A Unified Framework for Non-Universal SNARKs

Helger Lipmaa

Simula UiB, Bergen, Norway

Abstract. We propose a general framework for non-universal SNARKs. It contains (1) knowledge-sound and non-black-box any-simulation-extractable (ASE), (2) zero-knowledge and subversion-zero knowledge SNARKs for the well-known QAP, SAP, QSP, and QSP constraint languages that all by design have *relatively* simple security proofs. The knowledge-sound zero-knowledge SNARK is similar to Groth's SNARK from EUROCRYPT 2016, except having fewer trapdoors, while the ASE subversion-zero knowledge SNARK relies on few additional conditions. We prove security in a weaker, more realistic version of the algebraic group model. We characterize SAP, SSP, and QSP in terms of QAP; this allows one to use a SNARK for QAP directly for other languages. Our results allow us to construct a family of SNARKs for different languages and with different security properties following the same proof template. Some of the new SNARKs are more efficient than prior ones. In other cases, the new SNARKs cover gaps in the landscape, e.g., there was no previous ASE or Sub-ZK SNARK for SSP or QSP.

Keywords: NIZK, QAP, QSP, SNARK, SAP, SSP, simulation-extractability, subversion zero-knowledge

1 Introduction

There are many different SNARKs [22,30,31,21,36,23] that differ in the target language and the security objectives. Common target languages correspond to specific quadratic constraint satisfaction systems, and the choice of language depends on the application. The languages QAP [21] and SAP [23,25] are useful when arguing about arithmetic circuits, while QSP [21,31] and SSP [13] are handy when arguing about Boolean circuits.[1] While QAP, providing efficient reductions to arithmetic circuits, is the most useful language in general applications like cryptocurrencies [8], other languages have their applications. In particular, SSP is widely used in applications where Boolean circuits come naturally like in, say, shuffle arguments, [16].

The choice of security objectives depends on the application. Knowledge-soundness is often sufficient, but simulation-extractability (SE) is needed to get

[1] Within this paper, we *always* (though implicitly, without mentioning it) refer to the "strong" versions of these languages as defined in [21]. First, such versions are most useful and needed in applications. Second, modern SNARKs like [23] and the ones discussed in the current paper are for "strong' variants.' We omit further discussions.

UC-security [12]. On the other hand, not having SE can be beneficial in applications that need malleability. Finally, security properties evolve. Both Sub-ZK (subversion zero-knowledge [7,1,17,3]; the argument stays zero-knowledge even if the CRS is not trusted) and non-black-box SE [25] for SNARKs were defined in 2017, after most of the mentioned zk-SNARKs were proposed. [1,17,3] showed that the most efficient known SNARK by Groth [23] is Sub-ZK.

This has resulted in an era of SNARK proliferation: there exist knowledge-sound SNARKs for the mentioned four languages, *some* of which are Sub-ZK or SE. Groth and Maller [25] proposed a non-black-box strong any-simulation-extractable (SASE) SNARK that is only slightly less efficient than Groth's SNARK [23]. Recall that knowledge-soundness means that a successful prover must know the witness, and SE means that the knowledge-soundness holds even if the prover had access to the simulation oracle, [37]. Dodis *et al.* [15] defined different variants of SE, see Section 2 for more information. Intuitively, in an ASE SNARK, one is allowed to maul an argument to a different argument for the same statement, while this is not allowed in a SASE SNARK. (Non-)black-box SE means that a (non-)black-box extractor extracts the witness. Black-box ASE is sufficient to obtain UC security.

However, the Groth-Maller SNARK is for the SAP language [23,25]. Since SAP has an efficient reduction from arithmetic circuits with squaring gates instead of general multiplication gates, the SNARK from [25] works with approximately two times larger circuits than SNARKs for the QAP language. While non-black-box SASE is insufficient to obtain UC security, it is a stronger security notion than knowledge-soundness. In particular, a much simpler transformation suffices to obtain UC security when one starts with non-black-box SE SNARKs [5]. Due to the use of SAP, this transformation is twice as costly as the ones starting from SE SNARKs for QAP. Other known simulation-extractable Sub-ZK SNARKs include [10], which works in the random oracle model, and [4], based on updatable signature schemes.

Recently, [6] showed that Groth's SNARK [23] satisfies the weaker non-black-box any-simulation-property ASE. As argued in [29,6], (black-box or non-black-box) ASE is sufficient in many applications. The only known SE SNARKs are for QAP and SAP, and no previous *efficient* SE or Sub-ZK SNARKs are known for SSP or QSP.

Finally, [1,3] proved the knowledge-soundness of Groth's SNARK in the generic group model (GGM) with hashing. The "with hashing" part means that one allows the adversaries to use (say) elliptic curve hashing to create random group elements without knowing their discrete logarithms. More modern knowledge-soundness (and ASE) proofs of SNARKs are given in the algebraic group model (AGM, [19]). Unfortunately, the AGM proof of Groth's SNARK in [19] does not allow the adversaries to hash. Proving the knowledge-soundness of Groth's SNARK in the AGM "with hashing" seems to be still an open problem.

We aim to consolidate SNARK research by investigating how the choice of security properties and target language influences an argument system's design. This is important as only a few researchers have in-depth knowledge of *secure*

SNARK design. It is easy for even well-established research groups to err in such an endeavor; see, for example, [35,11,20,18] for related cryptanalysis. The resulting complexity can be seen when following through the soundness proofs in say [23,25]. Each existing SNARK has a tailored construction with a tailored security proof in its specific security models, and even verifying all the security proofs for all mentioned SNARKs is probably well beyond the most talented cryptographer's capability.

This brings us to the main goal of this paper:

> *Construct a SNARK framework for a multitude of languages (e.g., QAP, SAP, QSP, and SSP) and satisfying a multitude of security objectives (knowledge-soundness vs. ASE, ZK vs. Sub-ZK) that allows for (1) a (relatively) simple security proof that can be easily modified to cover all the languages and security objectives, and (2) results in ASE and Sub-ZK SNARKs that are almost as efficient as the most efficient known knowledge-sound non-Sub-ZK SNARKs. Additionally, (3) prove their security in a realistic version of AGM "with hashing".*

Our Contributions. We propose a family of $2 \cdot 2 \cdot 4 = 16$ SNARKs that contains both knowledge-sound and ASE, and both ZK and Sub-ZK SNARKs, for all four mentioned languages (QAP, SAP, QSP, SSP). While the derivation of the first two SNARKs (namely, knowledge-sound no-Sub-ZK and its ASE version) is complicated, we obtain the other fourteen SNARKs with minor additional work. Thus, we obtain a framework for efficient random-oracle-less pairing-based SNARKs for both arithmetic and Boolean circuits. Previous knowledge-sound SNARKs for all four languages were each published in a separate paper, with corresponding ASE and Sub-ZK versions being proposed later, if at all.

The new knowledge-sound zk-SNARK $\mathsf{S}_{\mathsf{qap}}$ for QAP is similar to Groth's SNARK [23], except it has only two trapdoors instead of five. We replace 3 trapdoors with a well-chosen power of one trapdoor. After an even more careful choice of the powers, we also achieve CRS-verifiability [1,3] and thus Sub-ZK; otherwise, the Sub-ZK version is precisely the same and thus also as efficient. Unlike Groth, who proposed his SNARK without explaining how he arrived at the construction, we thoroughly motivate each step of it. This enables researchers aiming for a different goal to deviate from the construction at the appropriate point. Importantly, we provide a simpler knowledge-soundness proof.

To prove ASE, we observe that due to the structure of the new SNARKs, an ASE adversary can successfully use at most one simulation query answer in the forgery attempt. We show that if the adversary used one query answer, this was necessarily a SASE and not an ASE attack. The ASE of $\mathsf{S}_{\mathsf{qap}}$ follows. It is non-trivial that one-time ASE suffices. Moreover, unexpectedly, all powers of the trapdoor that result in $\mathsf{S}_{\mathsf{qap}}$ being knowledge-sound result in it also being ASE.

We prove knowledge-soundness and ASE in a more realistic version of the AGM. The knowledge-soundness proof in [23] was given in the generic group model, while [19] provided an AGM proof. However, [19] considers adversaries that are purely algebraic and in particular do not have a capability to create random group elements without knowing their discrete logarithms. In our proofs,

Table 1. Efficiency comparison of QAP/SAP/SSP/QSP-based random-oracle-less SNARKs Ψ. m (or $\tilde{m}$) and n (or $\tilde{n}$) denote the number of wires and gates (or constraints) in the solutions. "✓" ("$\approx$") means that the corresponding SNARK (its slight modification) is Sub-ZK, with a citation to the Sub-ZK construction if needed. "$\mathfrak{m}_\iota$" ("$\mathfrak{a}_\iota$") denotes scalar multiplication (addition) in group $\mathbb{G}_\iota$, "$\mathfrak{p}$" denotes pairing, and $\mathfrak{g}_\iota$ denotes the representation length of a $\mathbb{G}_\iota$ element in bits. In the case of $|\mathsf{crs}|$ and P's computation, we omit constant or m_0-dependent addends like $+(m_0+3)\mathfrak{g}_1$. We omit field operations and membership tests since they are dominated by significantly costlier group operations. $\mathsf{S}_{\mathsf{sap}}$, $\mathsf{S}_{\mathsf{ssp}}$, and $\mathsf{S}_{\mathsf{ssp}}$ are described in the full version, [33].

Ψ	security	$\|\mathsf{crs}\|$	P computation	$\|\pi\|$	V computation	Sub-ZK
QAP-based (arithmetic circuit, with n gates), $\tilde{m}=m$						
[23]	KS/ASE [6]	$(m+2n)\mathfrak{g}_1+n\mathfrak{g}_2$	$(m+3n)\mathfrak{m}_1+n\mathfrak{m}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$3\mathfrak{p}+m_0\mathfrak{m}_1$	✓[1,17,3]
$\mathsf{S}_{\mathsf{qap}}$ § 3	ASE	$(m+2n)\mathfrak{g}_1+n\mathfrak{g}_2$	$(m+3n)\mathfrak{m}_1+n\mathfrak{m}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$3\mathfrak{p}+m_0\mathfrak{m}_1$	✓
SAP-based (arithmetic circuit, with $\tilde{n}$ squaring gates): $u=v$, $\tilde{n}\approx 2n$, $\tilde{m}\approx 2m$						
[25]	SASE	$(\tilde{m}+2\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$(\tilde{m}+2\tilde{n})\mathfrak{m}_1+\tilde{n}\mathfrak{m}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$5\mathfrak{p}+m_0\mathfrak{m}_1$	$\approx$ [26]
$\mathsf{S}_{\mathsf{sap}}$	ASE	$(\tilde{m}+2\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$(\tilde{m}+2\tilde{n})\mathfrak{m}_1+\tilde{n}\mathfrak{m}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$3\mathfrak{p}+m_0\mathfrak{m}_1$	✓
SSP-based (Boolean circuit with n gates): $u=v=w$, $\tilde{n}=m+n$						
[13]	KS	$(m+\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$2m\mathfrak{a}_1+\tilde{n}\mathfrak{m}_1+m\mathfrak{a}_2$	$3\mathfrak{g}_1+1\mathfrak{g}_2$	$6\mathfrak{p}+m_0\mathfrak{a}_1$	–
$\mathsf{S}_{\mathsf{ssp}}$	ASE	$(m+2\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$2m\mathfrak{a}_1+\tilde{n}\mathfrak{m}_1+m\mathfrak{a}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$3\mathfrak{p}+m_0\mathfrak{a}_1$	✓
QSP-based (Boolean circuit with n gates): $w=0$, $\tilde{n}\approx 14n$ [31]						
[31]	KS	–	–	–	–	–
$\mathsf{S}_{\mathsf{qsp}}$	ASE	$(\tilde{m}+2\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$4\tilde{m}\mathfrak{a}_1+\tilde{n}\mathfrak{m}_1+\tilde{m}\mathfrak{a}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$3\mathfrak{p}+m_0\mathfrak{a}_1$	✓

the adversary has such a capacity. We consider this proof (and the corresponding realistic version of the AGM) to be another major contribution of this paper.

Based on an observation about algebraic relations between the languages, we modify $\mathsf{S}_{\mathsf{qap}}$ to cover SAP, QSP, and SSP. Hence, almost automatically, we obtain a family of knowledge-sound (or ASE), and zero-knowledge (or Sub-ZK) SNARKs for four different languages.

Table 1 compares the efficiency of random-oracle-less SNARKs. It is fair to compare SNARKs for the same language; a comparison of SNARKs for different languages (for example, QAP vs. SAP) has to account for the complexity of the reduction from circuits to these languages. Note that [31] described a reduction from Boolean circuits to QSP and a linear PCP [9] for QSP but did not describe a SNARK. In all constructions, most of the prover's scalar multiplications in Table 1 are multi scalar-multiplications. As seen from the table, the new ASE SNARK for SAP is more efficient than the (SASE) SNARK for SAP by Groth and Maller. No previous SE or Sub-ZK SNARKs were known for SSP or QSP, and Groth's SNARK for QAP was only proven to be ASE in [6].

1.1 Technical Overview

In Section 3, we propose a knowledge-sound zk-SNARK $\mathsf{S_{qap}}$ for QAP. The argument consists of evaluations[2] $[A(x,y)]_1, [B(x,y)]_2, [C_s(x,y)]_1$ of three bivariate polynomials $A(X,Y), B(X,Y), C_s(X,Y)$ at a random point (x,y). Here, $[A(x,y)]_1, [B(x,y)]_2$ commit to the vector of left and right inputs to all gates, while $[C_s(x,y)]_1$ combines a commitment to the vector of all output wires with the rest of the argument. The verifier checks that a bivariate polynomial $\mathcal{V}$, that depends in a known way on A, B, C_s, evaluates to 0 at the same point.

As in [23], we aim to make $[C_s(x,y)]_1$ to be computable only by the honest prover. The prover has access to the CRS that contains the evaluation of well-chosen polynomials at (x,y) in both $\mathbb{G}_1$ and $\mathbb{G}_2$. We optimize to get an efficient SNARK while not sacrificing (much) in the knowledge-soundness proof's simplicity. $\mathsf{S_{qap}}$ is very similar to Groth's SNARK [23]; however, it uses only two trapdoors instead of five. This distinction is important: in [23], only two out of five trapdoors are used in simulation; thus, the other three trapdoors seem not to be needed. In general, it is important to minimize the number of components to the bare minimum so that the importance of each component is well understood. In $\mathsf{S_{qap}}$, we use well-chosen powers of one trapdoor y as substitutes for four out of the five trapdoors of Groth's SNARK. (A similar technique to use one trapdoor to align "interesting" monomials together was used, e.g., in [24].)

Knowledge-Soundness Proof And A More Realistic Variant of The AGM. The knowledge-soundness proof is in the algebraic group model (AGM [19]). In the AGM, one considers algebraic adversaries that always know a linear relationship between their output and input group elements. As an important difference with the AGM of [19], we additionally allow the cheating prover to sample random elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. Such an extension of the generic group model is well-known, [7,1,3], but not established in the case of the AGM. It is also well understood why this extension is needed since otherwise, one can prove the security of false knowledge assumptions. Really, without this extension, one can prove that if an adversary on input $[1]_1$ outputs $[y]_1$, it must know y. This assumption does not hold since it is easy to generate random group elements by using hash-then-increment or elliptic curve hashing.

Fuchsbauer *et al.* [19] give an adversary $\mathcal{A}$ access to a programmable random oracle [34] $\mathcal{O}$. $\mathcal{A}$ can create a random group element by querying $\mathcal{O}$ that returns a uniformly random group element. In the security proof, one allows the reduction to program $\mathcal{O}$ by creating random group elements together with their discrete logarithms. Unfortunately, since the reduction knows the discrete logarithms, also in this model, one can prove the security of the above false knowledge assumption. We overcome this issue by using a different oracle simulation strategy by defining two adversaries (one for each trapdoor x and y) and by using two different oracle programming strategies. This results in the first known

[2] We use the by now standard additive bracket notation for group elements, by fixing first a bilinear group $\mathsf{p} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, and then denoting say $[a]_\iota = aP_\iota \in \mathbb{G}_\iota$ for a fixed generator $P_\iota \in \mathbb{G}_\iota$. See Section 2 for more information.

knowledge-soundness and ASE proof of (a version) of Groth's SNARK [23] in a variant of the AGM with hashing where false knowledge assumptions like the above cannot be proven. This result is of independent importance.

Choosing Powers of y. The way we choose the powers of y is interesting by itself. In the security proof, A, B, C_s are chosen maliciously and depend on additional indeterminates. Let Y be an indeterminate corresponding to y and $\boldsymbol{X}^*$ be the vector of all indeterminates, *except* Y, in the knowledge-soundness or ASE proof. $\boldsymbol{X}^*$ includes X (the indeterminate corresponding to x), indeterminates *created when the adversary samples* random group elements, and (in the case of ASE) indeterminates created by simulator queries. Since the adversary is algebraic, the polynomials $A(X)$, $B(X)$, and $C_s(X)$ belong to the span of the polynomials in the CRS, the random oracle answers, and (in the case of the ASE) the simulator answers. We use the AGM extractor to extract their maliciously chosen coefficients in this span, allowing us to recover the coefficients of the (Laurent) polynomial $\mathcal{V}$. The verification guarantees that $\mathcal{V}(\boldsymbol{x}^*, y) = 0$, where the trapdoor $\boldsymbol{x}^*$ instantiates the indeterminate $\boldsymbol{X}^*$.

The knowledge-soundness proof considers two cases, when $\mathcal{V}(\boldsymbol{X}^*, Y) = 0$ and $\mathcal{V}(\boldsymbol{X}^*, Y) \neq 0$ as a polynomial. Consider the first case. Then, $\mathcal{V}(\boldsymbol{X}^*, Y) = \sum \mathcal{V}_{Y^i}(\boldsymbol{X}^*) Y^i$ for known polynomials $\mathcal{V}_{Y^i}(\boldsymbol{X}^*)$, where i is a linear combination of the coefficients of a public but initially undetermined integer tuple $\boldsymbol{\Delta} = (\alpha, \beta, \gamma, \delta, \eta)$. We prove that an algebraic prover is honest iff $\mathcal{V}_{Y^i}(\boldsymbol{X}^*) = 0$ for six *critical values* i. (In Groth's security proof, the number of critical values is significantly larger.) We choose $\boldsymbol{\Delta}$ so that the corresponding six critical values i are distinct from each other and all other non-critical values j; in this case, we say that $\boldsymbol{\Delta}$ is *soundness-friendly*. Moreover, we choose $\boldsymbol{\Delta}$ so that the SNARK is relatively efficient. For example, we require that for all critical i, $|i|$ is as small as possible, and check if there is a way to make some non-critical values j to coincide (this can shorten the CRS).

Finding a suitable $\boldsymbol{\Delta}$, satisfying all the restrictions, is a moderately complex optimization problem. In particular, the number of non-zero coefficients of $V_{Y^i}(\boldsymbol{X}^*)$ (even in the knowledge-soundness proof and without allowing the adversary to create new indeterminates) is at least 30, depending on the SNARK. Because of the complexity of the problem, we used an exhaustive computer search to find $\boldsymbol{\Delta}$. Due to the use of exhaustive search, exponents in the resulting SNARKs (see Eq. (11) for a recommended value of $\boldsymbol{\Delta}$ and Eq. (12) for the description of the CRS when using this value of $\boldsymbol{\Delta}$) may look somewhat obscure. However, the soundness-friendliness of the results of the exhaustive search are easy to verify manually (intuitively, this corresponds to checking that when $\boldsymbol{\Delta}$ is instantiated as in Eq. (11), then the critical six entries in Eq. (10) are different from each other and all other entries). It is easy to find suboptimal choices of the exponents; however, such choices will usually not be sufficient for Sub-ZK. We feel that using exhaustive search adds to the strength of this paper.

Other Results. In Section 4, we prove that $\mathsf{S}_{\mathsf{qap}}$ is ASE. We use the same proof strategy as in the case of knowledge-soundness. By analyzing the coefficients of $\mathcal{V}$, we get that the ASE adversary can use the result of at most one simulation query in the forgery attempt. If she used none, ASE follows from the knowledge-soundness. If she used one, then, due to an easily satisfiable additional requirement on the QAP instance, she was performing a SASE attack that is not an attack in the sense of ASE. For this proof to work, one needs $\boldsymbol{\Delta}$ to satisfy additional restrictions on $\boldsymbol{\Delta}$; however, we will show that any soundness-friendly $\boldsymbol{\Delta}$ satisfies these requirements. Thus, *any* version of $\mathsf{S}_{\mathsf{qap}}$ that is knowledge-sound is ASE, modulo a small, easily satisfiable, technical restriction.

As we mentioned before, $\mathsf{S}_{\mathsf{qap}}$ is very similar to Groth's SNARK. Groth proved knowledge-soundness in the case of symmetric pairings, and this implies knowledge-soundness in the case of asymmetric pairing. Asymmetric pairings are much more efficient than symmetric pairings and thus strongly preferred in practice. We obtain a simpler direct knowledge-soundness proof by explicitly assuming that the pairing is asymmetric. One corollary of our knowledge-sound proof is the up to our knowledge novel observation that Groth's SNARK has a simple knowledge-soundness proof given that one uses asymmetric pairings. *Having simpler (or alternative) security proofs is important by itself due to the easier verifiability; simpler proofs can also result in the construction of other protocols.* We also use a more realistic variant of the AGM to prove knowledge-soundness. (The use of this variant of the AGM makes the security proof somewhat more complex again.) Moreover, we emphasize that the number of critical values i is much larger when one follows Groth's original proof.

Our goal was *not* to duplicate Groth's SNARK but to construct an efficient SNARK with a simple knowledge-soundness proof. Our exposition of the derivation of $\mathsf{S}_{\mathsf{qap}}$ can also be seen as an intuitive pedagogical re-derivation of (a slight variant of) the most efficient existing pairing-based SNARK.

We make $\mathsf{S}_{\mathsf{qap}}$ subversion-zero knowledge (Sub-ZK). According to the template of [1,3], we construct a public CRS verification algorithm that checks that the CRS corresponds to *some* trapdoor, and then use a knowledge assumption to recover the trapdoor and simulate the argument. For the CRS-verifiability, we restrict the choice of $\boldsymbol{\Delta}$ even more. This suffices: all new SNARKs are Sub-ZK when choosing $\boldsymbol{\Delta}$ carefully. We then use the standard BDH-KE [1,3] knowledge assumption to recover the trapdoor and simulate the argument.

	U	V	W
QAP			
SAP		$= U$	
SSP		$= U$	$= U$
QSP			$= 0$

Fig. 1. Algebraic relations between languages.

In the full version [33], we consider the languages SAP [23,25], SSP [13], and QSP [21,31]. We explain their algebraic relation to QAP, and use it to lift $\mathsf{S}_{\mathsf{qap}}$ to the setting of the corresponding languages. In the case of SSP and QSP, the algebraic relation is not obvious; we explain it in detail in the full version [33]. See Fig. 1 for a brief summary. This summary becomes clear later (e.g., QAP states that $U\mathbb{z} \circ V\mathbb{z} = W\mathbb{z}$ for an input-witness vector $\mathbb{z}$, while SAP states that $U\mathbb{z} \circ U\mathbb{z} = W\mathbb{z}$ since $U = V$; here, U, V, and W are relation-dependent matrices that characterize the languages as constraint satisfaction problems), but we decided to have it here for an early reference.[3]

Our SNARK for SAP (and SSP) has a slightly different ASE proof compared to the SNARK for QAP. Previous research handled all four languages separately, and our (simple) relations seem to be novel in the case of SSP and QSP. We propose the first known either Sub-ZK or ASE SNARKs for SSP and QSP, and more generally, for Boolean circuits. Importantly, the new Sub-ZK ASE SNARK for SSP is more efficient than the knowledge-sound non-Sub-ZK SNARK of [13].

This work supersedes [32]. While the idea of using only two trapdoors is already present in [32], there are too many changes to enlist.

1.2 Further Work

Applications. We concentrate on the construction of the SNARKs themselves and leave possible applications for future work. The most evident efficiency benefit is in the case of the SSP, where the verifier computes only 3 pairings instead of 6 in [13]. This may result in more efficient shuffle arguments [16] that rely on SNARKs for SSP. The ASE and Sub-ZK properties of the new SNARKs, on the other hand, have the potential to guarantee the same properties in similar applications. For example, given the new ASE SNARK for SSP, it may be possible (but we leave it to future work) to construct an ASE shuffle argument.

Universal SNARKs. There is an even more significant SNARK proliferation when one also considers universal SNARKs. Within this paper, we only study SNARKs with circuit-dependent CRSs. Universal SNARKs deserve their own several papers, especially since much less is known in that scenario. (E.g., efficient SE universal SNARKs have only been proposed in a recent eprint [28].) However, some of the results of the current paper (like the relation between QAP, SAP, SSP, and QSP) are also interesting in the context of universal SNARKs. We are not aware, e.g., of any *efficient* universal SNARKs for SSP.

2 Preliminaries

For a matrix $\boldsymbol{A}$, $\boldsymbol{A}_i$ denotes its ith row and $\boldsymbol{A}^{(j)}$ denotes its jth column. Let $\mathrm{vect}(\boldsymbol{A})$ be the vectorization of matrix $\boldsymbol{A} \in \mathbb{Z}_p^{n \times m}$, $\mathrm{vect}(\boldsymbol{A}) = (A_{11}, A_{12}, \ldots, A_{1m}, A_{21}, \ldots, A_{nm})$. $\mathbb{Z}_p^{(\leq d)}[X]$ denotes the set of univariate polynomials of degree $\leq d$ over $\mathbb{Z}_p$. PPT denotes probabilistic polynomial-time; $\lambda \in \mathbb{N}$ is the security parameter. Let $\mathsf{negl}(\lambda)$ be an arbitrary negligible function, and $\mathsf{poly}(\lambda)$ be an arbitrary polynomial function. We write $i \approx_\lambda j$ if $|i - j| \leq \mathsf{negl}(\lambda)$. For an algorithm $\mathcal{A}$, $\mathrm{im}(\mathcal{A})$ is the image of $\mathcal{A}$, that is, the set of valid outputs of $\mathcal{A}$. $\mathsf{RND}_\lambda(\mathcal{A})$ denotes the random tape of $\mathcal{A}$ (for given λ), and $r \leftarrow_{\$} \mathsf{RND}_\lambda(\mathcal{A})$ denotes the uniformly random choice of r from $\mathsf{RND}_\lambda(\mathcal{A})$. By $y \leftarrow \mathcal{A}(\mathbb{x}; r)$ we denote the fact that $\mathcal{A}$, given an input $\mathbb{x}$ and a randomizer r, outputs y.

[3] Our definitions of SSP and QSP are very slight variations of the standard SSP and QSP. They are functionally equivalent but, to our mind, slightly more elegant. See the full version [33] for more discussion.

Assume n is a power of two. Let ω be the nth primitive root of unity modulo p. (ω exists, given that $n \mid (p-1)$.) Then, $Z(X) := \prod_{i=1}^{n}(X-\omega^{i-1})$ is the unique degree n monic polynomial such that $Z(\omega^{i-1}) = 0$ for all $i \in [1,n]$. For $i \in [1,n]$, let $\ell_i(X)$ be the *ith Lagrange polynomial*, the unique degree $n-1$ polynomial such that $\ell_i(\omega^{i-1}) = 1$ and $\ell_i(\omega^{j-1}) = 0$ for $i \neq j$. Given $\chi \in \mathbb{Z}_p$, $\ell_i(\chi)$ for $i \in [1,n]$ can be computed efficiently. Clearly, $L_{\boldsymbol{k}}(X) := \sum_{i=1}^{n} k_i \ell_i(X)$ is the interpolating polynomial of $\boldsymbol{k}$ at points ω^{i-1}, with $L_{\boldsymbol{k}}(\omega^{i-1}) = k_i$.

Bilinear Groups. Let $n \in \mathbb{N}_{>0}$ be an upper bound of the size of a circuit in the SNARKs. A bilinear group generator $\mathsf{Pgen}(1^\lambda, n)$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three additive cyclic groups of prime order p, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear pairing. Assume $n \mid (p-1)$. As in say [7], we assume that Pgen is deterministic and cannot be subverted. (In practice, one can use a standardized curve.) We require the bilinear pairing to be Type-3; that is, there is no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. We use the standard bracket notation, writing $[c]_\iota$ to denote cP_ι where P_ι is a fixed generator of $\mathbb{G}_\iota$. Note that P_ι is not given in p. We denote $\hat{e}([a]_1, [b]_2)$ by $[a]_1 \bullet [b]_2$. We use freely the bracket notation together with matrix notation, for example, $\boldsymbol{AB} = \boldsymbol{C}$ iff $[\boldsymbol{A}]_1 \bullet [\boldsymbol{B}]_2 = [\boldsymbol{C}]_T$.

Assumptions. Let $\mathcal{T}_1, \mathcal{T}_2$ be sets of small integers. Pgen is $(\mathcal{T}_1, \mathcal{T}_2)$-*PDL (Power Discrete Logarithm) secure* if for any non-uniform PPT adversary $\mathcal{A}$,

$$\Pr[\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda, n), x \leftarrow\!\!\$\, \mathbb{Z}_p^* : \mathcal{A}(\mathsf{p}; [x^i : i \in \mathcal{T}_1]_1, [x^i : i \in \mathcal{T}_2]_2) = x] \approx_\lambda 0 \enspace .$$

If $\mathcal{T}_1 = [0,n]$, then we talk about the $(n, \mathcal{T}_2)$-PDL assumption. The case $\mathcal{T}_2 = [0,n]$ is dual.

The BDH-KE assumption [1,3] holds for Pgen, if for every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathsf{Ext}_\mathcal{A}$, such that

$$\Pr\left[\begin{matrix}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); r \leftarrow \mathsf{RND}_\lambda(\mathcal{A}); ([y]_1, [z]_2) \leftarrow \mathcal{A}(\mathsf{p}; r); \\ y^* \leftarrow \mathsf{Ext}_\mathcal{A}(\mathsf{p}; r) : y = z \wedge y^* \neq y\end{matrix}\right] = \mathsf{negl}(\lambda) \enspace .$$

BDH-KE is one of the weakest known knowledge assumptions in the asymmetric pairing-based setting.

Algebraic Group Model (AGM). AGM is a new idealized model [19] used to prove the security of a cryptographic assumption, protocol, or a primitive. In addition, [19] proposed to combine the random oracle (RO) model with the AGM, allowing the adversary to create random group elements. Essentially, in the AGM with random oracles, one assumes that each PPT algorithm $\mathcal{A}$ is algebraic in the following sense. Assume $\mathcal{A}$'s input includes $[\boldsymbol{x}_\iota]_\iota$ and no other elements from the group $\mathbb{G}_\iota$. Moreover, $\mathcal{A}$ has an access to random oracles $\mathcal{O}_\iota$, $\iota \in \{1,2\}$, such that $\mathcal{O}_\iota$ samples and outputs a random element $[q_{\iota k}]_\iota$ from $\mathbb{G}_\iota$. The oracle access models the ability of $\mathcal{A}$ to create random group elements without knowing their discrete logarithms $q_{\iota k}$. However, a reduction can program [34] the random oracle so that it knows $q_{\iota k}$. Intuitively, one assumes that if $\mathcal{A}$ outputs group elements $[\boldsymbol{y}_\iota]_\iota$, then $\mathcal{A}$ knows matrices $\boldsymbol{N}_\iota$ and $([\boldsymbol{q}_1, \boldsymbol{q}_2]_1)$, such that $\boldsymbol{y}_\iota = \boldsymbol{N}_\iota \left(\begin{smallmatrix} \boldsymbol{x}_\iota \\ \boldsymbol{q}_\iota \end{smallmatrix}\right)$ while the reduction also knows $\boldsymbol{q}_\iota$.

Formally, a PPT algorithm $\mathcal{A}$ is *(*Pgen*-)algebraic* if there exists an efficient extractor $\mathsf{Ext}_\mathcal{A}$, such that for any PPT-sampleable distribution family $\mathcal{D} = (\mathcal{D}_\mathsf{p})_{\mathsf{p} \in \mathsf{Pgen}(1^\lambda)}$, $\mathsf{Adv}^{\mathrm{agm}}_{\mathsf{Pgen}, \mathcal{D}, \mathcal{A}, \mathsf{Ext}_\mathcal{A}}(\lambda) :=$

$$\Pr \left[\begin{array}{l} \mathsf{p} \leftarrow_\$ \mathsf{Pgen}(1^\lambda); \mathbb{x} = ([\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2) \leftarrow_\$ \mathcal{D}_\mathsf{p}; r \leftarrow_\$ \mathsf{RND}_\lambda(\mathcal{A}); \\ ([\boldsymbol{y}_1]_1, [\boldsymbol{y}_2]_2) \leftarrow_\$ \mathcal{A}^{(\mathcal{O}_1, \mathcal{O}_2)}(\mathbb{x}; r); (\boldsymbol{N}_1, \boldsymbol{N}_2) \leftarrow \mathsf{Ext}_\mathcal{A}(\mathbb{x}; r) : \\ (\boldsymbol{y}_1 \neq \boldsymbol{N}_1 \left(\begin{smallmatrix} \boldsymbol{x}_1 \\ \boldsymbol{q}_1 \end{smallmatrix}\right) \vee \boldsymbol{y}_2 \neq \boldsymbol{N}_2 \left(\begin{smallmatrix} \boldsymbol{x}_2 \\ \boldsymbol{q}_2 \end{smallmatrix}\right)) \end{array} \right] = \mathsf{negl}(\lambda) \enspace .$$

$\mathcal{O}_\iota$, $\iota \in \{1, 2\}$ is an oracle that samples and returns a random element from $\mathbb{G}_\iota$. $[\boldsymbol{q}_\iota]_\iota$ is the list of all elements output by $\mathcal{O}_\iota$. We denote the version of the AGM where the reduction can program $\mathcal{O}_\iota$, by first sampling a random element $q_{\iota k}$ from $\mathbb{Z}_p$ and then returning $\boldsymbol{q}_{\iota k}$, as RO$_\mathsf{fkl}$-AGM. The RO$_\mathsf{fkl}$-AGM states that, given such programmable random oracles, $\mathsf{Adv}^{\mathrm{agm}}_{\mathsf{Pgen}, \mathcal{D}, \mathcal{A}, \mathsf{Ext}_\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ for any PPT-sampleable $\mathcal{D}$ and PPT algebraic $\mathcal{A}$.

SNARKs. Let RG be a relation generator, such that $\mathsf{RG}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\mathbb{x}, \mathbb{w})\}$ together with auxiliary information p. Here, $\mathbb{x}$ is a statement, and $\mathbb{w}$ is a witness. We assume that λ is explicitly deductible from the description of $\mathbf{R}$. Intuitively, $(\mathsf{p}, \mathbf{R})$ is the common auxiliary input to the honest parties, the adversary, and the corresponding extractor. We assume that $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda, n)$ for a well-defined n. (Recall that the choice of p and thus of the groups $\mathbb{G}_\iota$ depends on n and that p is not subvertible.) Let $\mathcal{L}_\mathbf{R} = \{\mathbb{x} : \exists \mathbb{w} \text{ such that } (\mathbb{x}, \mathbb{w}) \in \mathbf{R}\}$ be an NP-language.

A *non-interactive zero-knowledge (NIZK) argument system* Ψ for RG consists of five PPT algorithms: First, a probabilistic CRS generator G that, given $(\mathsf{p}, \mathbf{R}) \in \mathrm{im}(\mathsf{RG}(1^\lambda))$, outputs $(\mathsf{crs}, \mathsf{td})$ where crs is a CRS and td is a simulation trapdoor. Otherwise, it outputs a special symbol $\perp$. For the sake of efficiency and readability, we divide crs into crs_P (the part needed by the prover) and crs_V (the part needed by the verifier). Within this paper, crs explicitly encodes $\mathbf{R}$. We also implicitly assume that crs encodes p. Second, a probabilistic CRS verifier CV that, given crs, returns either 0 (the CRS is malformed) or 1 (the CRS is well-formed). CV is only required to exist in the case of Sub-ZK argument systems. Third, a probabilistic prover P that, given $(\mathsf{crs}_\mathsf{P}, \mathbb{x}, \mathbb{w})$ for $(\mathbb{x}, \mathbb{w}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\perp$. Fourth, a probabilistic verifier V that, given $(\mathsf{crs}_\mathsf{V}, \mathbb{x}, \pi)$, returns either 0 (reject) or 1 (accept). Fifth, a probabilistic simulator Sim that, given $(\mathsf{crs}, \mathsf{td}, \mathbb{x})$, outputs an argument π.

A NIZK argument system must be complete (an honest verifier accepts an honest verifier), knowledge-sound (if a prover makes an honest verifier accept, then one can extract from the prover a witness $\mathbb{w}$), and zero-knowledge (there exists a simulator that, knowing the CRS trapdoor but not the witness, can produce accepting statements with the verifier's view being indistinguishable from the view when interacting with an honest prover). A Sub-ZK argument system [1,3] must additionally satisfy Sub-ZK (zero-knowledge holds even if the

CRS is maliciously generated); for this, one requires CRS-verifiability (CV only accepts a CRS if there exists a trapdoor td corresponding to it).

We will now give the formal definitions. Let Ψ be a non-interactive argument. Ψ is *perfectly complete for* RG, if for all λ, $(\mathsf{p}, \mathbf{R}) \in \mathrm{im}(\mathsf{RG}(1^\lambda))$, and $(\mathbb{x}, \mathbb{w}) \in \mathbf{R}$,

$$\Pr\left[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{G}(\mathsf{p}, \mathbf{R}) : \mathsf{V}(\mathsf{crs}_\mathsf{V}, \mathbb{x}, \mathsf{P}(\mathsf{crs}_\mathsf{P}, \mathbb{x}, \mathbb{w})) = 1\right] = 1 \enspace .$$

Ψ is computationally (adaptively) *knowledge-sound for* RG, if for every PPT $\mathcal{A}$, there exists a PPT extractor $\mathsf{Ext}_\mathcal{A}$, such that for all λ,

$$\Pr\left[\begin{matrix}(\mathsf{p}, \mathbf{R}) \leftarrow \mathsf{RG}(1^\lambda); (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{G}(\mathsf{p}, \mathbf{R}); r \leftarrow\!\!\$\, \mathsf{RND}_\lambda(\mathcal{A}); \\ (\mathbb{x}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}; r); \mathbb{w} \leftarrow \mathsf{Ext}_\mathcal{A}(\mathsf{crs}; r) : (\mathbb{x}, \mathbb{w}) \not\in \mathbf{R} \wedge \mathsf{V}(\mathsf{crs}_\mathsf{V}, \mathbb{x}, \pi) = 1\end{matrix}\right] \approx_\lambda 0 \enspace .$$

A knowledge-sound argument system is called an *argument of knowledge.*

Ψ is *statistically composable zero-knowledge for* RG, if for all λ, $(\mathsf{p}, \mathbf{R}) \in \mathrm{im}(\mathsf{RG}(1^\lambda))$, and computationally unbounded $\mathcal{A}$, $\varepsilon_0^{zk} \approx_\lambda \varepsilon_1^{zk}$, where

$$\varepsilon_b^{zk} := \Pr\left[\begin{matrix}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{KGen}(\mathsf{p}, \mathbf{R}), (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{td}); \pi_0 \leftarrow \mathsf{P}(\mathsf{crs}_\mathsf{P}, \mathbb{x}, \mathbb{w}); \\ \pi_1 \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, \mathbb{x}) : (\mathbb{x}, \mathbb{w}) \in \mathbf{R} \wedge \mathcal{A}(\pi_b) = 1\end{matrix}\right] \enspace .$$

Ψ is *perfectly composable Sub-ZK for* RG if one requires that $\varepsilon_0^{zk} = \varepsilon_1^{zk}$.

Ψ is *statistically composable Sub-ZK for* RG, if for any PPT subverter $\mathcal{S}$ there exists a PPT $\mathsf{Ext}_\mathcal{S}$, such that for all λ, all $(\mathsf{p}, \mathbf{R}) \in \mathrm{im}(\mathsf{RG}(1^\lambda))$, and all computationally unbounded $\mathcal{A}$, $\varepsilon_0^{zk} \approx_\lambda \varepsilon_1^{zk}$, where

$$\varepsilon_b^{zk} := \Pr\left[\begin{matrix}r \leftarrow\!\!\$\, \mathsf{RND}_\lambda(\mathcal{S}); (\mathsf{crs}, \mathsf{z}_\mathcal{S}) \leftarrow \mathcal{S}(\mathsf{p}, \mathbf{R}; r); \mathsf{td} \leftarrow \mathsf{Ext}_\mathcal{S}(\mathsf{p}, \mathbf{R}; r); \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{z}_\mathcal{S}); \pi_0 \leftarrow \mathsf{P}(\mathsf{crs}_\mathsf{P}, \mathbb{x}, \mathbb{w}); \pi_1 \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, \mathbb{x}); \\ (\mathbb{x}, \mathbb{w}) \in \mathbf{R} \wedge \mathsf{CV}(\mathsf{crs}) = 1 \wedge \mathcal{A}(\pi_b) = 1\end{matrix}\right] \enspace .$$

Ψ is *perfectly composable Sub-ZK for* RG if one requires that $\varepsilon_0^{zk} = \varepsilon_1^{zk}$.

A *SNARK (succinct non-interactive argument of knowledge)* is a NIZK argument system where the argument is sublinear in the input size.

Simulation-Extractability (SE). An SE argument system [37,14] stays knowledge-sound even if the soundness adversary has access to the simulation oracle. SE is motivated by applications like non-malleability and UC security.

Dodis *et al.* [15] differentiated between several favors of SE. In the case of any-simulation-extractability (ASE), the simulator can be queried with any (potentially false) statements while in the case of true-simulation-extractability (TSE), the simulator can only be queried with true statements. The adversary wins if she can come up with a new argument for a statement she has not queried a simulation for. In the case of strong any-simulation-extractability (SASE), the adversary wins even if she can come up with a new argument for a statement she has queried a simulation for. ASE suffices for UC security.

Groth and Maller [25] define SE SNARKs, where one requires that for each PPT knowledge-soundness adversary $\mathcal{A}$ with oracle access to the simulator, there

```
MAIN $\mathbf{Exp}^{\boxed{\mathsf{s}}\mathrm{ase}}_{\Psi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$
$\mathcal{Q} \leftarrow \emptyset; (\mathsf{p}, \mathbf{R}) \leftarrow \mathsf{RG}(1^\lambda); (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{G}(\mathsf{p}, \mathbf{R});$
$r \leftarrow \mathsf{RND}_\lambda(\mathcal{A}); (\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{Sim}^{\boxed{\mathsf{s}}\mathrm{ase}}_{\mathsf{crs},\mathsf{td}}}(\mathsf{crs}; r); \mathbb{w} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{crs}; r);$
if $\mathsf{V}(\mathsf{crs}_\mathsf{V}, \mathbb{x}, \pi) = 1 \wedge (\mathbb{x}\boxed{, \pi}) \notin \mathcal{Q} \wedge (\mathbb{x}, \mathbb{w}) \notin \mathbf{R}$
 then return $1$; else return $0$; fi

$\mathsf{Sim}^{\boxed{\mathsf{s}}\mathrm{ase}}_{\mathsf{crs},\mathsf{td}}(\mathbb{x}_j)$
$\pi_j \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, \mathbb{x}_j); \mathcal{Q} \leftarrow \mathcal{Q} \cup \{(\mathbb{x}_j\boxed{, \pi_j})\};$ return $\pi_j$;
```

Fig. 2. Any-simulation (ASE) and strong any-simulation (SASE) experiments. The $\boxed{\textit{boxed}}$ part is only present in the boxed (i.e., SASE) experiment.

exists a non-black-box extractor $\mathsf{Ext}_{\mathcal{A}}$ that can extract the witness. [25]'s definition of SE corresponds to *non-black-box SASE*, [15]. We assume implicitly SE means non-black-box SE. [25] proved that the argument of any (non-black-box) SASE SNARK consists of at least three group elements and that there should be at least two verification equations. They proposed a SASE SNARK for the SAP (Square Arithmetic Program) language that meets the lower bounds.

The following definition of the SASE property corresponds to the definition of SE SNARKs in [25, Definition 2.10]. All definitions are inspired by the corresponding black-box definitions from [15].

Let Ψ be a SNARK for the relation $\mathbf{R}$. Let $\mathsf{x} \in \{\mathsf{ase}, \mathsf{sase}\}$. Define $\mathsf{Adv}^{\mathsf{x}}_{\Psi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) := \Pr[\mathbf{Exp}^{\mathsf{x}}_{\Psi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)]$, where the experiment $\mathbf{Exp}^{\mathsf{x}}_{\Psi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ is depicted in Fig. 2. Then, (i) Ψ is *non-black-box any-simulation-extractable (ASE)* if for any PPT $\mathcal{A}$ there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, such that $\mathsf{Adv}^{\mathrm{ase}}_{\Psi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \approx_\lambda 0$. (ii) Ψ is *non-black-box strong any-simulation-extractable (SASE)* if for any PPT $\mathcal{A}$ there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, such that $\mathsf{Adv}^{\mathrm{sase}}_{\Psi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \approx_\lambda 0$.

3 Knowledge-Sound SNARK for QAP

Next, we will describe the new knowledge-sound SNARK $\mathsf{S}_{\mathsf{qap}}$. Its construction emphasizes two objectives: (i) simple soundness proof in the AGM and (ii) efficiency. $\mathsf{S}_{\mathsf{qap}}$ is similar to Groth's SNARK from EUROCRYPT 2016 [23] (shown to be Sub-ZK in [17]), with two major differences: (1) the use of only two trapdoors instead of five, and (2) an alternative, much more straightforward, knowledge-soundness proof in the case of asymmetric pairings. On the other hand, Groth provided a more complex knowledge-soundness proof that is valid for both asymmetric and symmetric pairings.

QAP. Quadratic Arithmetic Program (QAP) was introduced in [21] as a language where for an input $\mathbb{x}$ and witness $\mathbb{w}$, $(\mathbb{x}, \mathbb{w}) \in \mathbf{R}$ can be verified by using a parallel quadratic check. QAP has an efficient reduction from the (either Boolean or Arithmetic) CIRCUIT-SAT. Thus, an efficient zk-SNARK for QAP results in an efficient zk-SNARK for CIRCUIT-SAT.

We consider arithmetic circuits that consist only of fan-in-2 multiplication gates, but either input of each multiplication gate can be any weighted sum of wire values, [21]. Let $m_0 < m$ be a non-negative integer. For an arithmetic circuit, let n be the number of multiplication gates, m be the number of wires, and m_0 be the number of public inputs.

Let $\mathbb{F} = \mathbb{Z}_p$. For the sake of efficiency, we require the existence of the nth primitive root of unity modulo p, denoted by ω. Let U, V, and W be instance-dependent matrices and let $\mathbb{z}$ be a witness. A QAP is characterized by the constraint $U\mathbb{z} \circ V\mathbb{z} = W\mathbb{z}$. For $j \in [1, m]$, define $u_j(X) := L_{U^{(j)}}(X)$, $v_j(X) := L_{V^{(j)}}(X)$, and $w_j(X) := L_{W^{(j)}}(X)$ to be interpolating polynomials of the jth column of the corresponding matrix. Thus, $u_j, v_j, w_j \in \mathbb{Z}_p^{(\leq n-1)}[X]$. Let $u(X) = \sum \mathbb{z}_j u_j(X)$, $v(X) = \sum \mathbb{z}_j v_j(X)$, and $w(X) = \sum \mathbb{z}_j w_j(X)$. Then $U\mathbb{z} \circ V\mathbb{z} = W\mathbb{z}$ iff $Z(X) \mid u(X)v(X) - w(X)$ iff $u(X)v(X) \equiv w(X) \pmod{Z(X)}$ iff there exists a polynomial $h(X)$ such that $u(X)v(X) - w(X) = h(X)Z(X)$.

An QAP instance $\mathcal{I}_{\mathsf{qap}}$ is equal to $(\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=1}^m)$. This instance defines the following relation:

$$\mathbf{R}_{\mathcal{I}_{\mathsf{qap}}} = \left\{ \begin{matrix} (\mathbb{x}, \mathbb{w}) \colon \mathbb{x} = (\mathbb{z}_1, \dots, \mathbb{z}_{m_0})^\top \wedge \mathbb{w} = (\mathbb{z}_{m_0+1}, \dots, \mathbb{z}_m)^\top \wedge \\ u(X)v(X) \equiv w(X) \pmod{Z(X)} \end{matrix} \right\} \tag{1}$$

where $u(X) = \sum_{j=1}^m \mathbb{z}_j u_j(X)$, $v(X) = \sum_{j=1}^m \mathbb{z}_j v_j(X)$, and $w(X) = \sum_{j=1}^m \mathbb{z}_j w_j(X)$ as above. That is, $(\mathbb{x}, \mathbb{w}) \in \mathbf{R} = \mathbf{R}_{\mathcal{I}_{\mathsf{qap}}}$ if there exists a (degree $\leq n-2$) polynomial $h(X)$, such that the following key equation holds:

$$\chi(X) := u(X)v(X) - w(X) - h(X)Z(X) = 0 \ , \tag{2}$$

On top of checking Eq. (2), the verifier also needs to check that $u(X)$, $v(X)$, and $w(X)$ are correctly computed: that is, that (i) the first m_0 coefficients $\mathbb{z}_j$ in $u(X)$ are equal to the public inputs, and (ii) $u(X)$, $v(X)$, and $w(X)$ are all computed by using the same coefficients $\mathbb{z}_j$ for $j \leq m$.

SNARK Derivation. Let $u(X)$, $v(X)$, $w(X)$, and $\chi(X)$ be as in Section 2. Recall from Eq. (2) that the key equation of QAP states that the prover is honest iff $\chi(X) = 0$, that is, $h(X) := (u(X)v(X) - w(X))/Z(X)$ is a polynomial. We will use bivariate polynomials like $A(X, Y)$. The indeterminate X is related to the definition of QAP. The indeterminate Y groups together correct X-polynomials in the security proof; such a grouping approach was also used in say [24]. The argument in the new template consists of three elements, $\pi = ([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2)$, where $\mathsf{a} = A(x, y)$, $\mathsf{b} = B(x, y)$, and $\mathsf{c}_s = C_s(x, y)$ for well-defined polynomials $A(X, Y)$, $B(X, Y)$, and $C_s(X, Y)$. Intuitively, $[\mathsf{a}]_1$ is a succinct commitment to $u(X)$, $[\mathsf{b}]_2$ is a succinct commitment to $v(X)$, and $[\mathsf{c}_s]_1$ is the "actual" argument that at the same time commits to $w(X)$.

As in all most efficient random-oracle-less zk-SNARKs [21,36,31,23], we aim to make $[\mathsf{c}_s]_1$ to be computable only by the honest prover. The prover has access to the CRS that contains the evaluation of well-chosen polynomials at (x, y) in both $\mathbb{G}_1$ and $\mathbb{G}_2$. The knowledge-soundness proof is in the AGM. There, we show that if the verification polynomial $\mathcal{V}(X, Y) = 0$, and $A(X, Y)$, $B(X, Y)$, and $C_s(X, Y)$ are in the span of the polynomials in the CRS, then it must hold that $\chi(X) = 0$ and thus the prover is honest.

More precisely, let $\boldsymbol{\Delta} := (\alpha, \beta, \gamma, \delta, \eta)$ be a tuple of small integers chosen later. We will give a complete derivation of the new SNARK. We will also derive the conditions $\boldsymbol{\Delta}$ has to satisfy for the SNARK to be knowledge-sound; in Sections 4 and 5, we add more conditions to achieve both CRS-verifiability (and thus Sub-ZK) and ASE. We find it instructional to go first through the process with unfixed $\boldsymbol{\Delta}$. In Eq. (11), we propose a setting of $\boldsymbol{\Delta}$ that is sufficient to obtain all knowledge-soundness, ASE, and CRS-verifiability.

For randomizers r_a and r_b needed to make the commitment hiding, define

$$A(X, Y) := r_a Y^\alpha + u(X) Y^\beta \quad , \quad B(X, Y) := r_b Y^\alpha + v(X) Y^\beta \tag{3}$$

to be "commitments" to $u(X)$ and $v(X)$. We use different powers of Y to separate the randomness from the committed values. Define also

$$\begin{aligned} C(X, Y) :=&(A(X, Y) + Y^\gamma)(B(X, Y) + Y^\delta) - Y^{\gamma+\delta} \\ =&u(X) Y^{\beta+\delta} + v(X) Y^{\beta+\gamma} + u(X) v(X) Y^{2\beta} + R(X, Y) Y^\alpha \\ =&P(X, Y) + (u(X) v(X) - w(X)) Y^{2\beta} + R(X, Y) Y^\alpha \end{aligned} \tag{4}$$

where $P(X, Y) := u(X) Y^{\beta+\delta} + v(X) Y^{\beta+\gamma} + w(X) Y^{2\beta}$ and $R(X, Y) := r_b(A(X, Y) + Y^\gamma) + r_a(v(X) Y^\beta + Y^\delta)$.

The inclusion of Y^γ and Y^δ in the definition of $C(X, Y)$ serves three goals. First, it introduces the addend $P(X, Y) = \sum_{j=1}^{m} \mathbb{z}_j P_j(X, Y)$, where

$$P_j(X, Y) := u_j(X) Y^{\beta+\delta} + v_j(X) Y^{\beta+\gamma} + w_j(X) Y^{2\beta} \quad ; \tag{5}$$

this makes it easier to verify that P uses the same coefficients $\mathbb{z}_j$ when computing $[\mathsf{a}]_1$, $[\mathsf{b}]_2$, and $[\mathsf{c}_s]_1$. Second, it makes it possible to verify that P uses the correct public input. Third, the coefficient of $Y^{2\beta}$, $u(X)v(X) - w(X)$, divides by $Z(X)$ iff the prover is honest. That is, it is $h(X)Z(X)$ for some polynomial $h(X)$ iff the prover is honest and thus $\mathbb{x} \in \mathcal{L}_{\mathcal{I}_{\mathsf{qap}}}$.

On top of $\chi(X) = 0$, it must be possible to check that the public input $(\mathbb{z}_j)_{j=1}^{m_0}$ is correct. To this end, we define polynomials $C_s(X, Y)$ and $C_p(X, Y)$, s.t. $C(X, Y) = C_p(X, Y) Y^\eta + C_s(X, Y) Y^\alpha$. Here, $[\mathsf{c}_p]_1 = [C_p(x, y)]_1$ is recomputed by the verifier and thus $C_p(X, Y)$ must not depend on $\mathbb{z}_j$ for $j > m_0$ (i.e., on the secret information). To minimize the verifier's computation, $C_p(X, Y)$ has only m_0 addends. C_s depends both on public and secret inputs, and only an honest prover should be able to compute $[\mathsf{c}_s]_1 = [C_s(x, y)]_1$. Thus, we define

$$\begin{aligned} C_p(X, Y) :=& \textstyle\sum_{j=1}^{m_0} \mathbb{z}_j P_j(X, Y) Y^{-\eta} \\ C_s(X, Y) :=& \textstyle\sum_{j=m_0+1}^{m} \mathbb{z}_j P_j(X, Y) Y^{-\alpha} + (u(X) v(X) - w(X)) Y^{2\beta-\alpha} + R(X, Y) \enspace . \end{aligned} \tag{6}$$

```
$\mathsf{G}(\mathsf{p}, \mathbf{R})$: Sample $x, y \leftarrow_{\$} \mathbb{Z}_p^*$ such that $x^n \neq 1$, let $\mathsf{td} \leftarrow (x, y)$. Let
    $\mathsf{crs_P} \leftarrow \begin{pmatrix} [\{P_j(x,y)y^{-\alpha}\}_{j=m_0+1}^{m}, y^{\alpha}, \{x^j y^{\beta}\}_{j=0}^{n-1}, \{x^i Z(x) y^{2\beta-\alpha}\}_{j=0}^{n-2}, y^{\gamma}, y^{\delta}]_1, \\ [y^{\alpha}, \{x^j y^{\beta}\}_{j=0}^{n-1}]_2 \end{pmatrix}$ ;
    $\mathsf{crs_V} \leftarrow \left([\{P_j(x,y)y^{-\eta}\}_{j=1}^{m_0}, y^{\gamma}]_1, [y^{\alpha}, y^{\delta}, y^{\eta}]_2, [y^{\gamma+\delta}]_T\right)$ ;
    $\mathsf{crs} \leftarrow (\mathsf{crs_P}, \mathsf{crs_V})$; return $(\mathsf{crs}, \mathsf{td})$;
$\mathsf{P}(\mathsf{crs_P}, (\mathbb{z}_j)_{j=1}^{m_0}, (\mathbb{z}_j)_{j=m_0+1}^{m})$:
    $u(X) \leftarrow \sum_{j=1}^{m} \mathbb{z}_j u_j(X)$; $v(X) \leftarrow \sum_{j=1}^{m} \mathbb{z}_j v_j(X)$; $w(X) \leftarrow \sum_{j=1}^{m} \mathbb{z}_j w_j(X)$;
    $h(X) \leftarrow (u(X)v(X) - w(X))/Z(X)$;
    $(r_a, r_b) \leftarrow_{\$} \mathbb{Z}_p^2$; $[\mathsf{a}]_1 \leftarrow r_a[y^{\alpha}]_1 + [u(x)y^{\beta}]_1$; $[\mathsf{b}]_2 \leftarrow r_b[y^{\alpha}]_2 + [v(x)y^{\beta}]_2$;
    $[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^{m} \mathbb{z}_j [P_j(x,y)y^{-\alpha}]_1 + [h(x)Z(x)y^{2\beta-\alpha}]_1 + r_b([\mathsf{a}]_1 + [y^{\gamma}]_1) + r_a([y^{\delta}]_1 + [v(x)y^{\beta}]_1)$;
    return $\pi \leftarrow ([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2)$;
$\mathsf{V}(\mathsf{crs_V}, (\mathbb{z}_j)_{j=1}^{m_0}, \pi = ([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2))$:
    $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} \mathbb{z}_j [P_j(x,y)y^{-\eta}]_1$; Check that
        $[\mathsf{c}_p]_1 \bullet [y^{\eta}]_2 + [\mathsf{c}_s]_1 \bullet [y^{\alpha}]_2 = [\mathsf{a} + y^{\gamma}]_1 \bullet [\mathsf{b} + y^{\delta}]_2 - [y^{\gamma+\delta}]_T$ .    (7)
$\mathsf{Sim}(\mathsf{crs}, \mathsf{td} = (x, y), \mathbb{x} = (\mathbb{z}_j)_{j=1}^{m_0})$:    // $x$ is not used by the simulator
    $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} \mathbb{z}_j [P_j(x,y)y^{-\eta}]_1$; $d \leftarrow_{\$} \mathbb{Z}_p$; $e \leftarrow_{\$} \mathbb{Z}_p$; $[\mathsf{a}]_1 \leftarrow d[1]_1$; $[\mathsf{b}]_2 \leftarrow e[1]_2$;
    $[\mathsf{c}_s]_1 \leftarrow y^{-\alpha}((de + y^{\delta}d + y^{\gamma}e)[1]_1 - y^{\eta}[\mathsf{c}_p]_1)$;
    return $\pi \leftarrow ([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2)$;
```

Fig. 3. The new SNARK $\mathsf{S_{qap}}$. Moreover, $\mathsf{S_{qsp}}$ is exactly like $\mathsf{S_{qap}}$, except $w_j(X) = 0$.

Here, we use the factors Y^{η} and Y^{α} to separate the public input and the witness in the security proof. For efficiency reasons, we use Y^{α}, instead of a new power of Y: now $C_s(X, Y)$ has an addend $r_b A(X, Y)$ that reuses the value $A(X, Y)$.

As mentioned before, the SNARK argument is $\pi = ([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2)$. The verifier recomputes $[\mathsf{c}_p]_1 \leftarrow [C_p(x, y)]_1$ and $[C(x, y)]_T \leftarrow [\mathsf{c}_p]_1 \bullet [y^{\eta}]_2 + [\mathsf{c}_s]_1 \bullet [y^{\alpha}]_2$. Then, the verifier checks that $C(x, y)$ is computed correctly by checking that $C(x, y) = (A(x, y) + y^{\gamma})(B(x, y) + y^{\delta}) - y^{\gamma+\delta}$.

We are now ready to describe the SNARK $\mathsf{S_{qap}}$, see Fig. 3. The CRS consists of elements needed by the honest prover, the honest verifier, and the simulator. We will explain the simulator in the proof of Theorem 1. The CRS has two trapdoors (x and y), but the simulator uses only one of them (y). ([1,3] formalized the difference by defining two different types of trapdoors, CRS trapdoors $\mathsf{td_{crs}}$ and simulation trapdoors $\mathsf{td_{sim}}$. In $\mathsf{S_{qap}}$, $\mathsf{td_{crs}} = (x, y)$ and $\mathsf{td_{sim}} = y$.)

Security Intuition. We prove knowledge-soundness in the AGM with random oracles. Recall that an algebraic adversary can use the oracle $\mathcal{O}_{\iota}$, $\iota \in \{1, 2\}$, to create new random group elements $[q_{1i}]_{\iota}$. Let $\boldsymbol{Q}_{\iota}$ be the vector of corresponding indeterminates in $\mathbb{G}_{\iota}$. Let $\boldsymbol{X} = (X, \boldsymbol{Q}_1, \boldsymbol{Q}_2, Y)$ (resp., $\boldsymbol{x} = (x, \boldsymbol{q}_1, \boldsymbol{q}_2, y)$) be the tuple of all indeterminates (resp., corresponding random integers).

Write the CRS in Fig. 3 as $\mathsf{crs} = (\mathsf{crs}_1, \mathsf{crs}_2)$, where $\mathsf{crs}_\iota = [(f(x, y))_{f \in \Gamma_\iota}]_\iota$ for a public set Γ_ι of polynomials. For example, $\Gamma_2 = \{Y^\alpha, Y^\delta, Y^\eta\} \cup \{X^j Y^\beta\}_{j=0}^{n-1}$. (As an optimization, the CRS of $\mathsf{S}_{\mathsf{qap}}$ also includes $[y^{\gamma+\delta}]_T$, but it can be recomputed from the available elements in $\mathbb{G}_1$ and $\mathbb{G}_2$.) Since we work in the AGM, the malicious prover is algebraic and thus we can extract matrices $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$, such that $\left(\begin{smallmatrix} \mathsf{a} \\ \mathsf{c}_s \end{smallmatrix}\right) = \boldsymbol{N}_1\left(\begin{smallmatrix} \mathsf{crs}_1 \\ \boldsymbol{q}_1 \end{smallmatrix}\right)$ and $\mathsf{b} = \boldsymbol{N}_2\left(\begin{smallmatrix} \mathsf{crs}_2 \\ \boldsymbol{q}_2 \end{smallmatrix}\right)$. This means, that we can write $\mathsf{a} = A^\dagger(\boldsymbol{x})$, $\mathsf{b} = B^\dagger(\boldsymbol{x})$, and $\mathsf{c}_s = C_s^\dagger(\boldsymbol{x})$, where $A^\dagger(\boldsymbol{X})$, $B^\dagger(\boldsymbol{X})$, and $C_s^\dagger(\boldsymbol{X})$ are maliciously computed polynomials with known coefficients. We can recover all coefficients of $A^\dagger(\boldsymbol{X})$, $B^\dagger(\boldsymbol{X})$, and $C_s^\dagger(\boldsymbol{X})$ from $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$, as follows:

$$\begin{aligned} A^\dagger(\boldsymbol{X}) :=& \textstyle\sum_{j=1}^{m_0} a_j^* P_j(X, Y) Y^{-\eta} + \sum_{j=m_0+1}^{m} a_j^* P_j(X, Y) Y^{-\alpha} + r_a Y^\alpha + \\ & u_a(X) Y^\beta + h_a(X) Z(X) Y^{2\beta-\alpha} + a_\gamma Y^\gamma + a_\delta Y^\delta + \textstyle\sum_k q_{ak} Q_{1k} \enspace , \\ C_s^\dagger(\boldsymbol{X}) :=& \textstyle\sum_{j=1}^{m_0} c_j^* P_j(X, Y) Y^{-\eta} + \sum_{j=m_0+1}^{m} c_j^* P_j(X, Y) Y^{-\alpha} + r_c Y^\alpha + \\ & u_c(X) Y^\beta + h_c(X) Z(X) Y^{2\beta-\alpha} + c_\gamma Y^\gamma + c_\delta Y^\delta + \textstyle\sum_k q_{ck} Q_{1k} \enspace , \\ B^\dagger(\boldsymbol{X}) :=& r_b Y^\alpha + v_b(X) Y^\beta + b_\delta Y^\delta + b_\eta Y^\eta + \textstyle\sum_k b_{qk} Q_{2k} \enspace , \end{aligned} \tag{8}$$

where, say $a_j^* \in \mathbb{Z}_p$, $u_a(X) \in \mathbb{Z}_p^{(\leq n-1)}[X]$, and $h_a(X) \in \mathbb{Z}_p^{(\leq n-2)}[X]$.

The verification equation Eq. (7) guarantees $\mathcal{V}(\boldsymbol{x}) = 0$, where

$$\mathcal{V}(\boldsymbol{X}) := (A^\dagger(\boldsymbol{X}) + Y^\gamma)(B^\dagger(\boldsymbol{X}) + Y^\delta) - Y^{\gamma+\delta} - C_p(X, Y) Y^\eta - C_s^\dagger(\boldsymbol{X}) Y^\alpha \enspace . \tag{9}$$

Note that C_p is honestly computed. Since we know all coefficients of polynomials like $A^\dagger(\boldsymbol{X})$, we also know all coefficients of $\mathcal{V}(\boldsymbol{X})$.

On the Use of AGM. In the knowledge-soundness proof, we assume that the knowledge-soundness adversary $\mathcal{A}$ is algebraic and then break the PDL assumption. More precisely, with use the AGM with random oracles. However, we note that $\mathrm{RO}_{\mathsf{fkl}}$-AGM is not realistic since it allows to prove the security of false knowledge assumptions. [4] Really, consider the assumption that any PPT adversary $\mathcal{A}$, that on input $[1]_1$ generates $[x]_1$, must know x. This assumption is false in the settings where $\mathcal{A}$ has access to an efficient method (e.g., hash-and-increment or elliptic curve hashing) of creating random group elements without knowing their discrete logarithms. However, in the $\mathrm{RO}_{\mathsf{fkl}}$-AGM, one can extract an integer vector $\boldsymbol{N}$ and group element vector $[\boldsymbol{q}]_1$, such that $[x]_1 = \boldsymbol{N}^\top \left[\begin{smallmatrix} 1 \\ \boldsymbol{q} \end{smallmatrix}\right]_1 = N_1[1]_1 + \sum_{i \geq 1} N_{1+i}[q_i]_1$. Moreover, the reduction can program the random oracle by first creating the discrete logarithms q_k of each coordinate of $[\boldsymbol{q}]_1$. Then, $[x]_1 = (N_1 + \sum_{i \geq 1} N_{1+i})[1]_1$ and thus the reduction can output its discrete logarithm $x \leftarrow N_1 + \sum_{i \geq 1} N_{1+i}$. One has exactly the same issue when using AGM without random oracles (in this case, $\boldsymbol{q}$ has length 0).

The problem is that the reduction knows $\boldsymbol{q}$ and can thus compute x. The knowledge of $\boldsymbol{q}$ should be impossible if $\mathcal{A}$ has created $[q_k]_1$ by using elliptic curve hashing. We modify the AGM with random oracles so that one can still prove the

[4] This is probably one reason why [19] uses AGM with random oracles in the case where the analyzed protocol itself uses random oracles. [19] proves the knowledge-soundness of Groth's SNARK in the AGM *without* random oracles.

security of (thought to be) secure knowledge assumptions but not of assumptions of the above type. The first idea is to restrict the way the reduction is allowed to program the random oracle: given that the input of the reduction (who aims to break the PDL assumption) is $\mathbb{x}_{\mathcal{A}} = (\mathsf{p}; [x^i : i \in \mathcal{T}_1]_1, [x^i : i \in \mathcal{T}_2]_2)$, we require that the reduction programs the random oracle $\mathcal{O}_\iota$ by creating random integers $s, t \leftarrow\$ \mathbb{Z}_p$ and then outputting $s[x]_\iota + t$. Such "linear programming" was already used in [19] but in a different context. For example, it was used to implicitly create other CRS trapdoors from $\mathbb{x}_{\mathcal{A}}$ and in one case (the security proof of the RO-model BLS signature) also to program the random oracle. However, our usage of this strategy is in a novel context and for a novel goal.

We modify the strategy of AGM with random oracles of [19] even further. When using the described "linear programming" strategy to construct a PDL adversary $\mathcal{B}$ that obtains input, depending on one trapdoor (say, x), and then uses this to create a multivariate crs for the knowledge-soundness adversary $\mathcal{A}$. For the reduction to be successful, $\mathcal{B}$ creates other trapdoors (notably, including $q_{\iota k}$) implicitly as linear functions of x. E.g., $\mathcal{B}$ sets $[y]_1 \leftarrow s_y[x]_1 + t_y[1]_1$, for random s_y and t_y, and similarly $[y^i]_1 \leftarrow [(s_y x + t_y)^i]_1$; this assums that $[1, x, \ldots, x^i]_1$ are given in the CRS. In the security proof, this means that one can write $\mathcal{V}$ as a univariate (Laurent) polynomial $\mathcal{V}_x(X) = \mathcal{V}(\boldsymbol{X})$ and then use a polynomial factorization algorithm to compute x in the case $\mathcal{V}(\boldsymbol{X}) \neq 0$ but $\mathcal{V}(\boldsymbol{x}) = 0$.

This strategy has some undesirable properties. First, for every monomial $[x^i y^j]_\iota$ in the CRS, we need to give $[x^{i+j}]_\iota$ as an input to the PDL adversary. Since $\max i, \max j < \max(i + j)$ and $(n + 1, n')$-PDL is stronger than (n, n')-PDL in the AGM, one uses a stronger PDL assumption. Second, this strategy is challenging to implement when, as in our case, the CRS depends on the negative powers of some trapdoors. Really, given $[1/x^i]_1$ for various i-s, it is presumably hard to compute $[1/(sx + t)^j]_1$ for $j > 1$ and random s and t; due to this reason, *the "linear programming" strategy cannot be used to prove the knowledge-soundness of* $\mathsf{S}_{\mathsf{qap}}$ *(or Groth's SNARK since it also involves negative powers of trapdoors).*[5] Finally, the degree of $\mathcal{V}_x$ is related to the *total* degree of $\mathcal{V}$.

We use a different strategy. We define two different adversaries, one aiming to compute x (given a PDL input that depends on x) and another aiming to compute y (given a PDL input that depends on y). Both adversaries generate the second trapdoor randomly. The reduction programs the oracles differently, by using the "linear programming" strategy in one case and the $\mathrm{RO}_{\mathsf{fkl}}$ strategy in another case. (This is detailed in Fig. 4.) As a direct benefit, inside the reduction, we deal with polynomials of smaller degrees. Moreover, instead of giving $[x^{i+j}]_\iota$ to the adversary, we give $[x^i]_\iota$ as an input to one adversary and $[y^j]_\iota$ to another adversary. Hence, we can potentially rely on a weaker PDL assumption. Finally,

[5] In the case of the original Groth's SNARK, this holds true since there are two different trapdoors that are given in negative power in the CRS. One can solve this issue by modifying Groth's SNARK: for example, one can multiply all its CRS elements with a positive power of such trapdoors (but then one has to be carefully check that Sub-ZK still holds); [19] solved this issue by having an additional game inside the knowledge-soundness proof that modified the CRS correspondingly.

since the second adversary ($\mathcal{B}^y$ in Fig. 4) uses the $\mathrm{RO}_{\mathsf{fkl}}$ strategy, it is easy to handle CRS elements of type $[y^{-1}]_1$ since one chooses y randomly. On the other hand, since the first adversary uses the "linear programming" strategy, one cannot prove the security of the false knowledge assumption described above.

On the Choice of Exponents. Another complicated part of the knowledge-soundness proof is the analysis of what happens if $\mathcal{V}(\boldsymbol{X}) \neq 0$ as a Laurent polynomial, but the verification succeeds, that is, $\mathcal{V}(\boldsymbol{x}) = 0$. Let $\boldsymbol{X}^* = (X, \boldsymbol{Q}_1, \boldsymbol{Q}_2)$ and $\boldsymbol{x}^* = (x, \boldsymbol{q}_1, \boldsymbol{q}_2)$. Writing $\mathcal{V}(\boldsymbol{X}) = \sum_i \mathcal{V}_{Y^i}(\boldsymbol{X}^*) Y^i$ for known Laurent polynomials $\mathcal{V}_{Y^i}(\boldsymbol{X}^*)$, we get $\mathcal{V}_{Y^i}(\boldsymbol{X}^*) = 0$ for each i. There are 29 non-trivial coefficients $\mathcal{V}_{Y^i}(\boldsymbol{X}^*)$, for $i \in$

$$\begin{aligned}&\{2\alpha, 2\beta, \alpha+\beta, 3\beta-\alpha, \alpha+\gamma, \beta+\gamma, -\alpha+2\beta+\gamma, 2\delta, \alpha+\delta, \beta+\delta,\\ &-\alpha+2\beta+\delta, \gamma+\delta, -\alpha+\beta+\gamma+\delta, -\alpha+\beta+2\delta, \alpha+2\beta-\eta, 3\beta-\eta,\\ &\alpha+\beta+\gamma-\eta, 2\beta+\gamma-\eta, \alpha+\beta+\delta-\eta, 2\beta+\delta-\eta, \beta+\gamma+\delta-\eta, \beta+2\delta-\eta,\\ &\alpha+\eta, \beta+\eta, -\alpha+2\beta+\eta, \gamma+\eta, -\alpha+\beta+\gamma+\eta, \delta+\eta, -\alpha+\beta+\delta+\eta\}\ .\end{aligned} \tag{10}$$

It is possible but very tedious to show that from $\mathcal{V}_{Y^i}(\boldsymbol{X}^*) = 0$ for each twenty nine i-s, we get that $\chi(X) = 0$ and thus, the prover is honest. To simplify the knowledge-soundness proof, we constructed $\mathsf{S}_{\mathsf{qap}}$ so that there exists a small set Crit of *six* elements, such that $\chi(X) = 0$ follows from $\mathcal{V}_{Y^i}(\boldsymbol{X}^*) = 0$ for $Y^i \in \mathsf{Crit}$.

For this idea to work, we need to restrict the choice of $\boldsymbol{\Delta}$: namely, $\boldsymbol{\Delta}$ has to be such that the exponents in Crit are different from each other and all other exponents of Y in $\mathcal{V}(\boldsymbol{X})$. More precisely, define $\mathsf{Coeff} := \{Y^i : \mathcal{V}_{Y^i}(\boldsymbol{X}^*) \neq 0\}$,

$$\mathsf{Crit} := \{Y^{2\beta}, Y^{\beta+\gamma}, Y^{\beta+\delta}, Y^{\gamma+\delta}, Y^{\gamma+\eta}, Y^{2\delta}\}\ ,$$

and let $\overline{\mathsf{Crit}} := \mathsf{Coeff} \setminus \mathsf{Crit}$ be the "symbolic" complement of Crit; that is, $Y^j \in \overline{\mathsf{Crit}}$ if j is *symbolically* not the same as one of the exponents in Crit, so $|\mathsf{Coeff}| = 29$ and $|\overline{\mathsf{Crit}}| = 29 - 6 = 23$. We highlighted the 6 critical coefficients in Eq. (10), not highlighted coefficients correspond to coefficients in $\overline{\mathsf{Crit}}$.

We say that $\boldsymbol{\Delta}$ is *soundness-friendly* if Crit consists of mutually different powers of Y ($|\mathsf{Crit}| = 6$) and $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$. We will give a concrete soundness-friendly suggestion for $\boldsymbol{\Delta}$ in Eq. (11). We depict the critical coefficients $\mathcal{V}_{Y^i}(\boldsymbol{X}^*)$, $Y^i \in \mathsf{Crit}$, in Table 2. (The last rows in Table 2 are only relevant for the ASE proof in Section 4.) In the knowledge-soundness proof of Theorem 1, we show that if $\mathcal{V}_{Y^i}(\boldsymbol{X}^*) = 0$ for $Y^i \in \mathsf{Crit}$, then $\chi(X) = 0$ and thus the prover is honest.

3.1 Security Theorem

Theorem 1. *Let $\mathcal{I}_{\mathsf{qap}} = (\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=1}^m)$ be a QAP instance. Let $\mathsf{S}_{\mathsf{qap}}$ be the SNARK in Fig. 3. Let $\mathcal{T}_\iota^x$ be the minimal set of exponents i such that the CRS of $\mathsf{S}_{\mathsf{qap}}$ in Fig. 3 can be computed by an algebraic adversary given $[x^i : i \in \mathcal{T}_1^x]_1, [x^i : i \in \mathcal{T}_2^x]_2$ and y. We define $\mathcal{T}_\iota^y$ dually.*
(1) Assume $\boldsymbol{\Delta}$ is soundness-friendly. Then, $\mathsf{S}_{\mathsf{qap}}$ is knowledge-sound in the AGM under the $(\mathcal{T}_1^x, \mathcal{T}_2^x)$-PDL and the $(\mathcal{T}_1^y, \mathcal{T}_2^y)$-PDL assumptions.
(2) $\mathsf{S}_{\mathsf{qap}}$ is perfectly zero-knowledge.

Table 2. $\mathsf{S}_{\mathsf{qap}}$: the critical coefficients in the knowledge-soundness proof (up, left), addends to the same coefficients in the ASE proof (up, right), and coefficients that only occur in the ASE proof (bottom). Here, $\tilde{\mathbb{z}}_j = \mathbb{z}_j - b_\eta a_j^*$ for $j \le m_0$, $\tilde{\mathbb{z}}_j = c_j^* - r_b a_j^*$ for $j > m_0$, $u(X) = \sum_{j=1}^m \tilde{\mathbb{z}}_j u_j(X)$, $v(X) = \sum_{j=1}^m \tilde{\mathbb{z}}_j v_j(X)$, $w(X) = \sum_{j=1}^m \tilde{\mathbb{z}}_j w_j(X)$, and $h(X) = h_c(X) - r_b h_a(X)$.

$Y^i \cdots$	$\mathcal{V}_{Y^i \ldots}(\boldsymbol{X}^*)$ (KS and ASE)	$\hat{\mathcal{V}}_{Y^{i_1} \ldots}(\boldsymbol{X}^*)$ (ASE only)
$Y^{\gamma+\delta}$	$(a_\gamma + 1)(b_\delta + 1) - 1$	
$Y^{\gamma+\eta}$	$(a_\gamma + 1)b_\eta$	
$Y^{2\delta}$	$(b_\delta + 1)a_\delta$	
$Y^{\beta+\delta}$	$(b_\delta + 1)u_a(X) + a_\delta v_b(X) - u(X)$	$\sum_k (s_{c2k} - r_b s_{a2k}) \sum_j \sigma_{kj} u_j(X)$
$Y^{\beta+\gamma}$	$(a_\gamma + 1)v_b(X) - v(X)$	$\sum_k (s_{c2k} - r_b s_{a2k}) \sum_j \sigma_{kj} v_j(X)$
$Y^{2\beta}$	$u_a(X)v_b(X) - w(X) - h(X)Z(X)$	$\sum_k (s_{c2k} - r_b s_{a2k}) \sum_j \sigma_{kj} w_j(X)$
Used only in the ASE proof		
$Y^{-\alpha+2\delta}D_k$		$(b_\delta + 1)s_{a2k}$
$Y^\gamma E_k$		$r_b s_{a2k} + (a_\gamma + 1)s_{bk} - s_{c2k}$
$D_k E_k$		$r_b s_{a2k} + s_{a1k} s_{bk} - s_{c2k}$
$Y^\delta D_k$		$r_b s_{a2k} + (b_\delta + 1)s_{a1k} - s_{c2k}$
Used only in the case (ii) in the ASE proof, if $s_{a1k} = a_\gamma + 1$ and $s_{c2k} = (a_\gamma + 1)s_{bk}$		
$D_{k_1} E_{k_2}$, $k_1 \ne k_2$		$s_{a1k_1} s_{bk_2}$
$Y^\beta E_k$		$u_a(X)s_{bk}$

Here, $\mathcal{T}_1^x = [0, 2n-2]$, $\mathcal{T}_2^x = [0, n-1]$, $\mathcal{T}_1^y = \{\beta - \alpha + \delta, \beta - \alpha + \gamma, 2\beta - \alpha, \alpha, \beta, 2\beta - \alpha, \gamma, \delta, \beta - \eta + \delta, \beta - \eta + \gamma, 2\beta - \eta\}$, and $\mathcal{T}_2^y = \{\alpha, \beta, \delta, \eta\}$. This can be contrasted to [19] that provided an AGM knowledge-soundness proof under the stronger $([1, 2n-1], [1, 2n-1])$-PDL assumption.

We emphasize that the following knowledge-soundness proof depends minimally on the concrete SNARK: the only intrinsically $\mathsf{S}_{\mathsf{qap}}$-dependent part is the analysis of the abort probability. The rest of the proof can essentially be copied to the knowledge-soundness (*and ASE*) proofs of all following SNARKs.

Proof. **(1: knowledge-soundness)** Let $\mathcal{A}$ be an algebraic knowledge-soundness adversary. Assume that $\mathcal{A}^{(\mathcal{O}_1, \mathcal{O}_2)}(\mathsf{crs}; r_{\mathcal{A}})$ outputs $(\mathbb{x}, \pi)$, such that V accepts with a non-negligible probability $\varepsilon_{\mathcal{A}}$. Let $\mathsf{crs} = (\mathsf{crs}_1, \mathsf{crs}_2)$, with $\mathsf{crs}_\iota = [\{f(\boldsymbol{x})\}_{f \in \Gamma_\iota}]_\iota$, as before. Since $\mathcal{A}$ is algebraic and the distribution $\mathcal{D}_{\mathsf{p}}$ of crs is PPT-sampleable, there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$, such that with probability $\varepsilon_{\mathcal{A}} - \varepsilon_{\mathsf{Ext}}$, where $\varepsilon_{\mathsf{Ext}} = \mathsf{Adv}^{\mathrm{agm}}_{\mathsf{Pgen}, \mathcal{D}, \mathcal{A}, \mathsf{Ext}_{\mathcal{A}}}(\lambda) = \mathsf{negl}(\lambda)$, $\mathsf{Ext}_{\mathcal{A}}(\mathsf{crs}; r_{\mathcal{A}})$ succeeds.

We construct two different PDL adversaries, $\mathcal{B}^x$ and $\mathcal{B}^y$, see Fig. 4. Intuitively, the main difference between them is that they use the knowledge-soundness adversary $\mathcal{A}$, whose input depends on either x or y, to break PDL with respect to x or y, correspondingly.

Let $z \in \{x, y\}$ and $Z \in \{X, Y\}$, correspondingly. $\mathcal{B}^z$ obtains an input $\mathbb{x}_z = ([z^k : k \in \mathcal{T}_1^z]_1, [z^k : k \in \mathcal{T}_2^z]_2)$. Intuitively, $\mathcal{B}^z$ reduces the actions of $\mathcal{A}$ to a univariate case by sampling the second trapdoor (y or x) uniformly at random. The verification equation states that $\mathcal{V}(\boldsymbol{x}^*, y) = 0$, where $\mathcal{V}(\boldsymbol{X}^*, Y)$ is a

```
$\boxed{\mathcal{B}^y(\mathsf{p}, \mathbf{R}, \mathbb{x}_y)}$ $\dbox{\mathcal{B}^x(\mathsf{p}, \mathbf{R}, \mathbb{x}_x)}$   // $\mathbb{x}_z = ([z^k : k \in \mathcal{T}_1^z]_1, [z^k : k \in \mathcal{T}_2^z]_2)$
$\boldsymbol{q}_1 \leftarrow \emptyset; \boldsymbol{q}_2 \leftarrow \emptyset; \xi_1 \leftarrow 0; \xi_2 \leftarrow 0;$
$\boxed{x \leftarrow_\$ \mathbb{Z}_p^*}\dbox{y \leftarrow_\$ \mathbb{Z}_p^*}$; Create $\mathsf{crs}$ from $(\mathsf{p}, \mathbf{R}, \boxed{\mathbb{x}_y, x}\dbox{\mathbb{x}_x, y});$
$r_\mathcal{A} \leftarrow_\$ \mathsf{RND}_\lambda(\mathcal{A}); ([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2) \leftarrow \mathcal{A}^{(\mathcal{O}_1, \mathcal{O}_2)}(\mathsf{p}, \mathbf{R}; \mathsf{crs}, r_\mathcal{A});$
$(\boldsymbol{N}_1, \boldsymbol{N}_2) \leftarrow \mathsf{Ext}_\mathcal{A}(\mathsf{crs}; r_\mathcal{A});$
if $\mathsf{Ext}_\mathcal{A}$ does not succeed then abort fi ;   // Abort probability: $\varepsilon_{\mathsf{Ext}}$
Compute the coefficients of $\mathcal{V}(\boldsymbol{X}^*, Y)$ from $\boldsymbol{N}_\iota$;
$(*)$if $\mathcal{V}(\boldsymbol{X}^*, Y) = 0$ then abort fi ;   // Abort prob.: 0
Let $\mathsf{bad}$ be the event $\mathcal{V}(\boldsymbol{x}^*, Y) = 0$;
  [full box]
  if $\mathsf{bad}$ then abort fi ;
    // Now, $\mathcal{V}(\boldsymbol{x}^*, Y) \neq 0$
  $\{y_j\} \leftarrow roots(\mathcal{V}(\boldsymbol{x}^*, Y), Y);$
  $y \leftarrow y_j$ s.t. $[y_j^\delta]_1 = [y^\delta]_1;$
  return $y$;
  [dashed box]
  if $\overline{\mathsf{bad}}$ then abort fi ;
    // Now, $\mathcal{V}(\boldsymbol{x}^*, Y) = 0$
  Write $\mathcal{V}(\boldsymbol{X}^*, Y) = \sum \mathcal{V}_i(\boldsymbol{X}^*) Y^i$;
  Let $i$ be s.t. $\mathcal{V}_i(\boldsymbol{X}^*) \neq 0$ but $\mathcal{V}_i(\boldsymbol{x}^*) \neq 0$;
  $\{x_j\} \leftarrow roots(\mathcal{V}_i(\boldsymbol{X}^*), X);$
  return $x \leftarrow x_j$ s.t. $[x_j]_1 = [x]_1$;

$\mathcal{O}_\iota$   // $\iota \in \{1, 2\}$
$\xi_\iota \leftarrow \xi_\iota + 1; \boxed{q_{\iota\xi_\iota} \leftarrow_\$ \mathbb{Z}_p}; \dbox{s_{\iota\xi_\iota}, t_{\iota\xi_\iota} \leftarrow_\$ \mathbb{Z}_p; [q_{\iota\xi_\iota}]_\iota \leftarrow s_{\iota\xi_\iota}[x]_\iota + t_{\iota\xi_\iota}[1]_1}$; return $[q_{\iota\xi_\iota}]_\iota$;
```

Fig. 4. The adversarjes $\mathcal{B}^z(\mathsf{p}, \mathbf{R}, \mathbb{x}_y)$, $z \in \{x, y\}$, and how they emulate $\mathcal{O}_\iota$ to $\mathcal{A}$ in the proof of Theorem 1. The parts where the two adversaries differ are boxed. $\boxed{\text{Full-boxed}}$ entries are only in $\mathcal{B}^y$ and its emulation, and $\dbox{\text{dash-boxed}}$ entries are only in $\mathcal{B}^x$ and its emulation. E.g., $\mathcal{B}^y$ samples a random x and $\mathcal{B}^y$ samples a random y.

known Laurent polynomial due to the use of the AGM. The adversary aborts if $\mathcal{V}(\boldsymbol{X}^*, Y) = 0$ as a Laurent polynomial. The most complicated part of the proof is to show that if $\mathcal{A}$ is successful, then $\mathcal{V}(\boldsymbol{X}^*, Y) \neq 0$ and thus the abort on this step is never executed. (For this, we need to analyze the six critical coefficients of $\mathcal{V}$, and we will do it at the end of the proof.)

Otherwise, we choose a polynomial $f(Z)$, such that $f(Z) \neq 0$ but $f(z) = 0$. Note that $\mathcal{B}^y$ samples the oracle answers $q_{\iota k}$ uniformly at random, while $\mathcal{B}^x$ sets implicitly $q_{\iota k} \leftarrow s_{\iota k} x + t_{\iota k}$. (Differently from [19], we only use this technique in the case of $\mathcal{B}^x$.) Thus, $\boldsymbol{Q}_\iota = \boldsymbol{s}_\iota X + \boldsymbol{t}_\iota$. If $\mathcal{V}(\boldsymbol{X}^*, Y) \neq 0$ but $\mathcal{V}(\boldsymbol{x}^*, Y) = 0$, then $\mathcal{V}'(X, Y) := \mathcal{V}(X, \boldsymbol{s}_1 X + \boldsymbol{t}_1, \boldsymbol{s}_2 X + \boldsymbol{t}_2, Y)$ satisfies $\mathcal{V}'(x, Y) = 0$. We set $f(X)$ to be equal to some non-zero coefficient $\mathcal{V}'_i(X) \neq 0$ of $\mathcal{V}'(X, Y) = \sum \mathcal{V}'_i(X) Y^i$.

$\mathcal{B}^z$ finds all the roots of $f(Z)$ and then checks which of the roots is equal to z by using information given in her input. For this, we define event $\mathsf{bad} = 1$ if $\mathcal{V}(\boldsymbol{x}^*, Y) = 0$ as a Laurent polynomial, where x is either the value imminent in the input of $\mathcal{B}^x$ or sampled by $\mathcal{B}^y$. $\mathcal{B}^y$ aborts if $\mathsf{bad} = 1$ and otherwise finds y. $\mathcal{B}^x$ aborts if $\mathsf{bad} = 0$ and otherwise finds x. Clearly,

$$\begin{aligned} \Pr[\mathcal{A} \text{ succeeds}] &\leq \Pr[\mathsf{Ext}_\mathcal{A} \text{ failed}] + \Pr[\mathsf{Ext}_\mathcal{A} \text{ succeeds}|\mathsf{bad}] + \Pr[\mathsf{Ext}_\mathcal{A} \text{ succeeds}|\overline{\mathsf{bad}}] \\ &\leq \Pr[\mathsf{Ext}_\mathcal{A} \text{ failed}] + \Pr[\mathcal{B}^x \text{ succeeds}|\mathsf{bad}] + \Pr[\mathcal{B}^y \text{ succeeds}|\overline{\mathsf{bad}}] \enspace . \end{aligned}$$

Analysis of the abort probability in step (*). Both $\mathcal{B}^x$ and $\mathcal{B}^y$ abort if $\mathcal{V}(\boldsymbol{X}^*, Y) = 0$ as a Laurent polynomial. Assume now that $\mathcal{V}(\boldsymbol{X}) = 0$, thus $\mathcal{V}_{Y^i}(\boldsymbol{X}^*) = 0$ for $Y^i \in \mathsf{Crit}$. We must show that (a) the critical coefficients are as in Table 2 and (b) from "$\mathcal{V}_{Y^i}(\boldsymbol{X}^*) = 0$ for $Y^i \in \mathsf{Crit}$" it follows that $\chi(X) = 0$.

One can derive a by inspection (we verified it by using computer algebra), assuming that Crit satisfies the theorem conditions. For example, the coefficient of $Y^{\gamma+\delta}$ in $\mathcal{V}(\boldsymbol{X})$ is $(a_\gamma+1)(b_\delta+1)-1$ since the coefficient of $Y^{\gamma+\delta}$ in $(A^\dagger(\boldsymbol{X}) + Y^\gamma)(B^\dagger(\boldsymbol{X}) + Y^\delta)$ is $(a_\gamma+1)(b_\delta+1)$. Other coefficients can be checked similarly.

Now, b follows. Really, since $\mathcal{V}_{Y^{\gamma+\delta}}(\boldsymbol{X}^*) = b_\delta + a_\gamma(b_\delta+1) = 0$, we get $a_\gamma = -b_\delta/(b_\delta+1)$. Thus, $a_\gamma, b_\delta \neq -1$ and $(a_\gamma+1)(b_\delta+1) = 1$. Since $\mathcal{V}_{Y^{\gamma+\eta}}(\boldsymbol{X}^*) = (a_\gamma+1)b_\eta = 0$ and $a_\gamma \neq -1$, we get $b_\eta = 0$. Thus, $\tilde{\mathbb{z}}_j = \mathbb{z}_j - b_\eta a_j^* = \mathbb{z}_j$ for $j \le m_0$. Since $\mathcal{V}_{Y^{2\delta}}(\boldsymbol{X}^*) = (b_\delta+1)a_\delta = 0$ and $b_\delta \neq -1$, we get $a_\delta = 0$. From the remaining coefficients, we get $(b_\delta+1)u_a(X) = u(X)$, $(a_\gamma+1)v_b(X) = v(X)$, and $u(X)v(X) - w(X) = Z(X)h(X)$. Thus, $(\mathbb{x}, \mathbb{w}) \in \mathbf{R}_{\mathcal{I}_{\mathsf{qap}}}$.

(2: zero-knowledge) To see that V accepts, note that $(\mathsf{a} + y^\gamma)(\mathsf{b} + y^\delta) - \mathsf{c}_s y^\alpha - \mathsf{c}_p y^\eta - y^{\gamma+\delta} = de + dy^\delta + ey^\gamma - (de + dy^\delta + ey^\gamma - \mathsf{c}_p y^\eta) - \mathsf{c}_p y^\eta = 0$. Sim's output comes from the correct distribution since a and b are individually uniform in $\mathbb{Z}_p$, and c is chosen so that V accepts. □

Efficiency. Compared to [23], see Table 1, $\mathsf{S}_{\mathsf{qap}}$ has fewer trapdoors but otherwise the same complexity. For example, crs_P has $(m-m_0)+1+n+(n-1)+1 = m+2n-m_0+1$ elements from $\mathbb{G}_1$ and $n+2$ elements from $\mathbb{G}_2$. Moreover, crs_V has m_0+1 elements from $\mathbb{G}_1$, 3 elements from $\mathbb{G}_2$, and one element from $\mathbb{G}_T$. Since crs_P and crs_V have one common element in $\mathbb{G}_1$ then $|\mathsf{crs}| = (m+2n+2)\mathfrak{g}_1 + (n+4)\mathfrak{g}_2 + \mathfrak{g}_T$. (Recall that $\mathfrak{g}_\iota$ denotes the representation length of an element of $\mathbb{G}_\iota$.) Clearly, $[\mathsf{a}]_1$ can be computed from $[y^\alpha]_1$ and $[x^i y^\beta]_1$ by using $n+1$ scalar multiplications. It takes $\approx m+2n$ additional scalar multiplications to compute $[\mathsf{c}]_1$.

A Soundness-Friendly Choice of $\boldsymbol{\varDelta}$. Recall that we need to find values for $\boldsymbol{\varDelta} = (\alpha, \ldots)$, such that $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$ and $|\mathsf{Crit}| = 6$. We require that both sets $\varGamma_1$ and $\varGamma_2$ contain a non-zero monomial corresponding to $Y^0 = 1$ (then we can publish $[1]_1$ and $[1]_2$) and that the values i, for which $i \in \mathcal{T}_1^y \cup \mathcal{T}_2^y$, have as small absolute values as possible. The latter makes the PDL assumption somewhat more reasonable and additionally enables us to construct a CRS verification algorithm and thus prove Sub-ZK [1,3] in Section 5. We are also interested in minimizing the CRS length.

Since there are many coefficients to take into account, we have a moderately hard optimization problem. We used a computer search to find all possible values for $\alpha, \beta, \ldots$ under the restriction that each has an absolute value at most 7. See Table 3 for the full list of found tuples $\boldsymbol{\varDelta}$. Note that for each $\boldsymbol{\varDelta} = (\alpha, \beta, \ldots)$, this table contains also $-\boldsymbol{\varDelta} = (-\alpha, -\beta, \ldots)$.

We recommend to use the following setting:

$$\alpha = 0, \;\; \beta = -2, \;\; \gamma = -3, \;\; \delta = 7, \;\; \eta = 2. \tag{11}$$

Table 3. Soundness-friendly values of $\boldsymbol{\Delta}$ with each parameter having absolute value ≤ 7. "✓" in the last column means that this choice of $\boldsymbol{\Delta}$ results in a Sub-ZK SNARK

α	β	γ	δ	η	Sub-ZK
-1	0	-7	3	-2	
0	-1	6	-4	1	
0	-1	7	-4	1	
0	-1	7	-5	1	
0	-2	-3	7	2	✓

α	β	γ	δ	η	Sub-ZK
0	-2	6	7	2	✓
0	-3	5	7	1	
0	1	-6	4	-1	
0	1	-7	4	-1	
0	1	-7	5	-1	

α	β	γ	δ	η	Sub-ZK
0	2	-6	-7	-2	✓
0	2	3	-7	-2	✓
0	3	-5	-7	-1	✓
1	0	7	-3	2	

As we will see in Sections 4 and 5, this is one of the settings that allow obtaining both ASE and Sub-ZK security. Assuming the setting of Eq. (11), $\mathsf{Crit} = \{Y^{-4}, Y^{-5}, Y^{5}, Y^{4}, Y^{-1}, Y^{14}\}$ and

$$\begin{aligned} \mathsf{crs_P} &= \begin{pmatrix} [\{u_j(x)y^5 + v_j(x)y^{-5} + w_j(x)y^{-4}\}_{j=m_0+1}^{m}, y^0, \{x^j y^{-2}\}_{j=0}^{n-1}]_1, \\ [\{x^i Z(x) y^{-4}\}_{j=0}^{n-2}, y^{-3}, y^7]_1, [y^0, \{x^j y^{-2}\}_{j=0}^{n-1}]_2 \end{pmatrix} , \\ \mathsf{crs_V} &= \left([\{u_j(x)y^3 + v_j(x)y^{-7} + w_j(x)y^{-6}\}_{j=1}^{m_0}, y^{-3}]_1, [y^0, y^7, y^2]_2, [y^4]_T\right) . \end{aligned} \tag{12}$$

In addition, our computer search tries to minimize the CRS length, but none of the choices of $\boldsymbol{\Delta}$ in Table 3 results in a shorter CRS.

On 2-Phase Updatability. Each of $Y^{\alpha}, Y^{\beta}, \ldots$ can be changed to an independent indeterminant, $Y_{\alpha}, Y_{\beta}, \ldots$, without invalidating the knowledge-soundness (or ASE) proof. This offers us the flexibility of choosing the number of trapdoors. In particular, Kohlweiss *et al.* proved recently [27] that Groth's SNARK [23] is two-phase updatable. Similarly, $\mathsf{S_{qap}}$ is two-phase updatable, when one defines three trapdoors, x, y, z, and uses well-chosen powers of z instead of y^{α} and y^{η} throughout the construction of $\mathsf{S_{qap}}$. Then, one can update x and y in the first and z in the second phase. We will omit further discussion.

4 Any-Simulation Extractability of $\mathsf{S_{qap}}$

Next, we prove that $\mathsf{S_{qap}}$ is ASE. The ASE proof is similar to the knowledge-soundness proof Theorem 1. The main difference is the handling of the case when $\mathcal{V}(\boldsymbol{X}) = 0$ as a Laurent polynomial. We use some monomials of $\mathcal{V}(\boldsymbol{X})$ to simplify the formulas and then arrive at a crossroad: in one case, the adversary did not use simulation query results, and thus we are back to the knowledge-soundness proof. In the second case, the adversary used some of the query results; then, we use specific coefficients of $\mathcal{V}(\boldsymbol{X})$ to argue that she used the result of precisely one query. After that, we show that the adversary used the same input to the simulator in this query as in the forgery attempt. (This result relies on an additional assumption that each $u_j(X)$, for $j \leq m_0$, is linearly independent of all other $u_i(X)$, $i \leq m$. This assumption can be easily satisfied by adding to the QAP m_0 dummy constraints $u_j \cdot 1 = u_j$, similarly to [21].) Hence, this is not an ASE but a *SASE* attack, and thus not valid in our context. Thus, $\mathsf{S_{qap}}$ is ASE.

In the ASE proof, the algebraic adversary $\mathcal{A}$ also sees the outputs of the simulator. Thus, $\mathcal{A}$ has more inputs than in the knowledge-soundness proof. Let $\boldsymbol{\sigma}_k = (\sigma_{kj})_{j=1}^{m_0}$ be the maliciously chosen simulator input that the adversary used, instead of $(\mathbb{z}_j)_{j=1}^{m_0}$, during the kth query. Let $\boldsymbol{X} = (X, \boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{D}, \boldsymbol{E}, Y)$ and $\boldsymbol{X}^* = (X, \boldsymbol{Q}_1, \boldsymbol{Q}_2)$, where D_k (resp., E_k) is the indeterminate corresponding to the trapdoor $d = d_k$ (resp., $e = e_k$) generated by the simulator during the kth query. Observing Fig. 3, Sim answers with $([d_k, y^{-\alpha}((d_k e_k + y^\delta d_k + y^\gamma e_k) - \sum_{j=1}^{m_0} \sigma_{kj} P_j(x, y))]_1, [e_k]_2)$. Thus, in the ASE proof, $A^\dagger(\boldsymbol{X})$, $B^\dagger(\boldsymbol{X})$, and $C_s^\dagger(\boldsymbol{X})$ have the following additional addends:

$$
\begin{aligned}
A^\dagger(\boldsymbol{X}) &= \ldots + \textstyle\sum_k s_{a1k} D_k + \sum_k s_{a2k} Y^{-\alpha}((D_k E_k + Y^\delta D_k + Y^\gamma E_k) - \sum_{j=1}^{m_0} \sigma_{kj} P_j(X, Y)) \ , \\
C_s^\dagger(\boldsymbol{X}) &= \ldots + \textstyle\sum_k s_{c1k} D_k + \sum_k s_{c2k} Y^{-\alpha}((D_k E_k + Y^\delta D_k + Y^\gamma E_k) - \sum_{j=1}^{m_0} \sigma_{kj} P_j(X, Y)) \ , \\
B^\dagger(\boldsymbol{X}) &= \ldots + \textstyle\sum_k s_{bk} E_k \ .
\end{aligned}
$$

Here, the coefficients like s_{a1k} are chosen by the adversary. Let $\mathcal{V}(\boldsymbol{X}) = \sum_{i_1, i_2, i_3, i_4, k_1, k_2, k_3} \mathcal{V}_{Y^{i_1} D_{k_1}^{i_2} E_{k_2}^{i_3} E_{k_3}^{i_4}}(\boldsymbol{X^*}) Y^{i_1} D_{k_1}^{i_2} E_{k_2}^{i_3} E_{k_3}^{i_4}$. The addition of new addends to polynomials like $A^\dagger(\boldsymbol{X})$ means that the existing critical coefficients of $\mathcal{V}_{Y^{i_1} \ldots}$ of $\mathcal{V}(\boldsymbol{X})$ change by extra addends; we have denoted these extras by $\hat{\mathcal{V}}_{Y^i \ldots}$ in Table 2. Moreover, there are a number of new critical coefficients, depicted in the bottom of Table 2. For example, $\mathcal{V}_{Y^{\beta+\delta}}(\boldsymbol{X}^*) = (b_\delta + 1) u_a(X) + a_\delta v_b(X) - u(X) + \sum_k (s_{c2k} - r_b s_{a2k}) \sum_j \sigma_{kj} u_j(X)$ and, for any k, $\mathcal{V}_{Y^\gamma E_k}(\boldsymbol{X}^*) = r_b s_{a2k} + (a_\gamma + 1) s_{bk} - s_{c2k}$. Since here, the index $Y^{i_1} D_{k_1}^{i_2} E_{k_2}^{i_3} E_{k_3}^{i_4}$ of $\mathcal{V}_{Y^{i_1} \ldots}$ depends on a non-constant number of indeterminates, here both $\mathsf{Coeff}_{se} := \{Y^{i_1} D_{k_1}^{i_2} E_{k_2}^{i_3} E_{k_3}^{i_4} : \mathcal{V}_{Y^{i_1} D_{k_1}^{i_2} E_{k_2}^{i_3} E_{k_3}^{i_4}}(\boldsymbol{X}^*) \neq 0\}$ and

$$
\begin{aligned}
\mathsf{Crit}_{se} = &\{Y^{2\beta}, Y^{\beta+\gamma}, Y^{\beta+\delta}, Y^{\gamma+\delta}, Y^{\gamma+\eta}, Y^{2\delta}\} \cup \{Y^{-\alpha+2\delta} D_k\}_k \cup \{Y^\gamma E_k\}_k \cup \\
&\{D_{k_1} E_{k_2}\}_{k_1, k_2} \cup \{Y^\delta D_k\}_k \cup \{Y^\beta E_k\}_k
\end{aligned}
$$

also contain a non-constant number of coefficients. For example, Crit_{se} contains $D_{k_1} E_{k_2}$ for any $k_1, k_2 \leq q_s$, where q_s is the number of simulation queries. However, there are only 12 “families” of critical coefficients, and the members of the same family (say $D_1 E_2$ and $D_7 E_2$) can be analyzed similarly.

For Crit_{se} to consist of different monomials and for $\mathsf{Crit}_{se} \cap \overline{\mathsf{Crit}}_{se}$, the new critical monomials $Y^{i_1} D_k^{i_2} E_k^{i_3}$ (see Table 2, the last 6 monomials) must be different from all other monomials. We say that $\boldsymbol{\Delta}$ is *ASE-friendly* if these conditions are satisfied. While the number of additional monomials in Crit and Coeff is huge, ascertaining that the new critical monomials are unique is relatively easy, even if tedious, since one needs to guarantee that for each fixed (i_2, i_3), if $Y^{i_1} D_k^{i_2} E_k^{i_3} \in \mathsf{Crit}_{se}$ and $Y^{i'_1} D_k^{i_2} E_k^{i_3} \in \mathsf{Coeff}_{se}$ then $i_1 \neq i'_1$. By inspection, one can establish that it means the following.

(a) When $i_2 = 1$ and $i_3 = 0$, we need $-\alpha + 2\delta \neq \delta$ (i.e., $\delta \neq \alpha$, which follows from the fact that $Y^{\beta+\delta} \in \mathsf{Crit}$ and $Y^{\alpha+\beta} \in \overline{\mathsf{Crit}}$) and $-\alpha + 2\delta, \delta \notin \{\alpha, \beta, -\alpha + \beta + \delta, \eta, -\alpha + \delta + \eta\}$.
This guarantees, say, that $Y^{-\alpha+2\delta} D_k$ (which is a critical monomial) is not equal to $Y^{-\alpha+\delta+\eta} D_k$.

(b) When $i_2 = 0$ and $i_3 = 1$, we need $\gamma \neq \beta$ and $\gamma, \beta \notin \{\alpha, -\alpha + 2\beta, -\alpha + \beta + \gamma, \delta, -\alpha + \beta + \delta, -\alpha + \gamma + \delta, 2\beta - \eta, \beta + \gamma - \eta, \beta + \delta - \eta, -\alpha + \gamma + \eta\}$.

(c) When $i_2 = 1$ and $i_3 = 1$, we need $0 \notin \{-\alpha + \beta, -\alpha + \delta, -\alpha + \eta\}$.

By simple but tedious case analysis, one can prove the following lemma.

Lemma 1. *If $\boldsymbol{\Delta}$ is soundness-friendly, then it is also ASE-friendly.*

Proof. **(a)** Here, $-\alpha + 2\delta \neq \delta$ (i.e., $\delta \neq \alpha$) follows from the fact that $Y^{\beta+\delta} \in \mathsf{Crit}$ and $Y^{\alpha+\beta} \in \overline{\mathsf{Crit}}$. Moreover, $-\alpha + 2\delta \neq \alpha$ and $\delta \neq \alpha$ follow since $\alpha \neq \delta$, $-\alpha + 2\delta \neq \beta$ follows since $Y^{2\delta} \in \mathsf{Crit}$ and $Y^{\alpha+\beta} \in \overline{\mathsf{Crit}}$, $\delta \neq \beta$ follows since $Y^{2\beta}, Y^{2\delta} \in \mathsf{Crit}$, $-\alpha + 2\delta \neq -\alpha + \beta + \delta$ follows since $\beta \neq \delta$, $\delta \neq -\alpha + \beta + \delta$ follows since $\alpha \neq \delta$, $-\alpha + 2\delta \neq \eta$ follows from $Y^{2\delta} \in \mathsf{Crit}$ and $Y^{\alpha+\eta} \in \overline{\mathsf{Crit}}$, $\delta \neq \eta$ follows from $Y^{\gamma+\delta}, Y^{\gamma+\eta} \in \mathsf{Crit}$, $-\alpha + 2\delta \neq -\alpha + \delta + \eta$ follows from $\delta \neq \eta$, $\delta \neq -\alpha + \delta + \eta$ follows form $Y^{\gamma+\eta} \in \mathsf{Crit}$ and $Y^{\alpha+\gamma} \in \overline{\mathsf{Crit}}$.

(b) Next, $\gamma \neq \beta$ follows from $Y^{2\beta}, Y^{\beta+\gamma} \in \mathsf{Crit}$, $\gamma \neq \alpha$ follows from $Y^{\beta+\gamma} \in \mathsf{Crit}$ and $Y^{\alpha+\beta} \in \overline{\mathsf{Crit}}$, $\beta \neq \alpha$ follows from $Y^{2\beta} \in \mathsf{Crit}$ and $Y^{\alpha+\beta} \in \overline{\mathsf{Crit}}$, $\gamma \neq -\alpha + 2\beta$ follows from $Y^{2\beta} \in \mathsf{Crit}$ and $Y^{\alpha+\gamma} \in \overline{\mathsf{Crit}}$, $\beta \neq -\alpha + 2\beta$ follows from $\alpha \neq \beta$, $\gamma \neq -\alpha + \beta + \gamma$ follows from $\alpha \neq \beta$, $\beta \neq -\alpha + \beta + \gamma$ follows from $\alpha \neq \gamma$, $\gamma \neq \delta$ follows from $Y^{\beta+\gamma}, Y^{\beta+\delta} \in \mathsf{Crit}$, $\beta \neq \delta$ is already proven, $\gamma \neq -\alpha + \beta + \delta$ follows from $Y^{\beta+\gamma} \in \mathsf{Crit}$ and $Y^{-\alpha+2\beta+\delta} \in \overline{\mathsf{Crit}}$, $\beta \neq -\alpha + \beta + \delta$ follows from $\alpha \neq \delta$, $\gamma \neq -\alpha + \gamma + \delta$ follows from $\alpha \neq \delta$, $\beta \neq -\alpha + \gamma + \delta$ follows from $Y^{\gamma+\delta} \in \mathsf{Crit}$ and $Y^{\alpha+\beta} \in \overline{\mathsf{Crit}}$, $\gamma \neq 2\beta - \eta$ follows from $Y^{2\beta}, Y^{\gamma+\eta} \in \mathsf{Crit}$, $\beta \neq 2\beta - \eta$ (i.e., $\beta \neq \eta$) follows from $Y^{\beta+\gamma}, Y^{\gamma+\eta} \in \mathsf{Crit}$, $\gamma \neq \beta + \gamma - \eta$ follows from $\beta \neq \eta$, $\beta \neq \beta + \gamma - \eta$ (i.e., $\gamma \neq \eta$) follows from $Y^{\beta+\delta} \in \mathsf{Crit}$ and $Y^{\beta+\gamma+\delta-\eta} \in \overline{\mathsf{Crit}}$, $\gamma \neq \beta + \delta - \eta$ follows from $Y^{\beta+\delta}, Y^{\gamma+\eta} \in \mathsf{Crit}$, $\beta \neq \beta + \delta - \eta$ follows from $\delta \neq \eta$, $\gamma \neq -\alpha + \gamma + \eta$ follows from $Y^{\gamma+\eta} \in \mathsf{Crit}$ and $Y^{\alpha+\gamma} \in \overline{\mathsf{Crit}}$, $\beta \neq -\alpha + \gamma + \eta$ follows from $Y^{\gamma+\eta} \in \mathsf{Crit}$ and $Y^{\alpha+\beta} \in \overline{\mathsf{Crit}}$.

(c) Finally, $\alpha \neq \beta$ and $\alpha \neq \delta$ is already known, and $\alpha \neq \eta$ follows from $Y^{\gamma+\eta} \in \mathsf{Crit}$ and $Y^{\alpha+\gamma} \in \overline{\mathsf{Crit}}$. □

Theorem 2. *Let $\mathcal{T}_\iota^x$ and $\mathcal{T}_\iota^y$ be as in Theorem 1. Let $\mathcal{I}_{\mathsf{qap}} = (\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=1}^m)$ be a QAP instance. Let $\mathsf{S}_{\mathsf{qap}}$ be the SNARK in Fig. 3. Assume $\boldsymbol{\Delta}$ is soundness-friendly. Assume $u_j(X)$, $j \leq m_0$, are linearly independent from each other and from other polynomials u_i for $i > m_0$. $\mathsf{S}_{\mathsf{qap}}$ is non-black-box ASE in the AGM under the $(\mathcal{T}_1^x, \mathcal{T}_2^x)$-PDL and $(\mathcal{T}_1^y, \mathcal{T}_2^y)$-PDL assumptions.*

Proof. The ASE proof is similar to the knowledge-soundness proof. There are two main differences. First, $\mathcal{B}$ also has to emulate Sim to $\mathcal{A}$. Second, the analysis of the abort probability is different due to the larger number of critical monomials.

Hence, we refer to the proof of Theorem 1, except that the full description of $\mathcal{B}^z$ in Fig. 5 contains also the emulation of simulation queries. (Obviously, there is more going on behind the scene: for example, $\mathcal{V}$ is defined differently, and $\boldsymbol{X}^*$ includes $\boldsymbol{D}, \boldsymbol{E}$, but we already explained that part.)

The only thing left to do now is the different (more complicated) analysis of the abort probability.

Analysis of the abort probability in step (*). Assume that $\mathcal{V}(\boldsymbol{X}) = 0$, thus also $\mathcal{V}_{Y^{i_1}\dots}(\boldsymbol{X}^*) = 0$ for all critical monomials (see Table 2). From the coefficient of

```
$\boxed{\mathcal{B}^y(\mathsf{p}, \mathbf{R}, \mathbb{x}_{\mathcal{B}})}$ $\mathcal{B}^x(\mathsf{p}, \mathbf{R}, \mathbb{x}_{\mathcal{B}})$
$[\boldsymbol{d}]_1 \leftarrow \emptyset; [\boldsymbol{e}]_2 \leftarrow \emptyset; \zeta \leftarrow 0;$
Run $\mathcal{B}^z(\mathsf{p}, \mathbf{R}, \mathbb{x}_{\mathcal{B}})$ in Fig. 4, except give $\mathcal{A}$ also access to $\mathcal{O}_{\mathsf{Sim}}$;

$\mathcal{O}_{\mathsf{Sim}}((\sigma_j)_{j=1}^{m_0})$
$[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} \sigma_j [P_j(x, y) y^{-\eta}]_1;$
$\zeta \leftarrow \zeta + 1; \boxed{d_\zeta, e_\zeta \leftarrow_\$ \mathbb{Z}_p}; s'_\zeta, s''_\zeta, t'_\zeta, t''_\zeta \leftarrow_\$ \mathbb{Z}_p;$
$[d_\zeta]_1 \leftarrow s'_\zeta [x]_1 + t'_\zeta [1]_1; [e_\zeta]_2 \leftarrow s''_\zeta [x]_2 + t''_\zeta [1]_2; [\mathsf{a}]_1 \leftarrow [d_\zeta]_1; [\mathsf{b}]_2 \leftarrow [e_\zeta]_2;$
$[\mathsf{c}_s]_1 \leftarrow y^{-\alpha}([\mathsf{a} e_\zeta]_1 + y^\delta [\mathsf{a}]_1 + y^\gamma [e_\zeta]_1 - y^\eta [\mathsf{c}_p]_1);$ return $[q_{\iota \xi_\iota}]_\iota$;
```

Fig. 5. $\mathcal{B}(\mathsf{p}, \mathbf{R}, \mathbb{x}_{\mathcal{B}})$ in the proof of Theorem 2, and the emulation of $\mathcal{O}_{\mathsf{Sim}}$. $\boxed{\text{Full-boxed}}$ and dashed-boxed are defined as in Fig. 4.

$Y^{\gamma+\delta}$ of $\mathcal{V}$, we get $b_\delta = -a_\gamma/(a_\gamma + 1)$ and thus $a_\gamma, b_\delta \neq -1$. From the coefficients of $Y^{\gamma+\eta}$ and $Y^{2\delta}$, and since $a_\gamma, b_\delta \neq -1$, we get $b_\eta = 0$ and $a_\delta = 0$. Up to now, the proof looks similar to that of Theorem 1. The rest of the coefficients have to be handled differently.

From the coefficients of $Y^{\beta+\delta}$ and $Y^{\beta+\gamma}$, we get

$$u_a(X) = (a_\gamma + 1)(u(X) + \textstyle\sum_j \left(\sum_{k=1}^{m_0} \sigma_{kj}(r_b s_{a2k} - s_{c2k})\right) u_j(X)) \enspace ,$$
$$v_b(X) = (v(X) + \textstyle\sum_j \left(\sum_{k=1}^{m_0} \sigma_{kj}(r_b s_{a2k} - s_{c2k})\right) v_j(X))/(a_\gamma + 1) \enspace .$$

From the coefficient of $Y^{-\alpha+2\delta} D_k$, we get $s_{a2k} = 0$. From the coefficients of $Y^\gamma E_k$ and $D_k E_k$, we get $s_{c2k} = (a_\gamma + 1) s_{bk} = s_{a1k} s_{bk}$. Thus, for all k, either (i) $s_{bk} = s_{c2k} = 0$ or (ii) $s_{a1k} = a_\gamma + 1 \neq 0$ and $s_{c2k} = (a_\gamma + 1) s_{bk} \neq 0$.

If the case (i) holds for all k, then the first three polynomials $\hat{V}_{Y^i}$ in Table 2 are 0 and we are back to the knowledge-soundness case. One can then follow the remaining proof of Theorem 1, and obtain knowledge-soundness and ASE. Note that then, from the coefficient of $Y^\delta D_k$, it follows that also $s_{a1k} = 0$ for all k. Thus, the adversary did not benefit from the simulation queries.

Consider the case when at least for one k, (ii) holds. From the coefficient of $Y^\delta D_k$ of this k, we get $0 = r_b s_{a2k} + (b_\delta + 1) s_{a1k} - s_{c2k} = 1 - (a_\gamma + 1) s_{bk}$ and thus $s_{bk} = 1/(a_\gamma + 1)$. From the coefficient of $D_{k_1} E_{k_2}$ for any $k_1 \neq k_2$, we get $s_{a1k_1} s_{bk_2} = 0$. Thus, if some $s_{a1k_2} \neq 0$, then (since we are in the case (ii)) also $s_{bk_2} \neq 0$, and thus $s_{a1k_1} = s_{bk_1} = s_{c2k_1} = 0$ for all $k_1 \neq k_2$. Hence, $r_b s_{a2k_1} - s_{c2k_1} = 0$, and thus making the k_1th simulation query, $k_1 \neq k_2$, does not help the adversary. Thus, we can assume that $\mathcal{A}$ makes only one query, say the k_2th one, with the simulator input $\boldsymbol{\sigma} = (\sigma_j)$.

From the coefficient of $Y^\beta E_{k_2}$, we get $s_{bk_2} u_a(X) = 0$. Since $s_{bk_2} \neq 0$ and $a_\gamma \neq -1$, $\sum_{j \leq m_0} (\sigma_j (r_b s_{a2k_2} - s_{c2k_2}) + \mathbb{z}_j) u_j(X) + \sum_{j > m_0} \tilde{\mathbb{z}}_j u_j(X) = 0$. Since $s_{a2k_2} = 0$ and $s_{c2k_2} = 1$, $\sum_{j \leq m_0} (\mathbb{z}_j - \sigma_j) u_j(X) + \sum_{j > m_0} \tilde{\mathbb{z}}_j u_j(X) = 0$. Since $u_j(X)$ are linearly independent for $j \leq m_0$, it means $\mathbb{z}_j = \sigma_j$ for all $j \leq m_0$. Thus, $\mathcal{A}$ made the only simulation query on the same input that she used to cheat on, and thus this corresponds to a SASE but not an ASE attack. Hence, $\mathcal{A}$ did not succeed in an ASE attack and thus $\chi(X) = 0$. □

On Lower-Bound of [25]. Groth and Maller proved that in any SASE SNARK, the verifier has to perform two verification equations. Our result does not contradict it since we achieve ASE, a weaker property. (Similarly, the ASE SNARK of [6] has only one verification equation.)

5 Subversion-Zero Knowledge

In a subversion zero-knowledge (Sub-ZK) SNARK [7,1,17,3], the goal is to obtain zero-knowledge even if the CRS creator cannot be trusted. As noted in [2], one has to use non-falsifiable assumptions to achieve Sub-ZK. Next, we show that $\mathsf{S_{qap}}$ is Sub-ZK (under the BDH-KE assumption), assuming $\boldsymbol{\Delta}$ satisfies some additional requirements. The same argument applies in the case of all other new SNARKs. In particular, five different choices of $\boldsymbol{\Delta}$ in Table 3 result in a Sub-ZK SNARK; this includes the setting of Eq. (11).

According to the blueprint of [1,17,3], one can follow the next steps to make a SNARK subversion-resistant:

1. Construct a public CRS verification algorithm CV that checks that the CRS is correct (that is, it corresponds to *some* trapdoor td).
2. To facilitate public verification, this can mean adding new elements to the CRS. Let us denote the set of new elements by $\mathsf{crs_{CV}}$. If $\mathsf{crs_{CV}}$ is non-empty, then one must reprove knowledge-soundness and/or simulation-extractability, taking $\mathsf{crs_{CV}}$ into account.
3. Under a reasonable knowledge assumption, extract td from the CRS.
4. Show how to simulate the argument by using the extracted trapdoor.

This blueprint is formalized in [3], and we refer the reader to it for a further discussion, including proof that trapdoor-extractability and ZK suffice to get Sub-ZK. Moreover, for trapdoor-extractability, it suffices to have CRS-verifiability and a strong enough extractability assumption.

Let us show that under the setting in Eq. (11) with CRS as in Eq. (12), the correctness (that is, that it corresponds to *some* choice of trapdoors) of the CRS of $\mathsf{S_{qap}}$ can be verified by using a public CV algorithm. Modelling after [1,3], CV needs to check that (1) all trapdoors belong to correct domain (for example, it checks $y \in \mathbb{Z}_p^*$ by checking that $[y]_1 \neq [0]_1$), and that (2) all CRS elements $[f(\boldsymbol{x})]_\iota$, where f is a public rational function, are correctly computed from trapdoors $\boldsymbol{x}$. The last verification can be done step by step, starting from simpler (for example, lower-degree) functions and then using the already verified values as helpers to verify more complex functions.

We present the CRS verification algorithm CV for $\mathsf{S_{qap}}$ in Fig. 6. Note that here we assume $u_j(X) = \sum u_{ji} X^i$, $v_j(X) = \sum v_{ji} X^i$, and $w_j(X) = \sum w_{ji} X^i$. It is easy (though tedious) to check that CV suffices to check that the CRS of $\mathsf{S_{qap}}$ has been correctly generated but for the following two exceptions:

```
$\mathsf{CV}(\mathsf{crs}, \mathsf{crs}_{\mathsf{CV}})$:
Check that the following holds:
    // Trapdoors are not 0 and $x^n \neq 1$:
  $[xy^\beta]_1 \neq [0]_1; [Z(x)y^{2\beta-\alpha}]_1 \neq [0]_1$;
    // The bracketed elements $y^4 = y^\delta, z, x^j y^\beta = x^j y$ in $\mathbb{G}_1$ and $\mathbb{G}_2$ are consistent:
  $[y^\delta]_1 \bullet [1]_2 = [1]_1 \bullet [y^\delta]_2$;
  for $j = 0$ to $n-1$ do $[x^j y^\beta]_1 \bullet [1]_2 = [1]_1 \bullet [x^j y^\beta]_2$; endfor
    // Degrees of $y^i$ are consistent: depends on $\boldsymbol{\Delta}$; recall $\alpha = 0, \beta = -2, \gamma = -3, \delta = 7, \eta = 2$
  $\boxed{[1]_1 \bullet [y^\eta]_2 = [y]_1 \bullet [y]_2; [y^\beta]_1 \bullet [y^\eta]_2 = [1]_1 \bullet [1]_2; [y^\gamma]_1 \bullet [y]_2 = [y^\beta]_1 \bullet [1]_2;}$
  $\boxed{[y^\gamma]_1 \bullet [y^\delta]_2 = [y^\eta]_1 \bullet [y^\eta]_2;}$
    // Degrees of $x^j y^\beta = x^j y$ are consistent:
  for $j = 1$ to $n-2$ do $[x^{j+1} y^\beta]_1 \bullet [y^\beta]_2 = [x^j y^\beta]_1 \bullet [xy^\beta]_2$; endfor
    // $x^j Z(x) y^{2\beta-\alpha} = x^j Z(x) y^2$ are consistent:
  $[Z(x)y^{2\beta-\alpha}]_1 \bullet [1]_2 = \left[\boxed{xy^{\beta-\alpha}}\right]_1 \bullet [x^{n-1}y^\beta]_2 - \left[\boxed{y^{\beta-\alpha}}\right]_1 \bullet [y^\beta]_2$;
  for $j = 0$ to $n-3$ do $[x^{j+1} Z(x) y^{2\beta-\alpha}]_1 \bullet [y^\beta]_2 = [x^j Z(x) y^{2\beta-\alpha}]_1 \bullet [xy^\beta]_2$; endfor
    // Polynomials $P_j(x,y)y^{-\eta} = u_j(x)y^{\beta-\eta+\delta} + v_j(x)y^{\beta-\eta+\gamma} + w_j(x)y^{2\beta-\eta}$ are consistent:
  for $j = 1$ to $m_0$ do
    $[P_j(x,y)y^{-\eta}]_1 \bullet [y^\eta]_2 =$
      $\sum_{i=0}^{n-1} u_{ji}[x^i y^\beta]_1 \bullet [y^\delta]_2 + [y^\gamma]_1 \bullet \sum_{i=0}^{n-1} v_{ji}[x^i y^\beta]_2 + \sum_{i=0}^{n-1} w_{ji}[x^i y^\beta]_1 \bullet [y^\beta]_2$;
  endfor
    // Polynomials $P_j(x,y)y^{-\alpha} = u_j(x)y^{\beta-\alpha+\delta} + v_j(x)y^{\beta-\alpha+\gamma} + w_j(x)y^{2\beta-\alpha}$ are consistent:
  for $j = m_0 + 1$ to $m$ do
    $[P_j(x,y)y^{-\alpha}]_1 \bullet [1]_2 =$
      $\sum_{i=0}^{n-1} u_{ji}[x^i y^\beta]_1 \bullet \left[\boxed{y^{-\alpha+\delta}}\right]_2 + \left[\boxed{y^{-\alpha+\gamma}}\right]_1 \bullet \sum_{i=0}^{n-1} v_{ji}[x^i y^\beta]_2 +$
      $\sum_{i=0}^{n-1} w_{ji}[x^i y^\beta]_1 \bullet \left[\boxed{y^{\beta-\alpha}}\right]_2$;
  endfor
```

Fig. 6. The CRS verification algorithm CV in $\mathsf{S}_{\mathsf{qap}}$. $\boxed{\text{dashed}}$ elements are guaranteed to be in the CRS if $\alpha = 0$. $\boxed{\text{dotted}}$ equalities and the integer exponents in comments depend on the concrete of $\boldsymbol{\Delta}$ (namely, Eq. (11))

1. The $\boxed{\text{dashed}}$ elements are not guaranteed to be in the CRS unless $\boldsymbol{\Delta}$ is well-chosen. A simple way of solving this problem is to set $\alpha \leftarrow 0$. This is not too restrictive, since 12 out of 14 $\boldsymbol{\Delta}$-s in Table 3 have $\alpha = 0$.
2. One must verify that, for some ι such that $[y^\kappa]_\iota$ is in the CRS, y^κ is correctly computed, where $\kappa \in \{\beta, \gamma, \delta, \eta\}$. (Recall that $\alpha = 0$.)
   This involves adding a small number of pairing equations of type $[y^i]_1 \bullet [y^j]_2 = [y^k]_2 \bullet [y^\ell]_2$, such that each equation introduces exactly one new degree (either i, j, k or ℓ) and reuses three degrees that are already "verified". For example, in the first equation $i, j, k \in \{0, 1\}$. In this case, one can use pairings to establish the correctness of y^ℓ for $\ell \in \{-1, 0, 1, 2\}$. This means we need to put additional restrictions on $\boldsymbol{\Delta}$.

Lemma 2. *From the* 14 *settings of* $\boldsymbol{\Delta}$ *in Table 2, the five ones marked with* ✓ *are CRS-verifiable.*

Proof. Intuitively, we just need to describe how we (manually) found which of the choices of $\boldsymbol{\Delta}$ from Table 3 satisfy both above restrictions. As already mentioned, the first restriction is straightforward to satisfy. Now, assuming that $\alpha = 0$, consider two cases of ℓ from the first pairing equation in the second restriction:

1. $\ell = -1$. In the second pairing equation, then (say) $i, j, k \in \{-1, 0, 1\}$. In this case, one can establish the correctness of y^ℓ for $\ell \in [-3, 3]$.
   To solve this, we look at the possible $\boldsymbol{\Delta}$-s in Table 3, such that $\alpha = 0$ and one of $\beta, \gamma, \delta, \eta$ is equal to either -1 or 2. This only weeds out one additional possibility (namely, $\boldsymbol{\Delta} = (0, -3, 5, 7, 1)$).
   In the case one of $\beta, \gamma, \delta, \eta$ is equal to -1, we will look at the cases when one of the three other values $\kappa \in \{\beta, \gamma, \delta, \eta\}$ belongs to $[-3, 3]$. This leaves still several possibilities, $\boldsymbol{\Delta} \in \{(0, -1, 6, -4, 1), (0, -1, 7, -4, 1), (0, -1, 7, -5, 1), \dots\}$.
   However, in only one case, $\boldsymbol{\Delta} = (0, 3, -5, -7, -1)$, it is possible to verify all 5 values y^κ for $\kappa \in \{\alpha, \beta, \gamma, \delta, \eta\}$: namely, by checking that (say) $[y^\eta]_1 \bullet [y]_2 = [1]_1 \bullet [1]_1$, $[y^\eta]_1 \bullet [y^\beta]_2 = [y]_1 \bullet [y]_1$, $[y^\gamma]_2 \bullet [y^\beta]_1 = [y^\eta]_1 \bullet [y^\eta]_1$, and $[y^\delta]_2 \bullet [y]_1 = [y^\gamma]_1 \bullet [y^\eta]_1$.
2. $\ell = 2$. Then, in the second equation, one can establish the correctness of y^ℓ for $\ell \in [-2, 3]$. W.l.o.g., we assume that $\ell \neq -1$ (otherwise we are back to the previous case). Thus, after two verification equations, we have the following cases left: $\boldsymbol{\Delta} \in \{(0, -2, -3, 7, 2), (-2, 6, 7, 2), (2, -6, -7, 2), (2, 3, -7, -2)\}$.
   A simple inspection establishes that in all the three cases, where both -2 and 2 are present, one has an efficient CRS-verification algorithm. For example, one can take $\boldsymbol{\Delta} = (-2, -3, 7, 2)$, that is, the setting in Eq. (11). Then, one has to verify that $[1]_1 \bullet [y^\eta]_2 = [y]_1 \bullet [y]_2$, $[y^\beta]_1 \bullet [y^\eta]_2 = [1]_1 \bullet [1]_2$, $[y^\gamma]_1 \bullet [y]_2 = [y^\beta]_1 \bullet [1]_2$, and $[y^\gamma]_1 \bullet [y^\beta]_2 = [y^\eta]_1 \bullet [y^\eta]_2$. (Those are the dotted equations in Fig. 6.) □

For the sake of concreteness, we recommend to choose $\boldsymbol{\Delta}$ as in Eq. (11). However, one can use any of the five checkmarked choices in Table 3.

One can significantly speed up CV in Fig. 6 by using batching techniques, as explained in [1,3]. CV for other new SNARKs are essentially the same, modulo some simplifications due to say $w_i(X) = 0$ in the case of the QSP.

Trapdoor-Extractability and Sub-ZK. Trapdoor-extractability [3] means that if CV accepts the CRS, then one can extract the simulation trapdoor. In all new SNARKs, the simulation trapdoor is equal to $\mathsf{td} = y$ since Sim does not use x. Clearly, in all new SNARKs, if CV accepts crs, one can use the BDH-KE assumption to extract y. Thus, BDH-KE guarantees trapdoor-extractability, and the CRS-verifiability and the trapdoor-extractability together guarantee that one can extract td. Hence, by the general result of [3], all new SNARKs are Sub-ZK, assuming that their CRS is verifiable and that the BDH-KE holds.

Corollary 1. *Under the five ✓-ed settings of $\boldsymbol{\Delta}$ in Table 2,* $\mathsf{S}_{\mathsf{qap}}$ *is statistically composable Sub-ZK under the BDH-KE assumption.*

Acknowledgment. We thank Markulf Kohlweiss, Janno Siim, and Mikhail Volkhov for helpful comments. The author was partially supported by the Estonian Research Council grant (PRG49).

References

1. Abdolmaleki, B., Baghery, K., Lipmaa, H., Zajac, M.: A subversion-resistant SNARK. In: ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 3–33
2. Abdolmaleki, B., Lipmaa, H., Siim, J., Zajac, M.: On QA-NIZK in the BPK model. In: PKC 2020, Part I. LNCS, vol. 12110, pp. 590–620
3. Abdolmaleki, B., Lipmaa, H., Siim, J., Zajac, M.: On Subversion-Resistant SNARKs. J. Cryptology **34**(3) (2021) pp. 1–42
4. Abdolmaleki, B., Ramacher, S., Slamanig, D.: Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In: ACM CCS 2020, pp. 1987–2005
5. Baghery, K.: On the efficiency of privacy-preserving smart contract systems. In: AFRICACRYPT 19. LNCS, vol. 11627, pp. 118–136
6. Baghery, K., Kohlweiss, M., Siim, J., Volkhov, M.: Another Look at Extraction and Randomization of Groth's zk-SNARK. In: FC 2021 (1). LNCS, vol. 12674, pp. 457–475
7. Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: Security in the face of parameter subversion. In: ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 777–804
8. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474
9. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: TCC 2013. LNCS, vol. 7785, pp. 315–333
10. Bowe, S., Gabizon, A.: Making groth's zk-SNARK simulation extractable in the random oracle model. Cryptology ePrint Archive, Report 2018/187 (2018) `https://eprint.iacr.org/2018/187`.
11. Campanelli, M., Gennaro, R., Goldfeder, S., Nizzardo, L.: Zero-knowledge contingent payments revisited: Attacks and payments for services. In: ACM CCS 2017, pp. 229–243
12. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145
13. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square span programs with applications to succinct NIZK arguments. In: ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 532–550
14. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: CRYPTO 2001. LNCS, vol. 2139, pp. 566–598
15. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631
16. Fauzi, P., Lipmaa, H., Siim, J., Zajac, M.: An efficient pairing-based shuffle argument. In: ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 97–127
17. Fuchsbauer, G.: Subversion-zero-knowledge SNARKs. In: PKC 2018, Part I. LNCS, vol. 10769, pp. 315–347

18. Fuchsbauer, G.: WI is not enough: Zero-knowledge contingent (service) payments revisited. In: ACM CCS 2019, pp. 49–62
19. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62
20. Gabizon, A.: On the security of the BCTV pinocchio zk-SNARK variant. Cryptology ePrint Archive, Report 2019/119 (2019) https://eprint.iacr.org/2019/119.
21. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645
22. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340
23. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326
24. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728
25. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In: CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 581–612
26. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. Cryptology ePrint Archive, Report 2017/540 (2017) https://eprint.iacr.org/2017/540.
27. Kohlweiss, M., Maller, M., Siim, J., Volkhov, M.: Snarky Ceremonies. In: ASIACRYPT 2021 (3). LNCS, vol. 13092, pp. 98–127
28. Kohlweiss, M., Zajac, M.: On Simulation-Extractability of Universal zkSNARKs. Technical Report 2021/511, IACR (2021) https://ia.cr/2021/511.
29. Kosba, A.E., Zhao, Z., Miller, A., Qian, Y., Chan, T.H., Papamanthou, C., Pass, R., Shelat, A., Shi, E.: C∅C∅: A Framework for Building Composable Zero-Knowledge Proofs. Technical Report 2015/1093, International Association for Cryptologic Research (2015) https://ia.cr/2015/1093.
30. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189
31. Lipmaa, H.: Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In: ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 41–60
32. Lipmaa, H.: Simulation-Extractable ZK-SNARKs Revisited. Technical Report 2019/612, IACR (2019) https://ia.cr/2019/612.
33. Lipmaa, H.: A Unified Framework for Non-Universal SNARKs. Technical report, IACR (2021) https://ia.cr/2021.
34. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: CRYPTO 2002. LNCS, vol. 2442, pp. 111–126
35. Parno, B.: A note on the unsoundness of vnTinyRAM's SNARK. Cryptology ePrint Archive, Report 2015/437 (2015) https://eprint.iacr.org/2015/437.
36. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, pp. 238–252
37. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS, pp. 543–553