_1$, $h \in [0, 2l_D - 2l - 2]$, $i \in [0, s_{\max} - 2]$ (Type-8)
6. $[v^{-1} y^i z^h]^{\mathbf{0}} := [t_{\mathcal{Z}}(z)]_1$, $h \in [2(l_D - l) - 3, 0]$, $i \in [0, 2s_{\max} - 2]$ (Type-8)
7. $[v^{-1} y^i z^h]^{\mathbf{0}} := [t_{\mathcal{Z}}(z)]_1$, $h \in [0, 2l_D - 2l - 2]$, $i \in [s_{\max} - 1, s_{\max} + 1]$ (Type-8)
8. $[\mu^{-1} y^i z^h]^{\mathbf{0}} := [t_{\mathcal{Y}}(y)]_1$, $h = 0$, $i \in [0, l]$ (Type-8)
9. $[\gamma^{-1}]^{\mathbf{0}} = G_1$,
10. $[\eta_0^{-1}]^{\mathbf{0}} = G_1$, $[\eta_1^{-1}]^{\mathbf{0}} = G_1$
11. $[\mu^2]^{\mathbf{0}} = G_1$, $[\mu^{-1}]^{\mathbf{0}} = G_1$
12. $H_{1i} := \gamma^{-1} L_0(y) o_i(x)$
    $[H_{1i}]^{\mathbf{0}} := [L_0(y) o_i(x)]_1$, $i \in [0, l_{in} - 1]$ (Type-10)
13. $H_{2i} := \gamma^{-1} L_{-1}(y) o_i(x)$
    $[H_{2i}]^{\mathbf{0}} := [L_{-1}(y) o_i(x)]_1$, $i \in [l_{in}, l - 1]$ (Type-10)
14. $H_{3ik} := \eta_1^{-1} L_i(y) o_k(x)$
    $[H_{3ik}]^{\mathbf{0}} := [L_i(y) o_k(x)]_1$, $i \in [0, s_{\max} - 1]$, $k \in [l, l_D - 1]$ (Type-10)
15. $H_{4ik} := \delta^{-1} L_i(y) o_k(x)$
    $[H_{4ik}]^{\mathbf{0}} := [L_i(y) o_k(x)]_1$, $i \in [0, s_{\max} - 1]$, $k \in [l_D, m_D - 1]$ (Type-10)
16. $H_{5ik} := \eta_0^{-1} L_i(y) o_k(x) \left(K_{k-1}^2(z) - 1\right)$
    $[H_{5ik}]^{\mathbf{0}} := \left[L_i(y) o_k(x) \left(K_{k-1}^2(z) - 1\right)\right]_1$, $i \in [0, s_{\max} - 1]$, $k \in [l, l_D - 1]$ (Type-10)
17. $H_6 := \eta_0^{-1} t_{\mathcal{Y}}(y) \sum_{k=0}^{l_D-1} o_k(x) \left(K_{k-l}^2(z) - 1\right)$
    $[H_6]^{\mathbf{0}} := \left[t_{\mathcal{Y}}(y) \sum_{k=0}^{l_D-1} o_k(x) \left(K_{k-l}^2(z) - 1\right)\right]_1$, (Type-10)
18. $H_7 := \eta_1^{-1} t_{\mathcal{Y}}(y) \sum_{k=l}^{l_D-1} o_k(x)$
    $[H_7]^{\mathbf{0}} := [t_{\mathcal{Y}}(y) \sum_{k=l}^{l_D-1} o_k(x)]_1$, (Type-10)
19. $H_8 := \mu^2 \sum_{k=l}^{l_D-1} o_k(x) K_{k-l}(z)$
    $[H_8]^{\mathbf{0}} := \left[\sum_{k=l}^{l_D-1} o_k(x) K_{k-l}(z)\right]_2$, (Type-10)
20. $H_{9ik} := \mu^{-1} L_i(y) K_k(z)$
    $[H_{9ik}]^{\mathbf{0}} := [L_i(y) K_k(z)]_1$, $i \in [0, s_{\max} - 1]$, $k \in [0, l_D - l - 1]$ (Type-10)
21. $H_{10ik} := \eta_0^{-1} L_i(y) M_k(x, z) t_{\mathcal{Z}}(z)$
    $[H_{10ik}]^{\mathbf{0}} := [L_i(y) M_k(x, z) t_{\mathcal{Z}}(z)]_1$, $i \in [0, s_{\max} - 1]$, $k \in [l, l_D - 1]$ (Type-10)
22. $H_{11} := \eta_0^{-1} t_{\mathcal{Y}}(y) \sum_{k=1}^{l_D-1} M_k(x, z) t_{\mathcal{Z}}(z)$
    $[H_{11}]^{\mathbf{0}} := \left[t_{\mathcal{Y}}(y) \sum_{k=1}^{l_D-1} M_k(x, z) t_{\mathcal{Z}}(z)\right]_1$ (Type-10)

Computations:

For $j \in [N]$, P_j outputs:

1. **Compute8**$([\delta^{-1}y^i x^h]^{\mathbf{j-1}}, [y^i x^h]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([\delta^{-1}y^i x^h]^{\mathbf{j}}, [y^i x^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [x^{h-1}]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, y_{\delta^{-1}j}, y_{ij}\big)$, $h \in [0, n-2]$, $i \in [0, s_{\max}-2]$
2. **Compute8**$([\delta^{-1}y^i x^h]^{\mathbf{j-1}}, [y^i x^h]^{\mathbf{j-1}}, [x]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([\delta^{-1}y^i x^h]^{\mathbf{j}}, [y^i x^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [x^{h-1}]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, y_{\delta^{-1}j}, y_{ij}\big)$, $h \in [0, 2n-2]$, $i \in [0, s_{\max}-2]$
3. **Compute8**$([v^{-1}y^i z^h]^{\mathbf{j-1}}, [y^i z^h]^{\mathbf{j-1}}, [z]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}\big)$, $h \in [0, 2l_D - 2l - 2]$, $i \in [0, s_{\max}-2]$
4. **Compute8**$([v^{-1}y^i z^h]^{\mathbf{j-1}}, [y^i z^h]^{\mathbf{j-1}}, [z]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}\big)$, $h \in [2(l_D - l) - 3, 0]$, $i \in [0, 2s_{\max}-2]$
5. **Compute8**$([v^{-1}y^i z^h]^{\mathbf{j-1}}, [y^i z^h]^{\mathbf{j-1}}, [z]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}\big)$, $h \in [0, 2l_D - 2l - 2]$, $i \in [s_{\max}-1, s_{\max}+1]$
6. **Compute8**$([\mu^{-1}y^i z^h]^{\mathbf{j-1}}, [y^i z^h]^{\mathbf{j-1}}, [z]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow$
$\big([\mu^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\mu^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}\big)$, $h = 0$, $i \in [0, l]$
7. **Compute10**$([H_{1i}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{1i}]^{\mathbf{j}}, [\gamma^{-1}]^{\mathbf{j}}, [\gamma_j^{-1}]_1, y_{\gamma^{-1}})$, $i \in [0, l_{in}-1]$
8. **Compute10**$([H_{2i}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{2i}]^{\mathbf{j}}, [\gamma^{-1}]^{\mathbf{j}}, [\gamma_j^{-1}]_1, y_{\gamma^{-1}})$, $i \in [l_{in}, l-1]$
9. **Compute10**$([H_{3ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{3ik}]^{\mathbf{j}}, [\eta_1^{-1}]^{\mathbf{j}}, [\eta_{1j}^{-1}]_1, y_{\eta_1^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D - 1]$
10. **Compute10**$([H_{4ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{4ik}]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, [\delta_j^{-1}]_1, y_{\delta^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [l_D, m_D - 1]$
11. **Compute10**$([H_{5ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{5ik}]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, y_{\eta_0^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D - 1]$
12. **Compute10**$([H_6]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_6]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, y_{\eta_0^{-1}})$,
13. **Compute10**$([H_7]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_7]^{\mathbf{j}}, [\eta_1^{-1}]^{\mathbf{j}}, [\eta_{1j}^{-1}]_1, y_{\eta_1^{-1}})$,
14. **Compute10**$([H_8]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_8]^{\mathbf{j}}, [\mu^2]^{\mathbf{j}}, [\mu_j^2]_1, y_{\mu^2})$,
15. **Compute10**$([H_{9ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{9ik}]^{\mathbf{j}}, [\mu^{-1}]^{\mathbf{j}}, [\mu_j^{-1}]_1, y_{\mu^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [0, l_D - l - 1]$
16. **Compute11**$([H_{10ik}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{10ik}]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, y_{\eta_0^{-1}})$, $i \in [0, s_{\max}-1]$, $k \in [l, l_D - 1]$
17. **Compute11**$([H_{11}]^{\mathbf{j-1}}, \mathrm{v}_{2,j-1}) \rightarrow ([H_{11}]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, y_{\eta_0^{-1}})$,

Verifications:

The protocol verifier runs the following algorithms for each $j \in [N]$,

1. **Verify8** $\big([\delta^{-1}y^i x^h]^{\mathbf{j}}, [y^i x^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [x^{h-1}]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, y_{\delta^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\big)$,
$h \in [0, n-2]$, $i \in [0, s_{\max}-2]$
2. **Verify8** $\big([\delta^{-1}y^i x^h]^{\mathbf{j}}, [y^i x^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [x^{h-1}]^{\mathbf{j}}, [x^h]^{\mathbf{j}}, [x]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, y_{\delta^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\big)$,
$h \in [0, 2n-2]$, $i \in [0, s_{\max}-2]$
3. **Verify8** $\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\big)$,
$h \in [0, 2l_D - 2l - 2]$, $i \in [0, s_{\max}-2]$
4. **Verify8** $\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\big)$,
$h \in [2(l_D - l) - 3, 0]$, $i \in [0, 2s_{\max}-2]$
5. **Verify8** $\big([v^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [v^{-1}]^{\mathbf{j}}, y_{v^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\big)$,
$h \in [0, 2l_D - 2l - 2]$, $i \in [s_{\max}-1, s_{\max}+1]$
6. **Verify8** $\big([\mu^{-1}y^i z^h]^{\mathbf{j}}, [y^i z^h]^{\mathbf{j}}[y^i]^{\mathbf{j}}, [z^{h-1}]^{\mathbf{j}}, [z^h]^{\mathbf{j}}, [z]^{\mathbf{j}}, [\mu^{-1}]^{\mathbf{j}}, y_{\mu^{-1}j}, y_{ij}, \mathrm{v}_{2,j-1}\big)$, $h = 0$, $i \in [0, l]$

7. **Verify10** $\left([H_{1i}]^{\mathbf{j-1}}, [H_{1i})]^{\mathbf{j}}, [\gamma^{-1}]^{\mathbf{j}}, [\gamma_j^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\gamma^{-1}}\right)$, $i \in [0, l_{in} - 1]$
8. **Verify10** $\left([H_{2i}]^{\mathbf{j-1}}, [H_{2i})]^{\mathbf{j}}, [\gamma^{-1}]^{\mathbf{j}}, [\gamma_j^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\gamma^{-1}}\right)$, $i \in [l_{in}, l - 1]$
9. **Verify10** $\left([H_{3ik}]^{\mathbf{j-1}}, [H_{3ik})]^{\mathbf{j}}, [\eta_1^{-1}]^{\mathbf{j}}, [\eta_{1j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_1^{-1}}\right)$, $i \in [0, s_{\max} - 1]$, $k \in [l, l_D - 1]$
10. **Verify10** $\left([H_{4ik}]^{\mathbf{j-1}}, [H_{4ik})]^{\mathbf{j}}, [\delta^{-1}]^{\mathbf{j}}, [\delta_j^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\delta^{-1}}\right)$, $i \in [0, s_{\max} - 1]$, $k \in [l_D, m_D - 1]$
11. **Verify10** $\left([H_{5ik}]^{\mathbf{j-1}}, [H_{5ik})]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_0^{-1}}\right)$, $i \in [0, s_{\max} - 1]$, $k \in [l, l_D - 1]$
12. **Verify10** $\left([H_6]^{\mathbf{j-1}}, [H_6)]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_0^{-1}}\right)$
13. **Verify10** $\left([H_7]^{\mathbf{j-1}}, [H_7)]^{\mathbf{j}}, [\eta_1^{-1}]^{\mathbf{j}}, [\eta_{1j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_1^{-1}}\right)$
14. **Verify10** $\left([H_8]^{\mathbf{j-1}}, [H_8)]^{\mathbf{j}}, [\mu^2]^{\mathbf{j}}, [\mu_j^2]_1, \mathrm{v}_{2,j-1}, y_{\mu^2}\right)$
15. **Verify10** $\left([H_{9ik}]^{\mathbf{j-1}}, [H_{9ik})]^{\mathbf{j}}, [\mu^{-1}]^{\mathbf{j}}, [\mu_j^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\mu^{-1}}\right)$, $i \in [0, s_{\max} - 1]$, $k \in [0, l_D - l - 1]$
16. **Verify11** $\left([H_{10ik}]^{\mathbf{j-1}}, [H_{10ik})]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_0^{-1}}\right)$, $i \in [0, s_{\max} - 1]$, $k \in [l, l_D - 1]$
17. **Verify11** $\left([H_{11}]^{\mathbf{j-1}}, [H_{11})]^{\mathbf{j}}, [\eta_0^{-1}]^{\mathbf{j}}, [\eta_{0j}^{-1}]_1, \mathrm{v}_{2,j-1}, y_{\eta_0^{-1}}\right)$

8 Security Analysis of the MPC setup for Tokamak scheme

In this section, we present a sketch security analysis of the MPC setup ceremony for the Tokamak zk-SNARK scheme. A key aspect of many SNARK protocols is the setup phase, which generates the SRS necessary for proving statements succinctly and efficiently. The security of this setup phase is important, as it underpins the reliability of the entire cryptographic system. A trusted setup for generating CRS or SRS is avoided, as multiparty computation can substitute for the need for trust. The SRS generation typically involves MPC protocols, which allow multiple participants to collaboratively create the SRS in a manner that ensures security even if some participants are dishonest. The setup phase's security is primarily concerned with preventing adversaries from tampering with the SRS, which could lead to catastrophic failures such as forging proofs or bypassing verification checks. Multi-party computation protocols for SRS generation address these concerns by distributing trust among multiple parties, ensuring that as long as at least one party remains honest, the setup remains secure.

In conducting the security analysis of the Tokamak SNARK setup ceremony protocol, we based our work on the [2, 3] mentioned above. Briefly speaking, BGM17 [2] is currently commonly used in practice, present a Groth16 setup ceremony scheme by proposing random beacon paradigm. Kohlweiss et al. [3] prove the security of the Groth16 SNARK with a setup ceremony of BGM17 that they do not require the use of a random beacon.

In this security analysis, we analyze the Tokamak SNARK setup ceremony with respect to algebraic adversaries [5] in the random oracle model. Already the original AGM paper [5] proved knowledge soundness of the Groth16 SNARK in the AGM model (assuming trusted SRS). The algebraic group model (AGM) is a computational model in which all adversaries are modeled as algebraic.

We will employ the AGM to demonstrate the security of Tokamak SNARK setup ceremony protocol. In this model, we focus exclusively on algebraic algorithms that can provide a linear representation for each group element they generate. Specifically, if an algorithm A_{alg} has already received group elements $G_1, \ldots, G_n \in \mathbb{G}$ and outputs a new group element $G_{n+1} \in \mathbb{G}$, it must also provide a vector of integer coefficients $\vec{C} = (c_1, \ldots, c_n)$ such that $G_{n+1} = \prod_{i=1}^{n} G_i^{c_i}$. We employ AGM in a pairing-based context where we distinguish between group elements from $\mathbb{G}1$ and $\mathbb{G}2$. Formally, the coefficients $\vec{C}$ are derived by using the algebraic extractor $\vec{C} \leftarrow \mathcal{E}_{\mathcal{A}}^{\mathrm{agm}}(\mathrm{view}\mathcal{A})$, which is guaranteed to exist for any algebraic adversary $\mathcal{A}$. This extractor is white-box, meaning it operates with full access to $\mathcal{A}$'s internal view.

$\mathrm{RO}_t(\phi)$ // Initially $Q_{\mathrm{RO}} = \emptyset$
if $Q_{\mathrm{RO}}[\phi] \neq \perp$ then $r \leftarrow Q_{\mathrm{RO}}[\phi]$;
else $r \leftarrow \$\mathbb{Z}_p$; $Q_{\mathrm{RO}}[\phi] \leftarrow r$;
if $t = 1$ then return r else return G^r

Figure 2: The transparent random oracle $\mathrm{RO}_0(\cdot) : \{0,1\}^* \to \mathbb{G}_1$, $\mathrm{RO}_1(\cdot) : \{0,1\}^* \to \mathbb{Z}_p$.

Fuchsbauer et al. [5] also describe how to integrate AGM with the random oracle (RO) model. Our interest lies in random oracles that output group elements. Group elements produced by $\mathrm{RO}(\phi)$ are incorporated into the set of received group elements. To simulate update proofs, we use a modified version of the programmable RO model, which we call a transparent RO, as shown in Figure 1. For simplicity, we define $\mathrm{RO}(\cdot) := \mathrm{RO}_0(\cdot)$. The simulator has access to $\mathrm{RO}_1(\cdot)$ and can determine the discrete logarithm r by querying $\mathrm{RO}_1(x)$. It may query $\mathrm{RO}_0(x)$ for G^r, but it can also calculate this value independently. In all security definitions, both the constructions and the adversary $\mathcal{A}$ only have access to the restricted oracle $\mathrm{RO}_0(\cdot)$.

We revisit the (q_1, q_2)-discrete logarithm assumption as described in [3]

Definition 8.1 ($(q_1, q_2) - dlog$)**.** *The (q_1, q_2)-discrete logarithm assumption holds for BGen if for any PPT $\mathcal{A}$, the following probability is negligible in λ,*

$$\Pr\left[\ bp \leftarrow BGen(1^\lambda); z \leftarrow \$\mathbb{Z}_p; z' \leftarrow \mathcal{A}(bp, \{G^{z^i}\}_{i=1}^{q_1}, \{H^{z^i}\}_{i=1}^{q_2}) : z = z'\ \right].$$

Definition 8.2 ($(q_1, q_2) - dlog$)**.** *The (q_1, q_2)-extended discrete logarithm assumption holds for BGen if for any PPT $\mathcal{A}$, the following probability is negligible in λ,*

$$\Pr\left[\begin{array}{c} bp \leftarrow BGen(1^\lambda); z, r, s \leftarrow \$\mathbb{Z}_p \ s.t.\ rz + s \neq 0; \\ z' \leftarrow \mathcal{A}\left(bp, \{G^{z^i}\}_{i=1}^{q_1}, \{H^{z^i}\}_{i=1}^{q_2}, r, s, G^{\frac{1}{rz+s}}, H^{\frac{1}{rz+s}}\right) : z = z' \end{array}\right].$$

Theorem 8.1. *If $(q_1 + 1, q_2 + 1)$-dlog assumption holds, then (q_1, q_2)-edlog assumption holds.*

We also present two lemmas that are frequently useful when working with AGM proofs.

Lemma 8.1 ([6])**.** *Let Q be a non-zero polynomial in $\mathbb{Z}_p[X_1, \ldots, X_n]$ of total degree d. Define $Q'(Z) := Q(R_1Z+S_1, \ldots, R_nZ+S_n)$ in the ring $(\mathbb{Z}_p[R_1, \ldots, R_n, S_1, \ldots, S_n])[Z]$. Then the coefficient of the highest degree monomial in $Q'(Z)$ is a degree d polynomial in $\mathbb{Z}_p[R_1, \ldots, R_n]$.*

Lemma 8.2 (Schwartz-Zippel)**.** *Let P be a non-zero polynomial in $\mathbb{Z}_p[X_1, \ldots, X_n]$ of total degree d. Then,*

$$\Pr[x_1, \ldots, x_n \leftarrow \$\mathbb{Z}_p : P(x_1, \ldots, x_n) = 0] \leq \frac{d}{p}.$$

The update proof of knowledge that ensures validity of each sequential SRS update is the central ingredient of the ceremony protocol.

$\mathcal{O}_{\mathrm{se}}(\phi)$	$\mathcal{O}_{\mathrm{poly}}^{\mathbb{G}_1}(f(Z_1, \ldots, Z_{d(\lambda)}))$	$\mathcal{O}_{\mathrm{poly}}^{\mathbb{G}_2}(g(Z_1, \ldots, Z_{d(\lambda)}))$
// Initially $Q = \emptyset$	if $\deg(f) > d(\lambda)$	if $\deg(g) > d(\lambda)$
$\pi \leftarrow \mathrm{Sim}_{\mathrm{RO}_1(\cdot)}(\phi)$	return $\perp$	return $\perp$
$Q \leftarrow Q \cup \{(\phi, \pi)\}$	else return $G^{f(z_1, \ldots, z_{d(\lambda)})}$	else return $H^{g(z_1, \ldots, z_{d(\lambda)})}$
return π		

Figure 3: Simulation-extraction oracle and two d-Poly oracles — for $\mathbb{G}_1$ and $\mathbb{G}_2$.

The proof of knowledge (PoK) protocol does not depend on a reference string; instead, it uses a random oracle as its setup. Therefore, we will enhance the standard NIZK definitions by incorporating $\mathrm{RO}_t(\cdot)$, as defined in Figure 2.

Even with enhanced capabilities, this adversary should not be able to generate a proof of knowledge unless it possesses the corresponding witness. Note that $\mathcal{O}_{\mathrm{poly}}$ is separate from the random oracle RO_t and does not provide the adversary with any information regarding the random oracle's outputs. Overall, $\mathcal{O}_{\mathrm{poly}}$ strictly enhances the adversary's capabilities.

Definition 8.3. *An argument Ψ for $\mathcal{R}$ is straight-line simulation extractable in the $(RO, d\text{-}Poly)$-model, if for all PPT $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that $\Pr\left[Game_{SE}^{\mathcal{A}}(1^\lambda) = 1\right] = negl(\lambda)$, where $Game_{SE}^{\mathcal{A}}(1^\lambda) =$*

$$\left[\begin{array}{l} Q \leftarrow \emptyset; z_1, \ldots, z_{d(\lambda)} \leftarrow \mathbb{Z}_p; \\ (\phi, \pi) \leftarrow \mathcal{A}^{\mathcal{O}_{se}, RO, \mathcal{O}_{poly}^{\mathbb{G}_1}, \mathcal{O}_{poly}^{\mathbb{G}_2}}(1^\lambda); \\ w \leftarrow \mathcal{E}_{\mathcal{A}}(view_{\mathcal{A}}); \\ Verify^{RO(\cdot)}(\phi, \pi) = 1 \wedge (\phi, w) \notin \mathcal{R} \wedge (\phi, \pi) \notin Q \end{array}\right]$$

The oracles $\mathcal{O}_{se}, \mathcal{O}_{poly}^{\mathbb{G}_1}, \mathcal{O}_{poly}^{\mathbb{G}_2}$ are defined on Figure 3

Essentially, the adversary wins if it can provide a valid statement and proof without knowing a corresponding witness, ensuring that the proof hasn't been obtained from a simulation oracle. Additionally, there can be up to $d(\lambda)$ random variables chosen initially, which allow the adversary to query an oracle for arbitrary polynomial evaluations up to a degree of $d(\lambda)$ in the group.

In terms of comparing this definition to more conventional ones, two points are notable. First, our definition operates as a white-box model (since the extractor $\mathcal{E}_{\mathcal{A}}$ uses the adversary's view, $\text{view}_{\mathcal{A}}$) and is strong (meaning proofs cannot be randomized). Second, our concept inherently includes strong simulation extractability (strong-SE) in the context of a random oracle (RO), corresponding to the case of Game_{sSE} without $\mathcal{O}\text{poly}$, thus closely resembling the standard strong-SE variant without an RO.

We permit the final SRS to deviate from a uniform random distribution as long as the adversary does not gain any significant advantage in compromising the soundness of the SNARK. Essentially, this extends the updatability definitions from [7] to ceremonies that involve multiple rounds. .

We examine NP-languages $\mathcal{L}$ and their associated relations $\mathcal{R} = (\phi, w)$, where w serves as an NP-witness for the statement $\phi \in \mathcal{L}$.

- The probabilistic polynomial-time (PPT) parameter generator algorithm $Setup(pp_\lambda, \mathcal{L})$ takes as input the bilinear pairing group $pp_\lambda = (\mathbb{H}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ and the subcircuit library $\mathcal{L} = \{u_j(X), v_j(X), w_j(X)\}_{j=0}^{m_D-1}$, picks uniform random parameters

$$\boldsymbol{\tau} := (x, y, z, \alpha, \beta, \gamma, \delta, \eta_0, \eta_1, \mu, \nu, \psi_0, \psi_1, \psi_2, \psi_3, \kappa) \xrightarrow{\S} (\mathbb{F}^*)^{16},$$

  and returns $\boldsymbol{\sigma} = ([\boldsymbol{\sigma}_{A,I}]_1, [\boldsymbol{\sigma}_C]_1, [\boldsymbol{\sigma}_{zk}]_1, [\boldsymbol{\sigma}_\nu]_2)$, where the security parameter 1^λ.

- A PPT SRS update algorithm Update that takes as input a phase number $\varphi \in 1, 2$, the current SRS srs, and the proofs of prior updates $\{\rho_i\}_i$. It outputs a new SRS srs$'$ along with an update proof ρ'. It is assumed that Update enforces a specific order of phases, such as a sequential order.

- A deterministic polynomial-time (DPT) SRS verification algorithm VerifySRS that accepts an SRS srs and update proofs $\{\rho_i\}_i$, and returns either 0 or 1.

- A PPT prover algorithm Prove that accepts an SRS srs, a statement ϕ, and a witness w, and generates a proof π.

- A DPT verification algorithm Verify that takes as input an SRS srs, a statement ϕ, and a proof π, and produces either 0 or 1.

The specification of a setup protocol includes a default setting $\text{srs}^{\text{d}} = (\text{srs}^{\text{d}}\varphi_1, \text{srs}^{\text{d}}\varphi_2)$. We require that a secure scheme upholds the properties of completeness, zero-knowledge, and knowledge soundness. Our definitions operate within the random oracle model, as the final SRS update protocol relies on *RO*-dependent proof of knowledge. Consequently, all algorithms in this section can access the random oracle, if needed by certain sub-components of a scheme.

We require that a secure scheme meets the following criteria for completeness, zero-knowledge, and knowledge soundness. We will prove that Tokamak ceremony scheme provides the following properties: completeness, zero-knowledge, and knowledge soundness. The Tokamak ceremony protocol for a relation $\mathcal{R}$ consists of the following algorithms.

Theorem 8.2. *Tokamak ceremony scheme is perfectly complete.*

Proof. The Tokamak scheme provides both Update completeness and Prover completeness. Completeness of Tokamak ceremony scheme requires that Update and Prove always satisfy verification. Let's begin with a general observation: if a certain bitstring $s = (\text{srs}, \{\rho_i\}_i)$ fulfills $\text{VerifySRS}(s) = 1$, then there exists a unique $\boldsymbol{\sigma}_{A,I}, \boldsymbol{\sigma}_C, \boldsymbol{\sigma}_{zk}$ and $\boldsymbol{\sigma}_V \in \mathbb{Z}_p^*$ that define a well-formed srs. Let A be an adversary that outputs $s = (\text{srs}, \{\rho_i\}_i)$ such that $\text{VerifySRS}(s) = 1$ and the probability of A is

$$\Pr\left[\begin{array}{c}(\varphi, \text{srs}, \{\rho_i\}_i) \leftarrow A(\mathbf{1}^\lambda), (\text{srs}', \rho') \leftarrow \text{Update}(\varphi, \text{srs}, \{\rho_i\}_i) : \\ \text{VerifySRS}(\text{srs}, \{\rho_i\}_i) = 1 \wedge \text{VerifySRS}(\text{srs}', \{\rho_i\}_i \cup \{\rho'\}) = 0\end{array}\right] = 0.$$

Again, we are analysing *Update* together with VerifySRS and Tokamak scheme satisfies Prover completeness because the probability of A is

$$\Pr\left[\begin{array}{c}(\text{srs}, \{\rho_i\}_i, \phi, w) \leftarrow A(\mathbf{1}^\lambda), (\pi) \leftarrow \text{Prove}(srs, \phi, w) : \\ \text{VerifySRS}(\text{srs}, \{\rho_i\}_i) = 1 \wedge (\phi, w) \in R \wedge \text{Verify}(\text{srs}, \phi, \pi) \neq 1\end{array}\right] = 0.$$

□

Our definition of subversion zero knowledge (Sub-ZK) is based on [8]. Intuitively, it asserts that an adversary who produces a well-formed SRS must know the simulation trapdoor τ, enabling them to simulate a proof independently, without needing the witness. Consequently, the proofs do not disclose any additional information. From a technical perspective, we split the adversary into two parts: an efficient SRS subverter $\mathcal{Z}$, which generates the SRS (since knowing τ is only meaningful for an efficient adversary), and an unbounded distinguisher $\mathcal{A}$. We allow $\mathcal{Z}$ to send st to communicate with $\mathcal{A}$.

$\mathbf{Prove}_{\mathbf{dl}}^{\mathbf{RO}(\cdot)}(\phi, w)$	$\mathbf{Verify}_{\mathbf{dl}}^{\mathbf{RO}(\cdot)}(\phi = (\cdot, G^{y_1}, H^{y_2}), \pi)$	$\mathbf{Sim}_{\mathbf{dl}}^{\mathbf{RO}_1(\cdot)}(\phi = (\cdot, G^{y_1}, H^{y_2}))$
$G^r \leftarrow \mathrm{RO}(\phi)$; return G^{rw};	$G^r \leftarrow \mathrm{RO}(\phi)$; Verify that $\hat{e}(G^{y_1}, H) = (G, H^{y_2}) \wedge$ $\hat{e}(\pi, H) = \hat{e}(G^r, H^{y_2})$;	Assert $\hat{e}(G^{y_1}, H) = (G, H^{y_2})$; $r_\phi \leftarrow \mathrm{RO}_1(\phi)$; return $\pi \leftarrow (G^{y_1})^{r_\phi}$;

Figure 4: A discrete logarithm proof of knowledge Π_{dl}

Theorem 8.3. *Tokamak ceremony scheme provides subversion zero knowledge (Sub-ZK) property.*

Proof. The Tokamak scheme is subversion zero knowledge since for all PPT subverters $\mathcal{Z}$, there exists a PPT extractor $\mathcal{E}_{\mathcal{Z}}$ such that for any (unbounded) adversary $\mathcal{A}$, the difference $|\epsilon_0 - \epsilon_1|$ is negligible in λ, where

$$\epsilon_b := \Pr\left[(\mathrm{srs}, \{\rho_i\}_i, st) \leftarrow \mathcal{Z}(1^\lambda), \tau \leftarrow \mathcal{E}_{\mathcal{Z}}(\mathrm{view}_{\mathcal{Z}}) : \mathrm{VerifySRS}(\mathrm{srs}, \{\rho_i\}_i) = 1 \wedge \mathcal{A}^{\mathcal{O}_b(\mathrm{srs}, \tau, \cdot)}(st) = 1\right].$$

Here, $\mathcal{O}_b$ is a proof oracle that accepts as input $(\mathrm{srs}, \tau, (\phi, w))$ and only proceeds if $(\phi, w) \in \mathcal{R}$. If $b = 0$, the oracle $\mathcal{O}_b$ returns an honest proof $\mathrm{Prove}(\mathrm{srs}, \phi, w)$, and if $b = 1$, it provides a simulated proof $\mathrm{Sim}(\mathrm{srs}, \tau, \phi)$. It is important to note that Sim is permitted access to the discrete logarithms of the random oracle (RO). Moreover, it is evident that Π_{dl} achieves perfect zero-knowledge relative to the simulator described in Figure 4. When the simulator receives an input $\phi = (m, G^w, H^w)$ (where $\phi \in \mathcal{L}$ by definition, meaning the exponent w is the same for both G^w and H^w), it retrieves r such that $G^r = \mathrm{RO}(\phi)$ using RO_1, and outputs G^{wr}. Because the simulated and honest proofs are identical, an adversary cannot differentiate between them.

We analyze security against algebraic adversaries $\mathcal{A}$. The statement elements ϕ (G^y, H^y) and the proof $\pi \in \mathbb{G}_1$ produced by $\mathcal{A}$ are expected to lie within the span of elements that $\mathcal{A}$ has queried from oracles. The coefficients of these spans are accessible within $\mathcal{A}$'s view $\mathrm{view}_{\mathcal{A}}$ because $\mathcal{A}$ is algebraic. We define an extractor $\mathcal{E}_{\mathcal{A}}$ that takes $\mathrm{view}_{\mathcal{A}}$ as input and extracts the coefficient k corresponding to the element $\mathrm{RO}(\phi) = G^r$. The remainder of the proof centers on demonstrating that k serves as the witness y. Broadly speaking, the approach involves designing a discrete logarithm adversary $\mathcal{C}$ that incorporates a (randomized) discrete logarithm challenge G^c into each of the random oracle queries made by $\mathcal{A}$. We show that that unless $k = y, C$ is able to compute the discrete logarithm c from viewA with an overwhelming probablity. □

Bellare et al. [9] demonstrated that it is feasible to achieve both soundness and subversion zero-knowledge simultaneously. However, they also proved that subversion soundness is incompatible with (even non-subversion) zero-knowledge. The concept of updatable knowledge soundness from [7] can be viewed as a relaxation of subversion soundness, designed to circumvent this impossibility result.

Theorem 8.4. *Tokamak ceremony scheme provides update knowledge soundness.*

Proof. The Tokamak scheme is update knowledge sound since for all PPT adversaries $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}$ such that $\Pr\left[\mathrm{Game}_{\mathrm{uks}}^{\mathcal{A}, \mathcal{E}_{\mathcal{A}}}(1^\lambda) = 1\right]$ is negligible in λ, where

$$\mathrm{Game}_{\mathrm{uks}}^{\mathcal{A}, \mathcal{E}_{\mathcal{A}}}(1^\lambda) := \left[\begin{array}{l} (\phi, \pi) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{srs}}(\cdot)}(1^\lambda);\ \text{get } (\mathrm{srs}, \varphi) \text{ from } \mathcal{O}_{\mathrm{srs}};\ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathrm{view}_{\mathcal{A}}); \\ \textbf{return } \mathrm{Verify}(\mathrm{srs}, \phi, \pi) = 1 \wedge (\phi, w) \notin \mathcal{R} \wedge \varphi > \varphi_{\max} \end{array}\right].$$

The SRS update oracle $\mathcal{O}_{\mathrm{srs}}$ is described in Algorithm 1. □

We extend the concept of update knowledge soundness to multiple phases of SRS generation. The SRS begins as empty (or can be initialized to a default value $\mathrm{srs}^{\mathrm{d}}$). In each phase φ, the adversary is responsible for setting a portion of the SRS, denoted by srs_φ, ultimately constructing the final SRS. The adversary can request honest updates for its proposed srs_φ^*, but it must pass the verification by VerifySRS. The adversary may query honest updates via the UPDATE query through a special oracle

$\mathcal{O}_{\text{srs}}$, as described in Algorithm 1. Eventually, the adversary can propose a candidate srs^*_φ with update proofs $\mathcal{Q}^*$ to be finalized using a FINALIZE query. The oracle finalizes srs^*_φ if $\mathcal{Q}^*$ includes at least one honest update proof from the current phase. Once this condition is met, srs_φ cannot be altered further, and phase $\varphi + 1$ commences. When the complete SRS is established, $\mathcal{A}$ outputs a statement ϕ and a proof π. The adversary succeeds if (srs, ϕ, π) passes verification, but no PPT extractor $\mathcal{E}_\mathcal{A}$ can extract a witness, even when given the view of $\mathcal{A}$.

References

[1] Jehyuk Jang and Jamie Judd. An efficient SNARK for field-programmable and RAM circuits. Cryptology ePrint Archive, Paper 2024/507, 2024. `https://eprint.iacr.org/2024/507`.

[2] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Paper 2017/1050, 2017. `https://eprint.iacr.org/2017/1050`.

[3] Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky ceremonies. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 98–127, Cham, 2021. Springer International Publishing.

[4] Jens Groth. On the size of pairing-based non-interactive arguments. In *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*, page 305–326, Berlin, Heidelberg, 2016. Springer-Verlag.

[5] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 33–62, Cham, 2018. Springer International Publishing.

[6] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 121–151, Cham, 2020. Springer International Publishing.

[7] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-snarks. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 698–728, Cham, 2018. Springer International Publishing.

[8] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. Cryptology ePrint Archive, Paper 2017/599, 2017.

[9] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. Nizks with an untrusted crs: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 777–804, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.