

# On Subversion-Resistant SNARKs<sup>\*</sup>

Behzad Abdolmaleki<sup>1</sup>, Helger Lipmaa<sup>1,2</sup>, Janno Siim<sup>1</sup>, and Michał Zając<sup>3</sup>

<sup>1</sup> University of Tartu, Tartu, Estonia

<sup>2</sup> Simula UiB, Bergen, Norway [helger.lipmaa@gmail.com](mailto:helger.lipmaa@gmail.com)

<sup>3</sup> Clearmatics, London, UK

**Abstract.** While NIZK arguments in the CRS model are widely studied, the question of what happens when the CRS was subverted has received little attention. In ASIACRYPT 2016, Bellare, Fuchsbaauer, and Scafuro showed the first negative and positive results in the case of NIZK, proving also that it is impossible to achieve subversion soundness and (even non-subversion) zero-knowledge at the same time. On the positive side, they constructed an involved sound and subversion-zero-knowledge (Sub-ZK) non-succinct NIZK argument for NP. We consider the practically very relevant case of zk-SNARKs. We make Groth’s zk-SNARK for CIRCUIT-SAT from EUROCRYPT 2016 computationally knowledge-sound and perfectly composable Sub-ZK with minimal changes. We only require the CRS trapdoor to be extractable and the CRS to be publicly verifiable. To achieve the latter, we add some new elements to the CRS and construct an efficient CRS verification algorithm. We also provide a definitional framework for knowledge-sound and Sub-ZK SNARKs.

**Keywords:** Common reference string, generic group model, non-interactive zero knowledge, subversion zero knowledge, SNARK

## 1 Introduction

Combined effort of a large number of recent research papers (to only mention a few, [DL08,Gro10,Lip12,GGPR13,PHGR13,Lip13,DFGK14,Gro16]) has made it possible to construct very efficient succinct non-interactive zero-knowledge arguments of knowledge (zk-SNARKs) for both the Boolean and the Arithmetic CIRCUIT-SAT and thus for NP. The most efficient known approach for constructing zk-SNARKs for the Arithmetic CIRCUIT-SAT is based on Quadratic Arithmetic Programs (QAP, [GGPR13]).

In a QAP, the prover builds a set of polynomial equations that are then checked by the verifier by using a small number of pairings. QAP-based zk-SNARKs have excellent efficiency properties that make them applicable in verifiable computation [GGP10,GGPR13,PHGR13] where the client outsources some computation to the server, who returns the computation result together with a

---

<sup>\*</sup> An earlier version of this paper, [ABLZ17], was published at Asiacrypt 2017. The current version has been significantly modified.

succinct efficiently-verifiable correctness argument. Possibly even more importantly, zk-SNARKs are used in cryptocurrencies [DFKP13,BCG<sup>+</sup>14,KMS<sup>+</sup>16]. See, e.g., [Gro16] for more references.

The currently most efficient zk-SNARK for Arithmetic CIRCUIT-SAT was proposed by Groth (EUROCRYPT 2016, [Gro16]) who proved it to be knowledge-sound in the generic bilinear group model (GBGM [Nec94,Sho97,Mau05,BBG05]) while slightly less efficient zk-SNARKs are known under concrete knowledge assumptions [Dam92]. In Groth’s zk-SNARK, the argument consists of only 3 bilinear group elements and the verifier has to check a single pairing equation, dominated by the computation of three bilinear (type-III [GPS08]) pairings and  $m_0$  exponentiations, where  $m_0$  is the statement size.

Motivated by applications in cryptocurrencies (but not only), there has been a surge of interest in constructing cryptographic primitives and protocols secure against active subversion. In the context of zk-SNARKs, while the common reference string (CRS) model [BFM88] is widely used, one has to be very careful to guarantee that the CRS was created correctly. In [BFS16], Bellare, Fuchsbauer, and Scafuro tackled this problem by studying how much security one can still achieve when the CRS generator cannot be trusted. They proved several negative and positive results. In particular, they showed that it is impossible to achieve subversion soundness (soundness even if the CRS is not trusted) and (even non-subversion) zero-knowledge simultaneously, essentially since the zero-knowledge simulator can be used to break subversion soundness.

In one of their positive solutions, Bellare *et al.* show that it is possible to get (non-subversion) soundness and computational subversion zero-knowledge (Sub-ZK, ZK even if the CRS is not trusted). Their main new idea is to use a knowledge assumption in the Sub-ZK proof so that the simulator can extract a “trapdoor” from the untrusted CRS and then use this trapdoor to simulate the argument. While this idea is neat, their resulting argument system is not efficient. Since it has linear communication, in the case of zk-SNARKs, one presumably has to employ different techniques. One also needs to take care to define and implement statistical Sub-ZK as compared to computational Sub-ZK in [BFS16].

**Our Contributions.** We will take Groth’s zk-SNARK from EUROCRYPT 2016 [Gro16] as a starting point since it is the most efficient known zk-SNARK. Groth [Gro10] observed that if a trusted setup for generating the CRS is not available, the verifier may generate the CRS herself, given that the prover can verify its well-formedness. However, Groth’s proposed solution guarantees only witness-indistinguishability, not zero-knowledge, and he did not describe how to verify the CRS. We propose a minimal modification to Groth’s zk-SNARK that makes it computationally knowledge-sound and statistical composable Sub-ZK. We provide two different versions of this argument: the first has a more straightforward proof strategy, whereas the second is slightly more efficient. Intuitively, the first version verifies the correctness of the whole CRS but the second version

verifies only the correctness of the part of CRS that is relevant for obtaining Sub-ZK. However, on a conceptual level, both versions use similar ideas.

We change Groth’s zk-SNARK by adding extra elements to the CRS in the CRS-generation phase so that the CRS becomes publicly verifiable; under appropriate knowledge assumptions, this minimal step (clearly, some public verifiability of the CRS is needed in the case the CRS generator cannot be trusted) is sufficient to obtain Sub-ZK. However, choosing which elements to add to the CRS is not straightforward since the zk-SNARK must remain knowledge-sound even given enlarged CRS; adding too many or just “wrong” elements to the CRS can break the knowledge-soundness. On the other hand, importantly, the prover and the verifier of the new zk-SNARK are unchanged compared to Groth’s SNARK [Gro16]. In the rest of the introduction, we only outline the novel properties of the new SNARK as compared to [Gro16].

We start by defining perfectly complete, perfectly CRS-verifiable, computationally adaptively knowledge-sound, and statistically unbounded (or composable) zero-knowledge Sub-ZK SNARKs. These definitions are similar to but different from the non-subversion security definitions as given in, say, [Gro06]. First, since one cannot check whether the subverter uses perfectly uniform coins (or, the *CRS trapdoor*) to generate the CRS, we divide the CRS generation into three different algorithms:

- the generation of the CRS trapdoor  $\text{tc}$  (a probabilistic algorithm  $\mathcal{K}_{\text{tc}}$ ),
- the creation of the CRS from  $\text{tc}$  (a deterministic algorithm  $\mathcal{K}_{\text{crs}}$ ), and
- the creation of the simulation trapdoor  $\text{ts}$  from  $\text{tc}$  (a deterministic algorithm  $\mathcal{K}_{\text{ts}}$ ).

While we cannot check that  $\mathcal{K}_{\text{tc}}$  works correctly, we will guarantee that given a fixed  $\text{crs}$  that is accepted by the CRS-verification algorithm  $\text{CV}$  (see below) as well-formed,  $\text{crs}$  has been created by  $\mathcal{K}_{\text{crs}}$  given *some*  $\text{tc}$  as the input. More precisely, we require that Sub-ZK SNARKs satisfy the following trapdoor-extractability property: if  $\text{CV}$  accepts  $\text{crs}$  then there exists an extractor that extracts  $\text{tc}$  such that  $\mathcal{K}_{\text{crs}}$  maps  $\text{tc}$  to  $\text{crs}$ . As we later discuss, extracting  $\text{tc}$  is not strictly necessary, and slightly better efficiency can be achieved by just extracting a valid  $\text{ts}$ . The use of such an extractability requirement means that we achieve Sub-ZK only under a knowledge assumption.

The use of non-falsifiable assumptions (e.g., a knowledge assumption or the generic model) in the knowledge-soundness proof is necessary due to the impossibility result of Gentry and Wichs [GW11]. In the proof of knowledge-soundness, we use (a variant of) the GBGM. Following [Bro01,SPMS02,BFS16], we weaken the usual definition of GBGM by allowing the generic adversary to create (under realistic restrictions) random elements in the source groups without knowing their discrete logarithms. We call the resulting somewhat weaker model the *generic bilinear group model with hashing* (GBGM-H<sup>4</sup>). Following Groth [Gro16] (the main difference being that modeling a more powerful generic adversary and

---

<sup>4</sup> In the conference version of the current paper [ABLZ17], this model was called subversion GBGM (*Sub-GBGM*).

taking into account new CRS elements will complicate the proof somewhat), we prove that the proposed SNARK is knowledge-sound in the GBGM-H even in the case of type-I pairings. This provides a hedge against possible future cryptanalysis that finds an efficient isomorphism between the two source groups.

We consider the case of an efficient PPT subverter. In the proof of statistical composable Sub-ZK, we use a simple knowledge assumption (a variant of which was used, e.g., in [DFGK14]) that we call BDH-KE. We prove that, in the case of type-III pairings, BDH-KE holds in the GBGM-H. The Sub-ZK proof of the only previously known non-interactive Sub-ZK argument system by Bellare *et al.* [BFS16] also relies on knowledge assumptions. (As shown in subsequent work [ALSZ20], it is indeed necessary to use non-falsifiable assumptions.) We follow the main idea of [BFS16] by first using BDH-KE to extract the CRS trapdoor  $\text{tc}$  from the CRS and then construct a non-subversion simulator that gets  $\text{tc}$  as an input to simulate the argument. However, since we construct a zk-SNARK, our concrete approach is different from [BFS16].

Here too, we rely on the existence of an efficient CRS verification algorithm CV. We show that if CV accepts a CRS  $\text{crs}$ , then  $\text{crs}$  has been computed correctly by  $\mathcal{K}_{\text{crs}}$  from some trapdoor  $\text{tc}$  bijectively fixed by  $\text{crs}$ . From this, it follows under the BDH-KE assumption that for any subverter that produces a  $\text{crs}$  accepted by CV, there is an extractor that outputs  $\text{tc}$  such that  $\mathcal{K}_{\text{crs}}$  given  $\text{tc}$  outputs  $\text{crs}$ .

Our security proofs of knowledge-soundness and Sub-ZK work in incomparable models. The knowledge-soundness proof uses the full power of GBGM-H in the case of type- $\{\text{I,II,III}\}$  pairings. The Sub-ZK proof uses a concrete standard-looking knowledge assumption BDH-KE that holds in the GBGM in the case of type- $\{\text{I,II,III}\}$  pairings and in the GBGM-H in the case of type-III pairings. This enables us to construct an efficient Sub-ZK SNARK that uses type-III pairings while guaranteeing its knowledge-soundness even in the case of type-I pairings. In a subsequent work, Lipmaa [Lip19] showed that in the case of type-III pairings, a variant of our Sub-ZK SNARK is knowledge-sound under a concrete knowledge-assumption.

**General Design Recommendations.** Constructing Sub-ZK SNARKs cannot be done automatically, in particular since our framework points to the necessity of making at least a part of the CRS publicly verifiable, which potentially means adding new elements to the CRS. Since knowledge-soundness proofs of many SNARKs are very subtle, it seems to be challenging to give a general “theorem” about which SNARKs can be modified to be Sub-ZK or even whether their CRS can be made verifiable without a significant reconstruction. Whether a SNARK remains sound after that must be proven separately in each case.

However, we can still give a few recommendations for designing a Sub-ZK SNARK from a non-subversion-secure SNARK (or from scratch) when using the same approach as the current paper:

1. Division of duties (in the setup phase): make sure that  $\mathcal{K}$  can be divided into randomized  $\mathcal{K}_{\text{tc}}$ , deterministic  $\mathcal{K}_{\text{ts}}$ , and deterministic  $\mathcal{K}_{\text{crs}}$  algorithms.

2. trapdoor-extractability: each trapdoor element in  $\text{tc}$  or at least in  $\text{ts}$ , must be extractable from the CRS. For this, one can use a knowledge assumption or the GBGM-H.
3. CRS verifiability: assure the CRS is publicly verifiable, i.e.,  $\text{crs} = \text{K}_{\text{crs}}(\text{tc})$ , where  $\text{tc}$  is the trapdoor extracted in the previous step. For better efficiency, one can only check the CRS elements used by the prover or the simulator since other elements cannot affect ZK.
4. Sound approach: make sure that the previous steps do not hurt the knowledge soundness. To achieve it, one should aim at designing a SNARK with a very simple CRS or where CRS verifiability comes naturally. Depending on the SNARK in question, this step may be the most difficult one.

We prove a general result showing that if the above approach is followed and the argument has perfect (composable) ZK, then we automatically achieve statistical unbounded (composable) Sub-ZK. The same is not necessarily true for arguments that have only statistical or computational ZK. Intuitively, the reason is that in statistical (or computational) ZK, there might be a small set of well-formed CRSs  $\mathcal{C}_{bad}$  that make the argument leak information. This does not affect zero-knowledge when the CRS is honestly chosen, assuming that CRSs from  $\mathcal{C}_{bad}$  occur only with a small probability. However, in the subversion setting, an adversary may pick a CRS from the set  $\mathcal{C}_{bad}$  with a much higher likelihood. In the case of a perfect ZK argument, we show that  $\mathcal{C}_{bad} = \emptyset$ , or equivalently, we show that perfect ZK holds even respect to an arbitrary fixed CRS. This makes CRS verifiability and trapdoor extractability sufficient for simulating proofs. We use this result to prove Sub-ZK of our SNARK construction by noting that Groth’s SNARK has perfect ZK.

Finally, we show that the above strategy does not always give the best efficiency. Namely, following an idea from [ALSZ20], we give a slightly more efficient version of the new Sub-ZK SNARK by including the following changes: (i) we do not verify CRS elements used only by the verifier and not by the prover or the simulator (clearly, those elements cannot affect the Sub-ZK property), (ii) we do not extract the whole CRS trapdoor  $\text{tc}$ , but only a smaller simulation trapdoor  $\text{ts}$ . These two changes result in a slightly a more efficient CV algorithm, but we do not achieve trapdoor-extractability anymore. Thus, we cannot rely on the general result in this case and have to prove Sub-ZK from scratch.

**On Efficiency.** Since the new zk-SNARK is closely based on the most efficient known non-subversion zk-SNARK of Groth [Gro16], it has comparable efficiency. Importantly, since we achieve Sub-ZK and non-subversion knowledge-soundness, the CRS verification algorithm CV has to be executed only by the prover and only once in all applications where the prover uses the same  $\text{crs}$  (e.g., throughout the use of Zcash, [BCG<sup>+</sup>14]). This means that it suffices for CV to have the same (or even larger) computational complexity as the prover’s algorithm. The initial CV we describe in Fig. 2 is quite inefficient. However, we show in Section 10 that by using batching techniques, the CV algorithm can be sped up to be faster than

the prover’s algorithm (at the information-theoretical security level  $2^{-80}$ ) and even faster at the information-theoretical security level  $2^{-40}$ .

**Road-Map.** In Section 2, we discuss subsequent work. In Section 3, we describe the preliminaries. In Section 4, we give and motivate definitions of subversion-resistant SNARKs. In Section 5, we define trapdoor-extractability and show how to use it to obtain Sub-ZK. In Section 6, we describe GBGM and GBGM-H. In Section 7, we give the construction of the new Sub-ZK SNARKs. In Section 8, we prove that they are knowledge-sound, and in Section 9, that they are Sub-ZK. In Section 10, we discuss the efficiency of the new SNARKs. In Appendix A, we enlist the most important changes as compared to the conference version; however, this list is not exhaustive.

## 2 Subsequent Work

Bellare *et al.* [BFS16] showed that any NIWI argument like [GOS06] is subversion-soundness and subversion-WI. Subsequently, Fuchsbauer and Orrù [FO18] constructed a subversion-knowledge-sound NIWI. Unfortunately, the known NIWI argument systems are not succinct. Construction of more efficient NIWI argument systems is a very interesting open question.

Independently from our work, Fuchsbauer [Fuc18] constructed subversion-resistant SNARKs. Fuchsbauer’s approach is similar to the approach in [ABLZ17], but he modifies subtly the simulator of Groth’s SNARK so that it does not require the knowledge of trapdoor elements  $\alpha$  and  $\beta$ . Then he shows that the original Groth’s SNARK, without adding extra element to the CRS, is knowledge-sound and Sub-ZK under a somewhat weaker, yet clearly sufficient definition of Sub-ZK. We have adapted a version of Fuchsbauer’s (very natural) simulation technique and Sub-ZK definition in the current version of this paper; however, for simplicity, we use the same knowledge assumption as in [ABLZ17].

Abdolmaleki *et al.* [ALSZ20] showed that the notion of Sub-ZK in the CRS model is equivalent to the notion of *no-auxiliary-string non-black-box zero-knowledge* [GO94] in the weaker BPK model [CGGM00, MR01], where the verifier — or a party, trusted by her (but not by the prover) — itself can create the CRS and the authority is just trusted to store (but not verify) it. Hence, Sub-ZK SNARKs in the CRS model should just be known as zk-SNARKs in the BPK model. Due to the known impossibility results [GO94], it follows that Sub-ZK for NP cannot be based on falsifiable assumptions.

Subsequent work has shown that the GBGM-H is unnecessarily strong. First, the generic group model can be replaced with the algebraic group model (AGM) proposed by Fuchsbauer *et al.* [FKL18], where one only assumes that all adversaries are algebraic, that is, they know (a uniquely defined) linear relation between their input and output group elements. Lipmaa [Lip19] weakened the AGM even more, by showing that one can prove the knowledge-soundness of a version of the SNARKs of the current paper under a concrete knowledge assumption and a computational assumption. Moreover, as shown in [Lip19], one

does not have to assume that the new group elements created by the generic adversary by using hashing are uniformly random; instead, it suffices for them to have high min-entropy.

Groth *et al.* [GKM<sup>+</sup>18] introduced the important notion of *updatable zk-SNARKs* where any party can update the SNARK's CRS. The updated CRSs form a sequence  $\text{crs}_0, \text{crs}_1, \dots, \text{crs}_k$ . The promise is that if at least one of the parties involved in the updating process was honest, then the system is sound. Furthermore, by involving techniques similar to the CV algorithm explained in this paper, each CRS can be verified for its well-formedness. Updatable zk-SNARKs are crucially also Sub-ZK SNARKs; in particular, by the results of [ALSZ20], an updatable zk-SNARK is no-auxiliary-string non-black-box SNARK in the BPK model. In such an “updatable BPK” model, the prover does not have to trust the CRS at all while the verifier only has to trust one of the updaters.

### 3 Preliminaries

For a matrix  $M \in \mathbb{Z}_p^{n \times m}$ , we denote by  $\mathbf{M}_i$  the  $i$ -th row of  $M$  and by  $\mathbf{M}^{(j)}$  the  $j$ -th column of  $M$ . PPT stands for probabilistic polynomial-time, and NUPPT stands for non-uniform PPT. Let  $\lambda$  be the security parameter. We denote by  $\text{negl}(\lambda)$  an arbitrary negligible function. We write  $f(\lambda) \approx_\lambda g(\lambda)$ , if  $f(\lambda) - g(\lambda)$  is negligible as a function of  $\lambda$ . For an algorithm  $\mathcal{A}$ , let  $\text{range}(\mathcal{A})$  be the set of all outputs of  $\mathcal{A}$  that occur with non-zero probability. For an algorithm  $\mathcal{A}$ , let  $\text{RND}(\mathcal{A})$  denote the random tape of  $\mathcal{A}$ , and let  $r \leftarrow_s \text{RND}(\mathcal{A})$  denote the random choice of the randomizer  $r$  from  $\text{RND}(\mathcal{A})$ . By  $y \leftarrow \mathcal{A}(x; r)$  we denote the fact that  $\mathcal{A}$ , given an input  $x$  and a randomizer  $r$ , outputs  $y$ . When we use this notation, then  $r$  represents the full random tape of  $\mathcal{A}$ . For algorithms  $\mathcal{A}$  and  $\text{Ext}_{\mathcal{A}}$ , we write  $(y \parallel y') \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(\chi; r)$  as a shorthand for  $y \leftarrow \mathcal{A}(\chi; r)$ ,  $y' \leftarrow \text{Ext}_{\mathcal{A}}(\chi; r)$ . In both cases,  $\text{Ext}_{\mathcal{A}}$  and  $\mathcal{A}$  use internally the same randomness  $r$ .

In the new argument systems, we will use a small number of indeterminates, and we assume that each indeterminate has a canonical concrete value (a uniformly random element of  $\mathbb{Z}_p$  or  $\mathbb{Z}_p^*$ ). The canonical value of  $X, X_\alpha, X_\beta, X_\gamma, X_\delta$  (resp.,  $Y_i$ ) will be  $\chi, \alpha, \beta, \gamma, \delta$  (resp.,  $v_i$ ). The canonical value of the vector  $\mathbf{X} = (X, X_\alpha, X_\beta, X_\gamma, X_\delta)$  (resp.,  $\mathbf{Y} = (Y_1, \dots)$ ) will be  $\mathbf{x} = (\chi, \alpha, \beta, \gamma, \delta)$  (resp.,  $\mathbf{v} = (v_1, \dots)$ ).

**Lemma 1 (Schwartz-Zippel [Zip79, Sch80]).** *Let  $f \in \mathbb{F}[X_1, \dots, X_n]$  be a non-zero polynomial of total degree  $d \geq 0$  over a field  $\mathbb{F}$ . Let  $S$  be a finite subset of  $\mathbb{F}$ , and let  $x_1, \dots, x_n$  be selected at random independently and uniformly from  $S$ . Then  $\Pr[f(x_1, \dots, x_n) = 0] \leq d/|S|$ .*

**Interpolation.** Assume  $n$  is a power of two, and let  $\omega$  be the  $n$ th primitive root of unity modulo  $p$ . Such  $\omega$  exists, given that  $n \mid (p - 1)$ . Let  $(\omega^0, \omega^1, \dots, \omega^{n-1})$  be the interpolation points. Then,

- $\ell(X) := \prod_{i=1}^n (X - \omega^{i-1}) = X^n - 1$  is the unique degree  $n$  monic polynomial, such that  $\ell(\omega^{i-1}) = 0$  for all  $i \in [1..n]$ .

<pre> <b>complag</b>(<math>\chi, n</math>) ----- <math>\zeta \leftarrow (\chi^n - 1)/n; \omega' \leftarrow 1;</math> <b>if</b> <math>\chi = \omega'</math> <b>then</b> <math>\ell_1(\chi) \leftarrow 1</math>; <b>else</b> <math>\ell_1(\chi) \leftarrow \zeta/(\chi - \omega')</math>; <b>fi</b> <b>for</b> <math>i = 2</math> <b>to</b> <math>n</math> <b>do</b>   <math>\zeta \leftarrow \omega\zeta; \omega' \leftarrow \omega\omega';</math>   <b>if</b> <math>\chi = \omega'</math> <b>then</b> <math>\ell_i(\chi) \leftarrow 1</math>; <b>else</b> <math>\ell_i(\chi) \leftarrow \zeta/(\chi - \omega')</math>; <b>fi</b> <b>endfor</b> </pre>
---

**Fig. 1.** Computing  $(\ell_i(\chi))_{i=1}^n$

- For  $i \in [1..n]$ , let  $\ell_i(X)$  be the  $i$ th Lagrange basis polynomial, i.e., the unique degree  $n-1$  polynomial, s.t.  $\ell_i(\omega^{i-1}) = 1$  and  $\ell_i(\omega^{j-1}) = 0$  for  $i \neq j$ . Clearly,

$$\ell_i(X) := \frac{\ell(X)}{\ell'(\omega^{i-1})(X - \omega^{i-1})} = \frac{(X^n - 1)\omega^{i-1}}{n(X - \omega^{i-1})}. \quad (1)$$

Thus,  $\ell_i(\omega^{i-1}) = 1$  while  $\ell_i(\chi) = (\chi^n - 1)\omega^{i-1}/(n(\chi - \omega^{i-1}))$  for  $\chi \neq \omega^{i-1}$ .

Given any  $\chi \in \mathbb{Z}_p$ , `complag` in Fig. 1 (see, e.g., [BCG<sup>+</sup>13]) computes  $\ell_i(\chi)$  for  $i \in [1..n]$ . It can be implemented in  $4n - 2$  multiplications and divisions in  $\mathbb{Z}_p$ .

Clearly,  $L_{\mathbf{a}}(X) := \sum_{i=1}^n a_i \ell_i(X)$  is the interpolating polynomial of  $\mathbf{a}$  at points  $\omega^{i-1}$ , with  $L_{\mathbf{a}}(\omega^{i-1}) = a_i$ , and its coefficients can thus be computed by executing an inverse Fast Fourier Transform in time  $\Theta(n \log n)$ .

**Elliptic Curves and Bilinear Maps.** On input  $1^\lambda$ , a *bilinear map generator* `Pgen` returns  $\mathbf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, P_1, P_2)$ , where  $\mathbb{G}_1, \mathbb{G}_2$  and  $\mathbb{G}_T$  are three additive cyclic groups of prime order  $p$  (with  $\log p = \Omega(\lambda)$ ) and  $P_\nu$  is a randomly sampled generator of  $\mathbb{G}_\nu$  for  $\nu \in \{1, 2, T\}$ . As in [BFS16] (but not as in the conference version [ABLZ17] of the current paper), we will explicitly assume that the system parameters  $\mathbf{pp}$  are generated deterministically from  $\lambda$ . This means that in particular, the choice of  $\mathbf{pp}$  cannot be subverted. Moreover, for relevant security levels  $\lambda$  (say,  $\lambda = 128$ ), there usually exists a standard recommendation of which elliptic curve to use. (The current recommendation for  $\lambda = 128$  is the curve BLS12-381, [Bow17].) Additionally,  $\hat{e}$  is an efficient bilinear map  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  that, in particular, satisfies the following two properties: (i)  $\hat{e}(P_1, P_2) \neq 1$ , and (ii)  $\hat{e}(aP_1, bP_2) = (ab) \cdot \hat{e}(P_1, P_2)$ .

We give `Pgen` another input,  $n$  (intuitively, the input length), and allow  $p$  to depend on  $n$ . This is needed for efficient interpolation, for which we will need the existence of the  $n$ th, where  $n = 2^s$  is a power of 2, primitive root of unity modulo  $p$ . For this, it suffices that  $2^s \mid (p - 1)$  (recall that  $p$  is the elliptic curve group order). The curve BLS12-381 satisfies  $2^{32} \mid (p - 1)$ . All algorithms that handle group elements verify by default that their inputs belong to corresponding groups and reject if not. Usually, arithmetic in  $\mathbb{G}_1$  is considerably cheaper than in  $\mathbb{G}_2$ ; thus, we count exponentiations separately in both groups.

We use the by-now standard bracket notation [EHK<sup>+</sup>13]. That is, for an integer  $x$ , we denote  $[x]_\nu := xP_\nu$  even when  $x$  is unknown. We denote  $[x]_1 \bullet [y]_2 := \hat{e}([x]_1, [y]_2)$ , hence  $[xy]_T = [x]_1 \bullet [y]_2$ . We denote  $[a_1, \dots, a_s]_\nu = ([a_1]_\nu, \dots, [a_s]_\nu)$ .

**Quadratic Arithmetic Programs.** Gennaro *et al.* [GGPR13] introduced Quadratic Arithmetic Program (QAP) as a language where for an input  $x$  and witness  $w$  predicate  $(x, w) \in \mathbf{R}$  can be verified by using a parallel quadratic check and has an efficient reduction from the language (either Boolean or Arithmetic) CIRCUIT-SAT. Hence, an efficient zk-SNARK for QAP results in an efficient zk-SNARK for CIRCUIT-SAT.

In the case of an arithmetic circuit,  $n$  is the number of multiplication gates and  $m$  is the number of wires in the circuit. As in [GGPR13], we consider arithmetic circuits that consist only of fan-in-2 multiplication gates, where either input of each multiplication gate is a weighted sum of some wire values.

Let  $\mathbb{F} = \mathbb{Z}_p$ , such that  $\omega$  is the  $n$ th primitive root of unity modulo  $p$ . (This requirement is needed for the sake of efficiency, and we will make it implicitly throughout the current paper.) An QAP instance  $\mathcal{Q}$  is specified by  $(\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=1}^m)$ , where  $u_j(X) = \sum_{i=1}^n U_{ij} \ell_i(X)$ ,  $v_j(X) = \sum_{i=1}^n V_{ij} \ell_i(X)$ , and  $w_j(X) = \sum_{i=1}^n W_{ij} \ell_i(X)$  are degree- $\leq (n-1)$  polynomials from  $\mathbb{Z}_p[X]$ . Here,  $U, V, W \in \mathbb{Z}_p^{n \times m}$  are public sparse matrices. The instance  $\mathcal{Q}$  defines the following relation:

$$\mathbf{R}_{\mathcal{Q}} = \left\{ (x, w) : x = (A_1, \dots, A_{m_0})^\top \wedge w = (A_{m_0+1}, \dots, A_m)^\top \wedge \left( \sum_{j=1}^m A_j u_j(X) \right) \left( \sum_{j=1}^m A_j v_j(X) \right) \equiv \sum_{j=1}^m A_j w_j(X) \pmod{\ell(X)} \right\}.$$

Alternatively,  $(x, w) \in \mathbf{R}$  if there exists a degree- $\leq (n-2)$  polynomial  $h(X)$ , s.t.

$$\left( \sum_{j=1}^m A_j u_j(X) \right) \left( \sum_{j=1}^m A_j v_j(X) \right) - \sum_{j=1}^m A_j w_j(X) = h(X) \ell(X).$$

## 4 Definitions: SNARKs and Subversion Zero Knowledge

Next, we define subversion zero-knowledge (Sub-ZK) SNARKs and all their properties. To achieve Sub-ZK, we augment a zk-SNARK by requiring the existence of an *efficient* CRS verification (CV) algorithm. As outlined in Section 1, we also subdivide the CRS generation algorithm into two efficient algorithms.

Our definition of (statistical unbounded) Sub-ZK for SNARKs is motivated by the definition of [BFS16]. We also define statistical composable Sub-ZK. As in [Gro06], the definition of unbounded zero-knowledge guarantees security for any (polynomial) number of queries to the prover or simulator, while the definition of composable zero-knowledge guarantees security only in the case of a single query. Following [Gro06] we prove that a statistical composable Sub-ZK argument system is also statistical unbounded Sub-ZK. This allows us to use the simpler definition of statistical composable Sub-ZK in the rest of the paper.

## 4.1 Syntax

Let  $\text{RelGen}$  be a relation generator, such that  $\text{RelGen}(1^\lambda)$  returns a polynomial-time decidable binary relation  $\mathbf{R} = \{(x, w)\}$ . Here,  $x$  is the statement, and  $w$  is the witness. We assume that  $\lambda$  is explicitly deducible from the description of  $\mathbf{R}$ . The relation generator also outputs auxiliary information  $\mathbf{z}_R$ . As in [Gro16, Sect. 2.3],  $\mathbf{z}_R$  equals  $\text{pp} \leftarrow \text{Pgen}(1^\lambda, n)$  for a well-defined  $n$ . Because of this, we give  $\mathbf{z}_R$  as an input to all (including honest or adversarial) parties. According to the terminology of [BCPR14],  $\tilde{\mathbf{z}}_R := (\mathbf{R}, \mathbf{z}_R)$  is the common auxiliary input to an adversary and the corresponding extractor. Let  $\mathcal{L}_R = \{x : \exists w \text{ s.t. } |w| = \text{poly}(|x|), (x, w) \in \mathbf{R}\}$  be an NP-language.

A subversion-resistant *non-interactive zero-knowledge (NIZK) argument system*  $\Psi$  for  $\text{RelGen}$  consists of six PPT algorithms:

**CRS trapdoor generator**  $K_{\text{tc}}$ : a probabilistic algorithm that, given  $\tilde{\mathbf{z}}_R \in \text{range}(\text{RelGen}(1^\lambda))$ , outputs a *CRS trapdoor*  $\text{tc}$ . Otherwise, it outputs a special symbol  $\perp$ .

**Simulation trapdoor generator**  $K_{\text{ts}}$ : a *deterministic* algorithm that, given  $(\tilde{\mathbf{z}}_R, \text{tc})$  where  $\tilde{\mathbf{z}}_R \in \text{range}(\text{RelGen}(1^\lambda))$  and  $\text{tc} \in \text{range}(K_{\text{tc}}(\tilde{\mathbf{z}}_R)) \setminus \{\perp\}$ , outputs the *simulation trapdoor*  $\text{ts}$ . Otherwise, it outputs  $\perp$ .

**CRS generator**  $K_{\text{crs}}$ : a *deterministic* algorithm that, given  $(\tilde{\mathbf{z}}_R, \text{tc})$ , where  $\tilde{\mathbf{z}}_R \in \text{range}(\text{RelGen}(1^\lambda))$  and  $\text{tc} \in \text{range}(K_{\text{tc}}(\tilde{\mathbf{z}}_R)) \setminus \{\perp\}$ , outputs  $\text{crs}$ . Otherwise, it outputs  $\perp$ . For the sake of efficiency and readability, we divide  $\text{crs}$  into  $\text{crs}_P$  (the part needed to execute the honest prover),  $\text{crs}_V$  (the part needed to execute the honest verifier), and  $\text{crs}_{CV}$  (the part needed only to executed CV but not needed by P or V). If an element is used by both P and V then it will belong to both  $\text{crs}_P$  and  $\text{crs}_V$ .

**CRS verifier**  $\text{CV}$ : a probabilistic algorithm that, given  $(\tilde{\mathbf{z}}_R, \text{crs})$ , returns either 0 (the CRS is malformed) or 1 (the CRS is well-formed),

**Prover**  $P$ : a probabilistic algorithm that, given  $(\tilde{\mathbf{z}}_R, \text{crs}_P, x, w)$ , outputs an argument  $\pi$  if  $\text{CV}(\tilde{\mathbf{z}}_R, \text{crs}) = 1$  and  $(x, w) \in \mathbf{R}$ . Otherwise, it outputs  $\perp$ .

**Verifier**  $V$ : a probabilistic algorithm that, given  $(\tilde{\mathbf{z}}_R, \text{crs}_V, x, \pi)$ , returns either 0 (reject) or 1 (accept).

**Simulator**  $\text{Sim}$ : a probabilistic algorithm that, given  $(\tilde{\mathbf{z}}_R, \text{crs}, \text{ts}, x)$  where  $\text{CV}(\tilde{\mathbf{z}}_R, \text{crs}) = 1$ , outputs an argument  $\pi$ . Otherwise, it outputs  $\perp$ .

Let  $K(\tilde{\mathbf{z}}_R)$  be the combined CRS generation algorithm that first sets  $\text{tc} \leftarrow K_{\text{tc}}(\tilde{\mathbf{z}}_R)$ ,  $\text{crs} \leftarrow K_{\text{crs}}(\tilde{\mathbf{z}}_R, \text{tc})$ ,  $\text{ts} \leftarrow K_{\text{ts}}(\tilde{\mathbf{z}}_R, \text{tc})$ , and then outputs  $(\text{crs}, \text{ts})$ .

**SNARKs.** A non-interactive argument system is *succinct* if the proof size is polynomial in  $\lambda$  and the verifier runs in time polynomial in  $\lambda + |x|$ . A succinct non-interactive argument of knowledge is usually called *SNARK*. A zero-knowledge SNARK is abbreviated to *zk-SNARK*.

**Discussions.** A non-interactive zero-knowledge argument system is usually defined as a tuple  $(K, P, V, \text{Sim})$ , see, e.g., [Gro16]. Next, we will briefly motivate the

differences compared to the established syntax of non-interactive zero-knowledge argument systems. Section 4.2 will give formal security definitions where the above syntactic definition is an important part.

The division of  $K$  into 3 algorithms  $K_{tc}$ ,  $K_{ts}$ , and  $K_{crs}$  is usually not needed. Still,  $K$  of many known non-interactive zero-knowledge argument systems (and all zk-SNARKs that we know) satisfies such a division.  $K_{tc}$  just generates all randomness ( $tc$ ), needed to compute  $crs$  and  $ts$ , and then  $K_{crs}$  and  $K_{ts}$  compute  $crs$  and  $ts$  deterministically from  $tc$ . (Often,  $ts = tc$ .) Such division can be formalized by requiring the  $crs$  to be witness-sampleable [JR13] that also seems to be a reasonable requirement in case one is interested in subversion-resistance. Really, an important subgoal of Sub-ZK is to guarantee that the subverted CRS is consistent with at least some choice of  $tc$ . That is, for each  $tc$ , there must exist corresponding  $crs$  (accepted by CV) and  $ts$  (that can be used by the simulator to simulate subverted  $crs$  corresponding to  $tc$ ).

The existence of *efficient* CV will be crucial to obtain Sub-ZK. To guarantee Sub-ZK, it is intuitively clear that an honest prover should *at least* check the correctness of the CRS. Efficiency-wise, since the prover in known zk-SNARK constructions (including say [GGPR13, Gro16] and the current paper) takes superlinear time, this is fine unless the CRS verification will be even slower. We do not assume, however, that CV is a part of the P *algorithm*; instead, we assume that an honest prover first runs CV and given that it accepts, runs P. Such a separation is natural since, in practice, one can execute many zero-knowledge arguments by using the same CRS; it makes sense that then the prover runs CV only once. On the other hand, since an honest V trusts the CRS generator, she does not have to execute CRS verification.

Finally,  $crs_P$  (resp.,  $crs_V$ ) is the part of the CRS given to an honest prover (resp., an honest verifier), and  $crs_{CV}$  is the part of CRS not needed by the prover or the verifier except to run CV. The distinction between  $crs_P$ ,  $crs_V$ , and  $crs_{CV}$  is not significant from the security point of view, but in many cases  $crs_V$  is significantly shorter than  $crs_P$ . Keeping  $crs_{CV}$  separate helps one to evaluate better the additional efficiency penalty introduced by subversion security.

## 4.2 Security Definitions

A Sub-ZK SNARK has to satisfy various security definitions. The most important ones are *completeness* (an honest prover convinces an honest verifier), *CRS-verifiability* (an honestly generated CRS passes the CRS verification test), *computational knowledge-soundness* (if a prover convinces an honest verifier, then he knows the corresponding witness), and *statistical Sub-ZK* (given a possibly subverted CRS, an argument created by the honest prover reveals no side information). Next, we will define those properties in a way that guarantees both compositability and subversion resistance.

To keep the new security definitions as close to the accepted security definitions of zk-SNARKs as possible, we will start with non-subversion-resistant security definitions from [Gro16] (that will be stated below for the sake of

completeness), albeit by using our notation, and modify them by adding elements of subversion as in Bellare *et al.* [BFS16]. To ease the reading, we will *emphasize* the differences between non-subversion and subversion definitions. We use the division of CRS generation into three different algorithms also in the non-subversion-resistant case. As motivated in Section 4.1, we also give  $\mathbf{z}_R$  as an input to all honest parties. Finally, the notions of unbounded (ZK is guaranteed against an adversary who can arbitrarily query an oracle that outputs either proofs or simulations) and composable (ZK is insured against an adversary who has to distinguish a single argument from a simulation) zero-knowledge follow the definitions given in [Gro06] and are in fact equal in our case.

**Definition 1 (Perfect Completeness [Gro16]).** *A non-interactive argument  $\Psi$  is perfectly complete for RelGen, if for all  $\lambda$ , all  $\tilde{\mathbf{z}}_R \in \text{range}(\text{RelGen}(1^\lambda))$ ,  $\text{tc} \in \text{range}(\text{K}_{\text{tc}}(\tilde{\mathbf{z}}_R)) \setminus \{\perp\}$ , and  $(\mathbf{x}, \mathbf{w}) \in \mathbf{R}$ ,*

$$\Pr[\text{crs} \leftarrow \text{K}_{\text{crs}}(\tilde{\mathbf{z}}_R, \text{tc}) : \text{V}(\tilde{\mathbf{z}}_R, \text{crs}_V, \mathbf{x}, \text{P}(\tilde{\mathbf{z}}_R, \text{crs}_P, \mathbf{x}, \mathbf{w})) = 1] = 1 .$$

**Definition 2 (Perfect CRS Verifiability).** *A non-interactive argument  $\Psi$  is perfectly CRS-verifiable for RelGen, if for all  $\lambda$ , all  $\tilde{\mathbf{z}}_R \in \text{range}(\text{RelGen}(1^\lambda))$ , and  $\text{tc} \in \text{range}(\text{K}_{\text{tc}}(\tilde{\mathbf{z}}_R)) \setminus \{\perp\}$ ,*

$$\Pr[\text{crs} \leftarrow \text{K}_{\text{crs}}(\tilde{\mathbf{z}}_R, \text{tc}) : \text{CV}(\tilde{\mathbf{z}}_R, \text{crs}) = 1] = 1 .$$

**Definition 3 (Computational Knowledge Soundness [Gro16]).**  *$\Psi$  is computationally (adaptively) knowledge-sound for RelGen, if for every NUPPT  $\mathcal{A}$ , there exists a NUPPT extractor  $\text{Ext}_{\mathcal{A}}$ , s.t. for all  $\lambda$ ,*

$$\Pr \left[ \begin{array}{l} \tilde{\mathbf{z}}_R \leftarrow \text{RelGen}(1^\lambda), (\text{crs}, \text{ts}) \leftarrow \text{K}(\tilde{\mathbf{z}}_R), \\ r \leftarrow_s \text{RND}(\mathcal{A}), ((\mathbf{x}, \pi) \parallel \mathbf{w}) \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(\tilde{\mathbf{z}}_R, \text{crs}; r) : \\ (\mathbf{x}, \mathbf{w}) \notin \mathbf{R} \wedge \text{V}(\tilde{\mathbf{z}}_R, \text{crs}_V, \mathbf{x}, \pi) = 1 \end{array} \right] \approx_\lambda 0 .$$

Here,  $\mathbf{z}_R$  can be seen as a common auxiliary input to  $\mathcal{A}$  and  $\text{Ext}_{\mathcal{A}}$  that is generated by using a benign [BCPR14] relation generator (see [BCPR14] for an analysis where RelGen is *not* benign). We recall that we just think of  $\mathbf{z}_R$  as being the description of a secure bilinear group, and thus RelGen is benign. A knowledge-sound argument system is called an *argument of knowledge*.

Next, we define statistically unbounded ZK. Unbounded (non-Sub) ZK was not defined in [Gro16], presumably because it is a corollary of composable non-Sub ZK as shown in [Gro06]. Hence, we will first give a modified version of the definition of non-Sub ZK from [Gro06] where  $\text{K}$  will only output a  $\text{crs}$  and a CRS simulator  $\text{Sim}_{\text{crs}}$  will return  $(\text{crs}, \text{ts})$ . As in [Gro16], we find it more convenient to let  $\text{Sim}_{\text{crs}}$  (whom we call  $\text{K}$ ) to generate also the honest  $\text{crs}$ .

**Definition 4 (Statistically Unbounded ZK [Gro06]).**  *$\Psi$  is statistically unbounded Sub-ZK for RelGen, if for all  $\lambda$ , all  $\tilde{\mathbf{z}}_R \in \text{range}(\text{RelGen}(1^\lambda))$ , and all computationally unbounded  $\mathcal{A}$ ,  $\varepsilon_0^{\text{unb}} \approx_\lambda \varepsilon_1^{\text{unb}}$ , where*

$$\varepsilon_b^{\text{unb}} = \Pr[(\text{crs}, \text{ts}) \leftarrow \text{K}(\tilde{\mathbf{z}}_R) : \mathcal{A}^{\text{O}_b(\cdot, \cdot)}(\tilde{\mathbf{z}}_R, \text{crs}) = 1] .$$

Here, the oracle  $O_0(x, w)$  returns  $\perp$  (reject) if  $(x, w) \notin \mathbf{R}$ , and otherwise it returns  $P(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}_{\mathcal{P}}, x, w)$ . Similarly,  $O_1(x, w)$  returns  $\perp$  (reject) if  $(x, w) \notin \mathbf{R}$ , and otherwise it returns  $\text{Sim}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \text{ts}, x)$ .  $\Psi$  is perfectly unbounded ZK for  $\text{RelGen}$  if  $\varepsilon_0^{\text{unb}} = \varepsilon_1^{\text{unb}}$ .

The following definition of unbounded Sub-ZK differs from this as follows. Since we allow the CRS to be subverted, the CRS is generated by a subverter  $\mathcal{S}$  who also returns an auxiliary string  $z_{\mathcal{S}}$ . The adversary's access to  $z_{\mathcal{S}}$  models the possibility that the subverter and the adversary collaborate. The extractor  $\text{Ext}_{\mathcal{S}}$  extracts  $\text{ts}$  from  $\mathcal{S}$  and gives it to the oracle  $O_1$ . In [Gro06],  $\text{ts}$  was not given to the adversary because  $\mathbf{K}$  was not required to return  $\text{tc}$  and thus was not guaranteed to exist; adding this input to the adversary only increases the power of the adversary. In our construction, this issue does not matter since a computationally unbounded  $\mathcal{A}$  could compute  $\text{tc}$  herself (see the description of  $\text{Sim}$  in Fig. 2).

**Definition 5 (Statistically Unbounded Sub-ZK).**  $\Psi$  is statistically unbounded Sub-ZK for  $\text{RelGen}$ , if for any PPT subverter  $\mathcal{S}$  there exists a PPT  $\text{Ext}_{\mathcal{S}}$ , such that for all  $\lambda$ , all  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ , and all computationally unbounded  $\mathcal{A}$ ,  $\varepsilon_0^{\text{unb}} \approx_{\lambda} \varepsilon_1^{\text{unb}}$ , where

$$\varepsilon_b^{\text{unb}} x = \Pr \left[ \begin{array}{l} r \leftarrow_{\$} \text{RND}(\mathcal{S}), (\text{crs}, z_{\mathcal{S}} \parallel \text{ts}) \leftarrow (\mathcal{S} \parallel \text{Ext}_{\mathcal{S}})(\tilde{\mathbf{z}}_{\mathbf{R}}; r) : \\ \text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1 \wedge \mathcal{A}^{\text{Ob}(\cdot, \cdot)}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, z_{\mathcal{S}}) = 1 \end{array} \right].$$

Here, the oracle  $O_0(x, w)$  returns  $\perp$  (reject) if  $(x, w) \notin \mathbf{R}$ , and otherwise it returns  $P(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}_{\mathcal{P}}, x, w)$ . Similarly,  $O_1(x, w)$  returns  $\perp$  (reject) if  $(x, w) \notin \mathbf{R}$ , and otherwise it returns  $\text{Sim}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \text{ts}, x)$ .  $\Psi$  is perfectly unbounded Sub-ZK for  $\text{RelGen}$  if one requires that  $\varepsilon_0^{\text{unb}} = \varepsilon_1^{\text{unb}}$ .

*Remark 1.* Following [BFS16] (and differently from the conference version [ABLZ17] of the current paper), we consider only *uniform* PPT  $\mathcal{S}$  and  $\text{Ext}_{\mathcal{S}}$  in the definition of Sub-ZK. On the other hand, we consider NUPPT  $\mathcal{A}$  and  $\text{Ext}_{\mathcal{A}}$  in the case of knowledge-soundness.

First, the extractor has to be at least as powerful as the corresponding adversary ( $\mathcal{S}$  or  $\mathcal{A}$ ); otherwise, it is difficult to argue how this extractor can perform its duties. Second, a non-uniform adversary can output a part of her auxiliary string [GO94], without knowing how it was generated; in this case, the extractor does not work. If the input of the adversary is randomized, for such an adversarial strategy to work, the auxiliary string should include an acceptable output for a non-negligible fraction of the possible inputs. (We emphasize we are talking about the input, not the random tape of the adversary.) In particular, if the entropy of the adversary's input is  $O(\log \lambda)$ , then the (poly-length) auxiliary string can contain her outputs corresponding to a non-negligible fraction of all possible random inputs and thus the extractor is not applicable.

In the case of knowledge-soundness, the input of the adversary includes correctly generated CRS; thus, assuming the random trapdoor length is at  $\omega(\log \lambda)$ ,

as it is in all CRS-based SNARKs, one can obtain security against NUPPT adversaries. In the case of Sub-ZK, the only “random” input of the subverter is  $\tilde{z}_{\mathbf{R}}$ . In most of the applications,  $\mathbf{R}$  is not randomized, and  $z_{\mathbf{R}}$  only contains the description of the bilinear group. If the bilinear group is randomly generated (from a large enough set of bilinear groups), then one can obtain Sub-ZK against NUPPT adversaries. However, in practice, the bilinear group is fixed in advance, with the current recommendation being BLS12-381, [Bow17], and thus the subverter might have no random input at all. Following Bellare *et al.* [BFS16], we assume that the bilinear group is selected deterministically, and thus one can only obtain Sub-ZK against PPT subverters. Such an assumption on Pgen was not made in [ABLZ17], and thus they could claim security against NUPPT subverters in the definition of Sub-ZK.

In the case of composable ZK, the adversary can only see one purported (i.e., real or simulated) argument  $\pi$ , instead of being able to make many queries to a purported prover oracle as in the case of unbounded ZK. If composable ZK is defined carefully, it will be at least as strong as unbounded ZK while potentially allowing for simpler security proofs, [Gro06]. We will show that the same holds in the case of Sub-ZK.

**Definition 6 (Statistically Composable ZK [Gro06]).**  $\Psi$  is statistically composable zero-knowledge for RelGen, if for all  $\lambda$ ,  $\tilde{z}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ , and all computationally unbounded  $\mathcal{A}$ ,  $\varepsilon_0^{\text{comp}} \approx_\lambda \varepsilon_1^{\text{comp}}$ , where  $\varepsilon_b^{\text{comp}} =$

$$\Pr \left[ (\text{crs}, \text{ts}) \leftarrow \mathbf{K}(\tilde{z}_{\mathbf{R}}), (x, w) \leftarrow \mathcal{A}(\tilde{z}_{\mathbf{R}}, \text{crs}, \text{ts}); \pi_0 \leftarrow \mathbf{P}(\tilde{z}_{\mathbf{R}}, \text{crs}_{\mathbf{P}}, x, w); \right. \\ \left. \pi_1 \leftarrow \text{Sim}(\tilde{z}_{\mathbf{R}}, \text{crs}, \text{ts}, x) : (x, w) \in \mathbf{R} \wedge \mathcal{A}(\pi_b) = 1 \right].$$

$\Psi$  is perfectly composable Sub-ZK for RelGen if one requires that  $\varepsilon_0^{\text{comp}} = \varepsilon_1^{\text{comp}}$ .

Since  $\mathcal{A}$  is unbounded, she can usually compute both  $\text{ts}$  and  $\text{tc}$  from  $\text{crs}$ .

Next, we define statistical composable subversion zero knowledge (Sub-ZK). This definition is related to but crucially different from the definition of (computational) composable ZK from [Gro06]. Most importantly, [Gro06] defines two properties, the first being *reference string indistinguishability*, meaning that the CRS generated by *honest*  $\mathbf{K}$  and the CRS simulated by a simulator  $\text{Sim}_{\text{crs}}$  should be indistinguishable. We will use the CRS generated by the subverter in both the real and the simulated case. The second property in [Gro06] is simulation indistinguishability. Our definition of composable Sub-ZK is similar to the definition of simulation indistinguishability in [Gro06]. However, instead of simulating the CRS, we use  $\text{crs}$  generated by the subverter and assume that an extractor extracts  $\text{ts}$  from  $\text{crs}$ .

Moreover, we differ from the definition of composable non-Sub ZK in [Gro16] as follows. The CRS is generated by  $\mathcal{S}$  who also returns  $z_{\mathcal{S}}$ .  $\mathcal{A}$ 's access to  $z_{\mathcal{S}}$  models the possibility that  $\mathcal{S}$  and  $\mathcal{A}$  collaborate; although of course  $\mathcal{A}$  could just recompute it. We also drop  $\text{ts}$  as a part of  $\mathcal{A}$ 's input as we now consider it implicitly to be a part of  $z_{\mathcal{S}}$ .

**Definition 7 (Statistically Composable Sub-ZK).**  $\Psi$  is statistically composable Sub-ZK for RelGen, if for any PPT subverter  $\mathcal{S}$  there exists a PPT  $\text{Ext}_{\mathcal{S}}$ , such that for all  $\lambda$ , all  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ , and all computationally unbounded  $\mathcal{A}$ ,  $\varepsilon_0^{\text{comp}} \approx_\lambda \varepsilon_1^{\text{comp}}$ , where  $\varepsilon_b^{\text{comp}} =$

$$\Pr \left[ \begin{array}{l} r \leftarrow_{\$} \text{RND}(\mathcal{S}), (\text{crs}, \text{z}_{\mathcal{S}} \parallel \text{ts}) \leftarrow (\mathcal{S} \parallel \text{Ext}_{\mathcal{S}})(\tilde{\mathbf{z}}_{\mathbf{R}}; r), \\ (\mathbf{x}, \mathbf{w}) \leftarrow \mathcal{A}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \text{z}_{\mathcal{S}}, \pi_0 \leftarrow \text{P}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}_{\mathbf{P}}, \mathbf{x}, \mathbf{w})); \\ \pi_1 \leftarrow \text{Sim}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \text{ts}, \mathbf{x}); (\mathbf{x}, \mathbf{w}) \in \mathbf{R} \wedge \text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1 \wedge \mathcal{A}(\pi_b) = 1 \end{array} \right].$$

$\Psi$  is perfectly composable Sub-ZK for RelGen if one requires that  $\varepsilon_0^{\text{comp}} = \varepsilon_1^{\text{comp}}$ .

*Remark 2 (Comparison to Sub-ZK Definition of [BFS16]).* Let  $\mathcal{O}_b$  be defined as before. In [BFS16], it is required that for any PPT subverter  $\mathcal{S}$  there exists a PPT simulator  $(\text{Ext}_{\mathcal{S}}, \text{Sim})$ , such that for all  $\lambda$ ,  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ , and all PPT  $\mathcal{A}$ ,  $\varepsilon_0^{\text{bfs}} \approx_\lambda \varepsilon_1^{\text{bfs}}$ , where

$$\varepsilon_b^{\text{bfs}} = \Pr \left[ \begin{array}{l} \text{if } b = 0 \text{ then } r \leftarrow_{\$} \text{RND}(\mathcal{S}), \text{crs} \leftarrow \mathcal{S}(\tilde{\mathbf{z}}_{\mathbf{R}}; r); \\ \text{else } (\text{crs}, r) \leftarrow \text{Ext}_{\mathcal{S}}(\tilde{\mathbf{z}}_{\mathbf{R}}); \text{fi} : \mathcal{A}^{\mathcal{O}_b(\cdot, \cdot)}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, r) \end{array} \right].$$

First, [BFS16] defines computational Sub-ZK while we define statistical Sub-ZK. This by itself changes several aspects of the definition. E.g., we can just let  $\mathcal{A}$  to compute  $\text{tc}$  and  $\text{z}_{\mathcal{S}}$  from  $\text{crs}$  instead of giving them as extra inputs to  $\mathcal{A}$ .

Second, compared to [BFS16], we give to  $\mathcal{A}$  extra information  $\text{z}_{\mathcal{S}}$  (that also, w.l.o.g., contains  $\text{tc}$ ). Having access to  $\text{tc}$  means that one can implement SNARKs for different relations using the same CRS [Gro06]. Really, given  $\text{tc}$ ,  $\mathcal{A}$  can both form and simulate arguments for any of the considered relations. Having access to  $\text{z}_{\mathcal{S}}$  models, as already mentioned, the possibility that  $\mathcal{S}$  and  $\mathcal{A}$  are collaborating. (Bellare *et al.* only allow the subverter to communicate  $r$ , the secret coins.) In this sense, our definition is stronger compared to [BFS16].

Third, Bellare *et al.* required that for each  $\mathcal{S}$  there exists a simulator  $(\text{Ext}_{\mathcal{S}}, \text{Sim})$ . We consider it to be more natural to think of  $\text{Ext}_{\mathcal{S}}$  as an extractor and allow only  $\text{Ext}_{\mathcal{S}}$  to depend on  $\mathcal{S}$ . In the SNARK literature, extractors usually depend on the adversary (here,  $\mathcal{S}$ ) while there is a single simulator that works for all adversaries. Also Bellare *et al.* used  $\text{Ext}_{\mathcal{S}}$  as an extractor in their construction. In this sense, our definition is stronger compared to [BFS16].

Fourth, we give to  $\text{Ext}_{\mathcal{S}}$  the same coins  $r$  as to  $\mathcal{S}$ , while Bellare *et al.* allow  $\text{Ext}_{\mathcal{S}}$  to generate its own coins, only requiring that the distribution of  $(\text{crs}, r)$  is computationally indistinguishable from the output of  $\mathcal{S}$ . Also in this sense, our definition seems to be stronger.

Finally, we use explicitly the syntax of subversion-resistant SNARKs from Section 4.1, assuming the existence of algorithms  $\text{K}_{\text{ts}}$  and  $\text{CV}$ . While it might seem to be restrictive, as argued in Section 4.1, existence of both  $\text{K}_{\text{ts}}$  and  $\text{CV}$  seems to be necessary for subversion-resistance.

Now, we will prove the following result that makes it possible to operate in the rest of the paper with the simpler composable Sub-ZK definition. It is

motivated by a similar result of Groth [Gro06] that considers computational non-subversion-resistant zero knowledge. As seen from below, we can establish the same result for statistical zero knowledge, but then we have to restrict the number of oracle calls to a polynomial number.

**Theorem 1.** (i) *Statistical composable Sub-ZK implies statistical unbounded Sub-ZK, assuming that  $\mathcal{A}$  is given access to polynomially many oracle calls.*  
(ii) *Perfect composable Sub-ZK implies perfect unbounded Sub-ZK, even if given access to an unbounded number of oracle calls.*

*Proof.* (I) STATISTICAL SUB-ZK. Assume that the adversary can make up to  $q(\lambda)$  oracle queries for some fixed polynomial  $q$ . We define a sequence of  $q(\lambda) + 1$  oracles  $\mathcal{O}'_0(x, w), \dots, \mathcal{O}'_{q(\lambda)}(x, w)$ . Given the  $j$ th adversarial query  $(x_j, w_j)$ , the oracle  $\mathcal{O}'_k(\cdot, \cdot)$  responds with  $\mathbf{O}_1(x_j, w_j)$  for  $j \in [1..k]$  and  $\mathbf{O}_0(x_j, w_j)$  for  $j \in [k + 1..q(\lambda)]$ . Hence,  $\mathcal{O}'_0 = \mathbf{O}_0$  and  $\mathcal{O}'_{q(\lambda)} = \mathbf{O}_1$ .

Due to the statistical composable Sub-ZK property, we get that for  $i \in [0..q(\lambda) - 1]$ ,  $\varepsilon_i \approx_\lambda \varepsilon_{i+1}$ , where  $\varepsilon_i$  is defined as

$$\Pr \left[ r \leftarrow_{\$} \text{RND}(\mathcal{S}), (\text{crs}, z_{\mathcal{S}} \parallel \text{ts}) \leftarrow (\mathcal{S} \parallel \text{Ext}_{\mathcal{S}})(\tilde{z}_{\mathbf{R}}; r) : \begin{array}{l} \text{CV}(\tilde{z}_{\mathbf{R}}, \text{crs}) = 1 \wedge \mathcal{A}^{\mathcal{O}'_i(\cdot, \cdot)}(\tilde{z}_{\mathbf{R}}, \text{crs}, z_{\mathcal{S}}) = 1 \end{array} \right].$$

The oracles can be efficiently implemented given  $\text{crs}$  and  $\text{ts}$ . Since  $\varepsilon_0 = \varepsilon_0^{unb}$ ,  $q$  is a polynomial, and  $\varepsilon_0^{unb} = \varepsilon_0 \approx_\lambda \dots \approx_\lambda \varepsilon_{q(\lambda)} = \varepsilon_1^{unb}$ , we get that  $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$  and hence the claim holds.

(II) PERFECT SUB-ZK. As above, but assume  $q$  is the actual (possibly unbounded) number of queries. We get  $\varepsilon_0^{unb} = \varepsilon_0 = \dots = \varepsilon_{q(\lambda)} = \varepsilon_1^{unb}$ , and hence  $\varepsilon_0^{unb} = \varepsilon_1^{unb}$ , and the claim holds.  $\square$

In [Gro06], composable ZK was a stronger requirement than unbounded ZK since in the case of composable ZK, (i) the simulated CRS was required to be indistinguishable from the real CRS, and (ii) the adversary got access to  $\text{tc}$ . In the case of our Sub-ZK definitions, there is no such difference and it is easy to see that composable Sub-ZK and unbounded Sub-ZK are in fact equal notions.

## 5 From Trapdoor-Extractability to Sub-ZK

Next, we propose a general strategy for constructing Sub-ZK NIZK arguments: (i) we start with any secure NIZK argument in the CRS model, (ii) we construct a CV algorithm that checks the well-formedness of the CRS and additionally shows that the subverter knows a trapdoor. We formalize the property (ii) with the notion called trapdoor extractability, which intuitively gives us a stronger version of witness-sampleability [JR13]. We require that if  $\mathcal{S}$  generates a CRS  $\text{crs}$  accepted by CV then there exists a PPT extractor  $\text{Ext}_{\mathcal{S}}$  that (given the random coins of  $\mathcal{S}$ ) extracts a CRS trapdoor  $\text{tc}$  that corresponds to  $\text{crs}$ ; that is,  $\mathcal{K}_{\text{crs}}$  on input  $\text{tc}$  outputs  $\text{crs}$ . Here, it is important that  $\text{Ext}_{\mathcal{S}}$  returns  $\text{tc}$  not  $\text{ts}$ .

**Definition 8 (Trapdoor-Extractability).**  $\Psi$  has trapdoor-extractability for  $\text{RelGen}$ , if for any PPT subverter  $\mathcal{S}$  there exists a PPT  $\text{Ext}_{\mathcal{S}}$ , s.t. for all  $\lambda$  and  $\tilde{z}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ ,

$$\Pr \left[ r \leftarrow_{\$} \text{RND}(\mathcal{S}), (\text{crs}, z_{\mathcal{S}} \parallel \text{tc}) \leftarrow (\mathcal{S} \parallel \text{Ext}_{\mathcal{S}})(\tilde{z}_{\mathbf{R}}; r) : \left[ \text{CV}(\tilde{z}_{\mathbf{R}}, \text{crs}) = 1 \wedge \text{K}_{\text{crs}}(\tilde{z}_{\mathbf{R}}, \text{tc}) \neq \text{crs} \right] \approx_{\lambda} 0 \right. .$$

Intuitively, since we can extract the simulation trapdoor from  $\mathcal{S}$  and then use the simulator (from non-subversion zero knowledge) to simulate proofs, the above strategy seems enough for achieving Sub-ZK. We show in this section that *this strategy does not always work for arguments with statistical (or computational) zero-knowledge, but does work for perfect zero-knowledge arguments*. In particular, the latter result implies that any existing perfect NIZK argument can be made Sub-ZK just by constructing a trapdoor-extractable CV algorithm. Later we use this result to show that the new SNARK  $\Pi$  is Sub-ZK (see Corollary 1).

We first observe that the strategy does not always work if zero-knowledge is statistical but not perfect. In this case, there might be a subset  $\mathcal{C}_{\text{bad}}$  of CRSs such that the distribution of the real proof is far from that of the simulated one. According to the definition of statistical ZK, such a situation is fine as long as the probability of picking a CRS from  $\mathcal{C}_{\text{bad}}$  is small. Unfortunately, as we can see in the following example, the subverter might be able to pick a CRS from  $\mathcal{C}_{\text{bad}}$  with a much higher probability and make the simulator fail.

*Example 1.* Suppose  $\Psi = (\text{K}_{\text{tc}}, \text{K}_{\text{ts}}, \text{K}_{\text{crs}}, \text{CV}, \text{P}, \text{V}, \text{Sim})$  is a trapdoor-extractable and statistical (or perfect) zero-knowledge argument system. We use it to construct a new argument  $\Psi' = (\text{K}'_{\text{tc}}, \text{K}'_{\text{ts}}, \text{K}'_{\text{crs}}, \text{CV}', \text{P}', \text{V}, \text{Sim}')$  such that,

- $\text{K}'_{\text{tc}}(\tilde{z}_{\mathbf{R}})$  outputs with a negligible probability  $\eta(\lambda)$  a special symbol  $\text{tc} = \nabla$ . Otherwise, it outputs  $\text{tc} \leftarrow \text{K}_{\text{tc}}(\tilde{z}_{\mathbf{R}})$ .
- If  $\text{tc} = \nabla$ , then  $\text{K}'_{\text{ts}}(\tilde{z}_{\mathbf{R}}, \text{tc}) = \nabla$ , else  $\text{K}'_{\text{ts}}(\tilde{z}_{\mathbf{R}}, \text{tc}) = \text{K}_{\text{ts}}(\tilde{z}_{\mathbf{R}}, \text{tc})$ .
- If  $\text{tc} = \nabla$ , then  $\text{K}'_{\text{crs}}(\tilde{z}_{\mathbf{R}}, \text{tc}) = \nabla$ , else  $\text{K}'_{\text{crs}}(\tilde{z}_{\mathbf{R}}, \text{tc}) = \text{K}_{\text{crs}}(\tilde{z}_{\mathbf{R}}, \text{tc})$ .
- If  $\text{crs} = \nabla$ , then  $\text{CV}'(\tilde{z}_{\mathbf{R}}, \text{crs}) = 1$ , else  $\text{CV}'(\tilde{z}_{\mathbf{R}}, \text{crs}) = \text{CV}(\tilde{z}_{\mathbf{R}}, \text{crs})$ .
- If  $\text{crs} = \nabla$ , then  $\text{P}'(\tilde{z}_{\mathbf{R}}, \text{crs}_{\text{P}}, x, w) = w$ , else  $\text{P}'(\tilde{z}_{\mathbf{R}}, \text{crs}_{\text{P}}, x, w) = \text{P}(\tilde{z}_{\mathbf{R}}, \text{crs}_{\text{P}}, x, w)$ .
- If  $\text{ts} = \text{crs} = \nabla$ , then  $\text{Sim}'(\tilde{z}_{\mathbf{R}}, \text{crs}, \text{ts}, x) = \nabla$ , else  $\text{Sim}'(\tilde{z}_{\mathbf{R}}, \text{crs}, \text{ts}, x) = \text{Sim}(\tilde{z}_{\mathbf{R}}, \text{crs}, \text{ts}, x)$ .

Given extractor for  $\Psi$  we construct an extractor for  $\Psi'$  that returns  $\nabla$  if  $\text{crs} = \nabla$ . Since the extractor on  $\text{crs} = \nabla$  outputs  $\text{tc} = \nabla$ ,  $\Psi'$  is still trapdoor-extractable. It also preserves statistical zero knowledge since adversary's input on  $\Psi'$  differs from the input on  $\Psi$  only when  $\text{crs} = \nabla$ , which happens with negligible probability  $\eta(\lambda)$  only. Thus, the statistical distance between the real argument and the simulation in  $\Psi'$  is  $\approx \eta(\lambda) + \epsilon(\lambda)$  where  $\epsilon(\lambda)$  is the simulation error in  $\Psi$ .

However, Sub-ZK does not hold for any non-trivial relation since the subverter can always output  $\nabla$  as the CRS which makes  $\text{P}$  output the witness. We note that in the example above,  $\text{CV}$  could easily reject the bad CRS  $\nabla$ , but it seems unclear whether this approach is always possible.

Next, we focus on the case of perfect ZK. First, we prove that if  $\Psi$  has perfect ZK, then it has perfect ZK even respect to a fixed CRS. Let  $\text{Game}_{b, \mathcal{A}}^{\text{zk}}$  and

$\text{Game}_{b,\mathcal{S},\mathcal{A}}^{\text{sub-zk}}$  be the games respectively for statistically composable ZK (see Definition 6) and statistically composable Sub-ZK (see Definition 7) where  $\mathcal{A}$  is the adversary and  $\mathcal{S}$  is the subverter. We denote

$$\varepsilon_{b,\mathcal{A}|\text{crs}}^{\text{zk}} := \Pr[\text{Game}_{b,\mathcal{A}}^{\text{zk}} = 1 \mid \text{K outputs crs}] .$$

**Lemma 2.** *Assume  $\Psi$  is perfect composable zero-knowledge. Then for all  $\lambda$ ,  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ ,  $\text{crs}$  in  $\text{range}(\text{K}(\tilde{\mathbf{z}}_{\mathbf{R}}))$ , and adversary  $\mathcal{A}$ ,  $\varepsilon_{0,\mathcal{A}|\text{crs}}^{\text{zk}} = \varepsilon_{1,\mathcal{A}|\text{crs}}^{\text{zk}}$ .*

*Proof.* Suppose there exist  $\lambda$ ,  $\mathbf{R}$ ,  $\mathbf{z}_{\mathbf{R}}$ ,  $\text{crs}^* \in \text{range}(\text{K}(\tilde{\mathbf{z}}_{\mathbf{R}}))$ , and adversary  $\mathcal{A}$  as stated above, but  $\varepsilon_{0,\mathcal{A}|\text{crs}^*}^{\text{zk}} \neq \varepsilon_{1,\mathcal{A}|\text{crs}^*}^{\text{zk}}$ . We construct a new (possibly computationally unbounded) zero-knowledge adversary  $\mathcal{B}$  as follows: given the input  $(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \text{ts})$ , adversary  $\mathcal{B}$  runs  $\mathcal{A}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \text{ts})$  and returns its output. Given the input  $\pi_b$ ,  $\mathcal{B}$  proceeds as follows.

$$\boxed{\begin{array}{l} \mathcal{B}(\pi_b) \\ \hline \text{if crs} = \text{crs}^* \text{ then return } \mathcal{A}(\pi_b); \text{ else return } b' \leftarrow_{\mathcal{S}} \{0, 1\}; \text{ fi} \end{array}}$$

Observe that by the definition of  $\mathcal{B}$ ,  $\varepsilon_{b,\mathcal{B}|\text{crs}^*}^{\text{zk}} = \varepsilon_{b,\mathcal{A}|\text{crs}^*}^{\text{zk}}$  and for any  $\text{crs} \neq \text{crs}^*$  it holds that  $\varepsilon_{b|\text{crs}}^{\text{zk},\mathcal{B}} = 1/2$ . We may express  $\Pr[\text{Game}_b^{\text{zk}}(\mathcal{B}) = 1] = \sum_{\text{crs}} \Pr[\text{K outputs crs}] \cdot \varepsilon_{b,\mathcal{B}|\text{crs}}^{\text{zk}}$  and thus,

$$\begin{aligned} & |\Pr[\text{Game}_0^{\text{zk}}(\mathcal{B}) = 1] - \Pr[\text{Game}_1^{\text{zk}}(\mathcal{B}) = 1]| \\ &= \sum_{\text{crs}} \Pr[\text{K outputs crs}] \cdot |(\varepsilon_{0,\mathcal{B}|\text{crs}}^{\text{zk}} - \varepsilon_{1,\mathcal{B}|\text{crs}}^{\text{zk}})| \\ &= \Pr[\text{K outputs crs}^*] \cdot |(\varepsilon_{0,\mathcal{A}|\text{crs}^*}^{\text{zk}} - \varepsilon_{1,\mathcal{A}|\text{crs}^*}^{\text{zk}})| \neq 0 . \end{aligned}$$

This contradicts the assumption that  $\Psi$  has perfect composable zero-knowledge.  $\square$

The next theorem establishes correctness of our strategy for perfect ZK.

**Theorem 2.** *If  $\Psi$  is trapdoor-extractable and composable perfect ZK, then  $\Psi$  is statistical composable Sub-ZK.*

*Proof.* Let us fix a PPT subverter  $\mathcal{S}$ , an unbounded adversary  $\mathcal{A}$ , and  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ . According to the definition of trapdoor-extractability, there exists a PPT extractor  $\text{Ext}_{\mathcal{S}}^{\text{tc}}$ , such that for  $(\text{crs}, \mathbf{z}_{\mathbf{S}} \parallel \text{tc}) \leftarrow (\mathcal{S} \parallel \text{Ext}_{\mathcal{S}}^{\text{tc}})(\tilde{\mathbf{z}}_{\mathbf{R}}; r)$ , the probability  $\varepsilon_{\text{Ext}}(\lambda)$  that  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1 \wedge \text{K}_{\text{crs}}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{tc}) \neq \text{crs}$  is negligible over the random coins  $r \leftarrow_{\mathcal{S}} \text{RND}(\mathcal{S})$ . Let us denote  $\varepsilon_b^k = \Pr[\text{Game}_b^k = 1]$  for the games  $\text{Game}_b^k$  below. We will not write adversary as a subindex since in each of those games, adversaries are fixed.

$\text{Game}_b^0$ : This is the original Sub-ZK game  $\text{Game}_b^{\text{sub-zk}}$ , with  $\mathcal{S}$  and  $\mathcal{A}$  fixed as above. However, since we need to provide  $\text{ts}$  and not  $\text{tc}$  to the simulator, then we define the extractor as  $\text{Ext}_{\mathcal{S}}(\tilde{\mathbf{z}}_{\mathbf{R}}; r) := \text{K}_{\text{ts}}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{Ext}_{\mathcal{S}}^{\text{tc}}(\tilde{\mathbf{z}}_{\mathbf{R}}; r))$ .

**Game<sub>b</sub><sup>1</sup>**: We substitute  $\mathcal{A}$  with an adversary  $\mathcal{B}$  that does not get an auxiliary input  $z_S$  from the subverter but simulates it instead. Essentially, we are moving towards the adversary in the  $\text{Game}_b^{z^k}$  but actually even weaker since we also do not give  $ts$  as input to  $\mathcal{B}$ . First, when adversary  $\mathcal{B}$  is supposed to output  $(x, w)$ , then it now only gets the input  $(\tilde{z}_R, crs)$ . Adversary  $\mathcal{B}$  finds out all the possible valid auxiliary inputs  $z_S$  by running the  $\mathcal{S}$  with different random coins. Finally, it picks one of them randomly with the same distribution as  $\mathcal{S}$  and returns the same output as  $\mathcal{A}$ . More formally,  $\mathcal{B}$  works as follows.

```

B( $\tilde{z}_R, crs$ )
-----
 $S \leftarrow \emptyset$ ;
for  $r' \in \text{RND}(\mathcal{S})$  do
   $(crs', z'_S) \leftarrow \mathcal{S}(\tilde{z}_R; r')$ ;
  if  $crs = crs'$  then  $S \leftarrow S \sqcup \{z'_S\}$ ; // Disjoint union
endfor
 $z_S^* \leftarrow_{\$} S$ ;
return  $\mathcal{A}(\tilde{z}_R, crs, z_S^*)$ ;

```

Secondly, on the input  $\pi_b$ , the adversary  $\mathcal{B}$  makes the same choice as  $\mathcal{A}$  simply by returning the output  $\mathcal{A}(\pi_b)$ . Note that the distributions of  $(\tilde{z}_R, crs, z_S^*)$  and  $(\tilde{z}_R, crs, z_S)$ , with  $z_S$  from the subverter, are equal, hence the inputs to  $\mathcal{A}$  in  $\text{Game}_b^0$  and  $\text{Game}_b^1$  have the same distribution; this implies  $\varepsilon_b^0 = \varepsilon_b^1$ .

**Game<sub>b</sub><sup>2</sup>**: In this game, we substitute  $\text{Ext}_S$  in  $\text{Game}_b^1$  with an unbounded extractor  $\text{Ext}_S^*$  that always extracts a valid trapdoor from a valid CRS. First it runs  $\text{Ext}_S$  and if it fails, then tries to find a valid  $ts$  by brute-force.

```

ExtS*( $\tilde{z}_R; r$ )
-----
 $(crs, z_S \parallel tc) \leftarrow (\mathcal{S} \parallel \text{Ext}_S)(\tilde{z}_R; r)$ ;
if  $K_{crs}(\tilde{z}_R, tc) = crs$  then return  $ts \leftarrow K_{ts}(\tilde{z}_R, tc)$ ; fi
for  $tc' \in \text{range}(K_{tc}(\tilde{z}_R))$  do // If  $\text{Ext}_S$  failed
  if  $K_{crs}(\tilde{z}_R, tc') = crs$  then return  $ts' \leftarrow K_{ts}(\tilde{z}_R, tc')$ ; fi
endfor
return  $\perp$ ;

```

Since the output of  $\text{Ext}_S^*$  can differ from the output of  $\text{Ext}_S$  only if  $\text{Ext}_S$  fails,  $|\varepsilon_b^1 - \varepsilon_b^2| \leq \varepsilon_{\text{Ext}}$ .

Next, we analyse the success probability of  $\text{Game}_b^2$ . We consider the following two cases.

*Case 1:* Suppose  $\mathcal{S}$  outputs  $crs^* \in \text{range}(K(\tilde{z}_R))$ . Then the following holds: (i)  $\text{Ext}_S^*$  always recovers a valid  $ts$ , i.e., there exists  $tc \in \text{range}(K_{tc}(\tilde{z}_R))$ , such that  $K_{crs}(\tilde{z}_R, tc) = crs$  and  $ts = K_{ts}(\tilde{z}_R, tc)$ , (ii) even if there exists more than one valid  $ts$ , the simulator's output distribution is the same for all of them since otherwise one would contradict composable zero-knowledge. Thus,

$$\Pr[\text{Game}_b^2 = 1 \mid \mathcal{S} \text{ outputs } crs^*] = \Pr[\text{Game}_b^{z^k}(\mathcal{B}) = 1 \mid K \text{ outputs } crs^*] .$$

This allows us to conclude from Lemma 2 that for any  $\text{crs}^*$  in  $\text{range}(\mathbf{K}(\tilde{\mathbf{z}}_{\mathbf{R}}))$ ,

$$\Pr[\text{Game}_0^2 = 1 \mid \mathcal{S} \text{ outputs } \text{crs}^*] = \Pr[\text{Game}_1^2 = 1 \mid \mathcal{S} \text{ outputs } \text{crs}^*] .$$

*Case 2.* Suppose that  $\text{crs}^*$  is not in the range of  $\mathbf{K}(\tilde{\mathbf{z}}_{\mathbf{R}})$ . Then for all  $\text{tc}$ ,  $\mathbf{K}_{\text{crs}}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{tc}) \neq \text{crs}^*$ . Thus, the probability that the subverter  $\mathcal{S}$  outputs a  $\text{crs}^*$  outside of  $\text{range}(\mathbf{K}(\tilde{\mathbf{z}}_{\mathbf{R}}))$ , such that  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}^*)$  accepts, is bounded by  $\varepsilon_{\text{Ext}}$ .

Applying the law of total probability, we get

$$\begin{aligned} \varepsilon_b^2 &= \sum_{\text{crs}^* \in \text{range}(\mathcal{S}(\tilde{\mathbf{z}}_{\mathbf{R}}))} \Pr[\text{Game}_b^2 = 1 \mid \mathcal{S} \text{ outputs } \text{crs}^*] \cdot \Pr[\mathcal{S} \text{ outputs } \text{crs}^*] = \\ &= \sum_{\substack{\text{crs}^* \in \text{range}(\mathcal{S}(\tilde{\mathbf{z}}_{\mathbf{R}})) \\ \cap \text{range}(\mathbf{K}(\tilde{\mathbf{z}}_{\mathbf{R}}))}} \Pr[\text{Game}_b^2 = 1 \mid \mathcal{S} \text{ outputs } \text{crs}^*] \cdot \Pr[\mathcal{S} \text{ outputs } \text{crs}^*] + \\ &\quad + \Pr[\text{Game}_b^2 = 1 \wedge \mathcal{S} \text{ outputs } \text{crs}^* \notin \text{range}(\mathbf{K}(\tilde{\mathbf{z}}_{\mathbf{R}}))] . \end{aligned}$$

By using the knowledge from *Case 1* and *Case 2*, we get

$$\begin{aligned} |\varepsilon_0^2 - \varepsilon_1^2| &= |\Pr[\text{Game}_0^2 = 1 \wedge \mathcal{S} \text{ outputs } \text{crs}^* \notin \text{range}(\mathcal{S}(\tilde{\mathbf{z}}_{\mathbf{R}}))] \\ &\quad - \Pr[\text{Game}_1^2 = 1 \wedge \mathcal{S} \text{ outputs } \text{crs}^* \notin \text{range}(\mathcal{S}(\tilde{\mathbf{z}}_{\mathbf{R}}))] | \leq \varepsilon_{\text{Ext}} . \end{aligned}$$

Given that  $\varepsilon_b^0 = \varepsilon_b^1$ ,  $|\varepsilon_b^1 - \varepsilon_b^2| \leq \varepsilon_{\text{Ext}}$ , and  $|\varepsilon_0^2 - \varepsilon_1^2| \leq \varepsilon_{\text{Ext}}$ , it follows from triangle inequality that  $|\varepsilon_0^0 - \varepsilon_1^0| \leq 3 \cdot \varepsilon_{\text{Ext}}$  and thus  $\Psi$  has statistical Sub-ZK.  $\square$

## 6 GBGM And GBGM-H

**Preliminaries: Generic Bilinear Group Model.** In Section 8, we will prove that the new zk-SNARK is knowledge-sound in the generic bilinear group model with hashing (GBGM-H). In the current subsection, we will introduce the GBGM [Nec94, Sho97, Mau05, BBG05], by following the exposition in [Mau05]. After that, we will introduce the GBGM-H.

We start by picking an asymmetric bilinear group  $\text{pp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \text{Pgen}(1^\lambda, n)$ . Consider a black box  $\mathbf{B}$  that can store values from additive groups  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$  in internal state variables  $\text{cell}_1, \text{cell}_2, \dots$ , where for simplicity we allow the storage space to be infinite (this only increases the power of a generic adversary). The initial state consists of some values  $(\text{cell}_1, \text{cell}_2, \dots, \text{cell}_{|\text{inp}|})$ , which are set according to some probability distribution. Each state variable  $\text{cell}_i$  has an accompanying type  $\text{type}_i \in \{1, 2, T, \perp\}$ . We assume initially  $\text{type}_i = \perp$  for  $i > |\text{inp}|$ . The black box allows computation operations on internal state variables and queries about the internal state. No other interaction with  $\mathbf{B}$  is possible.

Let  $\Pi$  be an allowed set of computation operations. A computation operation consists of selecting a (say,  $t$ -ary) operation  $f \in \Pi$  together with  $t + 1$  indices  $i_1, i_2, \dots, i_{t+1}$ . Assuming inputs have the correct type,  $\mathbf{B}$  computes  $f(\text{cell}_{i_1}, \dots, \text{cell}_{i_t})$  and stores the result in  $\text{cell}_{i_{t+1}}$ . For a set  $\Sigma$  of relations, a query consists of selecting a (say,  $t$ -ary) relation  $\varrho \in \Sigma$  together with  $t$  indices  $i_1, i_2, \dots, i_t$ . Assuming inputs have the correct type,  $\mathbf{B}$  replies to the query with  $\varrho(\text{cell}_{i_1}, \dots, \text{cell}_{i_t})$ . In the GBGM, we define  $\Pi = \{+, \hat{e}\}$  and  $\Sigma = \{=\}$ , where

1. On input  $(+, i_1, i_2, i_3)$ : if  $\text{type}_{i_1} = \text{type}_{i_2} \neq \perp$  then set  $\text{cell}_{i_3} \leftarrow \text{cell}_{i_1} + \text{cell}_{i_2}$  and  $\text{type}_{i_3} \leftarrow \text{type}_{i_1}$ .
2. On input  $(\hat{e}, i_1, i_2, i_3)$ : if  $\text{type}_{i_1} = 1$  and  $\text{type}_{i_2} = 2$  then set  $\text{cell}_{i_3} \leftarrow \hat{e}(\text{cell}_{i_1}, \text{cell}_{i_2})$  and  $\text{type}_{i_3} \leftarrow T$ .
3. On input  $(=, i_1, i_2)$ : if  $\text{type}_{i_1} = \text{type}_{i_2} \neq \perp$  and  $\text{cell}_{i_1} = \text{cell}_{i_2}$  then return 1. Otherwise return 0.

Since we are proving lower bounds, we will give a generic adversary  $\mathcal{A}$  additional power. We assume that all relation queries are for free. We also assume that  $\mathcal{A}$  is successful if after  $\tau$  operation queries, he makes an equality query  $(=, i_1, i_2)$ ,  $i_1 \neq i_2$ , that returns 1; at this point  $\mathcal{A}$  quits. Thus, if  $\text{type}_i \neq \perp$ , then  $\text{cell}_i = F_i(\text{cell}_1, \dots, \text{cell}_{|\text{inp}|})$  for a polynomial  $F_i$  known to  $\mathcal{A}$ .

**GBGM-H.** By following [Bro01,SPMS02,BFS16], we enhance the power of generic bilinear group model [Nec94,Sho97,Mau05,BBG05]. Since the power of the generic adversary will increase, security proofs in the resulting *GBGM with hashing* will be somewhat more realistic than in the GBGM model.

Brown [Bro01] noted that since the generic group model assumes group encodings are random, it is easy in the generic model to generate a random group element by just sampling a random encoding. Bellare *et al.* [BFS16] made it more concrete by noting that it is known how to hash into elliptic curves [Ica09] and thus create group elements without knowing their discrete logarithms. However, it is not known how to create four elements  $[1]_\nu$ ,  $[a]_\nu$ ,  $[b]_\nu$ , and  $[ab]_\nu$  without knowing either  $a$  or  $b$ . The corresponding assumption, that may also be true in the case of symmetric pairings, was named *DH-KE(A)* in [BFS16] (see also [Dam92]).

Asymmetric pairings are much more efficient than symmetric pairings. If we work in the setting of type-III pairings [GPS08] where there is no efficient isomorphism either from  $\mathbb{G}_1$  to  $\mathbb{G}_2$  or from  $\mathbb{G}_2$  to  $\mathbb{G}_1$ , then clearly an adversary cannot, given  $[a]_\nu$  for  $\nu \in \{1, 2\}$  and a random unknown  $a$ , compute  $[a]_{3-\nu}$ . Thus, it seems reasonable to make a stronger assumption (that we call *BDH-KE*, a simplification of the asymmetric PKE assumption of [DFGK14]) that if an adversary outputs  $[a]_1$  and  $[a]_2$ , then she knows  $a$ . Really, since there is no polynomial-time isomorphism from  $\mathbb{G}_1$  to  $\mathbb{G}_2$  (or back), it seems to be natural to assume that one does not have to worry about an adversary knowing some trapdoor that would break the BDH-KE assumption. Since BDH-KE is not a falsifiable assumption, it does not have to hold for each type-III pairing. Instead, the BDH-KE assumption can be interpreted as a *stronger* definition of the type-III pairing setting. We formalize the added adversarial power as follows.

We give the generic model adversary an additional power to effectively create new indeterminates  $Y_i$  in groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$  (e.g., by hashing into elliptic curves), without knowing their values. We note since  $[Y]_1 \bullet [1]_2 = [Y]_T$  and  $[1]_1 \bullet [Y]_2 = [Y]_T$ , the adversary that has generated an indeterminate  $Y$  in  $\mathbb{G}_z$  can also operate with  $Y$  in  $\mathbb{G}_T$ . Formally, this means that  $\Pi$  will contain one more operation create, with the following semantics:

4. On input (create,  $i, t$ ): if  $\text{type}_i = \perp$  and  $t \in \{1, 2, T\}$  then set  $\text{cell}_i \leftarrow \mathbb{Z}_p$  and  $\text{type}_i \leftarrow t$ .

The semantics of `create` dictates that the actual value of the indeterminate  $Y_i$  is uniformly random in  $\mathbb{Z}_p$ , that is, the adversary cannot create indeterminates for which she does not know the discrete logarithm and that yet are not random. This assumption is needed for the lower bound on the generic adversary's time to be provable in Theorem 4. However, as pointed out subsequently in [Lip19], it is not necessary for  $Y_i$  to be uniformly random; high min-entropy suffices.

In the type-III setting, this semantics does *not* allow the adversary to create the same indeterminate  $Y_i$  in both groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$ ; she can only create a representation of a known to her integer in both groups. We formalize this by making the following Bilinear Diffie-Hellman Knowledge of Exponents (*BDH-KE*) assumption: if the adversary, given random generators  $[1]_1 \in \mathbb{G}_1$  and  $[1]_2 \in \mathbb{G}_2$ , can generate elements  $[\alpha_1]_1 \in \mathbb{G}_1$  and  $[\alpha_2]_2 \in \mathbb{G}_2$ , such that  $[1]_1 \bullet [\alpha_2]_2 = [\alpha_1]_1 \bullet [1]_2$ , then the adversary knows the value  $\alpha_1 = \alpha_2$ . To simplify the further use of the BDH-KE assumption in security reductions, we give the adversary access to  $\tilde{\mathbf{z}}_{\mathbf{R}} = (\mathbf{R}, \mathbf{z}_{\mathbf{R}}) \in \text{range}(\text{RelGen}(1^\lambda))$ . As before,  $\mathbf{z}_{\mathbf{R}} = \text{pp}$ , that is, it is the description of the bilinear group together with  $[1]_1$  and  $[1]_2$ .

**Definition 9 (BDH-KE).** *We say that Pgen is BDH-KE secure for RelGen if for any  $\lambda$ ,  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ , and PPT adversary  $\mathcal{A}$  there exists a PPT extractor  $\text{Ext}_{\mathcal{A}}$ , such that*

$$\Pr \left[ r \leftarrow \text{RND}(\mathcal{A}), ([\alpha_1]_1, [\alpha_2]_2 \parallel a) \leftarrow (\mathcal{A} \parallel \text{Ext}_{\mathcal{A}})(\tilde{\mathbf{z}}_{\mathbf{R}}; r) : \begin{array}{l} [\alpha_1]_1 \bullet [1]_2 = [1]_1 \bullet [\alpha_2]_2 \wedge a \neq \alpha_1 \end{array} \right] \approx_\lambda 0 .$$

The BDH-KE assumption is a simple, specific case of the PKE assumption as used in the case of asymmetric pairings say in [DFGK14]. In the PKE assumption of [DFGK14], adversary is given as an input the tuple  $\{([\chi^i]_1, [\chi^i]_2)\}_{i=0}^n$  for some  $n \geq 0$ , and it is assumed that if an adversary outputs  $([\alpha]_1, [\alpha]_2)$  then she knows  $(a_0, a_1, \dots, a_n)$ , such that  $\alpha = \sum_{i=0}^n a_i \chi^i$ . In our case,  $n = 0$ . BDH-KE can also be seen as an asymmetric-pairing version of the original KE assumption [Dam92]. Since GBGM-H is a relatively unknown model, we will next provide a direct proof that BDH-KE holds in the GBGM-H.

**Lemma 3.** *In the case of type-III pairings, BDH-KE is secure in the GBGM-H.*

*Proof (Sketch).* Let  $\mathcal{A}$  work be a BDH-KE adversary that, given  $\tilde{\mathbf{z}}_{\mathbf{R}}$  (that contains  $[1]_1$  and  $[2]_2$ ) and  $r$ , outputs with high probability  $([\alpha_1]_1, [\alpha_2]_2)$ , such that  $[\alpha_1]_1 \bullet [1]_2 = [1]_1 \bullet [\alpha_2]_2$ . Since we work in the GBGM-H, we know coefficients  $a_\nu$  and  $b_{\nu j}$ , such that  $\alpha_\nu = a_\nu + \sum b_{\nu j} Y_{\nu j}$ , where  $Y_{\nu j}$  are various indeterminates generated by  $\mathcal{A}$  in group  $\mathbb{G}_\nu$  by using elliptic curve hashing. Let  $\mathbf{Y}_\nu = (Y_{\nu 1}, Y_{\nu 2}, \dots)$ . If  $\mathcal{A}$  is successful then  $\alpha_1 = \alpha_2$  and thus the verification equation stipulates that  $V(\mathbf{Y}_1, \mathbf{Y}_2) = (a_1 + \sum b_{1j} Y_{1j}) - (a_2 + \sum b_{2j} Y_{2j}) = 0$  as a polynomial. From this, it follows that all coefficients of the polynomial  $V$  are equal to 0, thus  $a_1 = a_2 =: a$  and  $b_{\nu j} = 0$ . Hence,  $\alpha_1 = a = \alpha_2$ ; the required extractor  $\text{Ext}_{\mathcal{A}}$  just outputs  $a$ .  $\square$

In the case of type-I pairings, BDH-KE is similarly secure in the GBGM but it is not secure in the GBGM-H. Really, in the case of type-I pairings, we get that  $\alpha_\nu = a_\nu + \sum b_{\nu j} Y_j$  (since  $\mathbb{G}_1 = \mathbb{G}_2$ , any indeterminate  $Y_j$  can be used in both  $\mathbb{G}_1$  and  $\mathbb{G}_2$ ). But then from  $\alpha_1 = \alpha_2$  it only follows that  $V(\mathbf{Y}) = (a_1 + \sum b_{1j} Y_j) - (a_2 + \sum b_{2j} Y_j) = 0$  as a polynomial. By setting each coefficient  $V(\mathbf{Y})$  to be 0 we get that  $a := a_1 = a_2$  and  $b_{1j} = b_{2j}$ . This does not force  $b_{\nu j}$  to be 0, and thus we are unable to extract an *integer*  $a$  such that  $\alpha_1 = a = \alpha_2$ ; instead, we can only extract  $a$  and  $b$ , such that  $[\alpha_\nu]_\nu = a[1]_\nu + b[v]_\nu$  for a group element  $[v]_\nu$  generated by using elliptic curve hashing.

We think that for the following reasons, the BDH-KE assumption is more natural than the DH-KE assumption by Bellare *et al.* [BFS16] which states that if the adversary can create elements  $[\alpha_1]_\nu$ ,  $[\alpha_2]_\nu$  and  $[\alpha_1 \alpha_2]_\nu$  of the group  $\mathbb{G}_\nu$  then she knows either  $\alpha_1$  or  $\alpha_2$ .

First, the BDH-KE assumption is well suited to type-III pairings that are by far the most efficient pairings. The DH-KE assumption is tailored to type-I pairings. In the case of type-III pairings, the DH-KE assumption can still be used, but it results in inefficient protocols. For example in [BFS16], in security proofs the authors employ an adversary that extracts either  $\alpha_1$  or  $\alpha_2$ . Since it is not known a priori which value will be extracted, several elements in the argument system have to be duplicated, for the case  $\alpha_1$  is extracted and for the case  $\alpha_2$  is extracted.

Second, most of the efficient SNARKs are constructed to be sound and zero-knowledge in the (most efficient) type-III setting. While the SNARK of Groth [Gro16] is known to be sound in the case of both symmetric and asymmetric pairings, in the case of symmetric pairings, it will be much less efficient. It is only natural to keep the type-III setting to take advantage of already known efficient SNARKs. In the current paper, we have the best of both worlds. As in the case of [Gro16], we construct a SNARK that uses type-III pairings. On the one hand, we prove it to be Sub-ZK solely under the BDH-KE assumption. On the other hand, we prove that it is (adaptively) knowledge-sound in the GBGM-H, independently of whether one uses type-I, type-II, or type-III pairings. This provides a partial hedge against cryptanalysis: even if one were to later find an efficient isomorphism between  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , this would only break the Sub-ZK property of the new SNARK but leave the soundness property intact.

## 7 New Sub-ZK Secure SNARK

Consider a QAP instance  $\mathcal{Q} = (\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=1}^m)$ . The goal of the prover of a QAP argument of knowledge [GGPR13, Gro16] is to show that for public  $(A_1, \dots, A_{m_0})$ , the prover knows  $(A_{m_0+1}, \dots, A_m)$  and a degree  $\leq n-2$  polynomial  $h(X)$ , such that

$$h(X) = \frac{a(X)b(X) - c(X)}{\ell(X)}, \quad (2)$$

where  $a(X) = \sum_{j=1}^m A_j u_j(X)$ ,  $b(X) = \sum_{j=1}^m A_j v_j(X)$ ,  $c(X) = \sum_{j=1}^m A_j w_j(X)$ .

## 7.1 Construction

In Fig. 2, we describe two Sub-ZK SNARKs for RelGen that are closely based on the (non-subversion-resistant) SNARK by Groth from EUROCRYPT 2016 [Gro16]. Note that  $\mathsf{P}$  and  $\mathsf{V}$  are unchanged from [Gro16]. As always, we implicitly assume that each algorithm checks that their inputs belong to correct groups and that  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ . The first new SNARK  $\mathit{II}$ , which includes the grey-background elements in Fig. 2, satisfies trapdoor-extractability. Given that Groth’s SNARK satisfies perfect composable ZK,  $\mathit{II}$  has Sub-ZK by Theorem 2. The second new SNARK  $\mathit{II}^*$ , which excludes the grey-background elements, has a slightly shorter CRS and significantly more efficient CV algorithm. However, it does not satisfy the trapdoor-extractability, and its Sub-ZK property is, therefore, more cumbersome to prove. Thus, even though the result of Theorem 2 is convenient to use, it does not necessarily give the most efficient approach to achieve Sub-ZK.

Like [Gro16], the new SNARKs use five trapdoors,  $\chi$ ,  $\alpha$ ,  $\beta$ ,  $\gamma$ , and  $\delta$ . Here,  $\alpha$  and  $\beta$  (and the inclusion of  $\alpha\beta$  in the verification equation) will guarantee that  $[a]_1$ ,  $[b]_2$ , and  $[c]_1$  are computed by using the same coefficients  $A_i$ . The role of  $\gamma$  and  $\delta$  is to make the three products in the verification equation “independent” of each other. Due to the lack of space, we omit a more precise intuition behind Groth’s SNARK and refer an interested reader to [Gro16].<sup>5</sup>

As we already mentioned, the new Sub-ZK SNARKs are closely based on Groth’s zk-SNARK. Really, differences between Groth’s SNARK and the new subversion-resistant SNARKs can be summarized very briefly:

- (i) We add to the CRS some new elements that are needed for CV to work efficiently (see the variable  $\text{crs}_{\text{CV}}$  in Fig. 2):  $n + 3$  elements in the case of  $\mathit{II}^*$  and  $n + 5$  in the case of  $\mathit{II}$ .
- (ii) We divide the CRS generation algorithm into three algorithms,  $\mathsf{K}_{\text{tc}}$ ,  $\mathsf{K}_{\text{ts}}$ , and  $\mathsf{K}_{\text{crs}}$ . Groth’s CRS generation algorithm returns  $\mathsf{K}_{\text{crs}}(\tilde{\mathbf{z}}_{\mathbf{R}}, \mathsf{K}_{\text{tc}}(\tilde{\mathbf{z}}_{\mathbf{R}}))$  (minus the mentioned  $\text{crs}_{\text{CV}}$  part) as the CRS and  $\mathsf{K}_{\text{ts}}(\tilde{\mathbf{z}}_{\mathbf{R}}, \mathsf{K}_{\text{tc}}(\tilde{\mathbf{z}}_{\mathbf{R}}))$  as the simulation trapdoor.
- (iii) We describe an efficient CRS verification algorithms CV (see Fig. 2) for both versions of the argument. In the argument  $\mathit{II}$ , CV verifies the whole CRS, and we show that  $\text{tc}$  is extractable. In the optimized argument  $\mathit{II}^*$ , we will not verify elements of  $\text{crs}_{\text{V}}$  since the Sub-ZK proof does not depend on it, and we will only show that the simulation trapdoor  $\text{ts} = (\chi, \delta)$  can be extracted.

We prove the completeness and CRS-verifiability of our new SNARKs in the rest of this section. We postpone the proof of knowledge-soundness to Section 8 and proof of zero knowledge to Section 9. We analyze the efficiency in Section 10.

Note that Groth’s SNARK can be simulated in many different ways. As noticed by Fuchsbauer [Fuc18], it suffices to know  $(\chi, \delta)$  to be able to simulate;

<sup>5</sup> Lipmaa [Lip19] has showed recently how to minimally modify Groth’s SNARK so that it only has two trapdoors.

$K_{tc}(\bar{\mathbf{z}}_{\mathbf{R}})$ : Sample  $\mathbf{tc} = (\chi, \alpha, \beta, \gamma, \delta) \leftarrow_{\$} (\mathbb{Z}_p^* \setminus \{\omega^{i-1}\}_{i=1}^n) \times (\mathbb{Z}_p^*)^4$ ;  
 $K_{ts}(\bar{\mathbf{z}}_{\mathbf{R}}, \mathbf{tc} = (\chi, \alpha, \beta, \gamma, \delta))$ : **return**  $\mathbf{ts} \leftarrow (\chi, \alpha, \beta, \gamma, \delta)$ ;  
 $K_{crs}(\bar{\mathbf{z}}_{\mathbf{R}}, \mathbf{tc} = (\chi, \alpha, \beta, \gamma, \delta))$ : Set  $(\ell_i(\chi))_{i=1}^n \leftarrow \mathbf{compLag}(\chi, n)$ ; For  $j \in [1..m]$ , set  
 $u_j(\chi) \leftarrow \sum_{i=1}^n U_{ij} \ell_i(\chi)$ ,  $v_j(\chi) \leftarrow \sum_{i=1}^n V_{ij} \ell_i(\chi)$ ,  $w_j(\chi) \leftarrow \sum_{i=1}^n W_{ij} \ell_i(\chi)$ ; Let

$$\text{crs}_{\mathbf{P}} \leftarrow \left( [\alpha, \beta, \delta, ((u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta)_{j=m_0+1}^m, (\chi^i \ell(\chi)/\delta)_{i=0}^{n-2}]_1, \right. \\ \left. [(u_j(\chi), v_j(\chi))_{j=1}^m]_1, [\beta, \delta, (v_j(\chi))_{j=1}^m]_2 \right),$$

$$\text{crs}_{\mathbf{V}} \leftarrow ([((u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\gamma)_{j=1}^{m_0}]_1, [\gamma, \delta]_2, [\alpha\beta]_T),$$

$$\text{crs}_{\mathbf{CV}} \leftarrow ([\gamma, \chi, (\ell_i(\chi))_{i=1}^n]_1, [\alpha, \chi, \chi^{n-1}]_2);$$

**return**  $\text{crs} \leftarrow (\text{crs}_{\mathbf{CV}}, \text{crs}_{\mathbf{P}}, \text{crs}_{\mathbf{V}})$ ;  
 $K(\bar{\mathbf{z}}_{\mathbf{R}})$ : Let  $\mathbf{tc} \leftarrow K_{tc}(\bar{\mathbf{z}}_{\mathbf{R}})$ ;  $\mathbf{ts} \leftarrow K_{ts}(\bar{\mathbf{z}}_{\mathbf{R}}, \mathbf{tc})$ ;  $\text{crs} \leftarrow K_{crs}(\bar{\mathbf{z}}_{\mathbf{R}}, \mathbf{tc})$ ; **return**  $(\text{crs}, \mathbf{ts})$ ;  
 $\text{CV}(\bar{\mathbf{z}}_{\mathbf{R}}, \text{crs})$ :

1. Check  $[\gamma]_2 \neq^? [0]_2$ ; For  $\xi \in \{\chi, \alpha, \beta, \delta, \ell(\chi)/\delta\}$ : check  $[\xi]_1 \neq^? [0]_1$ ;
2. For  $\xi \in \{\chi, \alpha, \beta, \gamma, \delta\}$ : check  $[\xi]_1 \bullet [1]_2 \stackrel{?}{=} [1]_1 \bullet [\xi]_2$ ;
3. For  $i = 1$  to  $n - 2$ : check  $[\chi^i \ell(\chi)/\delta]_1 \bullet [1]_2 \stackrel{?}{=} [\chi^{i-1} \ell(\chi)/\delta]_1 \bullet [\chi]_2$ ;
4. Check  $[\ell(\chi)/\delta]_1 \bullet [\chi^{n-1}]_2 \stackrel{?}{=} [\chi^{n-2} \ell(\chi)/\delta]_1 \bullet [\chi]_2$ ;
5. Check  $[\ell(\chi)/\delta]_1 \bullet [\delta]_2 \stackrel{?}{=} [\chi]_1 \bullet [\chi^{n-1}]_2 - [1]_T$ ;
6. Check  $([\ell_i(\chi)]_1)_{i=1}^n$  is correctly computed by using `checkLag` in Fig. 3;
7. For  $j = 1$  to  $m$ :
  - (a) Check  $[u_j(\chi)]_1 \stackrel{?}{=} \sum_{i=1}^n U_{ij} [\ell_i(\chi)]_1$  and  $[v_j(\chi)]_1 \stackrel{?}{=} \sum_{i=1}^n V_{ij} [\ell_i(\chi)]_1$ ;
  - (b) Set  $[w_j(\chi)]_1 \leftarrow \sum_{i=1}^n W_{ij} [\ell_i(\chi)]_1$ ;
  - (c) Check  $[v_j(\chi)]_1 \bullet [1]_2 \stackrel{?}{=} [1]_1 \bullet [v_j(\chi)]_2$ ;
8. For  $j = 1$  to  $m_0$ : check  $[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\gamma]_1 \bullet [\gamma]_2 \stackrel{?}{=} [u_j(\chi)]_1 \bullet [\beta]_2 + [\alpha]_1 \bullet [v_j(\chi)]_2 + [w_j(\chi)]_1 \bullet [1]_2$ ;
9. For  $j = m_0 + 1$  to  $m$ : check  $[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1 \bullet [\delta]_2 \stackrel{?}{=} [u_j(\chi)]_1 \bullet [\beta]_2 + [\alpha]_1 \bullet [v_j(\chi)]_2 + [w_j(\chi)]_1 \bullet [1]_2$ ;
10. Check  $[\alpha]_1 \bullet [\beta]_2 \stackrel{?}{=} [\alpha\beta]_T$ ;

$\mathbf{P}(\bar{\mathbf{z}}_{\mathbf{R}}, \text{crs}_{\mathbf{P}}, \mathbf{x} = (A_j)_{j=1}^{m_0}, \mathbf{w} = (A_j)_{j=m_0+1}^m)$ :  $a^\dagger(X) \leftarrow \sum_{j=1}^m A_j u_j(X)$ ;  $b^\dagger(X) \leftarrow \sum_{j=1}^m A_j v_j(X)$ ;  $c^\dagger(X) \leftarrow \sum_{j=1}^m A_j w_j(X)$ ;  
 $h(X) = \sum_{i=0}^{n-2} h_i X^i \leftarrow (a^\dagger(X)b^\dagger(X) - c^\dagger(X))/\ell(X)$ ;  $r_a \leftarrow_{\$} \mathbb{Z}_p$ ;  $r_b \leftarrow_{\$} \mathbb{Z}_p$ ;  
 $[h(\chi)\ell(\chi)/\delta]_1 \leftarrow \sum_{i=0}^{n-2} h_i [\chi^i \ell(\chi)/\delta]_1$ ;  $[\mathbf{a}]_1 \leftarrow \sum_{j=1}^m A_j [u_j(\chi)]_1 + [\alpha]_1 + r_a [\delta]_1$ ;  
 $[\mathbf{b}]_2 \leftarrow \sum_{j=1}^m A_j [v_j(\chi)]_2 + [\beta]_2 + r_b [\delta]_2$ ;  $[\mathbf{c}]_1 \leftarrow r_b [\mathbf{a}]_1 + r_a (\sum_{j=1}^m A_j [v_j(\chi)]_1 + [\beta]_1) + \sum_{j=m_0+1}^m A_j [(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1 + [h(\chi)\ell(\chi)/\delta]_1$ ;  
**return**  $\pi \leftarrow ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{c}]_1)$ ;  
 $\mathbf{V}(\bar{\mathbf{z}}_{\mathbf{R}}, \text{crs}_{\mathbf{V}}, \mathbf{x} = (A_j)_{j=1}^{m_0}, \pi = ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{c}]_1))$ :  
 Check  $[\mathbf{a}]_1 \bullet [\mathbf{b}]_2 \stackrel{?}{=} [\mathbf{c}]_1 \bullet [\delta]_2 + (\sum_{j=1}^{m_0} A_j [(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\gamma]_1) \bullet [\gamma]_2 + [\alpha\beta]_T$ ;  
 $\text{Sim}(\bar{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \mathbf{ts} = (\chi, \delta), \mathbf{x})$ :  $\mathbf{a}^* \leftarrow_{\$} \mathbb{Z}_p$ ;  $\mathbf{b}^* \leftarrow_{\$} \mathbb{Z}_p$ ;  $[\mathbf{a}]_1 \leftarrow [\mathbf{a}^*]_1 + [\alpha]_1$ ;  $[\mathbf{b}]_2 \leftarrow [\mathbf{b}^*]_2 + [\beta]_2$ ;  
 $[\mathbf{c}]_1 \leftarrow (\mathbf{a}^* \mathbf{b}^* [1]_1 + \mathbf{a}^* [\beta]_1 + \mathbf{b}^* [\alpha]_1 - \sum_{j=1}^{m_0} A_j (u_j(\chi)[\beta]_1 + v_j(\chi)[\alpha]_1 + w_j(\chi)[1]_1))/\delta$ ;  
**return**  $\pi \leftarrow ([\mathbf{a}]_1, [\mathbf{b}]_2, [\mathbf{c}]_1)$ ;

**Fig. 2.** The Sub-ZK SNARKs  $\Pi$  (including grey-background items) and  $\Pi^*$  (without grey-background items) for relation `RelGen`

```

checkLag( $[\chi, (a_i)_{i=1}^n]_1, [1, \chi, \chi^{n-1}]_2, [1]_T$ )
//  $i = 1$ 
 $[\zeta]_T \leftarrow ([\chi]_1 \bullet [\chi^{n-1}]_2 - [1]_T)/n; [\omega']_2 \leftarrow [1]_2;$ 
if  $[\chi]_2 = [\omega']_2$  then check  $[a_1]_1 = [1]_1;$ 
else check  $[a_1]_1 \bullet ([\chi]_2 - [\omega']_2) \stackrel{?}{=} [\zeta]_T;$ 
for  $i = 2$  to  $n$  do
   $[\zeta]_T \leftarrow \omega[\zeta]_T; [\omega']_2 \leftarrow \omega[\omega']_2;$ 
  if  $[\chi]_2 = [\omega']_2$  then check  $[a_i]_1 \stackrel{?}{=} [1]_1;$ 
  else check  $[a_i]_1 \bullet ([\chi]_2 - [\omega']_2) \stackrel{?}{=} [\zeta]_T;$  endfor

```

**Fig. 3.** Checking that  $[a_i]_1 = [\ell_i(\chi)]_1$  for  $i \in [1..n]$

this is also the approach we take in the current paper. (In the conference version [ABLZ17], we used Groth’s original simulation algorithm.) Alternatively, simulation can be done based on  $(\gamma, \delta)$ ; in the case, the simulation will be more efficient but CV will be slightly slower. More precisely, one would not be able to use most of the optimizations we use in  $\Pi^*$ . Finally, Lipmaa [Lip19] proposed a version of Groth’s SNARK that only uses two trapdoors  $x$  (corresponds to  $\chi$  in the current paper) and  $y$  (conflates other trapdoors in the current paper); then, one can simulate by only knowing  $y$ .

## 7.2 Completeness and CRS-Verifiability

In the proof of CRS-verifiability, we need the following result that might be of independent interest.

**Lemma 4.** *Given  $([\chi, (a_i)_{i=1}^n]_1, [1, \chi, \chi^{n-1}]_2, [1]_T)$  as an input, checkLag in Fig. 3 checks that  $[a_i]_1 = [\ell_i(\chi)]_1$  for all  $i \in [1..n]$ . It can be implemented using  $n + 1$  pairings,  $n - 1$  exponentiations in  $\mathbb{G}_2$ , and  $n - 1$  exponentiations in  $\mathbb{G}_T$ .*

*Proof.* Recalling from Eq. (1) that  $\ell_i(X) = (X^n - 1)\omega^{i-1}/(n(X - \omega^{i-1}))$ , the proof is straightforward. Really, by induction on  $i$ , at any concrete value of  $i$ ,  $\zeta = (\chi^n - 1)\omega^{i-1}/n$  and  $\omega' = \omega^{i-1}$ . Thus, assuming that  $\chi \neq \omega^{i-1}$ ,  $[a_i]_1 \bullet ([\chi]_2 - [\omega']_2) = [\zeta]_T$  implies  $[a_i]_T = [(\chi^n - 1)\omega^{i-1}/(n(\chi - \omega^{i-1}))]_T = [\ell_i(\chi)]_T$ . If  $\chi = \omega^{i-1}$ , then  $\ell_i(\chi)$  should be 1 which follows from  $[a_i]_1 \stackrel{?}{=} [1]_1$ .  $\square$

In the new SNARKs, one has  $\chi^n \neq 0$  (due to the checks  $\ell(\chi)/\delta \neq 0$  and  $[\ell(\chi)]_1 \bullet [\delta]_2 = [\chi]_1 \bullet [\chi^n - 1]_2 - [1]_T$ ) and thus, strictly speaking, the two checks  $[a_i]_1 \stackrel{?}{=} [1]_1$  are not necessary. We left them in since they are computationally efficient (and simplify the statement of Lemma 4).

**Theorem 3.** *The arguments  $\Pi$  and  $\Pi^*$  are perfectly complete and perfectly CRS-verifiable.*

*Proof.* It suffices to give a proof for argument  $\Pi$  since the prover and the verifier algorithms are the same for both arguments and all verification equations of algorithm CV of  $\Pi^*$  are included in the CV of  $\Pi$ . Recall  $\mathbf{x} = (\chi, \alpha, \beta, \gamma, \delta)$ .

CRS-VERIFIABILITY: This can be established by a simple but tedious calculation. Basically, CV accepts due to the properties of the bilinear map, and due to the definitions of  $\ell(X)$  and  $\ell_i(X)$ . Let us do this for one of the more challenging equations.

Step 9 of CV holds since  $[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1 \bullet [\delta]_2 = [u_j(\chi)]_1 \bullet [\beta]_2 + [\alpha]_1 \bullet [v_j(\chi)]_2 + [w_j(\chi)]_1 \bullet [1]_2$  iff  $[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta \cdot \delta]_T = [u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)]_T$  which is a tautology. Other equalities can be shown to be satisfied as easily.

Note that the check  $\ell(\chi)/\delta \neq 0$  guarantees that  $\chi \neq \omega^{i-1}$  for any  $i$ .

COMPLETENESS: Let  $a^\dagger(\chi) = \sum_{j=1}^m A_j u_j(\chi)$ ,  $b^\dagger(\chi) = \sum_{j=1}^m A_j v_j(\chi)$ , and  $c^\dagger(\chi) = \sum_{j=1}^m A_j w_j(\chi)$ . In the honest case, see Fig. 2,  $[\mathbf{a}]_1 = [A(\mathbf{x})]_1$ ,  $[\mathbf{b}]_2 = [B(\mathbf{x})]_2$ , and  $[\mathbf{c}]_1 = [C(\mathbf{x})]_1$ , where  $A(\mathbf{x}) = a^\dagger(\chi) + \alpha + r_a \delta$ ,  $B(\mathbf{x}) = b^\dagger(\chi) + \beta + r_b \delta$ , and  $C(\mathbf{x}) = R(\mathbf{x}) + \sum_{j=m_0+1}^m (u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta + h(\chi)\ell(\chi)/\delta$  for  $R(\mathbf{x}) = r_b A(\mathbf{x}) + r_a (B(\mathbf{x}) - r_b \delta)$ . Clearly,

$$\begin{aligned} A(\mathbf{x})B(\mathbf{x}) &= R(\mathbf{x}) + \frac{a^\dagger(\chi)\beta + b^\dagger(\chi)\alpha}{\delta} + \frac{a^\dagger(\chi)b^\dagger(\chi)}{\delta} + \frac{\alpha\beta}{\delta} \\ &= R(\mathbf{x}) + \frac{a^\dagger(\chi)\beta + b^\dagger(\chi)\alpha + c^\dagger(\chi)}{\delta} + \frac{a^\dagger(\chi)b^\dagger(\chi) - c^\dagger(\chi)}{\delta} + \frac{\alpha\beta}{\delta} \\ &= R(\mathbf{x}) + \frac{a^\dagger(\chi)\beta + b^\dagger(\chi)\alpha + c^\dagger(\chi)}{\delta} + \frac{h(\chi)\ell(\chi)}{\delta} + \frac{\alpha\beta}{\delta} \\ &= R(\mathbf{x}) + \frac{\sum_{j=1}^m A_j (u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))}{\delta} + \frac{h(\chi)\ell(\chi)}{\delta} + \frac{\alpha\beta}{\delta} \\ &= C(\mathbf{x}) + \frac{\sum_{j=1}^{m_0} A_j (u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))}{\delta} + \frac{\alpha\beta}{\delta}, \end{aligned}$$

where the third equation holds since the prover is honest and thus  $a^\dagger(\chi)b^\dagger(\chi) - c^\dagger(\chi) - h(\chi)\ell(\chi)$ . Thus, the verifier accepts.  $\square$

## 8 Proof of Knowledge-Soundness

Since we are proving *non-subversion* knowledge-soundness (that is, we assume here that the CRS was correctly generated), the following security proof is similar to the corresponding proof in [Gro16]. The main differences are the need to take into account new elements  $\text{crs}_{\text{CV}}$  in the CRS and to incorporate new indeterminates  $Y_i$  created by the adversary with the elliptic curve hashing. Thus, the proof will be slightly (but not much) more complicated than the proof in [Gro16]. We give a proof only for the unoptimized argument  $\Pi$ . Knowledge-soundness of  $\Pi^*$  follows from the same proof since the adversary has access only to some subset of the CRS elements available to the adversary in  $\Pi$ .

Like [Gro16], we will prove adaptive knowledge-soundness even in the case when one uses symmetric pairings. The use of symmetric pairings means that

the generic adversary gets additional power compared to asymmetric case: in the asymmetric variant of the following theorem, the generic adversary will be allowed to create, without knowing their discrete logarithms, new indeterminates  $Y_i$  both in  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . Since this increases the power of the generic adversary, it provides some hedge against future cryptanalytic attacks that say make it possible to compute an efficient isomorphism between  $\mathbb{G}_1$  and  $\mathbb{G}_2$ .

**Theorem 4 (Knowledge-soundness).** *The argument  $\Pi$  from Section 7.1 is adaptively knowledge-sound in the GBGM-H even in the case of symmetric pairings. More precisely, any generic adversary attacking the knowledge-soundness of  $\Pi$  in the symmetric setting has complexity  $\Omega(\sqrt{p/n})$ .*

*Proof.* Assume the symmetric setting,  $\mathbb{G}_1 = \mathbb{G}_2$ . Let  $\mathbf{X} = (X, X_\alpha, X_\beta, X_\gamma, X_\delta)$  be the vector of indeterminates created by  $\mathsf{K}_{\text{tc}}$ . Let  $\mathbf{Y} = (Y_1, \dots, Y_q)^\top$ , for  $q \in \mathbb{N}_{\geq 0}$ , be the vector of indeterminates created by the generic adversary by using elliptic curve hashing.

The three elements output by a generic adversary are equal to  $[\mathbf{a}]_1 = [A(\mathbf{x}, \mathbf{v})]_1$ ,  $[\mathbf{b}]_2 = [B(\mathbf{x}, \mathbf{v})]_2$ , and  $[\mathbf{c}]_1 = [C(\mathbf{x}, \mathbf{v})]_1$  (where  $\mathbf{v}$  is a concrete value of  $\mathbf{Y}$ ), where, for  $T \in \{A, B, C\}$ ,

$$\begin{aligned} T(\mathbf{X}, \mathbf{Y}) = & T_\alpha X_\alpha + T_\beta X_\beta + T_\gamma X_\gamma + T_\delta X_\delta + T_c(X) + \\ & \sum_{j=1}^{m_0} T_j \cdot \frac{u_j(X)X_\beta + v_j(X)X_\alpha + w_j(X)}{X_\gamma} + \\ & \sum_{j=m_0+1}^m T_j \cdot \frac{u_j(X)X_\beta + v_j(X)X_\alpha + w_j(X)}{X_\delta} + \frac{T_h(X)\ell(X)}{X_\delta} + \sum_{i=1}^q T_{y_i} Y_i, \end{aligned}$$

where coefficients like  $T_\alpha$  and polynomials like  $T_c(X)$  are chosen by the adversary but known in the proof since we work in the generic model. Here,  $T_c(X)$  is a degree  $\leq n-1$  and  $T_h(X)$  is a degree  $\leq n-2$  polynomial. Thus,  $T(\mathbf{X}, \mathbf{Y}) \cdot X_\gamma X_\delta$  is a degree  $(n-2) + n-1 + 2 = 2n-1$  polynomial.

Since the only difference compared to the knowledge-soundness proof in [Gro16] is the addition of the terms  $\sum_{i=1}^q T_{y_i} Y_i$  to polynomials  $A, B, C$ , it suffices to show that  $T_{y_i} = 0$  for  $T \in \{A, B, C\}$  and  $i \in [1..q]$ . Then, the knowledge-soundness of  $\Pi$  follows from the knowledge-soundness of Groth's SNARK.

Motivated by the verification equation in Fig. 2, define

$$\begin{aligned} V(\mathbf{X}, \mathbf{Y}) := & A(\mathbf{X}, \mathbf{Y})B(\mathbf{X}, \mathbf{Y}) - C(\mathbf{X}, \mathbf{Y})X_\delta - \\ & \sum_{j=1}^{m_0} A_j^* (u_j(X)X_\beta + v_j(X)X_\alpha + w_j(X)) - X_\alpha X_\beta, \end{aligned}$$

where the Laurent polynomials  $A(\mathbf{X}, \mathbf{Y})$ ,  $B(\mathbf{X}, \mathbf{Y})$ , and  $C(\mathbf{X}, \mathbf{Y})$  are as given before. Here, for clarity (i.e., not to mix them up with adversarially chosen coefficients  $A_j$ ), we denote the public input by  $A_j^*$ .

The verification equation states that  $[V(\mathbf{x}, \mathbf{v})]_T = [0]_T$  and hence in the case of a generic adversary,  $V'(\mathbf{X}, \mathbf{Y}) := V(\mathbf{X}, \mathbf{Y}) \cdot X_\gamma^2 X_\delta^2 = 0$  as a polynomial and thus  $V'(\mathbf{X}, \mathbf{Y}) = 0$  as a Laurent polynomial. Write  $V(\mathbf{X}, \mathbf{Y}) =$

$\sum V_i(X)X_\alpha^{i_\alpha}X_\beta^{i_\beta}X_\gamma^{i_\gamma}X_\delta^{i_\delta}\prod_{j=1}^q Y_j^{i_j^*}$ , where  $\mathbf{i} = (i_\alpha, \dots, i_q^*)$ . Note that each coefficient  $V_i(X)$  of  $V(\mathbf{X}, \mathbf{Y})$  is a Laurent polynomial in  $X$ . Then,  $V(\mathbf{X}, \mathbf{Y}) = 0$  (as a Laurent polynomial) is equivalent to  $V_i(X) = 0$  (as a Laurent polynomial) as a polynomial for each  $\mathbf{i}$ .

First, the coefficient of  $X_\alpha^2$  of  $V(\mathbf{X}, \mathbf{Y})$  is  $V_{2,0,\dots,0}(X) = A_\alpha B_\alpha$ . Thus, from  $V(\mathbf{X}, \mathbf{Y}) = 0$  it follows that  $A_\alpha B_\alpha = 0$ . Since  $A(\mathbf{X}, \mathbf{Y})$  and  $B(\mathbf{X}, \mathbf{Y})$  play dual roles in the symmetric case, we can assume, w.l.o.g., that  $B_\alpha = 0$  (the same assumption was made in [Gro16] to simplify the proof).

Because  $B_\alpha = 0$ , the following claims hold:

1. from the coefficient of  $X_\alpha X_\beta$ ,  $A_\alpha B_\beta + A_\beta B_\alpha - 1 = 0$ . Thus,  $A_\alpha B_\beta = 1$  (and in particular, neither of  $A_\alpha$ ,  $B_\beta$  is equal to 0).
2. from the coefficient of  $X_\alpha Y_j$ ,  $j \in [1..q]$ ,  $A_\alpha B_{yj} + A_{yj} B_\alpha = 0$ . Thus,  $B_{yj} = 0$ .
3. from the coefficient of  $X_\beta Y_j$ ,  $j \in [1..q]$ ,  $A_\beta B_{yj} + A_{yj} B_\beta = 0$ . Thus,  $A_{yj} = 0$ .
4. from the coefficient of  $X_\delta Y_j$ ,  $j \in [1..q]$ ,  $C_{yj} = B_{yj} r_a + A_{yj} r_b$ . Thus,  $C_{yj} = 0$ . Since the coefficients  $A_{yj}$ ,  $B_{yj}$ , and  $C_{yj}$  are the only coefficients that are new compared to the knowledge-soundness proof of [Gro16], the rest of the current proof follows from the knowledge-soundness proof of [Gro16]. However, we will give the full proof for the sake of completeness.
5. from the coefficient of  $X_\beta^2$ ,  $A_\beta B_\beta = 0$  and thus  $A_\beta = 0$ .
6. from the coefficient of  $X_\beta X_\gamma$ ,  $A_\gamma B_\beta + A_\beta B_\gamma = 0$  and thus  $A_\gamma = 0$ .
7. from the coefficient of  $X_\alpha X_\gamma$ ,  $A_\gamma B_\alpha + A_\alpha B_\gamma = 0$  and thus  $B_\gamma = 0$ .
8. from the coefficients of  $1/X_\delta^2$ ,  $X_\alpha/X_\delta^2$ ,  $X_\beta/X_\delta^2$ ,  $X_\alpha X_\beta/X_\delta^2$ ,  $X_\alpha^2/X_\delta^2$  and  $X_\beta^2/X_\delta^2$ ,

$$\left( A_h(\chi)\ell(\chi) + \sum_{j=m_0+1}^m A_j(u_j(\chi)X_\beta + v_j(\chi)X_\alpha + w_j(\chi)) \right) \cdot \left( B_h(\chi)\ell(\chi) + \sum_{j=m_0+1}^m B_j(u_j(\chi)X_\beta + v_j(\chi)X_\alpha + w_j(\chi)) \right) = 0 .$$

Because of symmetry, we can assume that

$$A_h(\chi)\ell(\chi) + \sum_{j=m_0+1}^m A_j(u_j(\chi)X_\beta + v_j(\chi)X_\alpha + w_j(\chi)) = 0 .$$

But then the remaining terms in  $A_\alpha X_\alpha \cdot (B_h(\chi)\ell(\chi) + B_j(u_j(\chi)X_\beta + v_j(\chi)X_\alpha + w_j(\chi)))/X_\delta = 0$  show that also

$$B_h(\chi)\ell(\chi) + \sum_{j=m_0+1}^m B_j(u_j(\chi)X_\beta + v_j(\chi)X_\alpha + w_j(\chi)) = 0 .$$

9. from the coefficients of  $1/X_\gamma^2$ ,  $X_\alpha/X_\gamma^2$ ,  $X_\beta/X_\gamma^2$ ,  $X_\alpha X_\beta/X_\gamma^2$ ,  $X_\alpha^2/X_\gamma^2$  and  $X_\beta^2/X_\gamma^2$ , we get

$$\left( \sum_{j=1}^{m_0} A_j(u_j(\chi)X_\beta + v_j(\chi)X_\alpha + w_j(\chi)) \right) \cdot$$

$$\left( \sum_{j=1}^{m_0} B_j(u_j(\chi)X_\beta + v_j(\chi)X_\alpha + w_j(\chi)) \right) = 0 .$$

Due to symmetry, we may assume that

$$\sum_{j=1}^{m_0} A_j(u_j(\chi)X_\beta + v_j(\chi)X_\alpha + w_j(\chi)) = 0 .$$

But the remaining terms in  $A_\alpha X_\alpha \cdot (\sum_{j=1}^{m_0} B_j(u_j(X)X_\beta + v_j(X)X_\alpha + w_j(X)))/X_\gamma = 0$  show that also

$$\sum_{j=1}^{m_0} B_j(u_j(X)X_\beta + v_j(X)X_\alpha + w_j(X)) = 0 .$$

After this step, we know that

$$A(\mathbf{X}, \mathbf{Y}) = A_c(X) + A_\alpha X_\alpha + r_a X_\delta , \quad B(\mathbf{X}, \mathbf{Y}) = B_c(X) + B_\beta X_\beta + r_b X_\delta .$$

Define  $A_j^* := C_j$  for  $j > m_0$ . From the coefficient of  $X_\alpha$ , we get

$$A_\alpha B_c(X) = \sum_{j=1}^m A_j^* v_j(X) .$$

From the coefficient of  $X_\beta$ , we get

$$B_\beta A_c(X) = \sum_{j=1}^m A_j^* u_j(X) .$$

But then the coefficient of 1 is

$$A_c(X)B_c(X) - \ell(X)C_h(X) - \sum_{j=1}^m A_j^* w_j(X) ,$$

and thus

$$C_h(X) = (A_c(X)B_c(X) - \sum_{j=1}^m A_j^* w_j(X))/\ell(X) .$$

Thus,

$$C_h(X) = \frac{\left( \sum_{j=1}^m A_j^* u_j(X) \right) \left( \sum_{j=1}^m A_j^* v_j(X) \right) - \sum_{j=1}^m A_j^* w_j(X)}{\ell(X)}$$

and thus  $((\sum_{j=1}^m A_j^* u_j(X))(\sum_{j=1}^m A_j^* v_j(X)) - \sum_{j=1}^m A_j^* w_j(X))$  divides by  $\ell(X)$ , as required.

Next, we compute a lower bound to the efficiency of a generic adversary (this was not done in [Gro16], our bound clearly also holds in the case of

Groth's SNARK). Assume that after some  $\tau$  steps, the adversary has made a successful equality query  $(=, i_1, i_2)$ , i.e.,  $\text{cell}_{i_1} = \text{cell}_{i_2}$  for  $i_1 \neq i_2$ . Thus, she has found a collision  $D_1(\mathbf{x}, \mathbf{v}) = D_2(\mathbf{x}, \mathbf{v})$ , such that  $D_1(\mathbf{X}, \mathbf{Y}) \neq D_2(\mathbf{X}, \mathbf{Y})$ . Redefine  $D_j(\mathbf{X}, \mathbf{Y}) := D_j(\mathbf{X}, \mathbf{Y}) \cdot X_\gamma X_\delta$  (if  $\text{type}_{i_1} \in \{1, 2\}$ ) and  $D_j(\mathbf{X}, \mathbf{Y}) := D_j(\mathbf{X}, \mathbf{Y}) \cdot X_\gamma^2 X_\delta^2$  (if  $\text{type}_{i_1} = T$ ) for  $j \in \{1, 2\}$ , this guarantees that  $D_j(\mathbf{X}, \mathbf{Y})$  is a polynomial. Thus,

$$D_1(\mathbf{x}, \mathbf{v}) - D_2(\mathbf{x}, \mathbf{v}) \equiv 0 \pmod{p} . \quad (3)$$

Note that

- If  $\text{type}_{i_1} = 1$ , then  $\deg D_j(\mathbf{X}, \mathbf{Y}) \leq 2n - 1 =: d_1$ ,
- If  $\text{type}_{i_1} = 2$ , then  $\deg D_j(\mathbf{X}, \mathbf{Y}) \leq 2n - 1 =: d_2$ , and thus
- If  $\text{type}_{i_1} = T$ , then  $\deg D_j(\mathbf{X}, \mathbf{Y}) \leq 2 \cdot (2n - 1) = 4n - 2 =: d_T$ .

Clearly,  $\mathbf{x} = (\chi, \alpha, \beta, \gamma, \delta)$  is chosen uniformly random from  $(\mathbb{Z}_p^* \setminus \{\omega^{i-1}\}_{i=1}^n) \times (\mathbb{Z}_p^*)^4$ . Due to the assumption that the canonical values of  $Y_i$  are uniformly random in  $\mathbb{Z}_p$ ,  $\mathbf{v} = (v_1, v_2, v_3)$  is a uniformly random value in  $\mathbb{Z}_p^3$ . Hence, due to the Schwartz-Zippel lemma and since  $D_1(\mathbf{X}, \mathbf{Y}) \neq D_2(\mathbf{X}, \mathbf{Y})$  as a polynomial, Eq. (3) holds with probability at most  $\deg D_j(\mathbf{X}, \mathbf{Y}) / (p - 1) \leq d_{\text{type}_{i_1}} / (p - 1)$ . Clearly, an adversary working in time  $\tau$  can generate up to  $\tau$  new group elements. Then the probability that there exists a collision between any two of those group elements is upper bounded by  $\binom{\tau}{2} \cdot \deg D_j(\mathbf{X}, \mathbf{Y}) / (p - 1) \leq \binom{\tau}{2} \cdot d_{\text{type}_{i_1}} / (p - 1) \leq \tau^2 / 2 \cdot d_{\text{type}_{i_1}} / (p - 1)$ . Thus, a successful adversary on average requires time at least  $\tau$ , where  $\tau^2 \geq 2(p - 1) / d_{\text{type}_{i_1}} \geq 2(p - 1) / d_T = 2(p - 1) / (4n - 2)$ , to produce a collision. Simplifying, we get  $\tau = \Omega(\sqrt{p/n})$ .  $\square$

## 9 Proof of Statistical Subversion ZK

Next, we prove Sub-ZK of both arguments from Section 7.1. In the case of  $\Pi$ , we need to prove trapdoor-extractability, and then Sub-ZK follows from Theorem 2. However,  $\Pi^*$  is not trapdoor-extractable (we do not even verify  $\text{crs}_V$ ), and thus the same strategy will not work. Instead, we prove that it is Sub-ZK by showing that just enough trapdoors are extractable and that just enough of the CRS is verified by CV to simulate the proof.

### 9.1 Sub-ZK of $\Pi$

Before proving trapdoor-extractability, we first prove the following helpful lemma that has a simple but somewhat tedious proof.

**Lemma 5.** *Consider the argument  $\Pi$ . Let  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ , and let  $\text{crs}$  be any CRS such that  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1$ . Then,  $\text{crs} = \text{K}_{\text{crs}}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{tc})$  for a tc bijectively corresponding to  $[\chi, \alpha, \beta, \gamma, \delta]_1 \subset \text{crs}$ .*

*Proof.* We will consider each line in the construction of CV in Fig. 2 separately, and write down the corollary from that line. We note that the CRS verification equations in Fig. 2 are written as if the CRS was already correctly formed; e.g., there we have a check that  $[\beta]_1 \bullet [1]_2 = [1]_1 \bullet [\beta]_2$  which may fail (and then obviously there exists no such  $\beta$ ). However, before these equations are checked, it is, of course, not known that  $\beta$  on the left-hand side (LHS) and on the right-hand side (RHS) are equal. Thus, in the steps below, we use  $D_1$  as the temporary name of the yet-unestablished LHS variable and  $D_2$  as the temporary name of the yet-unestablished RHS variable. Only exception to this is the element  $[\ell(\chi)/\delta]_1$  for which we denote the yet-unestablished value by  $[D_{\ell/\delta}]_1$ .

We assume that  $\xi$  in  $[\xi]_1$  is already established for  $\xi \in \{\chi, \alpha, \beta, \gamma, \delta\}$ . We can do this since  $[\xi]_1$  information-theoretically fixes  $\xi$ .

1. For  $\xi \in \{\chi, \alpha, \beta, \gamma, \delta, D_{\ell/\delta}\}$ : after checking  $[\xi]_\nu \neq [0]_\nu$  (where  $\nu = 2$  in the case of  $\xi = \gamma$  and  $\nu = 1$  otherwise) we know that  $\xi \neq 0$ , and in particular this implies a bijection between a valid  $\text{tc} \in (\mathbb{Z}_p^* \setminus \{\omega^{i-1}\}) \times (\mathbb{Z}_p^*)^4$  and the value hidden in  $[\chi, \alpha, \beta, \gamma, \delta]_1$ .
2. For  $\xi \in \{\chi, \alpha, \beta, \gamma, \delta\}$ : from  $[\xi]_1 \bullet [1]_2 = [1]_1 \bullet [D_2]_2$  we get  $[\xi]_T = [D_2]_T$ . This implies  $[\xi - D_2]_T = [0]_T$ . Since  $[1]_T$  is not the zero element,  $[D_2]_2 = [\xi]_2$ .
3. As mentioned, we denote the supposed element  $[\ell(\chi)/\delta]_1$  by  $[D_{\ell/\delta}]_1$ ; we will verify it later in step 5.

For  $i = 1$  to  $n - 2$ : we can assume by induction that we have already established  $[\chi^{i-1}D_{\ell/\delta}]_1$ . Since  $[D_1]_T = [\chi^{i-1}D_{\ell/\delta}]_1 \bullet [\chi]_2$ , we get  $[D_1]_1 = [\chi^i D_{\ell/\delta}]_1$ .

4. Since  $[D_{\ell/\delta}]_1 \bullet [D_1]_2 = [\chi^{n-2}D_{\ell/\delta}]_1 \bullet [\chi]_2$ , we get  $[D_{\ell/\delta}D_1]_T = [\chi^{n-1}D_{\ell/\delta}]_T$ . Since  $D_{\ell/\delta} \neq 0$ , it is also invertible and we conclude that  $[D_1]_T = [\chi^{n-1}]_T$ , which implies  $[D_1]_2 = [\chi^{n-1}]_2$ .
5.  $[D_{\ell/\delta}]_1 \bullet [\delta]_2 = [\chi]_1 \bullet [\chi^{n-1}]_2 - [1]_T$  implies that  $[D_{\ell/\delta}]_1 = [(\chi^n - 1)/\delta]_1 = [\ell(\chi)/\delta]_1$ .
6. For  $i = 1$  to  $n$ :  $\omega' = \omega^{i-1}$  and  $\zeta = (\chi^n - 1)\omega^{i-1}/n$  in `checkLag` are computed by the CRS verifier. For  $\chi \neq \omega^{i-1}$ , the equation  $[D_1]_1 \bullet ([\chi]_2 - [\omega']_2) = [\zeta]_T$  implies  $[D_1]_1 = [\zeta/(\chi - \omega')]_1 = [(\chi^n - 1)\omega^{i-1}/(n(\chi - \omega^{i-1}))]_1 = [\ell_i(\chi)]_1$ . Although this is not relevant for our proof, we can also show that this holds for  $\chi = \omega^{i-1}$ : then, get  $D_1 \cdot (\chi - \omega^{i-1}) = \zeta$ , since  $\omega^{i-1}$  is a root of unity and  $\zeta = (\chi^n - 1)\omega^{i-1}/n$ , thus both sides of the equation are 0.
7. For  $j = 1$  to  $m$ :
  - (a)  $[D_1]_1 = \sum_{i=1}^n U_{ij}[\ell_i(\chi)]_1$  implies  $[D_1]_1 = [\sum_{i=1}^n U_{ij}\ell_i(\chi)]_1 = [u_j(\chi)]_1$ ,  
 $[D_1]_1 = \sum_{i=1}^n V_{ij}[\ell_i(\chi)]_1$  implies  $[D_1]_1 = [\sum_{i=1}^n V_{ij}\ell_i(\chi)]_1 = [v_j(\chi)]_1$ .
  - (b) This just sets  $[w_j(\chi)]_1 \leftarrow \sum_{i=1}^n W_{ij}[\ell_i(\chi)]_1$ .
  - (c)  $[v_j(\chi)]_1 \bullet [1]_2 = [1]_1 \bullet [D_2]_2$  implies  $[D_2]_2 = [v_j(\chi)]_2$ .
8. For  $j = 1$  to  $m_0$ :  $[D_1]_1 \bullet [\gamma]_2 = [u_j(\chi)]_1 \bullet [\beta]_2 + [\alpha]_1 \bullet [v_j(\chi)]_2 + [w_j(\chi)]_1 \bullet [1]_2$  implies  $[D_1]_1 = [(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\gamma]_1$ .
9. For  $j = m_0 + 1$  to  $m$ :  $[D_1]_1 \bullet [\delta]_2 = [u_j(\chi)]_1 \bullet [\beta]_2 + [v_j(\chi)]_1 \bullet [\alpha]_2 + [w_j(\chi)]_1 \bullet [1]_2$  implies  $[D_1]_1 = [(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1$ .
10.  $[\alpha]_1 \bullet [\beta]_2 = [D_2]_T$  implies  $[D_2]_T = [\alpha\beta]_T$ .

By direct observation, it is clear that we have now established all elements of the  $\text{crs}$  in Fig. 2 and thus  $\text{K}_{\text{crs}}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{tc}) = \text{crs}$ .  $\square$

Now we can prove trapdoor-extractability.

**Theorem 5.**  *$\Pi$  is trapdoor-extractable under the BDH-KE assumption.*

*Proof.* Let  $\mathcal{S}$  be a subverter, and let  $r \leftarrow_{\$} \text{RND}(\mathcal{S})$ . Let  $\xi \in \{\chi, \alpha, \beta, \gamma, \delta\}$ . Let  $\mathcal{S}^{\xi}(\tilde{\mathbf{z}}_{\mathbf{R}}; r)$  be as in Fig. 4. Since  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs})$  accepts,  $\text{crs}$  must contain  $([\xi]_1, [\xi]_2)$ . Hence, by the BDH-KE assumption, there exists a PPT extractor  $\text{Ext}_{\mathcal{S}^{\xi}}$ , such that if  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1$  then  $\xi \leftarrow \text{Ext}_{\mathcal{S}^{\xi}}(\tilde{\mathbf{z}}_{\mathbf{R}}; r)$  with an overwhelming probability.

$\mathcal{S}^{\xi}(\tilde{\mathbf{z}}_{\mathbf{R}}; r):$	$\text{Ext}_{\mathcal{S}}(\tilde{\mathbf{z}}_{\mathbf{R}}; r):$
$(\text{crs}, \text{z}_{\mathcal{S}}) \leftarrow \mathcal{S}(\tilde{\mathbf{z}}_{\mathbf{R}}; r);$	<b>for</b> $\xi \in \{\chi, \alpha, \beta, \gamma, \delta\}$ <b>do</b> $\xi \leftarrow \text{Ext}_{\mathcal{S}^{\xi}}(\tilde{\mathbf{z}}_{\mathbf{R}}; r);$ <b>endfor</b>
<b>return</b> $([\xi]_1, [\xi]_2);$	<b>return</b> $\text{tc} \leftarrow (\chi, \alpha, \beta, \gamma, \delta);$

**Fig. 4.** Algorithms used in extraction, where  $\xi \in \{\chi, \alpha, \beta, \gamma, \delta\}$

Finally, we construct the PPT extractor  $\text{Ext}_{\mathcal{S}}(\tilde{\mathbf{z}}_{\mathbf{R}}; r)$  in Fig. 4. By the BDH-KE assumption, if  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1$  then  $\text{tc} \leftarrow \text{Ext}_{\mathcal{S}}(\tilde{\mathbf{z}}_{\mathbf{R}}; r)$  satisfies  $\text{tc} \in \text{range}(\text{K}_{\text{tc}}(\tilde{\mathbf{z}}_{\mathbf{R}}))$  with an overwhelming probability. To prove trapdoor-extractability, assume that  $\text{crs}$  is returned by  $\mathcal{S}$  and  $\text{tc}$  is returned by  $\text{Ext}_{\mathcal{S}}$ . In addition, assume that  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1$ . Claim follows from Lemma 5.  $\square$

**Theorem 6.** [Gro16] *The argument  $\Pi$  has composable perfect ZK.*

The corollary below follows immediately from Theorem 2.

**Corollary 1.**  *$\Pi$  is statistically composable Sub-ZK under the BDH-KE assumption.*

Moreover, from Theorem 1 and Corollary 1 we get the following.

**Corollary 2.**  *$\Pi$  is statistically unbounded Sub-ZK under the BDH-KE assumption given that the adversary can only make polynomially many oracle calls.*

## 9.2 Sub-ZK of $\Pi^*$

We start by proving that trapdoors  $\chi$  and  $\delta$  are extractable in  $\Pi^*$ .

**Lemma 6.** *Consider  $\Pi^*$ . Let us assume that BDH-KE assumption holds. Then for any PPT subverter  $\mathcal{S}$ , there exists a PPT extractor  $\text{Ext}_{\mathcal{S}}$ , s.t. for all  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^{\lambda}))$ ,*

$$\Pr \left[ r \leftarrow_{\$} \text{RND}(\mathcal{S}), (\text{crs}, \text{z}_{\mathcal{S}} \parallel (\chi', \delta')) \leftarrow (\mathcal{S} \parallel \text{Ext}_{\mathcal{S}})(\tilde{\mathbf{z}}_{\mathbf{R}}; r) : \left[ \text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1 \wedge (\chi'[1]_1 \neq [\chi]_1 \vee \delta'[1]_1 \neq [\delta]_1) \right] \approx_{\lambda} 0 \right],$$

where  $[\chi]_1$  and  $[\delta]_1$  are elements from  $\text{crs}$ .

*Proof.* Proof follows the same ideas as proof of Theorem 5. Since  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs})$  accepts, then  $\text{crs}$  must contain  $([\chi]_1, [\chi]_2)$  and  $([\delta]_1, [\delta]_2)$ . Therefore under the BDH-KE assumption, there exist extractors  $\text{Ext}_{\mathcal{S}^\times}$  and  $\text{Ext}_{\mathcal{S}^\delta}$  that can output, with an overwhelming probability,  $\chi$  and  $\delta$  when given the same input and random coins as the subverter  $\mathcal{S}$ . Extractor  $\text{Ext}_{\mathcal{S}}$  can simply run  $\text{Ext}_{\mathcal{S}^\times}$  and  $\text{Ext}_{\mathcal{S}^\delta}$  and return their outputs  $(\chi', \delta')$ .  $\square$

**Theorem 7.** *The argument  $\Pi^*$  is statistically composable Sub-ZK under the BDH-KE assumption.*

*Proof.* Let us fix an arbitrary PPT subverter  $\mathcal{S}$  and let  $\text{Ext}_{\mathcal{S}}$  be defined as in Lemma 6. Fix  $\lambda$ ,  $\tilde{\mathbf{z}}_{\mathbf{R}} \in \text{range}(\text{RelGen}(1^\lambda))$ , and an adversary  $\mathcal{A}$ . We use the simulator  $\text{Sim}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \text{ts}, \mathbf{x})$  as defined in Fig. 2. As in Def. 7, assume  $r \leftarrow_{\mathcal{S}} \text{RND}(\mathcal{S})$ ,  $(\text{crs}, \mathbf{z}_{\mathcal{S}}) \leftarrow \mathcal{S}(\tilde{\mathbf{z}}_{\mathbf{R}}; r)$  such that  $\text{CV}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}) = 1$ . Assume that  $\pi_0 \leftarrow \text{P}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}_{\mathcal{P}}, \mathbf{x}, \mathbf{w})$  (case  $b = 0$ ) and  $\pi_1 \leftarrow \text{Sim}(\tilde{\mathbf{z}}_{\mathbf{R}}, \text{crs}, \text{ts}, \mathbf{x})$  (case  $b = 1$ ). It is sufficient to show that  $\pi_0$  and  $\pi_1$  have the same distribution.

**Case  $b = 0$ .** The honest prover creates  $[a]_1 \leftarrow \dots + r_a [\delta]_1$  and  $[b]_2 \leftarrow \dots + r_b [\delta]_2$  for  $r_a, r_b \leftarrow_{\mathcal{S}} \mathbb{Z}_p$ . Since  $\delta \neq 0$  (this is guaranteed by CV accepting crs) and the pairing is non-degenerate,  $[a]_1$  and  $[b]_2$  are uniformly random.

Since CRS elements used to construct  $[a]_1$ ,  $[b]_2$ , and  $[c]_1$  are well-formed (they belong to  $\text{crs}_{\mathcal{P}}$  and are thus checked by CV), then

$$[a]_1 \bullet [b]_2 - [c]_1 \bullet [\delta]_2 = \left[ \sum_{j=1}^{m_0} A_j(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)) + \alpha\beta \right]_T.$$

In particular, we can then express  $[c]_1$  as

$$[c]_1 \bullet [1]_2 = ([a]_1 \bullet [b]_2) / \delta - \left[ \frac{\sum_{j=1}^{m_0} A_j(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)) + \alpha\beta}{\delta} \right]_T. \quad (4)$$

**Case  $b = 1$ .** Let us assume that  $\text{Ext}_{\mathcal{S}}(\tilde{\mathbf{z}}_{\mathbf{R}}; r)$  outputs the correct trapdoor  $\text{ts} = (\chi, \delta)$ . This happens with an overwhelming probability, as we showed in Lemma 6. Here, the simulator picks  $[a]_1$  and  $[b]_2$  as uniformly random elements. Finally, the simulator picks  $[c]_1$  exactly according to Eq. (4) (although, the simulator computes it directly from  $[a]_1$ ,  $[b]_2$ ,  $\chi$ , and  $\delta$ ) and thus  $[c]_1$  has precisely the same distribution as the honest proof.  $\square$

The following result can again be obtained from Theorem 1.

**Corollary 3.**  *$\Pi^*$  is statistically unbounded Sub-ZK under the BDH-KE assumption given that the adversary can make only polynomially many calls to the proving oracle.*

## 10 Efficiency

We analyze the efficiency of  $\Pi^*$  in this section.  $\Pi$  differs only by having a slightly larger CRS and slower CV. Hence, we will not consider its efficiency separately.

**CRS Length.** Not counting  $\text{pp}$ , which contains  $([1]_1, [1]_2)$ , the number of CRS elements in different groups is given by the following table. Hence, the total size of the CRS is  $4m + 2n + 9$  group elements.

	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	Total
$\text{crs}_P$	$3m + n - m_0 + 2$	$m + 2$	0	$4m + n - m_0 + 4$
$\text{crs}_V$	$m_0$	2	1	$m_0 + 3$
$\text{crs}_{CV}$	$n + 1$	2	0	$n + 3$
Total	$3m + 2n + 3$	$m + 5$	1	$4m + 2n + 9$

One element (namely,  $[\delta]_2$ ) belongs both to  $\text{crs}_P$  and  $\text{crs}_V$ , and thus the numbers in the “total” row are not equal to the sum of the numbers in previous rows.

In Groth’s zk-SNARK [Gro16] the CRS consists of  $m + 2n$  elements of  $\mathbb{G}_1$  and  $n$  elements of  $\mathbb{G}_2$ . On top of it, we added  $n + 3$  group elements to make the CRS verification possible and also some elements to speed up the prover’s computation and the verifier’s computation; the latter elements can alternatively be computed from the rest of the CRS. Note that this comparison to [Gro16] is however not precise: also in [Gro16], depending on implementation, one might want to store  $[(\ell_i(\chi))_{i=1}^n]_1$  as a part of the CRS.

**CRS Generation: Computational Complexity.** Assume that  $\text{pp}$  has already been computed. We compute  $\text{crs}$  by first computing the discrete logarithms of all CRS elements, and then their versions in  $\mathbb{G}_\nu$ . One can evaluate  $u_j(\chi)$ ,  $v_j(\chi)$ , and  $w_j(\chi)$  for each  $j \in [1..m]$  in time  $\Theta(n)$  by using precomputed values  $\ell_i(\chi)$  for  $i \in [1..n]$  and the fact that the matrices  $U, V, W$  contain  $\Theta(n)$  non-zero elements. (The latter is a standard assumption, already made in [GGPR13].) The rest of the CRS can be computed efficiently by using straightforward algorithms.

By using the algorithm `compLag` in Fig. 1, the whole CRS generation algorithm is dominated by  $3m + 2n + 3$  exponentiations in  $\mathbb{G}_1$ ,  $m + 5$  exponentiations in  $\mathbb{G}_2$ , 1 exponentiation in  $\mathbb{G}_T$  (thus, one exponentiation per each CRS element), and  $\Theta(n)$  multiplications/divisions in  $\mathbb{Z}_p$ .

**CV’s Computational Complexity.** We assume that it is difficult to subvert  $\text{pp}$ ; this makes sense assuming that the SNARK uses a fixed bilinear group (say, the BLS12-381 curve). Consider the CRS verification algorithm in Fig. 2. It is clear that all other steps but Step 6 are efficient (computable in  $\Theta(n)$  cryptographic operations); this follows from the fact that  $U, V$ , and  $W$  are sparse. Computation in those steps is dominated by  $6m + 2n - 4m_0 + 6$  pairings. On top of it, one has to execute  $s(U) + s(V) + s(W)$  exponentiations in  $\mathbb{G}_1$ , where  $s(M)$  is the number of “large” (i.e., large enough so that exponentiating with them is expensive) entries in the matrix  $M$ . Often,  $s(M)$  is very small.

By using `checkLag`, one can check that  $[\ell_i(\chi)]_1$  has been correctly computed for all  $i \in [1..n]$  in  $n + 1$  pairings,  $n - 1$  exponentiations in  $\mathbb{G}_2$ , and  $n$  exponentiations in  $\mathbb{G}_T$ . Hence, the whole CRS verification algorithm is dominated by

$6m + 3n - 4m_0 + 7$  pairings,  $s(U) + s(V) + s(W)$  exponentiations in  $\mathbb{G}_1$ ,  $n - 1$  exponentiations in  $\mathbb{G}_2$ , and  $n$  exponentiations in  $\mathbb{G}_T$ .

**Prover’s and Verifier’s Complexity.** As in [Gro16], the prover’s computational complexity is dominated by the need to compute  $h(X)$  (3 interpolations, 1 polynomial multiplication, and 1 polynomial division; in total  $\Theta(n \log n)$  non-cryptographic operations in  $\mathbb{Z}_p$ ), followed by  $(n - 1) + s(A) + 1 + s(A) + 1 + s(A_1, \dots, A_{m_0}) \leq n + 2s(A) + s(A_1, \dots, A_{m_0}) + 1$  exponentiations in  $\mathbb{G}_1$  and  $s(A) + 1$  exponentiations in  $\mathbb{G}_2$ , where  $s(A)$  is again the number of large elements in  $A$  (i.e., large enough so that exponentiating with them would be expensive). This means that the prover’s computation is dominated by  $\Theta(n \log n)$  non-cryptographic operations and  $\Theta(n)$  cryptographic operations.

The verifier executes a single pairing equation that consists of 3 pairings and  $m_0$  exponentiations in  $\mathbb{G}_1$ . The exponentiations can be done offline since they do not depend on the argument  $\pi$  but only on the common input  $(A_1, \dots, A_{m_0})$ . Hence, the verifier’s computation is dominated by  $\Theta(m_0)$  cryptographic operations but her online computation is only dominated by 3 pairings.

The argument consists of 2 elements from  $\mathbb{G}_1$  and 1 element from  $\mathbb{G}_2$ .

## 10.1 Batched CV

One can speed up CV further by using batching [BGR98]. Namely, it can be shown that if  $\sum_{i=1}^s (t_i [a_i]_1) \bullet [b_i]_2 = \sum_{i=1}^s t_i [c_i]_T$  for uniformly randomly and independently chosen  $t_i$ , then w.h.p.,  $[a_i]_1 \bullet [b_i]_2 = [c]_T$  for each individual  $i \in [1..s]$ . The speed-up follows from the use of bilinear properties and from the fact that exponentiation is faster than pairing. Moreover, one can further slightly optimize this by assuming  $t_s = 1$  [Lip16,FLZ16] and sampling  $t_i$  from (say)  $[1..2^{80}] \subset \mathbb{Z}_p$ . The full batched version of CV is described in Fig. 5.

We use the following lemma which originally comes from [Lip16,FLZ16], but is closely based on the small exponents test from [BGR98].

**Lemma 7.** *Assume  $1 < t < q$ . Assume  $\mathbf{t}$  is a vector chosen uniformly random from  $\{1\} \times [1..t]^{k-1}$ ,  $\boldsymbol{\chi}$  is a vector of integers in  $\mathbb{Z}_q$ , and  $f_i$  are some polynomials of degree  $\text{poly}(\lambda)$ . If  $f_i(\boldsymbol{\chi})([1]_1 \bullet [1]_2) \neq [0]_T$  for some  $i$ , then  $\sum_{i=1}^k f_i(\boldsymbol{\chi})t_i \cdot ([1]_1 \bullet [1]_2) = [0]_T$  with probability  $\leq \text{poly}(\lambda) / t$ .*

By direct observation it is easy to confirm that each of the “batched” equations in Fig. 5 can be written in the form  $\sum_{i=1}^k f_i(\boldsymbol{\chi})t_i \cdot ([1]_1 \bullet [1]_2) = [0]_T$  such that  $f_i(\boldsymbol{\chi})([1]_1 \bullet [1]_2) \neq [0]_T$  is equivalent to some equation in Fig. 2. Moreover, in such a way, all the equations in the original CV algorithm are covered by some equation in the batched CV algorithm. Therefore by Lemma 7, we may conclude that if crs would not pass the original CV algorithm, then it would pass the batched CV algorithm at most with probability  $\text{poly}(\lambda) / t$ .

In the batched CV algorithm, we execute Alg. 6 instead of `checkLag`. This algorithm is dominated by 3 pairings,  $2n$  exponentiations in  $\mathbb{G}_1$ , and

CV( $\bar{z}_R$ , crs): // batched CV

1. Check  $[\gamma]_2 \neq^? [0]_2$ ; For  $\xi \in \{\chi, \alpha, \beta, \delta, \ell(\chi)/\delta\}$ : check  $[\xi]_1 \neq^? [0]_1$ ;
2. // For  $\xi \in \{\chi, \beta, \delta\}$ :
  - (a) Generate  $t_1, t_2 \leftarrow_s \{1, \dots, 2^\lambda\}$ ;
  - (b) Check  $(t_1[\chi]_1 + t_2[\beta]_1 + [\delta]_1) \bullet [1]_2 \stackrel{?}{=} [1]_1 \bullet (t_1[\chi]_2 + t_2[\beta]_2 + [\delta]_2)$ ;
3. // For  $i = 1$  to  $n - 2$ :
  - (a) Generate  $t_i \leftarrow_s \{1, \dots, 2^\lambda\}$  for  $i = 0$  to  $n - 3$ , then set  $t_{n-2} \leftarrow 1$ .
  - (b) Check  $(\sum_{i=1}^{n-2} t_i[\chi^i \ell(\chi)/\delta]_1) \bullet [1]_2 \stackrel{?}{=} (\sum_{i=1}^{n-2} t_i[\chi^{i-1} \ell(\chi)/\delta]_1) \bullet [\chi]_2$ ;
4. (a) Generate  $t \leftarrow_s \{1, \dots, 2^\lambda\}$ ;
- (b) Check  $[\ell(\chi)/\delta]_1 \bullet (t[\delta]_2 + [\chi^{n-1}]_2) \stackrel{?}{=} t([\chi]_1 \bullet [\chi^{n-1}]_2 - [1]_T) + [\chi^{n-2} \ell(\chi)/\delta]_1 \bullet [\chi]_2$ .
5. Check that  $[(\ell_i(\chi))_{i=1}^n]_1$  is correctly computed by using Alg. 6,
6. For  $j = 1$  to  $m$ :
  - (a) Generate  $t \leftarrow_s \{1, \dots, 2^\lambda\}$ ;
  - (b) Check  $t[u_j(\chi)]_1 + [v_j(\chi)]_1 \stackrel{?}{=} \sum_{i=1}^n (tU_{ij} + V_{ij})[\ell_i(\chi)]_1$ .
  - (c) Set  $[w_j(\chi)]_1 \leftarrow \sum_{i=1}^n W_{ij}[\ell_i(\chi)]_1$ ;
7. // For  $j = 1$  to  $m$ :
  - (a) Generate  $t_j \leftarrow_s \{1, \dots, 2^\lambda\}$  for  $j = 1$  to  $m - 1$ , then set  $t_m \leftarrow 1$ .
  - (b) Check that  $(\sum_{j=1}^m t_j[v_j(\chi)]_1) \bullet [1]_2 = [1]_1 \bullet (\sum_{j=1}^m t_j[v_j(\chi)]_2)$ .
8. // For  $j = m_0 + 1$  to  $m$ :
  - (a) Generate  $t_j \leftarrow_s \{1, \dots, 2^\lambda\}$  for  $j = m_0 + 1$  to  $m - 1$ , then set  $t_m \leftarrow 1$ ;
  - (b) Check  $(\sum_{j=m_0+1}^m t_j[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1) \bullet [\delta]_2 \stackrel{?}{=} (\sum_{j=m_0+1}^m t_j[u_j(\chi)]_1) \bullet [\beta]_2 + [\alpha]_1 \bullet (\sum_{j=m_0+1}^m t_j[v_j(\chi)]_2) + (\sum_{j=m_0+1}^m t_j[w_j(\chi)]_1) \bullet [1]_2$ ;

Fig. 5. Batched CV for  $II^*$

```

checklagrange-batched( $[\chi, (a_i)_{i=1}^n]_1, [1, \chi, \chi^{n-1}]_2, [1]_T$ )
 $\omega_0 \leftarrow 1/\omega$ ;
 $[a]_1 \leftarrow [0]_1; [b]_1 \leftarrow [0]_1; c \leftarrow 0$ ;
for  $i = 1$  to  $n$  do
   $t_i \leftarrow_s \{1, \dots, 2^\lambda\}; \omega_i \leftarrow \omega \omega_{i-1}$ ;
  if  $[\chi]_1 = [\omega_i]_1$  then check  $[a_i]_1 = [1]_1$ ;
   $[a]_1 \leftarrow [a]_1 + t_i[a_i]_1; [b]_1 \leftarrow [b]_1 + t_i \omega_i [a_i]_1; c \leftarrow c + t_i \omega_i$ ;
endfor Check that  $[a]_1 \bullet [\chi]_2 - [b]_1 \bullet [1]_2 = c/n \cdot ([\chi]_1 \bullet [\chi^{n-1}]_2 - [1]_T)$ ;

```

Fig. 6. Batched version of checking  $[a_i]_1 = [\ell_i(\chi)]_1$  for  $i \in [1..n]$

1 exponentiation in  $\mathbb{G}_T$ . The rest of the CV can be computed in 13 pairings,  $5m - 4m_0 + 2n + s(U + V) + s(W) - 6$  (mostly, small-exponent) exponentiations in  $\mathbb{G}_1$ , and  $2m + m_0 + 1$  (mostly, small-exponent) exponentiations in  $\mathbb{G}_2$ , and 1 exponentiation in  $\mathbb{G}_T$ . This makes, in total, 16 pairings,  $5m - 4m_0 + 4n + s(U + V) + s(W) - 6$  (mostly, small-exponent) exponentiations in  $\mathbb{G}_1$ ,  $2m + m_0 + 1$  (mostly, small-exponent) exponentiations in  $\mathbb{G}_2$ , and 2 expo-

mentation in  $\mathbb{G}_T$ . Since exponentiation with a short exponent is significantly less costly than a pairing, this will decrease the execution time of CV significantly.

We note that after taking batching into account, CV will become a probabilistic algorithm, and will accept incorrect CRSs with negligible probability. However, this will not affect our main results.

**Theorem 8.** *After batching CV, the SNARK  $\Pi^*$  from Section 7.1 is statistically composable Sub-ZK under the BDH-KE assumption.*

**Acknowledgment.** We thank Karim Baghery for his contribution to the conference version of the current paper. This work was partially supported by the Estonian Research Council grant (PRG49).

## References

- ABLZ17. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70700-6\_1.
- ALSZ20. Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michał Zając. On QA-NIZK in the BPK Model. In Aggelos Kiayias, editor, *PKC 2020*, volume 12110 of *LNCS*, pages 1–31, Edinburgh, UK, May 4–7, 2020. Springer, Cham. doi:https://doi.org/10.1007/978-3-030-45374-9\_20.
- BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005. doi:10.1007/11426639\_26.
- BCG<sup>+</sup>13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1\_6.
- BCG<sup>+</sup>14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014. doi:10.1109/SP.2014.36.
- BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014. doi:10.1145/2591796.2591859.
- BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. doi:10.1145/62212.62222.
- BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume

- 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016. doi: 10.1007/978-3-662-53890-6\_26.
- BGR98. Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch verification with applications to cryptography and checking. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191. Springer, Heidelberg, April 1998.
- Bow17. Sean Bowe. BLS12-381: New zk-SNARK Elliptic Curve Construction. Blog post, <https://blog.z.cash/new-snark-curve/>, last accessed in July, 2018, March 11, 2017.
- Bro01. Daniel R. L. Brown. The exact security of ECDSA. Contributions to IEEE P1363a, January 2001. <http://grouper.ieee.org/groups/1363/>.
- CGGM00. Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *32nd ACM STOC*, pages 235–244. ACM Press, May 2000. doi:10.1145/335305.335334.
- Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992. doi:10.1007/3-540-46766-1\_36.
- DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8\_28.
- DFKP13. George Danezis, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. pages 27–30, Berlin, Germany, November 4, 2013. ACM.
- DL08. Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP Proofs from an Extractability Assumption. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *Computability in Europe, CIE 2008*, volume 5028 of *LNCS*, pages 175–185, Athens, Greece, June 15–20, 2008. Springer, Heidelberg.
- EHK<sup>+</sup>13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge L. Villar. An Algebraic Framework for Diffie-Hellman Assumptions. In Ran Canetti and Juan Garay, editors, *CRYPTO (2) 2013*, volume 8043 of *LNCS*, pages 129–147, Santa Barbara, California, USA, August 18–22, 2013. Springer, Heidelberg.
- FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96881-0\_2.
- FLZ16. Prastudy Fauzi, Helger Lipmaa, and Michal Zajac. A shuffle argument secure in the generic model. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 841–872. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6\_28.
- FO18. Georg Fuchsbauer and Michele Orrù. Non-interactive Zaps of Knowledge. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 2018*, volume 10892 of *LNCS*, pages 44–62, Leuven, Belgium, July 2–4, 2018. Springer, Heidelberg.

- Fuc18. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5\_11.
- GGP10. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7\_25.
- GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9\_37.
- GKM<sup>+</sup>18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0\_24.
- GO94. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. doi:10.1007/BF00195207.
- GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006. doi:10.1007/11818175\_6.
- GPS08. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for Cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
- Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006. doi:10.1007/11935230\_29.
- Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8\_19.
- Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5\_11.
- GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. doi:10.1145/1993636.1993651.
- Ica09. Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316. Springer, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8\_18.
- JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42033-7\_1.

- KMS<sup>+</sup>16. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016. doi: 10.1109/SP.2016.55.
- Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9\_10.
- Lip13. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42033-7\_3.
- Lip16. Helger Lipmaa. Prover-efficient commit-and-prove zero-knowledge SNARKs. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 185–206. Springer, Heidelberg, April 2016. doi:10.1007/978-3-319-31517-1\_10.
- Lip19. Helger Lipmaa. Simulation-Extractable ZK-SNARKs Revisited. Technical Report 2019/612, IACR, May 31, 2019. <https://eprint.iacr.org/2019/612>, updated on 8 Feb 2020.
- Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.
- MR01. Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 542–565. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8\_32.
- Nec94. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.
- PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. doi:10.1109/SP.2013.47.
- Sch80. Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980.
- Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT’97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. doi:10.1007/3-540-69053-0\_18.
- SPMS02. Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 93–110. Springer, Heidelberg, August 2002. doi:10.1007/3-540-45708-9\_7.
- Zip79. Richard Zippel. Probabilistic Algorithms for Sparse Polynomials. In Edward W. Ng, editor, *EUROSM 1979*, volume 72 of *LNCS*, pages 216–226, Marseille, France, June 1979. Springer, Heidelberg.

## A Changes Compared to the Conference Version

This version of the paper contains several new results compared to the conference and fixes couple of small errors that occurred in the conference version. We briefly list them below.

1. Improvements in the definitional framework:
  - Divided subversion-completeness into completeness and CRS-verifiability.
  - In the composable Sub-ZK definition, we now allow the adversary to pick  $(x, w)$  adaptively based on the CRS.
  - Changed non-uniform adversary to uniform in the composable Sub-ZK definition (we provide a long discussion after the definition explaining this change).
  - The conference version contained an additional definition of Sub-ZK, which had a computationally unbounded extractor. We have omitted this definition in the current version since it is unclear whether it provides the correct privacy level. In general, the simulator should not be unbounded, and the extractor can be thought of as a part of the simulator.
  - Renamed Sub-GBGM to GBGM-H and statistical CRS trapdoor-extractability to trapdoor extractability.
2. Section 5 contains new generic results for achieving composable Sub-ZK. Namely, we prove that if an argument is perfectly zero-knowledge in the (trusted) CRS model and satisfies the notion of trapdoor extractability, then it has composable Sub-ZK.
3. We construct two versions of the Sub-ZK SNARK. The first version is trapdoor extractable and uses the general result from Section 5. The second version is more efficient but requires a different proof strategy. Compared to the construction in the conference version, CRS of the second argument is shorter by approximately  $n$  group elements, and the CV algorithm is also more efficient.
4. Fixed several small errors and updated the related/subsequent work section (see Section 2) with some of the intermediate results.