UC-Secure CRS Generation for SNARKs

Behzad Abdolmaleki[1], Karim Baghery[1], Helger Lipmaa[1], Janno Siim[1], and Michał Zając[2]

[1] University of Tartu, Estonia
[2] Clearmatics, UK

Abstract. Zero-knowledge SNARKs (zk-SNARKs) have recently found various applications in verifiable computation and blockchain applications (Zerocash), but unfortunately they rely on a common reference string (CRS) that has to be generated by a trusted party. A standard suggestion, pursued by Ben Sasson et al. [IEEE S&P, 2015], is to generate CRS via a multi-party protocol. We enhance their CRS-generation protocol to achieve UC-security. This allows to safely compose the CRS-generation protocol with the zk-SNARK in a black-box manner with the insurance that the security of the zk-SNARK is not influenced. Differently from the previous work, the new CRS-generation protocol also avoids the random oracle model which is typically not required by zk-SNARKs themselves. As a case study, we apply the protocol to the state-of-the-art zk-SNARK by Groth [EUROCRYPT, 2016].

Keywords: CRS model, SNARK, subversion-security, UC security

1 Introduction

A zero-knowledge argument is a cryptographic protocol between a prover and a verifier where the objective is to prove the validity of some statement while not leaking any other information. In particular, such an argument should be *sound* (it should be impossible to prove false statements) and *zero-knowledge* (the only leaked information should be the validity of the statement). Practical applications often require a non-interactive zero-knowledge (NIZK) argument where the prover outputs a single message which can be checked by many different verifiers.

Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) are particularly efficient instantiations of NIZK, and have thus found numerous application ranging from verifiable computation [29] to privacy-preserving cryptocurrencies [5] and privacy-preserving smart contracts [25]. In most of such zk-SNARKs (see, e.g., [19,26,15,29,13,20]), the verifier's computation is dominated by a small number of exponentiations and pairings in a bilinear group, while the argument consists of a small number of group elements. Importantly, a zk-SNARK exists for any NP-language.

One drawback in the mentioned pairing-based zk-SNARKs is their reliance on the strong common reference string (CRS) model. It assumes that in the setup phase of the protocol a trusted party publishes a CRS, sampled from some specialized distribution, while not leaking any side information. Subverting the setup phase can make it easy to break the security, e.g., leaking a CRS trapdoor makes it trivial to prove false statements. This raises the obvious question of how to apply zk-SNARKs in practice without completely relying on a single trusted party. The issue is further amplified since in all of the mentioned zk-SNARKs, one has to generate a new CRS each time the relation changes.

Reducing trust on CRS generation is indeed a long-standing open question. Several different approaches for this are known, but each one has its own problems. Some recent papers [6,8,9] have proposed efficient CRS-generation multiparty computation protocols, where only 1 out of N_p parties has to be honest, for a large class of known zk-SNARKs (in fact, most of the efficient pairing-based zk-SNARKs belong to this class, possibly after the inclusion of a small number of new elements to their CRSs) for which the CRS can be computed by a fixed well-defined class $\mathcal{C}^{\mathsf{S}}$ of circuits. Following [6], we will call this class of zk-SNARKs $\mathcal{C}^{\mathsf{S}}$*-SNARKs*. However, the CRS-generation protocols of [6,8,9] have the following two weaknesses:

1. They are not secure in the universal composability (UC) setting [10]. Hence, they might not be secure while running in parallel with other protocols, as is often the case in real life scenarios. Moreover, some systems require a UC-secure NIZK [25,22], but up to now their CRS is still be generated in a standalone setting. We note that [6,9] do prove some form of simulatability but not for full UC-security. Protocol of [8] is for one specific zk-SNARK.
2. All use the random oracle model and [8,9] additionally use knowledge assumption. Non-falsifiable assumptions [28] (e.g., knowledge assumptions) and the random oracle model are controversial (in particular, the random oracle model is uninstantiable [12,17] and thus can only be thought of as a heuristic), and it is desirable to avoid them in situations where they are not known to circumvent impossibility results. Importantly, construction of zk-SNARKs under falsifiable assumptions is impossible [16] and hence they do rely on non-falsifiable assumptions but usually not on the random oracle model. Relying on the random oracle model in the setup phase means that the complete composed system (CRS-generation protocol + zk-SNARK) relies on both random oracle model and non-falsifiable assumptions. Hence, we end up depending on two undesirable assumptions rather than one.

Updatable CRS [21] is another recent solution to the problem. Essentially, this can be viewed as a single round MPC protocol where each party needs to participate just once in the CRS computation. Current zk-SNARKs in updatable CRS model [21,27] are still less efficient, than the state-of-the-art non-updatable counterparts like the zk-SNARK by Groth [20].

As a different approach, in order to minimize the trust of NIZKs in the setup phase, Bellare *et al.* [4] defined the notion of subversion-resistance, which guarantees that a security property (like soundness) holds even if the CRS generators are all malicious. As proven in [4], achieving subversion-soundness and (even non-subversion) zero knowledge at the same time is impossible for NIZK arguments. On the other hand, one can construct subversion-zero knowledge

(Sub-ZK) and sound NIZK arguments. Abdolmaleki *et al.* [2] showed how to design efficient Sub-ZK SNARKs: essentially, a zk-SNARK can be made Sub-ZK by constructing an efficient public CRS-verification algorithm CV that guarantees the well-formedness of its CRS. In particular, [2] did this for the most efficient known zk-SNARK by Groth [20] after inserting a small number of new elements to its CRS. Fuchsbauer [14] proved that Groth's zk-SNARK (with a slightly different simulation) is Sub-ZK even without changing its CRS.

Our Contributions. We propose a new UC-secure multi-party CRS-generation protocol for $\mathcal{C}^{\mathsf{S}}$-SNARKs that crucially relies only on *falsifiable assumptions* and *does not require a random oracle*. Conceptually, the new protocol follows similar ideas as the protocol of [6], but it does not use any proofs of knowledge. Instead, we use a discrete logarithm extractable (DL-extractable) UC commitment functionality $\mathcal{F}_{\mathsf{dlmcom}}$ that was recently defined by Abdolmaleki *et al.* [1]. A DL-extractable commitment scheme allows to commit to a field element x and open to the group element g^x. Since $\mathcal{F}_{\mathsf{dlmcom}}$ takes x as an input, the committer must know x and thus x can be extracted by the UC-simulator. As we will show, this is sufficient to prove UC-security of the new CRS-generation protocol.

In addition, we show that the Sub-ZK SNARK of [2] is a Sub-ZK $\mathcal{C}^{\mathsf{S}}$-SNARK after just adding some more elements to its CRS. We also improve the efficiency of the rest of the CRS-generation protocol by allowing different circuits for each group, considering special multiplication-division gates, and removing a number of NIZK proofs that are used in [6]. Like in the previous CRS-generation protocols [6,8,9], soundness and zero-knowledge will be guaranteed as long as 1 out of N_p parties participating in the CRS generation is honest. If SNARK is also Sub-ZK [2,14], then zero-knowledge is guaranteed even if all N_p parties are dishonest, given that the prover executes a public CRS verification algorithm.

Since it is impossible to construct UC commitments in the standard model [11], the new UC-secure CRS-generation protocol necessarily relies on some trust assumption. The DL-extractable commitment scheme of [1] is secure in the registered public key (RPK) model[3] that is a weaker trust model than the CRS model. However, we stay agnostic to the concrete implementation of $\mathcal{F}_{\mathsf{dlmcom}}$, proving the security of the CRS-generation protocol in the $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model. Thus, the trust assumption of the CRS-generation protocol is directly inherited from the trust assumption of the used DL-extractable commitment scheme. Constructing DL-extractable commitment schemes in a weaker model like the random string model or the multi-string model is an interesting open question. Note that CRS-s of known *efficient* $\mathcal{C}^{\mathsf{S}}$-SNARKs, with a few exceptions, contain $\Omega(\mathsf{n})$ group elements, where n is the circuit size (e.g., in the last CRS generation of Zcash [5], $\mathsf{n} \approx 2\,000\,000$ [4]). Hence, even a relatively inef-

[3] In the RPK model, each party registers his public key with an authority of his choosing. It is assumed that even authorities of untrusted parties are honest to the extent that they verify the knowledge (e.g., by using a standalone ZK proof) of the corresponding secret key.

[4] See `https://www.zfnd.org/blog/conclusion-of-powers-of-tau/`

ficient DL-extractable commitment scheme (that only has to be called once per CRS trapdoor) will not be the bottleneck in the CRS-generation protocol.

We proceed as follows. First, we describe an ideal functionality $\mathcal{F}_{\mathsf{mcrs}}$, an explicit multi-party version of the CRS generation functionality. Intuitively (the real functionality is slightly more complicated), first, N_p key-generators $\mathcal{G}_i$ send to $\mathcal{F}_{\mathsf{mcrs}}$ their shares of the trapdoors, s.t. the shares of the honest parties are guaranteed to be uniformly random. Second, $\mathcal{F}_{\mathsf{mcrs}}$ combines the shares to create the trapdoors and the CRS, and then sends the CRS to each $\mathcal{G}_i$.

We propose a protocol $\mathsf{K}_{\mathsf{mcrs}}$ that UC-realizes $\mathcal{F}_{\mathsf{mcrs}}$ in the $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model, i.e., assuming the availability of a UC-secure realization of $\mathcal{F}_{\mathsf{dlmcom}}$. In $\mathsf{K}_{\mathsf{mcrs}}$, the parties $\mathcal{G}_i$ first $\mathcal{F}_{\mathsf{dlmcom}}$-commit to their individual share of each trapdoor. After opening the commitments, $\mathcal{G}_i$ compute crs by combining their shares with a variation of the protocol from [6]. The structure of this part of the protocol makes it possible to publicly check that it was correctly followed.

Next, we prove that a $\mathcal{C}^{\mathsf{S}}$-SNARK that is complete, sound, and Sub-ZK in the CRS model is also complete, sound, and Sub-ZK in the $\mathcal{F}_{\mathsf{mcrs}}$-hybrid model. Sub-ZK holds even if all CRS creators were malicious, but for soundness we need at least one honest party. We then show that the Sub-ZK secure version [2,14] of the most efficient known zk-SNARK by Groth [20] remains sound and Sub-ZK if the CRS has been generated by using $\mathsf{K}_{\mathsf{mcrs}}$. The main technical issue here is that since Groth's zk-SNARK is not $\mathcal{C}^{\mathsf{S}}$-SNARK (see Section 3), we need to add some new elements to its CRS and then reprove its soundness against an adversary who is given access to the new CRS elements. We note that Bowe et al. [9] proposed a different modification of Groth's zk-SNARK together with a CRS-generation protocol, but under strong assumptions of random beacon model, random oracle model, and knowledge assumptions. Role of the commitment in their case is substituted with a random beacon which in particular means that they do not need to fix parties in the beginning of the protocol.

We constructed a UC-secure CRS-generation protocol $\mathsf{K}_{\mathsf{mcrs}}$ in the $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model for any $\mathcal{C}^{\mathsf{S}}$-SNARK and in particular proved that a small modification of Groth's zk-SNARK is secure when composed with $\mathsf{K}_{\mathsf{mcrs}}$. Moreover, the resulting CRS-generation protocol is essentially as efficient as the prior protocols from [6,8,9]. However, (i) we proved the UC-security of the new CRS-generation protocol, and (ii) the new protocol is falsifiable, i.e., it does not require either the random oracle model or any knowledge assumption.

2 Preliminaries

Let PPT denote probabilistic polynomial-time. Let $\lambda \in \mathbb{N}$ be the information-theoretic security parameter, in practice, e.g., $\lambda = 128$. All adversaries will be stateful. For an algorithm $\mathcal{A}$, let $\mathrm{im}(\mathcal{A})$ be the image of $\mathcal{A}$, i.e., the set of of valid outputs of $\mathcal{A}$, let $\mathsf{RND}(\mathcal{A})$ denote the random tape of $\mathcal{A}$, and let $r \leftarrow_{\$} \mathsf{RND}(\mathcal{A})$ denote sampling of a randomizer r of sufficient length for $\mathcal{A}$'s needs. By $y \leftarrow \mathcal{A}(x; r)$ we denote that $\mathcal{A}$, given an input x and a randomizer r, outputs y. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function, and by $\mathsf{poly}(\lambda)$ an arbitrary polynomial function. $A \approx_c B$ means that distributions A and B are computationally indistinguishable. We write $x \leftarrow_{\$} \mathcal{D}$ if x is sampled according to distribution $\mathcal{D}$ or uniformly in case $\mathcal{D}$ is a set. By $\mathrm{Supp}(\mathcal{D})$ we denote the set of all elements in $\mathcal{D}$ that have non-zero probability.

Assume that $\mathcal{G}_i$ are different parties of a protocol. Following previous work [6], we will make the following assumptions about the network and the adversary. It is possible that the new protocols can be implemented in the asynchronous model but this is out of scope of the current paper.

Synchronicity assumptions: We assume that the computation can be divided into clearly divided rounds. As it is well-known, synchronous computation can be simulated, assuming bounded delays and bounded time-drift. For the sake of simplicity, we omit formal treatment of UC-secure synchronous execution, see [23] for relevant background.

Authentication: we assume the existence of an authenticated broadcast between the parties. In particular, (i) if an honest party broadcasts a message, we assume that all parties (including, in the UC-setting, the simulator) receive it within some delay, and (ii) an honest party $\mathcal{G}_j$ accepts a message as coming from $\mathcal{G}_i$ only if it was sent by $\mathcal{G}_i$.

Covertness: We assume that an adversary in the multi-party protocols is covert, i.e., it will not produce outputs that will not pass public verification algorithms. In the protocols we write that honest parties will abort under such circumstances, but in the proofs we assume that adversary will not cause abortions.

For pairing-based groups we will use additive notation together with the bracket notation, i.e., in group $\mathbb{G}_\iota$, $[a]_\iota = a\,[1]_\iota$, where $[1]_\iota$ is a fixed generator of $\mathbb{G}_\iota$. A deterministic *bilinear group generator* $\mathsf{Pgen}(1^\lambda)$ returns $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where p (a large prime) is the order of cyclic abelian groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear pairing, s.t. $\hat{e}([a]_1, [b]_2) = [ab]_T$. Denote $[a]_1 \bullet [b]_2 = \hat{e}([a]_1, [b]_2)$; this extends to vectors in a natural way. Occasionally we write $[a]_\iota \bullet [b]_{3-\iota}$ for $\iota \in \{1, 2\}$ and ignore the fact that for $\iota = 2$ it should be written $[b]_{3-\iota} \bullet [a]_\iota$. Let $[a]_\star := ([a]_1, [a]_2)$. As in [4], we will implicitly assume that p is generated deterministically from λ; in particular, the choice of p cannot be subverted.

UC Security. We work in the standard universal composability framework of Canetti [10] with static corruptions of parties. The UC framework defines a PPT environment machine $\mathcal{Z}$ that oversees the execution of a protocol in one of two worlds. The "ideal world" execution involves "dummy parties" (some of whom may be corrupted by an ideal adversary/simulator Sim) interacting with a functionality $\mathcal{F}$. The "real world" execution involves PPT parties (some of whom may be corrupted by a PPT real world adversary $\mathcal{A}$) interacting only with each other in some protocol π. We refer to [10] for a detailed description of the executions, and a definition of the real world ensemble $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and the ideal world ensemble $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim}^{\mathcal{A}},\mathcal{Z}}$. A protocol π *UC-securely computes* $\mathcal{F}$ if there exists a PPT Sim such that for every non-uniform PPT $\mathcal{Z}$ and PPT $\mathcal{A}$, $\{\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim}^{\mathcal{A}},\mathcal{Z}}(\lambda, \mathsf{x})\}_{\lambda\in\mathbb{N},\mathsf{x}\in\{0,1\}^*} \approx_c \{\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, \mathsf{x})\}_{\lambda\in\mathbb{N},\mathsf{x}\in\{0,1\}^*}$.

$\mathcal{F}_{\mathsf{crs}}^{\mathcal{D},f}$ is parametrized by a distribution $\mathcal{D}$ and a function f. It proceeds as follows, running with parties $\mathcal{G}_i$ and an adversary Sim.

CRS generation: Sample $\mathsf{tc} \leftarrow_{\$} \mathcal{D}$; Set $\mathsf{crs} \leftarrow f(\mathsf{tc})$; Send $(\mathtt{crsOK?}, \mathsf{sid}, \mathsf{crs})$ to Sim; If Sim returns $(\mathtt{crsOK}, \mathsf{sid})$ then store $(\mathsf{sid}, \mathsf{crs})$.

Retrieval: upon receiving $(\mathtt{retrieve}, \mathsf{sid})$ from $\mathcal{G}_i$: If $(\mathsf{sid}, \mathsf{crs})$ is recorded for some crs then send $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs})$ to $\mathcal{G}_i$. Otherwise, ignore the message.

Fig. 1: Functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D},f}$

$\mathcal{F}_{\mathsf{dlmcom}}$, parametrized by $\mathcal{M} = \mathbb{Z}_p$ and group $\mathbb{G}_\iota$, interacts with $\mathcal{G}_1, \ldots, \mathcal{G}_{\mathsf{N}_p}$ as follows.

– Upon receiving $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ from $\mathcal{G}_i$, where $m \in \mathbb{Z}_p$: if a tuple $(\mathsf{sid}, \mathsf{cid}, \cdots)$ with the same $(\mathsf{sid}, \mathsf{cid})$ was previously recorded, do nothing. Otherwise, record $(\mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ and send $(\mathtt{rcpt}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j)$ to $\mathcal{G}_j$ and Sim.

– Upon receiving $(\mathsf{open}, \mathsf{sid}, \mathsf{cid})$ from $\mathcal{G}_i$, proceed as follows: if a tuple $(\mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ was previously recorded then send $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, y \leftarrow [m]_\iota)$ to $\mathcal{G}_j$ and Sim. Otherwise do nothing.

Fig. 2: Functionality $\mathcal{F}_{\mathsf{dlmcom}}$ for $\iota \in \{1, 2\}$

The importance of this definition is a composition theorem that states that any protocol that is universally composable is secure when run concurrently with many other arbitrary protocols; see [10] for discussions and definitions.

CRS functionality. The CRS model UC functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D},f}$ parameterized by a distribution $\mathcal{D}$ and a function f intuitively works as follows. Functionality samples a trapdoor tc from $\mathcal{D}$, computes $\mathsf{crs} = f(\mathsf{tc})$, and stores crs after a confirmation from the simulator. Subsequently on each retrieval query $(\mathtt{retrieve}, \mathsf{sid})$ it responds by sending $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs})$. For full details see Fig. 1.

DL-extractable UC Commitment. Abdolmaleki et al. [1] recently proposed a discrete logarithm extractable (DL-extractable) UC-commitment scheme. Differently from the usual UC-commitment, a committer will open the commitment to $[m]_1$, but the functionality also guarantees that the committer knows x. Hence, in the UC security proof it is possible to extract the discrete logarithm of $[m]_1$. Formally, the ideal functionality $\mathcal{F}_{\mathsf{dlmcom}}$ takes m as a commitment input (hence the user must know m), but on open signal only reveals $[m]_1$. See Fig. 2. We refer to [1] for a known implementation of $\mathcal{F}_{\mathsf{dlmcom}}$ in the RPK model.

Non-interactive zero-knowledge. Let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\mathsf{x}, \mathsf{w})\}$. Here, x is the statement and w is the witness. We assume that λ is explicitly deducible from the description of $\mathbf{R}$. The relation generator also outputs auxiliary information $\xi_{\mathbf{R}}$ that will be given to the honest parties and the adversary. As in [20,2], $\xi_{\mathbf{R}}$ is the value returned by $\mathsf{Pgen}(1^\lambda)$. Because of this, we also give $\xi_{\mathbf{R}}$ as an input to the honest parties; if needed, one can include an additional auxiliary input to the adversary. Let $\mathbf{L_R} = \{\mathsf{x} : \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathbf{R}\}$ be an $\mathbf{NP}$-language.

A (subversion-resistant) *non-interactive zero-knowledge argument system* [2] Ψ for $\mathcal{R}$ consists of six PPT algorithms:

CRS trapdoor generator: $\mathsf{K_{tc}}$ is a PPT algorithm that, given $(\mathbf{R}, \xi_{\mathbf{R}}) \in \operatorname{im}(\mathcal{R}(1^\lambda))$, outputs a *CRS trapdoor* tc. Otherwise, it outputs $\perp$.

CRS generator: $\mathsf{K_{crs}}$ is a *deterministic* algorithm that, given $(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{tc})$, where $(\mathbf{R}, \xi_{\mathbf{R}}) \in \operatorname{im}(\mathcal{R}(1^\lambda))$ and $\mathsf{tc} \in \operatorname{im}(\mathsf{K_{tc}}(\mathbf{R}, \xi_{\mathbf{R}})) \setminus \{\perp\}$, outputs crs. Otherwise, it outputs $\perp$. We distinguish three parts of crs: $\mathsf{crs_P}$ (needed by the prover), $\mathsf{crs_V}$ (needed by the verifier), and $\mathsf{crs_{CV}}$ (needed by CV algorithm).

CRS verifier: CV is a PPT algorithm that, given $(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs})$, returns either 0 (the CRS is ill-formed) or 1 (the CRS is well-formed).

Prover: P is a PPT algorithm that, given $(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs_P}, \mathsf{x}, \mathsf{w})$, where $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\perp$.

Verifier: V is a PPT algorithm that, given $(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs_V}, \mathsf{x}, \pi)$, returns either 0 (reject) or 1 (accept).

Simulator: Sim is a PPT algorithm that, given $(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}, \mathsf{tc}, \mathsf{x})$, outputs an argument π.

We also define the CRS generation algorithm $\mathsf{K}(\mathbf{R}, \xi_{\mathbf{R}})$ that first sets $\mathsf{tc} \leftarrow \mathsf{K_{tc}}(\mathbf{R}, \xi_{\mathbf{R}})$ and then outputs $\mathsf{crs} \leftarrow \mathsf{K_{crs}}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{tc})$.

Ψ is *perfectly complete for* $\mathcal{R}$, if for all λ, $(\mathbf{R}, \xi_{\mathbf{R}}) \in \operatorname{im}(\mathcal{R}(1^\lambda))$, $\mathsf{tc} \in \operatorname{im}(\mathsf{K_{tc}}(\mathbf{R}, \xi_{\mathbf{R}})) \setminus \{\perp\}$, and $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$,

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{K_{crs}}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{tc}) : \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs_V}, \mathsf{x}, \mathsf{P}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs_P}, \mathsf{x}, \mathsf{w})) = 1\right] = 1 \ .$$

Ψ is computationally adaptively *knowledge-sound for* $\mathcal{R}$ [20], if for every non-uniform PPT $\mathcal{A}$, there exists a non-uniform PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. $\forall$ λ,

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \xi_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda), (\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{K}(\mathbf{R}, \xi_{\mathbf{R}}), r \leftarrow_r \mathsf{RND}(\mathcal{A}), \\ (\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}; r), \mathsf{w} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}; r) : \\ (\mathsf{x}, \mathsf{w}) \not\in \mathbf{R} \wedge \mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs_V}, \mathsf{x}, \pi) = 1\end{array}\right] \approx_\lambda 0 \ .$$

Here, $\xi_{\mathbf{R}}$ can be seen as a common auxiliary input to $\mathcal{A}$ and $\mathsf{Ext}_{\mathcal{A}}$ that is generated by using a benign [7] relation generator; we recall that we just think of $\xi_{\mathbf{R}}$ as being the description of a secure bilinear group.

Ψ is *statistically unbounded ZK for* $\mathcal{R}$ [18], if for all λ, all $(\mathbf{R}, \xi_{\mathbf{R}}) \in \operatorname{im}(\mathcal{R}(1^\lambda))$, and all computationally unbounded $\mathcal{A}$, $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$, where

$$\varepsilon_b^{unb} = \Pr[(\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{K}(\mathbf{R}, \xi_{\mathbf{R}}) : \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}) = 1] \ .$$

Here, the oracle $\mathsf{O}_0(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}$, and otherwise it returns $\mathsf{P}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs_P}, \mathsf{x}, \mathsf{w})$. Similarly, $\mathsf{O}_1(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \not\in \mathbf{R}$, and otherwise it returns $\mathsf{Sim}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}, \mathsf{tc}, \mathsf{x})$. Ψ is *perfectly unbounded ZK for* $\mathcal{R}$ if one requires that $\varepsilon_0^{unb} = \varepsilon_1^{unb}$.

Ψ is *statistically unbounded Sub-ZK for* $\mathcal{R}$, if for any non-uniform PPT subverter X there exists a non-uniform PPT $\mathsf{Ext_X}$, such that for all λ, $(\mathbf{R}, \xi_{\mathbf{R}}) \in \operatorname{im}(\mathcal{R}(1^\lambda))$, and computationally unbounded $\mathcal{A}$, $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$, where

$$\varepsilon_b^{unb} = \Pr\left[\begin{array}{l} r \leftarrow_r \mathsf{RND}(\mathsf{X}), (\mathsf{crs}, \xi_{\mathsf{X}}) \leftarrow \mathsf{X}(\mathbf{R}, \xi_{\mathbf{R}}; r), \mathsf{tc} \leftarrow \mathsf{Ext_X}(\mathbf{R}, \xi_{\mathbf{R}}; r) : \\ \mathsf{CV}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}) = 1 \wedge \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}, \mathsf{tc}, \xi_{\mathsf{X}}) = 1 \end{array}\right] \ .$$

Here, the oracle $\mathsf{O}_0(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}$, and otherwise it returns $\mathsf{P}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}_\mathsf{P}, \mathsf{x}, \mathsf{w})$. Similarly, $\mathsf{O}_1(\mathsf{x}, \mathsf{w})$ returns $\perp$ (reject) if $(\mathsf{x}, \mathsf{w}) \notin \mathbf{R}$, and otherwise it returns $\mathsf{Sim}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}, \mathsf{tc}, \mathsf{x})$. Ψ is *perfectly unbounded Sub-ZK for $\mathcal{R}$* if one requires that $\varepsilon_0^{unb} = \varepsilon_1^{unb}$.

Intuitively the previous definition says that an argument is Sub-ZK when for any untrusted (efficient) CRS generator X, some well-formedness condition $\mathsf{CV}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}) = 1$ implies that X knows a trapdoor which would allow him to simulate the proof. Hence, to protect privacy from malicious CRS generators, the prover just needs to verify that the CRS satisfies the CV algorithm.

Finally, a non-interactive argument system is *succinct* if the argument length is polynomial in λ and the verifier runs in time polynomial in $\lambda + |\mathsf{x}|$.

3 Multi-Party CRS Generation

Recently, [6,8,9] proposed several multi-party CRS-generation protocols for SNARKs. In particular, [6] proposes a specific class of arithmetic circuits $\mathcal{C}^\mathsf{S}$, shows how to evaluate $\mathcal{C}^\mathsf{S}$-circuits in an MPC manner, and claims that $\mathcal{C}^\mathsf{S}$-circuits can be used to compute CRS-s for a broad class of SNARKs, in this paper called $\mathcal{C}^\mathsf{S}$-SNARKs. The CRS of each $\mathcal{C}^\mathsf{S}$-SNARK is an output of some $\mathcal{C}^\mathsf{S}$-circuit taken into exponent. The input of such circuit is the CRS trapdoor. In the following, we review and modify the framework of [6] redefining slightly the class $\mathcal{C}^\mathsf{S}$ and the CRS-generation protocol.

$\mathcal{C}^\mathsf{S}$-Circuits. For an arithmetic circuit C over a field $\mathbb{F}$, denote by $\mathsf{wires}(\mathsf{C})$ and $\mathsf{gates}(\mathsf{C})$ the set of wires and gates of C (each gate can have more than one output wire), and by $\mathsf{inputs}(\mathsf{C}), \mathsf{outputs}(\mathsf{C}) \subset \mathsf{wires}(\mathsf{C})$ the set of input and output wires of C. There can also be wires with hard-coded constant values, but these are not considered to be part of $\mathsf{inputs}(\mathsf{C})$. The size of C is $|\mathsf{inputs}(\mathsf{C})| + |\mathsf{gates}(\mathsf{C})|$. For a wire w we denote the value on the wire by $\bar{w}$; this notation also extends to tuples, say, $\overline{\mathsf{inputs}}(\mathsf{C})$ denotes the tuple of values of $\mathsf{inputs}(\mathsf{C})$.

For a gate g, $\mathsf{output}(g) = w$ is the output wire and the tuple of all input wires is denoted by $\mathsf{inputs}(g)$. Let g_w be the gate with $w = \mathsf{output}(g_w)$. We consider circuits with *addition* and *multiplication-division* gates. For an addition gate ($\mathsf{type}(g) = \mathsf{add}$), $\mathsf{inputs}(g) = (w_1, \ldots, w_f)$, $\mathsf{coeffs}(g) = (a_0, a_1, \ldots, a_f)$, and it outputs a value $\bar{w} = a_0 + \sum_{j=1}^{f} a_j \bar{w}_j$. For a $\mathsf{multdiv}$ gate ($\mathsf{type}(g) = \mathsf{multdiv}$), $\mathsf{inputs}(g) = (w_1, w_2, w_3)$, $\mathsf{L\text{-}input}(g) = w_1$ is the left multiplication input, $\mathsf{R\text{-}input}(g) = w_2$ is the right multiplication input, $\mathsf{D\text{-}input}(g) = w_3$ is the division input, and $\mathsf{coeffs}(g) = a$. The output wire w contains the value $\bar{w} = a\bar{w}_1\bar{w}_2/\bar{w}_3$. Previous works either only considered multiplication gates [6] or separate multiplication and division gates [9]. Using $\mathsf{multdiv}$ gates can, under some circumstances, reduce the circuit size compared to separate multiplication and division gates. A smaller circuit gives lower computation and communication cost for our CRS-generation protocol as we see later.

Class $\mathcal{C}^\mathsf{S}$ contains $\mathbb{F}$-arithmetic circuits $\mathsf{C} : \mathbb{F}^t \to \mathbb{F}^h$, such that:

(1) For any $w \in \mathsf{inputs}(\mathsf{C})$, there exists $g \in \mathsf{gates}(\mathsf{C})$ such that $\mathsf{type}(g) = \mathsf{multdiv}$, $\overline{\mathsf{inputs}}(g) = (1, \bar{w}, 1)$, and $\mathsf{coeffs}(g) = 1$. That is, each trapdoor itself should

be a part of the output of the circuit. Adding those $\mathsf{multdiv}$ gates corresponds to the MPC protocol combining the shares of trapdoor ts of each party to get $[\mathsf{tc}]_\iota$.

(2) For any $g \in \mathsf{gates}(\mathsf{C})$:
   (a) $\mathsf{output}(g) \in \mathsf{outputs}(\mathsf{C})$. Hence, each gate output is a CRS element.
   (b) If $\mathsf{type}(g) = \mathsf{multdiv}$ then $\mathsf{L\text{-}input}(g) \notin \mathsf{inputs}(C)$, $\mathsf{R\text{-}input}(g)$, $\mathsf{D\text{-}input}(g) \in \mathsf{inputs}(C)$. That is, the left multiplication input can be a constant or an output of a previous gate, the right multiplication and division inputs have to be one of the inputs of the circuit. This allows to easily verify the computation in the MPC. For convenience, we require further that constant value of $\mathsf{L\text{-}input}(g)$ can only be 1; from computational point of view nothing changes since $\mathsf{coeffs}(g)$ can be any constant.
   (c) If $\mathsf{type}(g) = \mathsf{add}$ then $\mathsf{inputs}(g) \cap \mathsf{inputs}(\mathsf{C}) = \emptyset$. Addition is done locally in MPC (does not require additional rounds) with the outputs of previous gates, since outputs correspond to publicly known CRS elements.

The *sampling depth* $\mathrm{depth}_\mathsf{S}$ of a gate $g \in \underline{\mathsf{gates}(\mathsf{C})}$ is defined as follows:

1. $\mathrm{depth}_\mathsf{S}(g) = 1$ if g is a $\mathsf{multdiv}$ gate and $\overline{\mathsf{L\text{-}input}}(g)$ is a constant.
2. $\mathrm{depth}_\mathsf{S}(g) = \max\{\mathrm{depth}_\mathsf{S}(g') : g' \text{ an input of } g\}$ for other $\mathsf{multdiv}$ gates,
3. $\mathrm{depth}_\mathsf{S}(g) = b_g + \max\{\mathrm{depth}_\mathsf{S}(g') : g' \text{ an input of } g\}$ for any add gate, where (i) $b_g = 0$ iff all the input gates of g are add gates. (ii) $b_g = 1$, otherwise.

Denote $\mathrm{depth}_\mathsf{S}(\mathsf{C}) := \max_g\{\mathrm{depth}_\mathsf{S}(g)\}$. We again defined $\mathrm{depth}_\mathsf{S}$ slightly differently compared to [6]; our definition emphasizes the fact that addition gates can be executed locally. Essentially, $\mathsf{N}_p \cdot \mathrm{depth}_\mathsf{S}(g_w)$ will be the number of rounds that it takes to compute $[\bar{w}]_\iota$ with our MPC protocol. The *multiplicative depth* of a circuit (denoted by $\mathrm{depth}_\mathsf{M}(\mathsf{C})$) is the maximum number of multiplication gates from any input to any output. An exemplary $\mathcal{C}^\mathsf{S}$-circuit is given in Fig. 3.

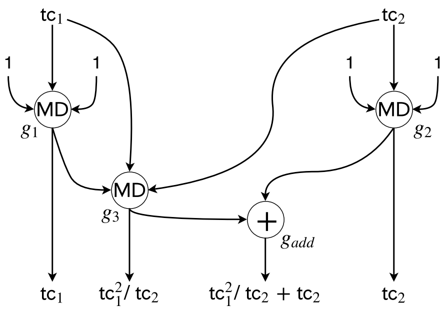

Fig. 3: Example $\mathcal{C}^\mathsf{S}$ circuit with inputs tc_1 and tc_2

Multi-Party Circuit Evaluation Protocol. We describe the circuit evaluation protocol, similar to the one in [6], that allows to evaluate any $\mathcal{C}^\mathsf{S}$-circuits "in the exponent". We assume there are N_p parties $\mathcal{G}_i$, each having published a share $[\mathsf{ts}_{i,s}]_\star \in \mathbb{F}^*$, for $s \in [1\,..\,t]$. The goal of the evaluation protocol is to output $\big([\mathsf{C}_1(\mathsf{tc})]_1, [\mathsf{C}_2(\mathsf{tc})]_2\big)$ where $\mathsf{C}_1, \mathsf{C}_2$ are $\mathcal{C}^\mathsf{S}$-circuits and $\mathsf{tc} = (\prod_j \mathsf{ts}_{j,1}, \ldots, \prod_j \mathsf{ts}_{j,t})$.

```
$\mathsf{C}_{\mathsf{md}}([b]_\iota, a, \mathsf{ts}_{i,s}, \mathsf{ts}_{i,k})$
return $(a \cdot (\mathsf{ts}_{i,s}/\mathsf{ts}_{i,k}))[b]_\iota$;

$\mathsf{V}_{\mathsf{md}}([b']_\iota, [b]_\iota, a, [\mathsf{ts}_{i,s}]_{3-\iota}, [\mathsf{ts}_{i,k}]_{3-\iota})$
return $(([b']_\iota = [0]_\iota) \vee$
    $([b']_\iota \bullet [\mathsf{ts}_{i,k}]_{3-\iota} \neq a[b]_\iota \bullet [\mathsf{ts}_{i,s}]_{3-\iota}))\,?\,0 : 1$;

$\mathsf{Eval}_{\mathsf{md}}(a, b, s, k)$
$[b_0]_\iota \leftarrow a[b]_\iota$; // Multiplication with $a$ is done by $\mathcal{G}_1$
for $i \in [1 \mathinner{..} \mathsf{N}_p]$ do $\mathcal{G}_i$ broadcasts $[b_i]_\iota \leftarrow \mathsf{C}_{\mathsf{md}}([b_{i-1}]_\iota, 1, \mathsf{ts}_{i,s}, \mathsf{ts}_{i,k})$;
return $[b_{\mathsf{N}_p}]_\iota$;
```

Fig. 4: Algorithms C_{md}, V_{md} and the protocol $\mathsf{Eval}_{\mathsf{md}}$ for $\iota \in \{1, 2\}$.

This protocol constructs a well-formed CRS, given that tc is the CRS trapdoor and $[\mathsf{C}_1(\mathsf{tc})]_1$, $[\mathsf{C}_2(\mathsf{tc})]_2$ are respectively all the $\mathbb{G}_1$ and $\mathbb{G}_2$ elements of the CRS. In Section 4, we combine the circuit evaluation protocol with a UC-secure commitment scheme to obtain a UC-secure CRS-generation protocol. Each step in the circuit evaluation protocol is publicly verifiable and hence, no trust is needed at all; except that to get the correct distribution we need to trust one party.

We make two significant changes to the circuit evaluation protocol compared to [6]: (i) we do not require that $\mathsf{C}_1 = \mathsf{C}_2$, allowing CRS elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ to be different, and (ii) instead of multiplication gates we evaluate $\mathsf{multdiv}$ gates.

Let us first describe the computation of $[\bar{w}]_\iota$ for a single gate g_w. For an add gate, given that all input gates have already been computed, that is, $[\bar{w}_1, \dots, \bar{w}_f]_\iota$ are already public, each $\mathcal{G}_i$ computes $[\bar{w}]_\iota = a_0 + \sum_{j=1}^{f} a_j [\bar{w}_j]_\iota$ locally. A $\mathsf{multdiv}$ gate g, with $\overline{\mathsf{inputs}}(g) = (b, \mathsf{tc}_s, \mathsf{tc}_k)$ and $\mathsf{coeffs}(g) = a$, can be implemented by the N_p-round protocol $\mathsf{Eval}_{\mathsf{md}}$ from Fig. 4. Here, each party $\mathcal{G}_i$ takes as input $[b]_\iota$ (the output of the preceeding gate or just $[1]_\iota$ if there is none), runs C_{md} procedure on $\mathsf{ts}_{i,s} \in \mathbb{F}$, $\mathsf{ts}_{i,k} \in \mathbb{F}$ (her shares of the trapdoor that are also g's inputs), and broadcasts its output. Note that $[b]_\iota$ corresponds to the left multiplication, $\mathsf{ts}_{i,s}$ to the right multiplication, and $\mathsf{ts}_{i,k}$ to the division input of g.

Importantly, since each party $\mathcal{G}_i$ published $[\{\mathsf{ts}_{i,j}\}_{j=1}^{t}]_\star$, everybody can verify that $\mathcal{G}_i$ executed C_{md} correctly by checking if $\mathsf{V}_{\mathsf{md}}([b_i]_\iota, [b_{i-1}]_\iota, a, [\mathsf{ts}_{i,s}, \mathsf{ts}_{i,k}]_{3-\iota}) = 1$, where $[b_i]_\iota$ is $\mathcal{G}_i$'s output and $[b_{i-1}]_\iota$ is her input (the output of the party $\mathcal{G}_{i-1}$). We assume $[b_0]_\iota = [1]_\iota$ to allow the parties to check the computations of $\mathcal{G}_1$. Just running $\mathsf{Eval}_{\mathsf{md}}$ to evaluate each $\mathsf{multdiv}$ gate in C would require $\approx \mathsf{N}_p \cdot \mathrm{depth}_{\mathsf{M}}(\mathsf{C})$ rounds. Next we see that computation can be parallelized to obtain $\mathsf{N}_p \cdot \mathrm{depth}_{\mathsf{S}}(\mathsf{C})$ rounds.

Optimised Multi-Party Circuit Evaluation Protocol. Before presenting the complete (parallelised) circuit evaluation protocol, we provide an illustrative example of how $\mathcal{C}^{\mathsf{S}}$-circuits can be evaluated efficiently using multiple parties. The idea behind this approach is to allow parties to evaluate the circuit not gate-by-gate but all the gates of the same sampling depth. We say that gates are in the same *layer* if they have the same $\mathrm{depth}_{\mathsf{S}}$. Following the definition of $\mathrm{depth}_{\mathsf{S}}$, layers are separated by add gates. That is, two gates, say g_1 and g_2 are in different layers if there is an add gate g_{add} such that g_1 (or g_2) depends on g_{add}'s output, while the other gate does not. In each layer, each gate is computed using only trapdoor elements and outputs from gates of some preceding layer.

Parties evaluate the layer in a round-robin manner broadcasting intermediate values which allows other parties to verify the computation.

This is how the optimised protocol and the naive MPC protocol differ. Since naive protocol evaluates circuit gate-by-gate, one gate's output can be another's input even if both share the same layer. For instance, consider gates g_1 and g_3 from Fig. 3. There, g_1's output is g_3's input and they are both in the same layer. Since the output of g_1 is computed before g_3 is evaluated, it can be used in the computation. On the other hand, in the optimised version of circuit evaluation all gates in the same layer are evaluated at the same time, thus g_3 is computed at the same time when g_1 is computed. To illustrate this idea we provide an example.

Example 1. Suppose we have parties $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$ that wish to compute $\mathsf{crs} = \{[\mathsf{tc}_1]_\star, [\mathsf{tc}_2]_\star, [\mathsf{tc}_1^2/\mathsf{tc}_2]_1, [\mathsf{tc}_1^2/\mathsf{tc}_2 + \mathsf{tc}_2]_1\}$. Let us only focus on the computation of $\mathbb{G}_1$ elements. This is represented by a $\mathcal{C}^\mathsf{S}$-circuit in Fig. 3 where we have (i) a $\mathsf{multdiv}$ gate g_1 with input values $(1, \mathsf{tc}_1, 1)$, (ii) a $\mathsf{multdiv}$ gate g_2 with input values $(1, \mathsf{tc}_2, 1)$, (iii) a $\mathsf{multdiv}$ gate g_3 that takes the output of g_1 as $\mathsf{L\text{-}input}$, the circuit's inputs tc_1 as $\mathsf{R\text{-}input}$, and tc_2 as $\mathsf{D\text{-}input}$, that is, the input values of g_3 are $(\mathsf{tc}_1, \mathsf{tc}_1, \mathsf{tc}_2)$, and (iv) an add gate g_{add} that adds outputs of g_2 and g_3. The parties respectively publish shares $[\mathsf{ts}_{1,1}, \mathsf{ts}_{1,2}]_\star$, $[\mathsf{ts}_{2,1}, \mathsf{ts}_{2,2}]_\star$, $[\mathsf{ts}_{3,1}, \mathsf{ts}_{3,2}]_\star$.

– *In the first round,* $\mathcal{G}_1$ broadcasts $[b^1_{g_1}]_1 \leftarrow [\mathsf{ts}_{1,1}]_1$ for gate g_1, $[b^1_{g_2}]_1 \leftarrow [\mathsf{ts}_{1,2}]_1$ for gate g_2, and $[b^1_{g_3}]_1 \leftarrow [\mathsf{ts}^2_{1,1}/\mathsf{ts}_{1,2}]_1$ for gate g_3.
– *In the second round,* $\mathcal{G}_2$ broadcasts $[b^2_{g_1}]_1 \leftarrow \mathsf{ts}_{2,1} \cdot [b^1_{g_1}]_1$ for gate g_1, $[b^2_{g_2}]_1 \leftarrow \mathsf{ts}_{2,2} \cdot [b^1_{g_2}]_1$ for gate g_2, and $[b^2_{g_3,1}]_1 \leftarrow \mathsf{ts}_{2,1} \cdot [b^1_{g_3}]_1$, $[b^2_{g_3,2}]_1 \leftarrow (\mathsf{ts}_{2,1}/\mathsf{ts}_{2,2}) \cdot [b^2_{g_3,1}]_1$ for g_3 (note that g_3 required two computations rather than one).
– *In the third round,* $\mathcal{G}_3$ broadcasts $[b^3_{g_1}]_1 \leftarrow \mathsf{ts}_{3,1} \cdot [b^2_{g_1}]_1$ for gate g_1, $[b^3_{g_2}]_1 \leftarrow \mathsf{ts}_{3,2} \cdot [b^2_{g_2}]_1$ for gate g_2, and $[b^3_{g_3,1}]_1 \leftarrow \mathsf{ts}_{3,1} \cdot [b^2_{g_3,2}]_1$, $[b^3_{g_3,2}]_1 \leftarrow (\mathsf{ts}_{3,1}/\mathsf{ts}_{3,2}) \cdot [b^3_{g_3,1}]_1$ for g_3. For g_{add} each party computes $[b_{g_{add}}]_1 \leftarrow [b^3_{g_2}]_1 + [b^3_{g_3,2}]_1$.

Finally, if we define $\mathsf{tc}_1 := \mathsf{ts}_{1,1} \cdot \mathsf{ts}_{2,1} \cdot \mathsf{ts}_{3,1}$ and $\mathsf{tc}_2 := \mathsf{ts}_{1,2} \cdot \mathsf{ts}_{2,2} \cdot \mathsf{ts}_{3,2}$, then the outputs of $\mathcal{G}_3$ contain $[b^3_{g_1}]_1 = [\mathsf{tc}_1]_1$, $[b^3_{g_2}]_1 = [\mathsf{tc}_2]_1$, and $[b^3_{g_3,2}]_1 = [\mathsf{tc}_1^2/\mathsf{tc}_2]_1$; moreover, $[b_{g_{add}}]_1 = [\mathsf{tc}_2 + \mathsf{tc}_1^2/\mathsf{tc}_2]_1$. Besides addition, each element is built up one share multiplication at a time and hence the computation can be verified with pairings, e.g, the last output $[b^2_{g_3,2}]_1$ of $\mathcal{G}_2$ is correctly computed exactly when $[b^2_{g_3,2}]_1 \bullet [\mathsf{ts}_{2,2}]_2 = [b^2_{g_3,1}]_1 \bullet [\mathsf{ts}_{2,1}]_2$. Computation of this example is illustrated in Fig. 5 $\square$

Motivated by the example above, we give the full and formal description of the circuit evaluation protocol. Let $\mathsf{C}_\iota \in \mathcal{C}^\mathsf{S}$, for $\iota \in \{1, 2\}$, and $\mathsf{C}_{\iota,d} \subseteq \mathsf{gates}(\mathsf{C})$ be a circuit layer that contains all $\mathsf{multdiv}$ gates g at sampling depth d. For any $g \in \mathsf{C}_{\iota,d}$ let $\mathsf{ExtractPath}(g, \mathsf{C}_{\iota,d})$ output the longest path $(g_1, \dots, g_q = g)$ such that each $g_j \in \mathsf{C}_{\iota,d}$, and, for $j < q$, $\mathsf{output}(g_j) = \mathsf{L\text{-}input}(g_{j+1})$. Intuitively, this is the path of gates in $\mathsf{C}_{\iota,d}$ that following only the left inputs lead up to the gate g, say, $\mathsf{ExtractPath}(g_3, \mathsf{C}_{1,1}) = (g_1, g_3)$ for the circuit C in Fig. 3. For simplicity, we describe a $\mathsf{multdiv}$ gate g by a tuple $([b]_\iota, a, s, k)$ where $[b]_\iota = [\mathsf{L\text{-}input}(g)]_\iota$ is the left input value, assumed already to be known by the parties, $a = \mathsf{coeffs}(g)$, $\mathsf{R\text{-}input}(g) = \mathsf{tc}_s$, and $\mathsf{D\text{-}input}(g) = \mathsf{tc}_k$.

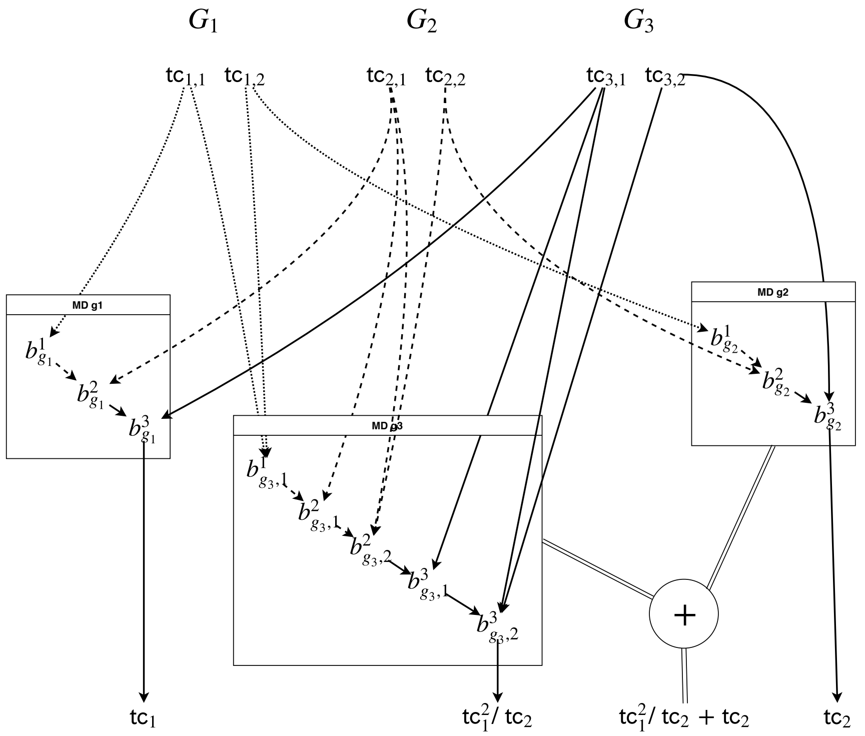

Fig. 5: Illustration for the Example 1. Dotted lines denote computation of $\mathcal{G}_1$, dashed lines computation by $\mathcal{G}_2$, and full lines computation by $\mathcal{G}_3$. Double lines denote computation made by the parties internally (g_{add} gate). Each container denotes a $\mathsf{multdiv}$ gate and shows consecutive stages of computation. Note that all $b^i_{g_j,k}$ are broadcast and $b^3_{g_1} = \mathsf{tc}_1$, $b^3_{g_2} = \mathsf{tc}_2$, $b^3_{g_3} = \frac{(\mathsf{tc}_1)^2}{\mathsf{tc}_2}$.

The parties evaluate $\mathsf{multdiv}$ gates of the circuit in order $\mathsf{C}_{\iota,1}, \mathsf{C}_{\iota,2}, \ldots, \mathsf{C}_{\iota,D_\iota}$, where D_ι is the sampling depth of C_ι. After each layer $C_{\iota,d}$ each party locally evaluates all the addition gates at depth $d+1$. The evaluation of $\mathsf{C}_{\iota,d}$ proceeds in a round-robin fashion. First, $\mathcal{G}_1$ evaluates $\mathsf{C}_{\iota,d}$ with her input shares $\mathsf{ts}_{1,k}$ alone. Next, $\mathcal{G}_2$ multiplies her shares $\mathsf{ts}_{2,k}$ to each output of $\mathcal{G}_1$. However, to make computation verifiable, if $\mathcal{G}_2$ is supposed to compute $[b_g^1 \cdot \mathsf{ts}_{2,\alpha_1} \cdot \ldots \mathsf{ts}_{2,\alpha_q}]_\iota$, where $[b_g^1]_\iota$ is some output of $\mathcal{G}_1$, then it is done one multiplication at a time. Namely, she outputs $[b_{g,1}^2]_\iota = [b_g^1 \cdot \mathsf{ts}_{2,\alpha_1}]_\iota$, $[b_{g,2}^2]_\iota = [b_{g,1}^2 \cdot \mathsf{ts}_{2,\alpha_2}]_\iota$, $\ldots$, $[b_{g,q}^2]_\iota = [b_{g,q-1}^2 \cdot \mathsf{ts}_{2,\alpha_q}]_\iota$. Each multiplication would correspond to exactly one gate in $\mathsf{ExtractPath}(g, \mathsf{C}_{\iota,d})$. The elements $[b_{g,1}^2, \ldots, b_{g,q-1}^2]_\iota$ are used only for verification; $[b_{g,q}^2]_\iota$ is additionally used by $\mathcal{G}_3$ to continue the computation. Each subsequent party $\mathcal{G}_i$ multiplies her shares to the output of $\mathcal{G}_{i-1}$ in a similar fashion. This protocol requires only $\mathsf{N}_p \cdot \mathrm{depth}_\mathsf{S}(\mathsf{C}_\iota)$ rounds.

Let $\mathsf{cert}^\iota = (\mathsf{cert}_1^\iota, \ldots, \mathsf{cert}_{D_\iota}^\iota)$ be the total transcript (certificate) in $\mathbb{G}_\iota$ corresponding to the output of the multi-party evaluation of C_ι where cert_r^ι is the transcript in round r. Denote $\mathsf{cert} := (\mathsf{cert}^1, \mathsf{cert}^2)$. All gates of depth r of C_ι are evaluated by a uniquely fixed party $\mathcal{G}_i$. In what follows, let $i = \mathsf{rndplayer}(r)$ be the index of this party.

The complete description of evaluation and verification of a layer $\mathsf{C}_{\iota,d}$ is given in Fig. 6 with function $\mathsf{C}_{\mathsf{layer}}$ and $\mathsf{V}_{\mathsf{layer}}$ that have the following interface. First, for $i = \mathsf{rndplayer}(r)$ and for both $\iota \in \{1,2\}$, in round r to compute $[\mathsf{C}_{\iota,d}(\mathsf{tc})]_\iota$, $\mathcal{G}_i$ computes $\mathsf{cert}_r^\iota \leftarrow \mathsf{C}_{\mathsf{layer}}(\mathsf{C}_{\iota,d}, \iota, i, r, \{\mathsf{ts}_{i,k}\}_{k=1}^t, \{\mathsf{cert}_j^\iota\}_{j=1}^{r-1})$, given a circuit layer $\mathsf{C}_{\iota,d}$, the shares $\mathsf{ts}_{i,k}$ for all t trapdoors of tc_k, and the transcript $\{\mathsf{cert}_j^\iota\}_{j=1}^{r-1}$ of all previous computation. Second, any party can verify, by using the algorithm $\mathsf{V}_{\mathsf{layer}}(\mathsf{C}_{\iota,d}, \iota, i, r, \{[\mathsf{ts}_{i,k}]_{3-\iota}\}_{k=1}^t, \{\mathsf{cert}_j^\iota\}_{j=1}^r)$, that the computation of the circuit layer $\mathsf{C}_{\iota,d}$ in round r has been performed correctly by $\mathcal{G}_i$. In particular, $\mathcal{G}_i$ checks that $\mathsf{V}_{\mathsf{layer}}$ outputs 1 for all rounds since $\mathcal{G}_i$'s previous round before executing $\mathsf{C}_{\mathsf{layer}}$ for her new round. Importantly, executing $\mathsf{V}_{\mathsf{layer}}$ does not assume the knowledge of any trapdoors.

On the Importance of $\mathsf{multdiv}$ Gates. Let us briefly discuss the importance of $\mathsf{multdiv}$ gates. Introduction of $\mathsf{multdiv}$ gates allows for more compact $\mathcal{C}^\mathsf{S}$ circuits which in the CRS generation protocol means that there are less computation and less communication. Suppose the CRS contains an element $[a \cdot (\alpha\beta/\gamma)]_1$, for some trapdoor elements α, β, γ and constant a; $\mathsf{multdiv}$ gates allow to compute the element using a single gate. On the other hand with a separate multiplication and division gate, computing elements like this would require two gates, one that outputs $a\alpha \cdot \beta$ and another one that outputs $(a\alpha\beta)/\gamma$.

Let us see what this means for the efficiency of the CRS generation protocol. Using the protocol $\mathsf{Eval}_{\mathsf{md}}$ in Fig. 4, a $\mathsf{multdiv}$ gate could be evaluated with N_p scalar multiplication and would produce a transcript of N_p group elements. Verification would roughly require $2\mathsf{N}_p$ pairings. Separate multiplication and division gates can also implemented using $\mathsf{Eval}_{\mathsf{md}}$ respectively by either fixing the division input to be 1 or the right multiplication input to be 1. In that case computing $[a \cdot (\alpha\beta/\gamma)]_1$ would double the number of scalar multiplication and pairings needed for computation and would also double the communication size.

$\mathsf{C}_{\mathsf{layer}}(\mathsf{C}_{\iota,d}, \iota, i, r, \{\mathsf{ts}_{i,k}\}_{k=1}^{t}, \{\mathsf{cert}_j^{\iota}\}_{j=1}^{r-1})$ // The following is executed by $\mathcal{G}_i$

```
Assert $i = \mathsf{rndplayer}(r); \mathsf{cert}_r^{\iota} \leftarrow \epsilon$;
for $g = ([b]_{\iota}, a, s, k) \in \mathsf{C}_{\iota,d}$ do// In topological order
  $\mathsf{cert}_{g,i} \leftarrow \epsilon; \quad (g_1, \dots, g_q) \leftarrow \mathsf{ExtractPath}(g, \mathsf{C}_{\iota,d})$;
  if $i = 1$ then
    if $q = 1$ then $[b']_{\iota} \leftarrow \mathsf{C}_{\mathsf{md}}([b]_{\iota}, a, \mathsf{ts}_{1,s}, \mathsf{ts}_{1,k})$;
    else Parse $\mathsf{cert}_{g_{q-1},1} = [b_L]_{\iota}$;
        $[b']_{\iota} \leftarrow \mathsf{C}_{\mathsf{md}}([b_L]_{\iota}, a, \mathsf{ts}_{1,s}, \mathsf{ts}_{1,k})$;
    $\mathsf{cert}_{g,1} \leftarrow [b']_{\iota}$;
  else Parse $\mathsf{cert}_{g,i-1} = [b_1, \dots, b_q]_{\iota}$; $[b']_{\iota} \leftarrow [b_q]_{\iota}$;
    for $j = 1, \dots, q$ do
      Parse $g_j = ([b^*]_{\iota}, a^*, s^*, k^*)$;
      $[b']_{\iota} \leftarrow \mathsf{C}_{\mathsf{md}}([b']_{\iota}, 1, \mathsf{ts}_{i,s^*}, \mathsf{ts}_{i,k^*})$; Append $[b']_{\iota}$ to $\mathsf{cert}_{g,i}$;
  Append $\mathsf{cert}_{g,i}$ to $\mathsf{cert}_r^{\iota}$;
return $\mathsf{cert}_r^{\iota}$;
```

$\mathsf{V}_{\mathsf{layer}}(\mathsf{C}_{\iota,d}, \iota, i, r, \{[\mathsf{ts}_{i,k}]_{3-\iota}\}_{k=1}^{t}, \{\mathsf{cert}_j^{\iota}\}_{j=1}^{r})$

```
Assert $i = \mathsf{rndplayer}(r)$;
for each evaluation of $[b']_{\iota} \leftarrow \mathsf{C}_{\mathsf{md}}([b]_{\iota}, a, \mathsf{ts}_{i,s}, \mathsf{ts}_{i,k})$ in round $r$ by $\mathcal{G}_i$ do
  Extract $[b']_{\iota}, [b]_{\iota}$ from $\{\mathsf{cert}_j^{\iota}\}_{j=1}^{r}$;
  if $\mathsf{V}_{\mathsf{md}}([b']_{\iota}, [b]_{\iota}, a, [\mathsf{ts}_{i,s}]_{3-\iota}, [\mathsf{ts}_{i,k}]_{3-\iota}) = 0$ then return 0;
return 1;
```

Fig. 6: $\mathsf{C}_{\mathsf{layer}}$ and $\mathsf{V}_{\mathsf{layer}}$ for $\iota \in \{1, 2\}$

4 UC-Secure CRS Generation

We propose a functionality $\mathcal{F}_{\mathsf{mcrs}}$ for multi-party CRS generation of any $\mathcal{C}^{\mathsf{S}}$-SNARK. Finally, we construct a protocol $\mathsf{K}_{\mathsf{mcrs}}$ that UC-realizes $\mathcal{F}_{\mathsf{mcrs}}$ in the $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model.

New Ideal Functionality. In Fig. 7, we define the new ideal functionality $\mathcal{F}_{\mathsf{mcrs}} = \mathcal{F}_{\mathsf{mcrs}}^{\mathsf{p},\mathsf{N}_p,\mathbf{C},\mathcal{D},\mathsf{comb}}$ for pairing-based (since it outputs elements from $\mathbb{G}_{\iota}$) multi-party CRS-generation protocol. The CRS is described by a t-input arithmetic circuits $\mathbf{C} := (\mathsf{C}_1, \mathsf{C}_2)$ over a field $\mathbb{F} = \mathbb{Z}_p$ such that $\mathsf{crs} = ([\mathsf{C}_1(\mathsf{tc})]_1, [\mathsf{C}_2(\mathsf{tc})]_2)$ for $\mathsf{tc} \leftarrow_{\$} \mathcal{D}$, where $\mathcal{D}$ is a samplable distribution over $\mathbb{Z}_p^t$.

The trapdoor tc is constructed by combining shares $\mathsf{ts}_i \in \mathrm{Supp}(\mathcal{D})$ of each party $\mathcal{G}_i$ by a function comb. For each honest party $\mathcal{G}_i$, the ideal functionality picks $\mathsf{ts}_i \leftarrow_{\$} \mathcal{D}$, whereas for malicious parties we only know $\mathsf{ts}_i \in \mathrm{Supp}(\mathcal{D})$. The function comb should be defined so that if there exists at least one honest party then $\mathsf{tc} \leftarrow \mathsf{comb}(\mathsf{ts}_1, \dots, \mathsf{ts}_{\mathsf{N}_p})$ is also distributed accordingly to $\mathcal{D}$. In such case we say that $\mathcal{D}$ is comb-*friendly*. It is true for example when comb is point-wise multiplication and $\mathcal{D}$ is a uniform distribution over $(\mathbb{Z}_p^*)^t$ as, e.g., in [6,8,9]. This guarantees the correct distribution of crs if at least one party is honest.

We believe $\mathcal{F}_{\mathsf{mcrs}}$ captures essentially any reasonable pairing-based multi-party CRS-generation protocol, where the trapdoor is shared between N_p parties. Note that specifying distinct honest and corrupted inputs to the functionality is

Parameters: p defines a bilinear pairing, $\mathbf{C} = (\mathsf{C}_1, \mathsf{C}_2)$ contains t-input arithmetic circuits over the field $\mathbb{Z}_p$, $\mathcal{D}$ is a distribution of trapdoor elements, and $\mathsf{comb} : (\mathbb{Z}_p^t)^{\mathsf{N}_p} \to \mathbb{Z}_p^t$. We have parties $\mathcal{G}_i$ for $i \in [1 \mathinner{..} \mathsf{N}_p]$.

Share collection phase:

1. Upon receiving $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_i)$ from an honest $\mathcal{G}_i$, store $\mathsf{ts}_i \leftarrow_{\$} \mathcal{D}$ and send $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_i)$ to Sim.
2. Upon receiving $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_i, \mathsf{ts}_i)$ from a dishonest $\mathcal{G}_i$, if $\mathsf{ts}_i \in \mathrm{Supp}(\mathcal{D})$, then store ts_i, else abort.

Only one message from each $\mathcal{G}_i$ is accepted.

CRS generation phase: Once ts_i is stored for each $\mathcal{G}_i$:

1. Compute $\mathsf{tc} \leftarrow \mathsf{comb}(\mathsf{ts}_1, \dots, \mathsf{ts}_{\mathsf{N}_p})$.
2. Set $\mathsf{crs} \leftarrow ([\mathsf{C}_1(\mathsf{tc})]_1, [\mathsf{C}_2(\mathsf{tc})]_2)$ and send $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs})$ to Sim.
3. If Sim returns $(\mathtt{CRS}, \mathsf{ok})$ then send $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs})$ to every party $\mathcal{G}_i$ for $i \in [1 \mathinner{..} \mathsf{N}_p]$.

Fig. 7: Ideal functionality $\mathcal{F}_{\mathsf{mcrs}}$

common in the UC literature, [3,24]. In Theorem 2, we will establish the relation between $\mathcal{F}_{\mathsf{crs}}$ and $\mathcal{F}_{\mathsf{mcrs}}$.

New Protocol. We define the new multi-party CRS-generation protocol $\mathsf{K}_{\mathsf{mcrs}} = \mathsf{K}_{\mathsf{mcrs}}^{\mathsf{p}, \mathsf{N}_p, \mathbf{C}, \mathcal{D}, \mathsf{comb}}$ (see Fig. 8) in the $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model. This allows us to instantiate the protocol with any DL-extractable commitment and, moreover, the only trust assumption that the protocol needs is the one inherited from the commitment scheme, e.g., using construction from [1] gives security in the RPK model. Given that D_ι is the sampling depth of C_ι, then $R = \mathsf{N}_p \cdot \max(D_1, D_2)$ is the number of rounds needed to evaluate both circuits in parallel. For the sake of simplicity, we assume cert_r^ι is the empty string for $r > \mathsf{N}_p \cdot D_\iota$.

$\mathsf{K}_{\mathsf{mcrs}}$ proceeds in rounds: (i) In round 1, each $\mathcal{G}_i$ gets a signal $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_i)$; parties commit to their shares of trapdoor tc. (ii) In round 2, each party $\mathcal{G}_i$ gets a signal $(\mathtt{mcrsopen}, \mathsf{sid})$; parties open their shares. (iii) In round $r \in [3 \mathinner{..} R + 2]$, $(\mathtt{mcrscertok}, \mathsf{sid}, \mathcal{G}_i, r)$ is triggered, where $i = \mathsf{rndplayer}(r)$; parties jointly compute crs from the trapdoor shares; before party $\mathcal{G}_i$ performs her computation, she checks if previous computation were done correctly. (iv) In round $R+3$, each party $\mathcal{G}_i$ gets the signal $(\mathtt{mcrsfinal}, \mathsf{sid}, \mathcal{G}_i)$ and extracts the crs from cert. The CRS will be output by $\mathcal{G}_i$ only if all the verifications succeeded. The signals $\mathtt{sc}$, $\mathtt{mcrsopen}$, $\mathtt{mcrscertok}$, and $\mathtt{mcrsfinal}$ can be sent either by a controller server or by the internal clock of $\mathcal{G}_i$. The construction uses a secure broadcast channel; thus, if a message is broadcast, then all parties are guaranteed to receive the same message. Note that after $\mathcal{G}_j$ obtains $(\mathtt{rcpt}, \mathsf{lbl}_{ijk})$, for $i \in [1 \mathinner{..} \mathsf{N}_p], j \neq i, k \in [1 \mathinner{..} t]$, she broadcasts $(\mathtt{mcrsreceipt}, \mathsf{lbl}_{ijk})$ since $\mathtt{rcpt}$ is not broadcast.

Security. To prove UC-security of $\mathsf{K}_{\mathsf{mcrs}}$, we restrict $\mathcal{F}_{\mathsf{mcrs}}$ as follows: (i) $\mathbf{C} = (\mathsf{C}_1, \mathsf{C}_2)$ such that $\mathsf{C}_\iota \in \mathcal{C}^{\mathsf{S}}$ for $\iota \in \{1, 2\}$. Note that this means that for any trapdoor element $\mathsf{tc}_k \in \mathsf{tc}$, $[\mathsf{tc}_k]_\star \in \mathsf{crs}$. (ii) $\mathcal{D}$ is the uniform distribution on $(\mathbb{Z}_p^*)^t$, (iii) $\mathsf{comb}(\mathsf{ts}_1, \dots, \mathsf{ts}_{\mathsf{N}_p}) := \mathsf{ts}_1 \circ \dots \circ \mathsf{ts}_{\mathsf{N}_p}$, where $\circ$ denotes point-wise multiplication, and ts_{ik} is $\mathcal{G}_i$'s share of tc_k.

```
**Share collection phase:** *Round 1:* upon receiving $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_i)$, $\mathcal{G}_i$ does the following.
for $k \in [1 \mathinner{..} t]$ do
  1. $\mathsf{ts}_{ik} \leftarrow_{\$} \mathbb{Z}_p^*$;
  2. for $j \neq i$ do
     – Send $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}_{ijk}, \mathcal{G}_i, \mathcal{G}_j, \mathsf{ts}_{ik})$ to $\mathcal{F}_{\mathsf{dlmcom}}$;
     – Upon receiving $(\mathtt{rcpt}, \mathsf{lbl}_{ijk} = (\mathsf{sid}, \mathsf{cid}_{ijk}, \mathcal{G}_i, \mathcal{G}_j))$, $\mathcal{G}_j$ broadcasts
       $(\mathtt{mcrsreceipt}, \mathsf{lbl}_{ijk})$;
     – Store $\mathsf{st}_{ij} \leftarrow (\mathsf{lbl}_{ijk}, \mathsf{ts}_{ik})_{k=1}^{t}$;
If by the end of the round 1, $\mathcal{G}_i$ does not receive $(\mathtt{mcrsreceipt}, \mathsf{sid}, \mathsf{cid}_{jj'k}, \mathcal{G}_j, \mathcal{G}_{j'})$ for
$k \in [1 \mathinner{..} t]$, $j \neq i$, $j' \neq i$, and $j' \neq j$ then $\mathcal{G}_i$ aborts.

*Round 2:* upon receiving $(\mathtt{mcrsopen}, \mathsf{sid})$, $\mathcal{G}_i$ does:
for $k \in [1 \mathinner{..} t]$ do
  1. for $j \neq i$ do
     – Send $(\mathtt{open}, \mathsf{sid}, \mathsf{cid}_{ijk})$ to $\mathcal{F}_{\mathsf{dlmcom}}$;
     – After receiving $(\mathtt{open}, \mathsf{lbl}_{ijk}, [\mathsf{ts}'_{ijk}]_1)$, where $\mathsf{lbl}_{ijk} = (\mathsf{sid}, \mathsf{cid}_{ijk}, \mathcal{G}_i, \mathcal{G}_j)$, from
       $\mathcal{F}_{\mathsf{dlmcom}}$, $\mathcal{G}_j$ stores $(\mathsf{lbl}_{ijk}, [\mathsf{ts}'_{ijk}]_1)$; // If $\mathcal{G}_i$ is honest then $\mathsf{ts}_{ik} = \mathsf{ts}'_{ijk}$
  2. Broadcast $(\mathtt{sbroadc}, \mathcal{G}_i, k, [\mathsf{ts}_{ik}]_1)$.
  3. Upon receiving $(\mathtt{sbroadc}, \mathcal{G}_i, k, [\mathsf{ts}_{ik}]_1)$ broadcast by $\mathcal{G}_i$, $\mathcal{G}_j$ does the following.
     – If $(\mathsf{lbl}_{ijk}, [\mathsf{ts}'_{ijk}]_1)$ is not stored for some $[\mathsf{ts}'_{ijk}]_1$ then abort.
     – Abort unless $[\mathsf{ts}_{ik}]_1 = [\mathsf{ts}'_{ijk}]_1 \neq [0]_1$.
     – If by the end of round 2, $\mathcal{G}_j$ has not received $(\mathtt{sbroadc}, \dots)$, $\forall j \neq i$, $\forall k$, then
       $\mathcal{G}_j$ aborts.

**CRS generation phase:** *Round $r = 3$ to $R + 2$:*
upon receiving $(\mathtt{mcrscertok}, \mathsf{sid}, \mathcal{G}_i, r)$, $\mathcal{G}_i$ does the following, for $i = \mathsf{rndplayer}(r)$.
  1. Extract $\mathsf{C}_{1,d}$, $\mathsf{C}_{2,d}$ corresponding to round $r$ from $\mathsf{C}_1$, $\mathsf{C}_2$;
  2. for $\iota \in \{1, 2\}$ do $\mathsf{cert}_r^{\iota} \leftarrow \mathsf{C}_{\mathsf{layer}}(\mathsf{C}_{\iota,d}, \iota, i, r, \{\mathsf{ts}_{i,k}\}_{k=1}^{t}, \{\mathsf{cert}_j^{\iota}\}_{j=1}^{r-1})$;
  3. $\mathsf{cert}_r \leftarrow (\mathsf{cert}_r^1, \mathsf{cert}_r^2)$; broadcast $(\mathtt{mcrscert}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, r, \mathsf{cert}_r)$;
  4. Any $j \neq i$ does after receiving $(\mathtt{mcrscert}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, r, \mathsf{cert}_r)$ from $\mathcal{G}_i$:
     (a) if $j \neq \mathsf{rndplayer}(r)$, $\mathsf{V}_{\mathsf{layer}}(\mathsf{C}_{1,d}, \iota, i, r, \{[\mathsf{ts}_{i,k}]_{3-\iota}\}_{k=1}^{t}, \{\mathsf{cert}_k^1\}_{k=1}^{r})) = 0$, or
         $\mathsf{V}_{\mathsf{layer}}(\mathsf{C}_{2,d}, \iota, i, r, \{[\mathsf{ts}_{i,k}]_{3-\iota}\}_{k=1}^{t}, \{\mathsf{cert}_j^2\}_{j=1}^{r})) = 0$ then abort;
     (b) Replace stored $(\mathsf{sid}, \mathsf{cid}, r-1, \{\mathsf{cert}_j^{\iota}\}_{j=1}^{r-1})$ with $(\mathsf{sid}, \mathsf{cid}, r, \{\mathsf{cert}_j^{\iota}\}_{j=1}^{r})$;
If by the end of round $r$, for any $i$, $\mathcal{G}_i$ has not stored $(\mathtt{mcrscert}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, r, \mathsf{cert}_r)$ then
$\mathcal{G}_i$ aborts.

*Round $R + 3$:* upon receiving $(\mathtt{mcrsfinal}, \mathsf{sid}, \mathcal{G}_i)$, $\mathcal{G}_i$ does the following.
  1. If $\mathcal{G}_i$ has already received this message then ignore;
  2. Extract $\mathsf{crs}$ from $\{\mathsf{cert}_k^1, \mathsf{cert}_k^2\}_{k=1}^{R}$. Write $(\mathtt{CRS}, \mathsf{crs})$ on the output tape.
```

Fig. 8: The protocol $\mathsf{K}_{\mathsf{mcrs}}$ in the $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model

Theorem 1. $\mathsf{K}_{\mathsf{mcrs}}$ *UC-realizes* $\mathcal{F}_{\mathsf{mcrs}}$ *in the* $\mathcal{F}_{\mathsf{dlmcom}}$*-hybrid model with perfect security against a static adversary. Formally, there exits a PPT simulator* $\mathsf{Sim}^{\mathcal{A}}$ *such that for every static (covert) PPT adversary* $\mathcal{A}$ *and for any non-uniform PPT environment* $\mathcal{Z}$*,* $\mathcal{Z}$ *cannot distinguish* $\mathsf{K}_{\mathsf{mcrs}}$ *composed with* $\mathcal{F}_{\mathsf{dlmcom}}$ *and* $\mathcal{A}$ *from* Sim *composed with* $\mathcal{F}_{\mathsf{mcrs}}$*. That is,* $\mathsf{HYBRID}_{\mathsf{K}_{\mathsf{mcrs}},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{dlmcom}}} = \mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mcrs}},\mathsf{Sim}^{\mathcal{A}},\mathcal{Z}}$.

Proof. Fix any adversary $\mathcal{A}$. To prove the above statement we construct a simulator Sim. Since we only allow static corruption, $\mathcal{A}$ has to corrupt parties before the protocol begins. First, assume that $\mathcal{A}$ corrupts all parties. Then, it is sufficient if Sim (with black-box access to $\mathcal{A}$) honestly simulates $\mathcal{F}_{\mathsf{dlmcom}}$ and forwards messages between $\mathcal{Z}$ and $\mathcal{A}$, and between different corrupted parties $\mathcal{G}_i$. There is no communication with the ideal functionality $\mathcal{F}_{\mathsf{mcrs}}$, hence there is no difference with the real world, and $\mathsf{HYBRID}^{\mathcal{F}_{\mathsf{dlmcom}}}_{\mathsf{K}_{\mathsf{mcrs}},\mathcal{A},\mathcal{Z}} = \mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mcrs}},\mathsf{Sim}^{\mathcal{A}},\mathcal{Z}}$.

In the following we assume that there is a non-empty set of honest parties $\mathfrak{H} = \{\mathcal{G}_{h_i}\} \neq \emptyset$ whose behaviour Sim needs to simulate. As usual, we consider a sequence of hybrid games where we change the rules of games step by step. We denote the changes by using gray background.

$\boxed{\mathsf{Game}_0 = \mathsf{HYBRID}^{\mathcal{F}_{\mathsf{dlmcom}}}_{\mathsf{K}_{\mathsf{mcrs}},\mathcal{A},\mathcal{Z}}\textbf{:}}$ the initial game corresponds to the real world in the $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model, where the real protocol is run among the parties that have access to ideal functionality $\mathcal{F}_{\mathsf{dlmcom}}$. The environment $\mathcal{Z}$ adaptively chooses the input for the honest parties and receives the honest parties' output. The adversary $\mathcal{A}$ attacks the real protocol in the real world; she can interact with the honest parties playing the role of the corrupted parties. For each corrupted party, $\mathcal{A}$ can read her initial inner state and act on her behalf. $\mathcal{Z}$ can control $\mathcal{A}$ and see the interactions between the honest parties and between the honest parties and $\mathcal{A}$ via the view of $\mathcal{A}$.

$\boxed{\mathsf{Game}_1\textbf{:}}$ Game_1 is exactly like game Game_0 except that Sim simulates $\mathcal{F}_{\mathsf{dlmcom}}$ and hence learns all committed shares.

Security analysis. since the behaviour of the protocol does not change, Game_0 is perfectly indistinguishable from Game_1.

$\boxed{\mathsf{Game}_2\textbf{:}}$ We change Game_1 as follows. We fix an arbitrary uncorrupted party $\mathcal{G}_h \in \mathfrak{H}$. During the share collection round, Sim collects all messages $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}_{ihk}, \mathcal{G}_i, \mathcal{G}_h, \mathsf{ts}_{ihk})$ for all $i \neq h$, $k \in [1\mathinner{..}t]$. If after the end of share collection, for some $i \neq h$ and k, there is no ts_{ihk} or $\mathsf{ts}_{ihk} = 0$, then protocol continues as in Game_1; otherwise we do the following.

In the CRS generation phase, Sim sets $\mathsf{tc} = (\mathsf{tc}_1, \dots, \mathsf{tc}_t) \leftarrow_{\$} (\mathbb{Z}_p^*)^t$ and $\mathsf{ts}^*_{hk} \leftarrow \mathsf{tc}_k / (\prod_{i \in [1\mathinner{..}\mathsf{N}_p] \setminus \{h\}} \mathsf{ts}_{ihk})$ for $k \in [1\mathinner{..}t]$. Sim substitutes ts_{hk} in the answer of $\mathcal{F}_{\mathsf{dlmcom}}$ and in the memory of $\mathcal{G}_h$ by ts^*_{hk}. It means that in the opening phase of $\mathcal{F}_{\mathsf{dlmcom}}$, the commitment of $\mathcal{G}_h$ will be opened to $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}_{hik}, \mathcal{G}_h, \mathcal{G}_i, [\mathsf{ts}^*_{hk}]_1)$ and in the remaining protocol interaction $\mathcal{G}_h$ will use ts^*_{hk}.

Security analysis. Since tc distributed uniformly in $(\mathbb{Z}_p^*)^t$, then the same is true for $(\mathsf{ts}^*_{h1}, \dots, \mathsf{ts}^*_{ht})$. Hence, Game_1 is perfectly indistinguishable from Game_2.

We make the following observation about the protocol output. After the opening phase of $\mathcal{F}_{\mathsf{dlmcom}}$ is finished, $\mathsf{K}_{\mathsf{mcrs}}$ is deterministic and all messages are securely broadcast to all parties. Thus, either (i) all honest parties abort since a malicious party did not broadcast the correct output or because some honest party aborted during the opening phase and will not participate in gate evaluations, or (ii) all honest parties write to their output tape $(\mathsf{CRS}, \mathsf{crs})$, where $\mathsf{crs} = ([\mathsf{C}_1(\mathsf{tc})]_1, [\mathsf{C}_2(\mathsf{tc})]_2)$.

$\boxed{\mathsf{Game}_3\textbf{:}}$ Game_3 is like Game_2, but instead of giving $\mathcal{F}_{\mathsf{dlmcom}}$ and $\mathcal{G}_h$ access to ts^*_{hk}, Sim computes them using $\mathsf{crs} \leftarrow ([\mathsf{C}_1(\mathsf{tc})]_1, [\mathsf{C}_2(\mathsf{tc})]_2)$ and trapdoor shares of other parties. First, since crs is computed by $\mathcal{C}^{\mathsf{S}}$-circuits, it contains $[\mathsf{tc}]_1$. Thus,

$\mathcal{F}_{\mathsf{dlmcom}}$ can open the commitment to $[\mathsf{ts}^*_{hk}]_1 \leftarrow [\mathsf{tc}_k]_1/(\prod_{i\in[1\,..\,\mathsf{N}_p]\setminus\{h\}} \mathsf{ts}_{ihk})$. Second, we change the computation of $\mathsf{cert}_{g,h}$ (See Fig. 6) each $\mathsf{multdiv}$ gate g for $\mathcal{G}_h$ as follows.

Let us denote $\boldsymbol{x}_{h-1} = (\mathsf{ts}_{1h1}, \ldots, \mathsf{ts}_{1ht}, \ldots, \mathsf{ts}_{h-1h1}, \ldots, \mathsf{ts}_{h-1ht})$, that is, $\boldsymbol{x}_{h-1}$ contains shares of all the parties $\mathcal{G}_1, \ldots, \mathcal{G}_{h-1}$. We observe that each element $[b_j]_\iota \in \mathsf{cert}_{g,h}$ can be written in the form $[b_j]_\iota = [b]_\iota \cdot \frac{M_1(\boldsymbol{x}_{h-1})\cdot\mathsf{ts}^*_{h,\alpha_1}\cdot\ldots\cdot\mathsf{ts}^*_{h,\alpha_q}}{M_2(\boldsymbol{x}_{h-1})\cdot\mathsf{ts}^*_{h\beta_1}\cdot\ldots\cdot\mathsf{ts}^*_{h\beta_r}}$ where M_1 and M_2 are some monomials and $[b]_\iota$ is either in crs or is a constant. However, according to the structure of $\mathcal{C}^{\mathsf{S}}$ circuit, then the crs will also contain an element $[\sigma]_\iota = [b]_\iota \cdot \frac{\mathsf{tc}_{\alpha_1}\cdot\ldots\cdot\mathsf{tc}_{\alpha_q}}{\mathsf{tc}_{\beta_1}\cdot\ldots\cdot\mathsf{tc}_{\beta_r}}$. Given that we know shares of all the other parties, we can easily compute $[\sigma_h]_\iota = [b]_\iota \cdot \frac{\mathsf{ts}^*_{h\alpha_1}\cdot\ldots\cdot\mathsf{ts}^*_{h\alpha_q}}{\mathsf{ts}^*_{h\beta_1}\cdot\ldots\cdot\mathsf{ts}^*_{h\beta_r}} = [\sigma]_\iota \cdot \frac{\prod_{i\in[1\,..\,\mathsf{N}_p]\setminus\{h\}} \mathsf{ts}_{ih\beta_1}\cdot\ldots\cdot\mathsf{ts}_{ih\beta_r}}{\prod_{i\in[1\,..\,\mathsf{N}_p]\setminus\{h\}} \mathsf{ts}_{ih\alpha_1}\cdot\ldots\cdot\mathsf{ts}_{ih\alpha_q}}$ and from there we can compute $[b_j]_\iota = [\sigma_h]_\iota \cdot \frac{M_1(\boldsymbol{x}_{h-1})}{M_2(\boldsymbol{x}_{h-1})}$. Hence we are able to simulate $\mathsf{cert}_{g,h}$.

Security analysis. Although in Game_3, the computation is different for the opening and $\mathsf{multdiv}$ gate evaluations, the result of each computation is the same as in Game_2. Thus, Game_2 and Game_3 are perfectly indistinguishable for $\mathcal{Z}$.

$\boxed{\mathsf{Game}_4 = \mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mcrs}},\mathsf{Sim}^{\mathcal{A}},\mathcal{Z}}\text{:}}$ We construct an ideal-world adversary Sim that runs a black-box simulation of the real-world adversary $\mathcal{A}$ by simulating the protocol execution and relaying messages between $\mathcal{A}$ and $\mathcal{Z}$. Sim acts as an interface between $\mathcal{A}$ and $\mathcal{Z}$ by imitating a copy of a real execution of π for $\mathcal{A}$, incorporating $\mathcal{Z}$'s ideal-model interactions and vice versa forwarding $\mathcal{A}$'s messages to $\mathcal{Z}$. Additionally, since we are in $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model, then Sim can simulate the ideal functionality $\mathcal{F}_{\mathsf{dlmcom}}$ for adversaries.

Sim interacts with $\mathcal{Z}$, the functionality $\mathcal{F}_{\mathsf{mcrs}}$, and an internal copy of $\mathcal{A}$. An honest party $\mathcal{G}_h$ is picked as before.

Simulating $\mathcal{F}_{\mathsf{dlmcom}}$ for corrupted committer $\mathcal{G}_i$: perfectly emulate $\mathcal{F}_{\mathsf{dlmcom}}$ but store the data. More precisely:

Upon receiving $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ from a corrupted $\mathcal{G}_i$, do the following. If a tuple $(\mathsf{sid}, \mathsf{cid}, \ldots)$ with the same $(\mathsf{sid}, \mathsf{cid})$ was previously recorded, do nothing. Otherwise, Sim records $(\mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ and sends $(\mathtt{rcpt}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j)$ to $\mathcal{G}_j$.

Upon receiving $(\mathsf{open}, \mathsf{sid}, \mathsf{cid})$ from a corrupted $\mathcal{G}_i$, proceed as follows: if a tuple $(\mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, \mathcal{G}_j, m)$ was previously recorded then send $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}, , \mathcal{G}_i, \mathcal{G}_j, [m]_1)$ to $\mathcal{G}_j$. Otherwise do nothing.

Simulating the share collection phase for uncorrupted $\mathcal{G}_i$: upon receiving $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_i)$ from $\mathcal{F}_{\mathsf{mcrs}}$ for some $\mathcal{G}_i \in \mathfrak{H}$, Sim does the following.

For $k \in [1\,..\,t]$: (1) Sim picks $\mathsf{ts}^*_{ik} \leftarrow_{\$} \mathbb{Z}_p^*$. (2) For $j \neq i$, Sim sends $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}_{ijk}, \mathcal{G}_i, \mathcal{G}_j, \mathsf{ts}^*_{ik})$ to $\mathcal{F}_{\mathsf{dlmcom}}$.

From this point onward, Sim simulates behaviour of all uncorrupted parties *but* $\mathcal{G}_h$ exactly as in the real protocol but with ts^*_{ik} replacing ts_{ik}.

Simulating the share collection phase for corrupted $\mathcal{G}_i$ and uncorrupted $\mathcal{G}_j$: Upon receiving $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}_{ihk}, \mathcal{G}_i, \mathcal{G}_h, \mathsf{ts}_{ihk})$ for $k \in [1 \,..\, t]$ from $\mathcal{G}_i$, set $\mathsf{ts}_i \leftarrow (\mathsf{ts}_{ih1}, \dots, \mathsf{ts}_{iht})$ and send $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_i, \mathsf{ts}_i)$ to $\mathcal{F}_{\mathsf{mcrs}}$.

Simulating the CRS generation phase for uncorrupted committer $\mathcal{G}_i$: upon obtaining $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs})$ from $\mathcal{F}_{\mathsf{mcrs}}$ just after the share collection phase ends, Sim does the following.

At this moment Sim knows the shares ts_{ihk} of all corrupted parties and the simulated shares ts^*_{ik} of all uncorrupted parties. Since crs is computed by $\mathcal{C}^{\mathsf{S}}$-circuits, it contains $[\mathsf{tc}]_1$. In the case $i = h$, Sim sets

$$[\mathsf{ts}^*_{hk}]_1 \leftarrow [\mathsf{tc}_k]_1 / (\textstyle\prod_{j \in [1 \,..\, \mathsf{N}_p] \setminus \{h\}} \mathsf{ts}'_{jk}) \quad , \tag{1}$$

where $[\mathsf{tc}_k]_1 \in [\mathsf{tc}]_1$, $\mathsf{ts}'_{jk} = \mathsf{ts}^*_{jk}$ for $\mathcal{G}_j \in \mathfrak{H}$ and otherwise $\mathsf{ts}'_{jk} = \mathsf{ts}_{jhk}$. Since tc_k is uniformly random in $\mathbb{Z}_p^*$ (tc_k is the product of shares where at least one share is uniformly random), therefore ts^*_{hk} is uniformly random in $\mathbb{Z}_p^*$.

In the rest of the CRS generation phase, Sim simulates $\mathcal{G}_i$, for honest $\mathcal{G}_i$, as in the protocol but using $[\mathsf{ts}^*_{ik}]_1$ instead of $[\mathsf{ts}_{ik}]_1$.

In particular, for any $j \neq h$: if $\mathcal{F}_{\mathsf{dlmcom}}$ sends $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_h, \mathcal{G}_j, [\mathsf{ts}_{hk}]_1)$ to $\mathcal{G}_j$ then Sim sends instead $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}, \hat{\mathcal{G}}_h, \mathcal{G}_j, [\mathsf{ts}^*_{hk}]_1)$ to $\mathcal{G}_j$. Sim also broadcasts the new value $(\mathtt{sbroadc}, \mathcal{G}_h, k, [\mathsf{ts}^*_{hk}]_1)$ for $k \in [1 \,..\, t]$.

Finally, as a verifier, an uncorrupted $\mathcal{G}_j$ executes the same verifications as $\mathcal{G}_j$ in $\mathsf{K}_{\mathsf{mcrs}}$. In particular, $\mathtt{sbroadc}$ message guarantees that malicious parties have to open to the same value that they broadcast (unless $\mathcal{A}$ is satisfied with parties aborting).

Upon receiving $(\mathtt{mcrscertok}, \mathsf{sid}, \mathcal{G}_i, r)$, Sim does the following, assuming $\mathcal{G}_i$ has not aborted before. The computation result of uncorrupted $\mathcal{G}_i$ in $\mathsf{K}_{\mathsf{mcrs}}$ is deterministic, and can be verified by using commitment openings.

Sim simulates each uncorrupted party $\mathcal{G}_i$, $i \neq h$, as in $\mathsf{K}_{\mathsf{mcrs}}$. What is left to show is that Sim can simulate the output transcript $\mathsf{cert}_{g,h}$ for each $\mathsf{multdiv}$ gate g that $\mathcal{G}_h$ needs to compute. Sim does this as was explained above.

After the protocol ends, parties are provided with an output crs'. Recall that since Sim sets the shares of $\mathcal{G}_h$ as in Eq. (1), trapdoor tc' corresponding to crs' is equal to trapdoor tc. Since CRS crs is computed by a $\mathcal{C}^{\mathsf{S}}$-circuit, that is crs is determined by tc and all outputs of $\mathsf{multdiv}$ gates of $\mathsf{C}_1, \mathsf{C}_2$ are parts of it, it holds that $\mathsf{crs}' = \mathsf{crs}$.

Simulating the CRS generation phase for corrupted $\mathcal{G}_i$ and uncorrupted receiver $\mathcal{G}_j$:

Upon receiving $(\mathsf{open}, \mathsf{sid}, \mathsf{cid}_{ijk}, \mathcal{G}_i, \mathcal{G}_j, [\mathsf{ts}_{ijk}]_1)$ from $\mathcal{G}_i$, Sim executes the checks of the verifier exactly as in $\mathsf{K}_{\mathsf{mcrs}}$. If the checks fail, Sim aborts. Otherwise, Sim sends $(\mathtt{mcrsopen}, \mathsf{sid}, \mathsf{cid})$ to $\mathcal{F}_{\mathsf{mcrs}}$.

Upon receiving $(\mathtt{mcrscert}, \mathsf{sid}, \mathsf{cid}, \mathcal{G}_i, r, \mathsf{cert}_r)$ from $\mathcal{G}_i$, Sim performs the checks of $\mathcal{G}_j$ as in $\mathsf{K}_{\mathsf{mcrs}}$. If $\mathcal{G}_j$ aborts then output whatever $\mathcal{G}_i$ outputs and then abort.

Finalization: When simulated protocol finished and no honest party aborted, then Sim sends $(\mathtt{CRS}, \mathsf{ok})$ to $\mathcal{F}_{\mathsf{mcrs}}$.

There are two differences between Game_3 and Game_4. Firstly, in Game_4 simulator does not pick crs itself, but instead it is given by $\mathcal{F}_{\mathsf{mcrs}}$. However, as at least one party is honest, then crs has exactly the same distribution.

Secondly, there are honest dummy parties that on the successful execution of the protocol write $(\mathtt{CRS}, \mathsf{crs})$ on their output tapes. We argued that same happens in Game_2 (and therefore also in the equivalent Game_3) for honest parties. We conclude that $\mathsf{HYBRID}^{\mathcal{F}_{\mathsf{dlmcom}}}_{\mathsf{K}_{\mathsf{mcrs}}, \mathcal{A}, \mathcal{Z}} = \mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mcrs}}, \mathsf{Sim}^{\mathcal{A}}, \mathcal{Z}}$. $\square$

Achieving adaptive security seems to need a non-trivial change in $\mathsf{K}_{\mathsf{mcrs}}$ or at least a very different proof strategy. In particular, the following attack works for the current simulation strategy. Suppose $\mathcal{G}_i$ is honest until he has broadcast $(\mathtt{sbroadc}, \mathcal{G}_i, k, [\mathsf{ts}_{ik}]_1)$ in Fig. 8. If the adversary now corrupts $\mathcal{G}_i$, then Sim has to provide adversary with all the internal knowledge of $\mathcal{G}_i$, including ts_{ik}. Hence, Sim cannot set $[\mathsf{ts}_{ik}]_1 \leftarrow [\mathsf{tc}_k]_1 / (\prod_{j \in [1 \mathinner{..} \mathsf{N}_p] \setminus \{i\}} \mathsf{ts}'_{jk})$, where $[\mathsf{tc}_k]_1$ is the output of the ideal functionality $\mathcal{F}_{\mathsf{mcrs}}$, since this would require computing discrete logarithm of $[\mathsf{tc}_k]_1$.

5 Secure MPC for NIZKs

Next, we show that $\mathsf{K}_{\mathsf{mcrs}}$ can be used to generate the CRS of any $\mathcal{C}^{\mathsf{S}}$-SNARK without harming the completeness, soundness, or (subversion) zero-knowledge properties. It could also be used to generate CRS of other primitives which can be represented by $\mathcal{C}^{\mathsf{S}}$-circuits, but it is especially well suited for the intricate structure of SNARK CRS. Finally, we apply the protocol to the Sub-ZK secure version [2,14] of the most efficient zk-SNARK by Groth [20].

NIZK in the MCRS model. Let Ψ be a NIZK argument system secure in the $\mathcal{F}_{\mathsf{crs}}$-hybrid model. We show that by instantiating $\mathcal{F}_{\mathsf{crs}}$ with $\mathcal{F}_{\mathsf{mcrs}}$, the NIZK remains complete, sound, and zero-knowledge, provided that the adversary $\mathcal{A}$ controls up to $\mathsf{N}_p - 1$ out of N_p parties. Here we require that $\mathcal{D}$ is comb-friendly. See Fig. 9 for the high-level description of MPC protocol for the CRS generation.

Theorem 2. *Let $\mathcal{D}$ and* $\mathsf{comb} : (\mathrm{Supp}(\mathcal{D}))^{\mathsf{N}_p} \to \mathrm{Supp}(\mathcal{D})$ *be such that $\mathcal{D}$ is* comb*-friendly.* $\mathsf{K}^{\mathcal{F}_{\mathsf{mcrs}}}_{\mathsf{crs}}$ *securely realizes* $\mathcal{F}^{\mathcal{D}, f}_{\mathsf{crs}}$ *in the* $\mathcal{F}_{\mathsf{mcrs}}$*-hybrid model given (covert) $\mathcal{A}$ corrupts up to $\mathsf{N}_p - 1$ out of N_p parties (i.e. CRS generators).*

Proof. As usual, we consider a sequence of hybrid games.

$\boxed{\mathsf{Game}_0 = \mathsf{HYBRID}^{\mathcal{F}_{\mathsf{mcrs}}}\textbf{:}}$ This game corresponds to the real world in the $\mathcal{F}_{\mathsf{mcrs}}$-hybrid model. (The rest of the description of this game is the same as in the proof of Theorem 1.)

$\boxed{\mathsf{Game}_1\textbf{:}}$ We construct an ideal-world adversary Sim that runs a black-box simulation of the real-world adversary $\mathcal{A}$ by simulating the protocol execution and relaying messages between $\mathcal{A}$ and the environment $\mathcal{Z}$. Sim acts as an interface between $\mathcal{A}$ and $\mathcal{Z}$ by imitating the real execution of $\mathsf{K}^{\mathcal{F}_{\mathsf{mcrs}}}_{\mathsf{crs}}$ for $\mathcal{A}$, incorporating $\mathcal{Z}$'s ideal-model interactions and vice versa forwarding $\mathcal{A}$'s messages to $\mathcal{Z}$.

$\mathsf{K}_{\mathsf{crs}}^{\mathcal{F}_{\mathsf{mcrs}}}$ proceeds as follows, running with a set $\{P_1, \ldots, P_{\mathsf{N}'_p}\}$ of parties, designated set $\{\mathcal{G}_1, \ldots, \mathcal{G}_{\mathsf{N}_p}\}$ of CRS generators, and an adversary Sim.
CRS generation: Send a signal to each $\mathcal{G}_i$ to execute the functionality $\mathcal{F}_{\mathsf{mcrs}}$. If $\mathcal{F}_{\mathsf{mcrs}}$ returns crs then $\mathcal{G}_i$ stores $(\mathsf{sid}, \mathsf{crs})$.
Retrieval: P_i sends $(\mathtt{retrieve}, \mathsf{sid})$ to each $\mathcal{G}_j$: If $(\mathsf{sid}, \mathsf{crs})$ is recorded for some crs then $\mathcal{G}_j$ sends $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs})$. If all N_p responses from $\mathcal{G}_j$ are the same, then P_i outputs $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs})$. Otherwise P_i aborts.

Fig. 9: Protocol $\mathsf{K}_{\mathsf{crs}}^{\mathcal{F}_{\mathsf{mcrs}}}$

Additionally, since we assumed $\mathcal{F}_{\mathsf{mcrs}}$-hybrid model, Sim can simulate the ideal functionality $\mathcal{F}_{\mathsf{mcrs}}$ for adversaries. The protocol handles adaptive corruption.

In the following, we describe Sim. We assume that there is a non-empty set of honest parties $\mathfrak{H} = \{\mathcal{G}_{h_i}\} \neq \emptyset$ whose behaviour Sim needs to simulate.

Simulating $\mathcal{F}_{\mathsf{mcrs}}$: For all parties Sim precisely imitates honest $\mathcal{F}_{\mathsf{mcrs}}$, unless we say otherwise. This functionality is available at all times, exactly as in the real protocol.

Simulating CRS generation: On $(\mathtt{crsOK?}, \mathsf{sid}, \mathsf{crs}')$ from $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D},f}$ send $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_i)$ to all $\mathcal{G}_i \in \mathfrak{H}$. Wait for $(\mathtt{sc}, \mathsf{sid}, \mathcal{G}_j, \mathsf{ts}_j)$ from all corrupted $\mathcal{G}_j$. If no such messages arrive, abort. Otherwise let $\mathcal{F}_{\mathsf{mcrs}}$ send $(\mathtt{CRS}, \mathsf{crs}')$ to all $\mathcal{G}_i$. Send $(\mathtt{crsOK}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{crs}}$.

Simulating CRS retrieval: On message $(\mathtt{retrieve}, \mathsf{sid})$ from P_i, send $(\mathtt{retrieve}, \mathsf{sid})$ to each $\mathcal{G}_j$. Abort if all $\mathcal{G}_j$ do not send the same message $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs}')$. Send $(\mathtt{CRS}, \mathsf{sid}, \mathsf{crs}')$ to P_j.

Let $\mathcal{G}_h$ be an honest party, whose trapdoor share is ts_h. Assume $\mathcal{Z}$ is able to distinguish Game_0 and Game_1 with an advantage ε. Since the simulator simulates the real-world protocol perfectly, the only advantage $\mathcal{Z}$ may gain comes from different distributions of CRSs in the games.

Denote by crs a CRS produced in Game_0. The corresponding trapdoor tc equals to $\mathsf{comb}(\mathsf{ts}_1, \ldots, \mathsf{ts}_{\mathsf{N}_p})$, where ts_h is unknown to $\mathcal{Z}$ and has distribution $\mathcal{D}$. In Game_1, the output CRS, crs', comes from functionality $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D},f}$ and the corresponding trapdoor tc' is distributed accordingly to $\mathcal{D}$. Since CRS is a deterministic function of trapdoor and the trapdoors tc, tc' share the same distribution, so do crs and crs'. Thus, the advantage ε of any $\mathcal{Z}$ in distinguishing which CRS was produced in which game is always 0. Moreover, since we consider only covert adversaries, then we can ignore the situations where parties would abort. Hence, Game_0 and Game_1 are indistinguishable. $\square$

Next corollary immediately follows from the universal composition theorem [10].

Corollary 1. *Let Ψ be a NIZK argument that is complete, sound, computationally ZK, and computationally Sub-ZK in the $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D},f}$-hybrid model. By instantiating $\mathcal{F}_{\mathsf{crs}}^{\mathcal{D},f}$ with $\mathsf{K}_{\mathsf{crs}}^{\mathcal{F}_{\mathsf{mcrs}}}$, the following holds:*

CRS / trapdoor: $\mathsf{tc} \leftarrow (\alpha, \beta, \gamma, \delta, \chi)$ and $\mathsf{crs} = (\mathsf{crs_P}, \mathsf{crs_V}, \mathsf{crs_{CV}})$, where

$$\mathsf{crs_P} \leftarrow \left(\begin{array}{l} \left[\alpha, \beta, \delta, \left((u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta\right)_{j=m_0+1}^{\mathsf{m}}\right]_1, \\ \left[\left(\chi^i \ell(\chi)/\delta\right)_{i=0}^{\mathsf{n}-2}, (u_j(\chi), v_j(\chi))_{j=0}^{\mathsf{m}}\right]_1, \left[\beta, \delta, (v_j(\chi))_{j=0}^{\mathsf{m}}\right]_2 \end{array} \right),$$

$$\mathsf{crs_V} \leftarrow \left(\left[\left((u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\gamma\right)_{j=0}^{m_0}\right]_1, [\gamma, \delta]_2\right),$$

$$\mathsf{crs_{CV}} \leftarrow ([\gamma, (\chi^i)_{i=1}^{\mathsf{n}-1}, (\ell_i(\chi))_{i=1}^{\mathsf{n}}]_1, [\alpha, \chi, \chi^{\mathsf{n}-1}]_2).$$

Fig. 10: CRS of Z^* Sub-ZK SNARK from [2]

1. *Ψ is complete, sound, and computationally zero-knowledge in the $\mathcal{F}_{\mathsf{mcrs}}$-hybrid model, given that (covert) $\mathcal{A}$ corrupts up to $\mathsf{N}_p - 1$ out of N_p parties.*
2. *Ψ is Sub-ZK in the $\mathcal{F}_{\mathsf{mcrs}}$-hybrid model, even if (covert) $\mathcal{A}$ corrupts all N_p parties.*
3. *If $\mathcal{D}$ is a uniform distribution over $(\mathbb{Z}_p^*)^t$, comb the point-wise multiplication and the CRS can be computed by $\mathcal{C}^{\mathsf{S}}$-circuits, then properties 1 and 2 hold in the $\mathcal{F}_{\mathsf{dlmcom}}$-hybrid model since $\mathsf{K}_{\mathsf{mcrs}}$ realizes $\mathcal{F}_{\mathsf{mcrs}}$ in that setting.*

Applying $\mathsf{K}_{\mathsf{mcrs}}$ to Groth's zk-SNARK. Fig. 10 contains the description of the CRS for the Sub-ZK version of Groth's zk-SNARK Z^* as was proposed in [2]. We have omitted the element $[\alpha\beta]_T$ that can be computed from $[\alpha]_1$ and $[\beta]_2$. The CRS from [2] differs from the original CRS for Groth's zk-SNARK [20] by the entries in $\mathsf{crs_{CV}}$ which make the CRS verifiable using a CV algorithm. Here, $\ell_i(X)$ are Lagrange basis polynomials and $\ell(X) = X^{\mathsf{n}} - 1$, $u_j(X)$, $v_j(X)$, $w_j(X)$ are publicly-known circuit-dependent polynomials.

We recall that Quadratic Arithmetic Program (QAP, [15]) is an **NP**-complete language with an efficient reduction from CIRCUIT-SAT. A QAP instance can be expressed as $\mathcal{Q}_p = (\mathbb{Z}_p, m_0, \ell(X), \{u_j(X), v_j(X), w_j(X)\}_{j=0}^{m})$.

The goal of the prover of a SNARK for QAP [15,20,2] is to prove that for public $(A_1, \ldots, A_{m_0})$ and $A_0 = 1$, he knows $(A_{m_0+1}, \ldots, A_m)$ and a degree $\leq n-2$ polynomial $h(X)$, such that $h(X) = (a(X)b(X) - c(X))/\ell(X)$, where $a(X) = \sum_{j=0}^{m} A_j u_j(X)$, $b(X) = \sum_{j=0}^{m} A_j v_j(X)$, $c(X) = \sum_{j=0}^{m} A_j w_j(X)$, and $\ell(X) = \prod_{i=1}^{\mathsf{n}}(X - \omega^{i-1})$, where ω is an n-th primitive root of unity modulo p, is a polynomial related to Lagrange interpolation.

Figure 11 describes the prover and verifier algorithm of Z^*. Intuitively the CV algorithm of Z^* checks that certain pairing equations hold for the CRS elements which guarantees that CRS is well-formed, that is, CRS could be computed from some valid $\mathsf{tc} \leftarrow (\alpha, \beta, \gamma, \delta, \chi)$. For the full description CV algorithm we refer to [2].

We recall that to use the algorithm $\mathsf{K}_{\mathsf{crs}}^{\mathcal{F}_{\mathsf{mcrs}}}$ the CRS has to be of the form $\mathsf{crs} = ([\mathsf{C}_1(\mathsf{tc})]_1, [\mathsf{C}_2(\mathsf{tc})]_2)$, where $\mathsf{C}_\iota \in \mathcal{C}^{\mathsf{S}}$. In Fig. 10, the highlighted entries cannot be computed from trapdoors by a $\mathcal{C}^{\mathsf{S}}$-circuit unless we add $\mathsf{crs_{TV}} = ([(w_j(\chi), \beta u_j(\chi), \alpha v_j(\chi))_{j=0}^{\mathsf{m}}, \chi^{\mathsf{n}}]_1, [(\ell_i(\chi))_{i=1}^{\mathsf{n}}, (\chi^k)_{k=1}^{\mathsf{n}-1}]_2)$ to the CRS. To obtain better efficiency we additionally add $[(\ell_i(\chi))_{i=1}^{\mathsf{n}}]_2$ to the CRS, although

```
$\mathsf{P}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{x} = (A_1, \dots, A_{m_0}), \mathsf{w} = (A_{m_0+1}, \dots, A_m))$
    1. Let $a^\dagger(\chi) \leftarrow \sum_{j=0}^m A_j u_j(\chi)$, $b^\dagger(\chi) \leftarrow \sum_{j=0}^m A_j v_j(\chi)$,
    2. Let $c^\dagger(\chi) \leftarrow \sum_{j=0}^m A_j w_j(\chi)$,
    3. Set $h(\chi) = \sum_{i=0}^{n-2} h_i \chi^i \leftarrow (a^\dagger(\chi) b^\dagger(\chi) - c^\dagger(\chi))/\ell(\chi)$,
    4. Set $[h(\chi)\ell(\chi)/\delta]_1 \leftarrow \sum_{i=0}^{n-2} h_i \left[\chi^i \ell(\chi)/\delta\right]_1$,
    5. Set $r_a \leftarrow_r \mathbb{Z}_p$; Set $\mathfrak{a} \leftarrow \sum_{j=0}^m A_j [u_j(\chi)]_1 + [\alpha]_1 + r_a [\delta]_1$,
    6. Set $r_b \leftarrow_r \mathbb{Z}_p$; Set $\mathfrak{b} \leftarrow \sum_{j=0}^m A_j [v_j(\chi)]_2 + [\beta]_2 + r_b [\delta]_2$,
    7. Set $\mathfrak{c} \leftarrow r_b \mathfrak{a} + r_a \left(\sum_{j=0}^m A_j [v_j(\chi)]_1 + [\beta]_1\right) +$
           $\sum_{j=m_0+1}^m A_j [(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1 + [h(\chi)\ell(\chi)/\delta]_1$,
    8. Return $\pi \leftarrow (\mathfrak{a}, \mathfrak{b}, \mathfrak{c})$.
$\mathsf{V}(\mathbf{R}, \xi_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{x} = (A_1, \dots, A_{m_0}), \pi = (\mathfrak{a}, \mathfrak{b}, \mathfrak{c}))$: assuming $A_0 = 1$, check that $\mathfrak{a} \bullet \mathfrak{b} =$
    $[\alpha\beta]_T + \mathfrak{c} \bullet [\delta]_2 + \left(\sum_{j=0}^{m_0} A_j [(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\gamma]_1\right) \bullet [\gamma]_2$.
```

Fig. 11: Prover and Verifier of Z^* Sub-ZK SNARK from [2]

they can be computed from the existing elements $[(\chi^k)_{k=1}^{\mathsf{n}-1}]_2$. However, since we are adding elements to the CRS, we also need to reprove the soundness. Since Groth's zk-SNARK for QAP was proven secure in the Generic Bilinear Group Model (GBGM, [20]), and this GBGM proof depends on the precise CRS, we have to prove the knowledge-soundness of the argument again. Abdolmaleki *et al.* proved in [2] that Z^* is sound in the Sub-GBGM model; in the latter the adversary is additionally allowed to create group elements without knowing their discrete logarithms. For the description of Sub-GBGM we refer the reader to [2].

Theorem 3. *Z^*, with $\mathsf{crs} = (\mathsf{crs}_{\mathsf{P}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{crs}_{\mathsf{CV}}, \mathsf{crs}_{\mathsf{TV}})$, has perfect completeness and perfect Sub-ZK. It has statistical knowledge soundness in sub-GBGM.*

Proof. Perfect completeness and zero-knowledge can be proved exactly as in [20]. Thus, we only need to reprove knowledge-soundness. For this, we use standard (Sub-)GBGM proof techniques.

Since Z^* is known to be knowledge-sound, it is sufficient to show that the elements in $\mathsf{crs}_{\mathsf{TV}}$ do not give the soundness adversary $\mathcal{A}$ any additional advantage. First, note that $u_j(X)$, $v_j(X)$, and $w_j(X)$ for $j \in [0 \,..\, \mathsf{m}]$ are in the span of $\{X^i\}_{i=0}^{\mathsf{n}-1}$. Hence, $[u_j(\chi)]_1$, $[v_j(\chi)]_1$, $[w_j(\chi)]_1$, and $[\ell_i(X)]_1$ can be efficiently computed from $(\left[\chi^i\right]_1)_{i=0}^{\mathsf{n}-1}$ and we can ignore them in the following analysis.

Due to the structure of the CRS, the use of Sub-GBGM, and the Schwartz-Zippel lemma, the discrete logarithm of any element $[T]_\iota$ for $\iota \in \{1, 2\}$ that the adversary outputs can be represented as the following polynomial

$$\begin{aligned} T(\boldsymbol{X}, \boldsymbol{Y}) = & T_1(X_\chi) + X_\alpha T_2(X_\chi) + X_\beta T_3(X_\chi) + T_\gamma X_\gamma + \\ & T_\delta X_\delta + \frac{\sum_{j=0}^{m_0} T_{\gamma:j} D_j}{X_\gamma} + \frac{\sum_{j=m_0+1}^{\mathsf{m}} T_{\delta:j} D_j}{X_\delta} + \frac{T_4(X_\chi)\ell(X_\chi)}{X_\delta} + \sum_{i=1}^q T_{y:i} Y_i \ , \end{aligned}$$

where each indeterminant X_s corresponds to the randomly picked secret $s \in \mathsf{tc}$. Here $T_1(X_\chi)$ is polynomial in the span of $\{X_\chi^i\}_{i=0}^{\mathsf{n}}$. $T_2(X_\chi)$ and $T_3(X_\chi)$ are polynomials in the span of $\{X_\chi^i\}_{i=0}^{\mathsf{n}-1}$. Elements D_j are equal to $u_j(X_\chi)X_\beta +$

$v_j(X_\chi)X_\alpha + w_j(X_\chi)$. Elements Y_i are considered for the case that adversary may query the GBGM oracle to give him random group elements.

Let $([A]_1, [B]_2, [C]_1)$ be the proof output by the adversary. Substituting the symbolic value T in the previously described polynomial, gives for the proof the following polynomial representation $A = A(\boldsymbol{X}, \boldsymbol{Y})$, $B = B(\boldsymbol{X}, \boldsymbol{Y})$, and $C = C(\boldsymbol{X}, \boldsymbol{Y})$. The verification equation is equivalent to checking that $V(\boldsymbol{X}, \boldsymbol{Y}) = AB - X_\alpha X_\beta - \sum_{j=0}^{m_0} A_j D_j - X_\delta C$ is a zero polynomial.

Let us first analyse the coefficients of $V(\boldsymbol{X}, \boldsymbol{Y})$:

- Coefficient of X_α^2 is $A_2(X_\chi)B_2(X_\chi) = 0$. Without loss of generality, let us assume that $B_2(X_\chi) = 0$.
- Coefficient of $X_\alpha X_\beta$ is $A_2(X_\chi)B_3(X_\chi) + B_2(X_\chi)A_3(X_\chi) = A_2(X_\chi)B_3(X_\chi) = 1$. Therefore both polynomials $A_2(X_\chi)$ and $B_3(X_\chi)$ are constants and for simplicity we may denote them by A_2 and B_3.
- Coefficient of X_β^2 is $B_3 A_3(X_\chi) = 0$, hence $A_3(X_\chi) = 0$.
- Coefficients of $X_\alpha X_\delta$ and $X_\beta X_\delta$ are respectively $A_2 B_\delta + C_2(X_\chi) = 0$ and $B_3 A_\delta + C_3(X_\chi) = 0$, hence polynomials $C_2(X_\chi)$ and $C_3(X_\chi)$ are constants.
- Coefficient next to Y_i^2 is $A_{y:i}B_{y:i}$. Thus $A_{y:i}$ or $B_{y:i}$ equals zero. Without loss of generality, assume $B_{y:i} = 0$.
- Coefficient of $Y_i X_\beta$ is $A_{y:i}B_3(X_\chi)$. Thus, $A_{y:i} = 0$.
- Coefficient of $Y_i X_\delta$ is $C_{y:i}$. Thus, $C_{y:i} = 0$.

With this we have shown that for A, B, C, the polynomials $T_2(X_\chi)$, $T_3(X_\chi)$ are constant. Hence, adversary could efficiently compute the proof $([A]_1, [B]_2, [C]_1)$ without the elements in $\mathsf{crs}_{\mathsf{TV}}$. Rest of the proof is exactly as in [2]. □

We give a brief description of the CRS-generation protocol for Z^* without explicitly describing the circuits C_1 and C_2. Without directly saying it, it is assumed that parties verify all the computations as shown in Fig. 8.

Share collection phase. Parties proceed as is in Fig. 8 to produce random and independent shares $[\mathsf{ts}_i]_\star = [\alpha_i, \beta_i, \gamma_i, \delta_i, \chi_i]_\star$ for each $\mathcal{G}_i$.

CRS generation phase. (i) On layers $\mathsf{C}_{1,1}, \mathsf{C}_{2,1}$ parties jointly compute $[\alpha, \beta, \gamma, \delta]_\star$, $[(\chi^k)_{k=1}^{\mathsf{n}-1}]_\star$ and $[\chi^{\mathsf{n}}]_1$. (ii) Each $\mathcal{G}_i$ locally computes $[(\ell_k(\chi))_{k=1}^{\mathsf{n}}]_\star$, $[(w_j(\chi), u_j(\chi))_{j=0}^{\mathsf{m}}]_1$, and $[(v_j(\chi))_{j=0}^{\mathsf{m}}]_\star$ using $[(\chi^k)_{k=1}^{\mathsf{n}-1}]_\star$; and also computes $[\ell(\chi)]_1 = [\chi^{\mathsf{n}}]_1 - [1]_1$. (iii) On layer $\mathsf{C}_{1,2}$, from input $[\ell(\chi)]_1$, parties jointly compute $[(\chi^k \ell(\chi)/\delta)_{k=0}^{\mathsf{n}-2}]_1$ using $\mathsf{n} - 1$ $\mathsf{multdiv}$ gates. Moreover, they compute $[(\beta u_l(\chi), \alpha v_l(\chi))_{l=0}^{\mathsf{m}}]_1$. (iv) Each party computes locally $[(\beta u_l(\chi) + \alpha v_l(\chi) + w_l(\chi))_{l=0}^{\mathsf{m}}]_1$. (v) On layer $\mathsf{C}_{1,3}$ parties compute jointly $[(\beta u_l(\chi) + \alpha v_l(\chi) + w_l(\chi)/\gamma)_{l=0}^{m_0}]_1$ and $[(\beta u_l(\chi) + \alpha v_l(\chi) + w_l(\chi)/\delta)_{l=m_0+1}^{\mathsf{m}}]_1$.

The cost of the CRS generation for Z^* can be summarised as follows: the circuits C_1 and C_2 have both sampling depth 3; the multi-party protocol for computing the crs takes $3\mathsf{N}_p + 6$ rounds and requires $3\mathsf{m} + 3\mathsf{n} + 9$ $\mathsf{multdiv}$ gates. Note that with separate multiplication and division gates one would need $2\mathsf{m} + 3\mathsf{n} + 8$ multiplication gates and $\mathsf{m} + \mathsf{n}$ division gates which would be less efficient.

Acknowledgement. The authors were supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No

780477 (project PRIViLEDGE), and by the Estonian Research Council grant PRG49. The work was done while Zając was working at the University of Tartu.

References

1. Abdolmaleki, B., Baghery, K., Lipmaa, H., Siim, J., Zajac, M.: DL-Extractable UC-Commitment Schemes. Technical Report 2019/201, IACR (2019) Available from `https://eprint.iacr.org/2019/201`.
2. Abdolmaleki, B., Baghery, K., Lipmaa, H., Zajac, M.: A subversion-resistant SNARK. In: ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 3–33
3. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: 45th FOCS, pp. 186–195
4. Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: Security in the face of parameter subversion. In: ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 777–804
5. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474
6. Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure sampling of public parameters for succinct zero knowledge proofs. In: 2015 IEEE Symposium on Security and Privacy, pp. 287–304
7. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. In: 46th ACM STOC, pp. 505–514
8. Bowe, S., Gabizon, A., Green, M.D.: A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. Cryptology ePrint Archive, Report 2017/602 (2017) `http://eprint.iacr.org/2017/602`.
9. Bowe, S., Gabizon, A., Miers, I.: Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050 (2017) `http://eprint.iacr.org/2017/1050`.
10. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145
11. Canetti, R., Fischlin, M.: Universally composable commitments. In: CRYPTO 2001. LNCS, vol. 2139, pp. 19–40
12. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC, pp. 209–218
13. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square span programs with applications to succinct NIZK arguments. In: ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 532–550
14. Fuchsbauer, G.: Subversion-zero-knowledge SNARKs. In: PKC 2018, Part I. LNCS, vol. 10769, pp. 315–347
15. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645
16. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: 43rd ACM STOC, pp. 99–108
17. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th FOCS, pp. 102–115
18. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459

19. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340
20. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326
21. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. LNCS, pp. 698–728
22. Juels, A., Kosba, A.E., Shi, E.: The ring of Gyges: Investigating the future of criminal smart contracts. In: ACM CCS 16, pp. 283–295
23. Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Universally composable synchronous computation. In: TCC 2013. LNCS, vol. 7785, pp. 477–498
24. Kidron, D., Lindell, Y.: Impossibility results for universal composability in public-key models and with fixed inputs. Journal of Cryptology **24**(3) (2011) pp. 517–544
25. Kosba, A.E., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy, pp. 839–858
26. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189
27. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. Cryptology ePrint Archive, Report 2019/099 (2019) `https://eprint.iacr.org/2019/099`.
28. Naor, M.: On Cryptographic Assumptions and Challenges. In: CRYPTO 2003. LNCS, vol. 2729, pp. 96–109
29. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, pp. 238–252