Simulation-Extractable SNARKs Revisited

July 13, 2019,

Helger Lipmaa[1,2]

[1] University of Tartu, Tartu, Estonia
[2] Simula UiB, Bergen, Norway
helger.lipmaa@gmail.com

Abstract. The most efficient SNARKs (e.g., Groth, 2016) have a brittle and difficult-to-verify knowledge-soundness proof in the generic model. This makes it nontrivial to modify such SNARKs to, e.g., satisfy simulation-extractability or to implement some other language instead of QAP (Quadratic Arithmetic Program). We propose knowledge-sound and non-black-box strong any-simulation-extractable (SASE) SNARKs for QAP that is designed to have a relatively simple security proof. The knowledge-sound SNARK is similar to the mentioned SNARK of Groth, except it has fewer trapdoors. To achieve SASE, we add to it a one-time simulation-extractable QA-NIZK for a subspace language. Moreover, we give a simple characterization of languages like SAP, SSP, and QSP in the terms of QAP and show how to modify the SNARK for QAP correspondingly. The only prior published efficient simulation-extractable SNARK was for the somewhat impractical SAP language. We prove security under subversion algebraic knowledge assumptions that are a concrete version of the (subversion) algebraic group model.

Keywords: Algebraic group model, NIZK, non-black-box, QAP, QSP, SNARK, SAP, SSP, simulation-extractability, zero-knowledge

1 Introduction

Zero-knowledge proof systems [GMR85] are fundamental for the theory and applications of cryptography. In particular, zero-knowledge proof systems are used to guarantee that participants of some protocol follow the protocol correctly. For zero-knowledge proof systems to be used in practice, one needs an "efficient" zero-knowledge proof system that satisfies "reasonable" security definitions under "reasonable" cryptographic and trust assumptions. Due to their performance and versatility, zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs, [Gro10,BCCT12,Lip12,GGPR13,PHGR13,BCCT13,Gro16]) have become one of the most widely researched and deployed proof systems, in particular because of their applicability in verifiable computation [PHGR13] and anonymous cryptographic currencies [DFKP13,BCG+14]. The mentioned zk-SNARKs are knowledge-sound in the CRS model [BFM88].

It is difficult to design SNARKs, and it is easy for even well-established research groups to err in such an endeavor (see, e.g., [Par15,CGGN17,Gab19] for

related cryptanalysis). One explanation for this is that for the proof system to be secure, one needs to carefully designed the constant number of proof elements and verification equations o that they satisfy a number of properties:

First, they need to encode an NP language. The most widely used language is that of a quadratic arithmetic program (QAP, [GGPR13]) which corresponds to the rank-1 quadratic constraint system of the popular `libSNARK` library. Other related languages are square arithmetic programs (SAP, [Gro16,GM17]), quadratic span programs (QSP, [GGPR13,Lip13]), and square span programs (SSP, [DFGK14]). Here, QSP and SSP (resp., SAP and QAP) are convenient in the case one works with Boolean (resp., arithmetic) circuits.

Second, for optimal efficiency, the NP witness and the argument need to be encoded into the smallest number of proof elements and verified via the smallest number of verification equations possible. This creates a new set of design constraints, and several (tight) lower bound are known, [Gro16,GM17].

Third, throughout this process, one needs to assure that the SNARK remains (at least) knowledge-sound and zero-knowledge. Due to known impossibility results [GW11], one has to use non-falsifiable assumptions like the knowledge assumptions [Dam92]. To facilitate better efficiency, the most efficient zk-SNARKs like [Gro16] are proven to be knowledge-sound in the generic model. Generic model proofs often require one to derive soundness from a solution of a complicated system of polynomial equations. Moreover, there exist constructions that are secure in the generic group model but cannot be instantiated given any efficient instantiation of the group encoding [Fis00,Den02].

Fourth, sometimes, knowledge-soundness is not sufficient and one desires to achieve simulation-extractability (SE, [Sah99,DDO$^+$01,GM17]). SE SNARKs guarantee that knowledge-soundness holds even after the adversary has seen many simulated proofs, a property that is needed in many applications including UC-security [Can01].

It has been studied how to achieve UC-security for SNARKs. Kosba *et al.* [KZM$^+$15] constructed a black-box simulation-extractable version of SNARKs; black-box simulation-extractability is sufficient to obtain UC-security, [Gro06]. However, their transformation results in quite a large overhead and, in particular, results in a linear-size commitment. Alternatively, Groth and Maller [GM17] proposed a non-black-box strong any-simulation-extractable (SASE) SNARK that is only slightly less efficient than the most efficient knowledge-sound SNARK of Groth [Gro16]. However, their SNARK is based on the SAP language [Gro16,GM17] and thus has a blowup of approximately two times in circuit size compared to the QAP language. (This is since SAP has an efficient reduction from arithmetic circuits that have squaring gates instead of general multiplication gates, [Gro16,GM17].) They also proved that their construction achieved the lower bound for the argument length for SASE SNARKs. While SASE is not sufficient to obtain UC-security, it is clearly a stronger security notion than knowledge-soundness. Based on this observation, Baghery [Bag19] recently noticed that a much simpler transformation is needed to obtain UC-security based on SASE SNARKs. However, due to the use of SAP,

this transformation is twice as costly when using the Groth-Maller SNARK as compared to (yet unknown) SASE SNARKs for QAP.

No other simulation-extractable SNARKs are known at this moment (except [BG18] that works in the random-oracle model), not even ones that are just non-black-box ASE (any-simulation-extractable, allows an adversary after seeing simulation queries to modify a valid argument to a different valid argument for the same statement) or non-black-box TSE (any-simulation-extractable, allows an adversary after seeing simulation queries to *true statement* to modify a valid argument to a different valid argument for the same statement). This brings us to the main question of this paper:

> *Is it possible to construct a general SNARK for a multitude of languages (like QAP, SAP, QSP, and SSP) that would simultaneously (i) satisfy SASE, (ii) have a simple soundness proof that does not use the whole power of the generic model, and (iii) be almost as efficient as the most efficient known knowledge-sound SNARKs.*

Our Contributions. We answer positively to the main question. The new knowledge-sound zk-SNARK $\mathsf{S}_{\mathsf{qap}}$ for QAP is similar to Groth's SNARK [Gro16] while the SASE version $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ of it is obtained from it by well-motivated modifications. Based on a simple observation about algebraic relations (summarized in Table 2) between QAP and other languages, we modify both $\mathsf{S}_{\mathsf{qap}}$ and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ to cover SAP, QSP, and SSP. See Table 1 for an efficiency comparison.[3]

Since the new SASE SNARKs are tag-based, we use (tautological) knowledge assumptions instead of the full-blown generic group model (GGM, [Nec94,Sho97]) to prove knowledge-soundness and SASE. We motivate these knowledge assumptions by analysing the algebraic group model (AGM, [FKL18]): essentially, AGM states that for any efficient algorithm $\mathcal{A}$ that on an input vector $[\boldsymbol{x}]_\iota$ of group elements outputs an output vector $[\boldsymbol{y}]_\iota$ of group elements (we use here the bracket notation of [EHK$^+$13]), there exists an efficient extractor $\mathsf{Ext}_{\mathcal{A}}$ that outputs a matrix $\boldsymbol{N}$ such that $\boldsymbol{y} = \boldsymbol{N}\boldsymbol{x}$.

A tautological *algebraic knowledge (AK) assumption*, see Section 3, states that the same holds only for a concrete distribution of $[\boldsymbol{x}]_\iota$ (e.g., the distribution of correctly formed CRSs). In Section 3, we will also study a "subversion" [BFS16] version SAK of the AK assumption: it is essentially an AGM version of the subversion generic model [SPMS02,BFS16,ABLZ17] where we require the extractor to output a vector of group elements $[\boldsymbol{q}]_1$ and a matrix $\boldsymbol{N}$, such that $\boldsymbol{y} = \boldsymbol{N}\left(\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{q}\end{smallmatrix}\right)$. Here, $[\boldsymbol{q}]_1$ are elements for which the adversary does not know discrete logarithm. In the new SNARKs, to be able to rely on an SAK assumption, we need $[\boldsymbol{q}]_1$ to come from a distribution of high min-entropy.

[3] We emphasize that it is only fair to compare SNARKs for the same language; to compare SNARKs for different languages, one also has to take into account the complexity of the reduction from circuits to these languages. Note that [Lip13] only described a reduction from Boolean circuits to QSP and a linear PCP [BCI$^+$13] for QSP, leaving out cryptographic details of constructing a SNARK.

Table 1. Efficiency comparison of QAP/SAP/SSP/QSP-based SNARKs. m, n (or n^*), and m^* denote the number of wires, gates, and constraints in the solutions. "$\mathfrak{e}_\iota$" ("$\mathfrak{m}_\iota$") denotes exponentiation (multiplication) 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 $+(m_0+3)\mathfrak{g}_1$.

Π	security	$\lvert\mathsf{crs}\rvert$	P computation	$\lvert\pi\rvert$	V computation
QAP-based (arithmetic circuit, with n gates), $m^* = m$					
[Gro16]	KS	$(m+2n)\mathfrak{g}_1 + n\mathfrak{g}_2$	$(m+3n)\mathfrak{e}_1 + n\mathfrak{e}_2$	$2\mathfrak{g}_1 + 1\mathfrak{g}_2$	$3\mathfrak{p} + m_0\mathfrak{e}_1$
$\mathsf{S}_{\mathsf{qap}}$ § 4	KS	$(m+2n)\mathfrak{g}_1 + n\mathfrak{g}_2$	$(m+3n)\mathfrak{e}_1 + n\mathfrak{e}_2$	$2\mathfrak{g}_1 + 1\mathfrak{g}_2$	$3\mathfrak{p} + m_0\mathfrak{e}_1$
$\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ § 4	SASE	$(m+3n)\mathfrak{g}_1 + n\mathfrak{g}_2$	$(m+4n)\mathfrak{e}_1 + n\mathfrak{e}_2$	$3\mathfrak{g}_1 + 1\mathfrak{g}_2$	$5\mathfrak{p} + (m_0+1)\mathfrak{e}_1$
SAP-based (arithmetic circuit, with n^* squaring gates): $u = v$, $n^* \approx 2n$, $m^* \approx 2m$					
[GM17]	SASE	$(m^*+2n^*)\mathfrak{g}_1 + n^*\mathfrak{g}_2$	$(m^*+2n^*)\mathfrak{e}_1 + n^*\mathfrak{e}_2$	$2\mathfrak{g}_1 + 1\mathfrak{g}_2$	$5\mathfrak{p} + m_0\mathfrak{e}_1$
$\mathsf{S}_{\mathsf{sap}}$ § 6	KS	$(m^*+2n^*)\mathfrak{g}_1 + n^*\mathfrak{g}_2$	$(m^*+2n^*)\mathfrak{e}_1 + n^*\mathfrak{e}_2$	$2\mathfrak{g}_1 + 1\mathfrak{g}_2$	$3\mathfrak{p} + m_0\mathfrak{e}_1$
$\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$ § 6	SASE	$(m^*+3n^*)\mathfrak{g}_1 + n^*\mathfrak{g}_2$	$(m^*+2n^*)\mathfrak{e}_1 + n^*\mathfrak{e}_2$	$2\mathfrak{g}_1 + 1\mathfrak{g}_2$	$5\mathfrak{p} + (m_0+4)\mathfrak{e}_1$
SSP-based (Boolean circuit with n gates): $u = v = w$, $n^* = m + n$					
[DFGK14]	KS	$(m+n^*)\mathfrak{g}_1 + n^*\mathfrak{g}_2$	$2m\mathfrak{m}_1 + n^*\mathfrak{e}_1 + m\mathfrak{m}_2$	$3\mathfrak{g}_1 + 1\mathfrak{g}_2$	$6\mathfrak{p} + m_0\mathfrak{m}_1$
$\mathsf{S}_{\mathsf{ssp}}$ § 7	KS	$(m+2n^*)\mathfrak{g}_1 + n^*\mathfrak{g}_2$	$2m\mathfrak{m}_1 + n^*\mathfrak{e}_1 + m\mathfrak{m}_2$	$2\mathfrak{g}_1 + 1\mathfrak{g}_2$	$3\mathfrak{p} + m_0\mathfrak{m}_1$
$\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ § 7	SASE	$(m+3n^*)\mathfrak{g}_1 + n^*\mathfrak{g}_2$	$3m\mathfrak{m}_1 + n^*\mathfrak{e}_1 + m\mathfrak{m}_2$	$2\mathfrak{g}_1 + 1\mathfrak{g}_2$	$5\mathfrak{p} + (m_0+4)\mathfrak{m}_1$
QSP-based (Boolean circuit with n gates): $w = 0$, $n^* \approx 14n$ [Lip13]					
[Lip13]	KS	–	–	–	–
$\mathsf{S}_{\mathsf{qsp}}$ § 8	KS	$(m^*+2n^*)\mathfrak{g}_1 + n^*\mathfrak{g}_2$	$4m^*\mathfrak{m}_1 + n^*\mathfrak{e}_1 + m^*\mathfrak{m}_2$	$2\mathfrak{g}_1 + 1\mathfrak{g}_2$	$3\mathfrak{p} + m_0^*\mathfrak{m}_1$
$\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ § 8	SASE	$(m^*+3n^*)\mathfrak{g}_1 + n^*\mathfrak{g}_2$	$5m^*\mathfrak{m}_1 + n^*\mathfrak{e}_1 + m^*\mathfrak{m}_2$	$3\mathfrak{g}_1 + 1\mathfrak{g}_2$	$5\mathfrak{p} + (m_0^*+1)\mathfrak{m}_1$

Similarly to subversion generic model and subversion algebraic model, a SAK assumption can explain the absence of attacks on existing efficient cryptographic protocols as they are, without having to decrease efficiency to obtain security under more standard assumptions like the knowledge-of-exponent assumptions [Dam92] or falsifiable assumptons. On the other hand, using a SAK assumption as compared to the generic model enables one to handle a larger variety of protocols (e.g., tag-based or protocols where one employs hashing from group elements to integers) and avoids some of the criticisms against the generic model [Fis00,Den02]. Thus, arguably, SAK assumptions hit a sweet spot, being minimal assumptions to prove security of maximally efficient protocols.

In Section 4, we propose a knowledge-sound zk-SNARK $\mathsf{S}_{\mathsf{qap}}$ for QAP. Recall that that the prover is honest (the statement belongs to the QAP language) iff $\chi(X) := u(X)v(X) - w(X) - h(X)\ell(X) = 0$ (see [GGPR13]) for some polynomial $h(X)$, where the polynomials $u(X)$, $v(X)$, and $w(X)$ depend on the concrete circuit and on the witness the prover is using, $\ell(X)$ is a public fixed polynomial.

We consider polynomials $A(X,Y)$, $B(X,Y)$ ("commitments" to $u(X)$ and $v(X)$, respectively), and $C(X,Y) = A(X,Y)B(X,Y)$, such that the coefficient of Y^κ (for a κ fixed later) in $C(X,Y)$ is $u(X)v(X) - w(X) = h(X)\ell(X)$ for some $h(X)$ iff the prover is honest, i.e., $\chi(X) = 0$. One can guarantee that $\chi(X) = 0$

in the case of an algebraic adversary by inserting to the CRS elements of type $[f(x)y^\kappa]_1$ only for polynomials $f(X)$ that divide by $\ell(X)$. On top of it, $\mathsf{S}_{\mathsf{qap}}$ needs to guarantee that (i) $u(X)$, $v(X)$, and $w(X)$ use the same witness, and (ii) the public input encoded into $u(X)$ is correct.

We use aggressive optimization to get an as efficient SNARK as possible while not sacrificing (much) in the simplicity of the knowledge-soundness proof. Somewhat surprisingly, $\mathsf{S}_{\mathsf{qap}}$ is very similar to Groth's SNARK from EUROCRYPT 2016 [Gro16]. However, it uses only two trapdoors instead of five. This distinction is important: for example, as noted in [ABLZ17,Fuc18], only two out of Groth's five trapdoors are needed for simulation; thus, it is logical or at least aesthetic to drop the other trapdoors. In $\mathsf{S}_{\mathsf{qap}}$, we use well-chosen powers of one trapdoor Y as substitutes of four out of the five trapdoors of Groth's SNARK.

The way we choose the powers of Y is interesting by itself. Let $\boldsymbol{X}^* = (X, \ldots)$ be the vector of all indeterminates, except Y, that are relevant in the knowledge-soundness (or SASE) proof. This includes X, Y, indeterminates created by the adversary by using elliptic curve hashing [Ica09], and (in the case of SASE) indeterminates created by simulator queries. Then, $V(\boldsymbol{X}^*, Y) = \sum V_i(\boldsymbol{X}^*)Y^i$ for known polynomials $V_i(\boldsymbol{X}^*)$, where i is a linear combination of an initially undetermined integer vector $\boldsymbol{\Delta} = (\alpha, \beta, \ldots)$. We show that in the case of $\mathsf{S}_{\mathsf{qap}}$, a generic prover is honest iff $V_i(\boldsymbol{X}^*) = 0$ for six *critical* values i. We then choose $\boldsymbol{\Delta}$ so that the corresponding six critical linear combinations i are distinct from each other and from all other non-critical linear combinations j. Moreover, we choose $\boldsymbol{\Delta}$ so that the SNARK is relatively efficient. E.g., 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 collapse (this can shorten the CRS). Since this is a moderately hard optimization problem for humans, we here use an exhaustive computer search. Due to this, exponents in the resulting SNARKs may look somewhat obscure, e.g., $A(X, Y) = r_a Y^3 + u(X)Y^4$.

In Section 5, we modify $\mathsf{S}_{\mathsf{qap}}$ to make it SASE. We establish that for any k, a SASE adversary has an attack vector by setting $A(X, Y) = s_{a1k}D_k + \ldots$ for non-zero s_{a1k}, where D_k is indeterminate generated during the kth simulation query. We eliminate this attack vector by letting the prover to use an efficient quasi-adaptive NIZK (QA-NIZK, [JR13]) to prove that $A(X, Y)$ is in the span of correct monomials. Since our goal is simulation-extractability, the QA-NIZK has to be simulation-extractable. While known (unbounded) simulation-extractable QA-NIZKs are not very efficient, we observe that $\mathsf{S}_{\mathsf{qap}}$ itself (without the added QA-NIZK) already guarantees that an acceptable argument can only depend on the answer of a single simulation query. Thus, quite surprisingly, it is sufficient to use a more efficient one-time simulation-extractable (OTSE) QA-NIZK. It is known how to construct the latter efficiently [KW15] by using tags. We construct an even more efficient OTSE QA-NIZK by relying on the specifics of $\mathsf{S}_{\mathsf{qap}}$ and on non-falsifiable assumptions. Adding this QA-NIZK increases the complexity of the SNARK only slightly compared to $\mathsf{S}_{\mathsf{qap}}$ (see Table 1). Since the Groth-Maller SASE zk-SNARK [GM17] is for SAP, the new SASE SNARK is more efficient.

Importantly, $\mathsf{S}_{\mathsf{qap}}$ has a simple Sub-GBGM knowledge-soundness proof where only the value of the six critical coefficients of V matters. The SASE proof of the SASE SNARK relies only a few more extra coefficients. This should be compared to Groth's SNARK [Gro16] (resp., the Groth-Maller SNARK [GM17]) that has a very complicated knowledge-soundness (resp., SASE) proof.

As we mentioned before, $\mathsf{S}_{\mathsf{qap}}$ is very similar to Groth's SNARK. We obtain a simpler knowledge-soundness proof by assuming that the pairing is asymmetric. (Asymmetric pairings are much more efficient than symmetric pairings and thus strongly preferred in practice.) On the other hand, Groth proved knowledge-soundness in the case of a symmetric pairing, which results in $A(\boldsymbol{X}^*, Y)$, $B(\boldsymbol{X}^*, Y)$, and $C(\boldsymbol{X}^*, Y)$ having more terms and thus $V(\boldsymbol{X}^*, Y)$ having more critical coefficients. Thus, one corollary of our knowledge-sound proof is the (up to our knowledge, novel) observation that Groth's SNARK has a very simple knowledge-soundness proof given that one uses asymmetric pairings. Our goal was *not* to duplicate Groth's SNARK but to construct an efficient SNARK that has a simple knowledge-soundness proof. Thus, 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 emphasize that, on the other hand, $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is novel. In particular, none of the previous simulation-extractable SNARKs [GM17,BG18] used tags.

After that, we consider languages SAP [Gro16,GM17], SSP [DFGK14], and QSP [GGPR13,Lip13] that have also been used in the pairing-based SNARK literature. We explain their algebraic relation to QAP, which helps us to lift both $\mathsf{S}_{\mathsf{qap}}$ and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ to the setting of the corresponding languages. In the previous research, all four languages are handled separately and our (simple) relation seems to be novel. In some of the cases, we improve on the efficiency of previous known SNARKs for the same language. IWe propose the first known SASE SNARKs for QAP, SSP, and SAP. In particular, we propose the first known (efficient) SASE SNARKs for Boolean circuits in general. We omit precise descriptions of the reduction between circuits and corresponding languages, giving only a brief explanation and then referring to original papers.

In Section 6, we describe a SNARK $\mathsf{S}_{\mathsf{sap}}$ for the language SAP (Square Arithmetic Program, [GM17]). As mentioned before, SAP has an efficient reduction from arithmetic circuits that use squaring gates instead of multiplication gates. Thus, one has to take into account that such a circuit has usually two times more gates and wires, since in general one needs two squaring gates to implement a multiplication gate. This is a difference in the reduction overhead between circuits and the corresponding language, not in the cryptographic construction of the SNARK. Algebraically, SAP is a variant of QAP with $v(X) = u(X)$; thus, $\chi(X) = u(X)^2 - w(X) - h(X)\ell(X)$. Thus, $\mathsf{S}_{\mathsf{sap}}$ itself is as efficient as $\mathsf{S}_{\mathsf{qap}}$. Since the honest argument contains $([\mathsf{a}]_1, [\mathsf{b}]_2)$ with $\mathsf{a} = \mathsf{b}$, we obtain a SASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$ by using a simpler tranformation than we used in the case of QAP.

In Section 7, we describe a SNARK $\mathsf{S}_{\mathsf{ssp}}$ for the SSP language [DFGK14] that has efficient reduction from Boolean Circuit-SAT. Algebraically, SSP is a variant of QAP, where one sets $u(X) = v(X) = w(X)$. Then, $\chi(X) =$

Table 2. Algebraic relations between languages: restrictions on $u(X)$, $v(X)$, and $v(X)$

	$u(X)$	$v(X)$	$w(X)$
QAP	general	general	general
SAP	general	$= u(X)$	general
SSP	general	$= u(X)$	$= u(X)$
QSP	general	general	$= 0$

$u(X)(u(X)-1) - h(X)\ell(X)$. $\mathsf{S}_{\mathsf{ssp}}$ is approximately as efficient as the SSP-based SNARK of [DFGK14] but it has a shorter argument with more efficient verification (only one verification equation instead of two). The new SASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ for SSP uses the same transformation as $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$; no previous SASE SNARKs for SSP were known. We are not aware of a previous observation that one can design SNARKs for SSP by starting with a SNARK for QAP and then just setting $u(X) = v(X) = w(X)$.

We emphasize that an efficient SNARK for SSP is well-suited in applications where one needs to use Boolean circuits. They are also useful in applications like shuffle arguments [FLZ16,FLSZ17], and SSP has been used as the basis for falsifiable SNARKs with long commitments [DGP+19].

Finally, in Section 8, we design a SNARK for QSP (Quadratic Span Programs, [GGPR13,Lip13]). Algebraically, QSP is a variant of QAP, where one sets $w(X) = 0$. QSP is interesting in theory since one can construct a 2-query linear PCP for it, [BCI+13,Lip13]. However, the reduction from Boolean circuits to QSP is relatively complex, with the need to implement span-program-based gate checkers and error-correcting-code-based wire checkers [GGPR13,Lip13]. The new SASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ for QSP uses the same transformation as $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$. $\mathsf{S}_{\mathsf{qsp}}$ is again more efficient than previously known knowledge-sound SNARKs for QSP, while there was no previously known SASE SNARK for QSP.

To construct eight different SNARKs and verify their sets of critical coefficients and also soundness, we used computer algebra and exhaustive search. We believe that the soundness of the SNARKs is obvious, assuming that the variables $\boldsymbol{\Delta} = (\alpha, \beta, \ldots)$ have been chosen so that exponents of Y corresponding to the critical coefficients are different from all other exponents. However, finding *small* values of these variables seems to require exhaustive search — the number of non-zero coefficients of $V_i(\boldsymbol{X}^*)$ (even in the knowledge-soundness proof and without allowing the algebraic adversary to create new indeterminates) is at least 30, depending on the SNARK. This issue can be solved by using more trapdoors as in [Gro16], but such a solution is not always acceptable.

Application: UC-Secure SNARKs. One can plug in $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ (instead of the Groth's SNARK as done in [KZM+15] or the Groth-Maller SNARK as done in [Bag19]) to the known transformation of non-black-box SASE SNARKs to bllack-box SASE SNARKs [Bag19] obtain better efficiency.

Further Work. Since our goal was to provide a simple, very general, template that allows for efficient soundness proofs, we did not fully optimize all eight new SNARKs. Moreover, we did not consider the important notion of subversion-security [BFS16,ABLZ17,Fuc18]: including all technical details for how to do it in the case of all 8 new zk-SNARKs would make the paper considerably longer.

Historic Remark. This version (from July 13, 2019) differs significantly from the first eprint version from May 31, 2019. The main difference is in the handling of simulation-extractability (SE): the earlier version achieved ASE but not SASE. In fact, its ASE security proofs contained a subtle error, introduced in the last moment during a submission rush. The current version of this paper achieves SASE by using tags; this changed the SE SNARKs somewhat but their efficiency remains comparable to the SE SNARKs in the earlier version. Due to the use of tags, we stopped using the full power of the generic bilinear group model in the soundness / SE proofs and added a lengthy description of the AGM and tautological knowledge assumptions.

2 Preliminaries

For a matrix $\boldsymbol{A}$, $\boldsymbol{A}_i$ denotes its ith row and $\boldsymbol{A}^{(j)}$ denotes its jth column.

Assume n is a power of two, and let ω be the n-th primitive root of unity modulo p. Such ω exists, given that $n \mid (p-1)$. Then, $\ell(X) := \prod_{i=1}^{n}(X - \omega^{i-1})$ is the unique degree n monic polynomial such that $\ell(\omega^{i-1}) = 0$ for all $i \in [1\,..\,n]$. For $i \in [1\,..\,n]$, let $\ell_i(X)$ be the *ith Lagrange basis polynomial*, i.e., the unique degree $n-1$ polynomial s.t. $\ell_i(\omega^{i-1}) = 1$ and $\ell_i(\omega^{j-1}) = 0$ for $i \neq j$. Given $\chi \in \mathbb{Z}_p$, there is an efficient algorithm (see, e.g., [BCG+13]) that computes $\ell_i(\chi)$ for $i \in [1\,..\,n]$. Clearly, $L_{\boldsymbol{a}}(X) := \sum_{i=1}^{n} a_i \ell_i(X)$ is the interpolating polynomial of $\boldsymbol{a}$ at points ω^{i-1}, with $L_{\boldsymbol{a}}(\omega^{i-1}) = a_i$, and its coefficients can thus be computed by executing an inverse Fast Fourier Transform in time $\Theta(n \log n)$. Moreover, $(\ell_j(\omega^{i-1}))_{i=1}^{n} = \boldsymbol{e}_j$ (the jth unit vector) and $(\ell(\omega^{i-1}))_{i=1}^{n} = \mathbf{0}_n$.

Let PPT denote probabilistic polynomial-time and let $\lambda \in \mathbb{N}$ be the security parameter. For an algorithm $\mathcal{A}$, range($\mathcal{A}$) is the range of $\mathcal{A}$, i.e., the set of of valid outputs of $\mathcal{A}$, $\mathsf{RND}(\mathcal{A})$ denotes the random tape of $\mathcal{A}$, and $r \leftarrow_{\$} \mathsf{RND}(\mathcal{A})$ denotes the uniformly random choice of the randomizer r from $\mathsf{RND}(\mathcal{A})$. By $y \leftarrow \mathcal{A}(\mathsf{x}; r)$ we denote the fact that $\mathcal{A}$, given an input x and a randomizer r, outputs y. Let $\mathsf{negl}(\lambda)$ be an arbitrary negligible function, and $\mathsf{poly}(\lambda)$ be an arbitrary polynomial function. We write $a \approx_\lambda b$ if $|a - b| \leq \mathsf{negl}(\lambda)$.

Bilinear Groups. 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. We assume that $n \mid (p-1)$. As in say [BFS16], we assume that Pgen is deterministic and cannot be subverted. We require the bilinear pairing to be Type-3 [GPS08], i.e., we assume that there is no

efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. At this moment, the curve BLS12-381 [BLS04,Bow17] is recommended at the 128-bit security level. We use the bracket notation of [EHK+13], i.e., we write $[a]_\iota$ to denote ag_ι where g_ι is a fixed generator of $\mathbb{G}_\iota$. We denote $\hat{e}([a]_1,[b]_2)$ by $[a]_1 \bullet [b]_2$. We use freely the bracket notation together with matrix notation, e.g., $\boldsymbol{AB}=\boldsymbol{C}$ iff $[\boldsymbol{A}]_1 \bullet [\boldsymbol{B}]_2 = [\boldsymbol{C}]_T$.

Let $d_1(n), d_2(n) \in poly(\lambda)$. Then, Pgen is $(d_1(n),d_2(n))$-*PDL (Power Discrete Logarithm, [Sta08,THS+09,JR10,Lip12]) secure* if for any non-uniform PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{A}}(\lambda)=\mathsf{negl}(\lambda)$, where $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\mathsf{p}\leftarrow\mathsf{Pgen}(1^\lambda,n), x\leftarrow_{\$}\mathbb{Z}_p : \mathcal{A}\left(\mathsf{p}; [(x^i)_{i=0}^{d_1(n)}]_1, [(x^i)_{i=0}^{d_2(n)}]_2\right)=x\right] \ .$$

QAP. Quadratic Arithmetic Program (QAP) was introduced in [GGPR13] as a language where for an input x and witness inp, $(\mathsf{x},\mathsf{inp})\in\mathbf{R}$ 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.

Let $m_0 < m$ be a non-negative integer. In the case of arithmetic circuits, n is the number of multiplication gates, m is the number of wires, and m_0 is the number of public inputs. 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, [GGPR13].

Let $\mathbb{F}=\mathbb{Z}_p$, such that ω is the n-th primitive root of unity modulo p. This requirement is needed for the sake of efficiency, and we will make it implicitly throughout the paper. However, it is not needed for the new SNARKs to work. A QAP is characterized by n constraints $(\sum_{j=1}^m U_{ij}u_j(X))(\sum_{j=1}^m V_{ij}u_j(X)) = \sum_{j=1}^m W_{ij}w_j(X)$, where $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ are instant-dependent matrices. For $j\in[1\,..\,m]$, define $u_j(X):=L_{\boldsymbol{U}^{(j)}}(X)$, $v_j(X):=L_{\boldsymbol{V}^{(j)}}(X)$, and $w_j(X):=L_{\boldsymbol{W}^{(j)}}(X)$ to be interpolating polynomials of the jth column of the corresponding matrix. Thus, $u_j,v_j,w_j\in\mathbb{Z}_p^{(\le n-1)}[X]$.

An QAP instance $\mathsf{Inst}_{\mathsf{qap}}$ is equal to $(\mathbb{Z}_p, m_0, \{u_j,v_j,w_j\}_{j=0}^m)$. This instance defines the following relation:

$$\mathbf{R}_{\mathsf{Inst}_{\mathsf{qap}}} = \left\{ \begin{array}{l} (\mathsf{x},\mathsf{inp}) \colon \mathsf{x}=(a_1,\ldots,a_{m_0})^\top \wedge \mathsf{inp}=(a_{m_0+1},\ldots,a_m)^\top \wedge \\ u(X)v(X)\equiv w(X) \pmod{\ell(X)} \end{array} \right\} \qquad (1)$$

where $u(X)=\sum_{j=0}^m a_j u_j(X)$, $v(X)=\sum_{j=0}^m a_j v_j(X)$, and $w(X)=\sum_{j=0}^m a_j w_j(X)$. Alternatively, $(\mathsf{x},\mathsf{inp})\in\mathbf{R}$ if there exists a (degree $\le n-2$) polynomial $h(X)$, such that the following key equation holds:

$$\chi(X) := u(X)v(X)-w(X)-h(X)\ell(X)=0 \ , \qquad (2)$$

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 a_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 a_j for $j\le m$.

SNARKs. Let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\mathsf{x}, \mathsf{inp})\}$. Here, x is a statement and inp is a witness. We assume that λ is explicitly deductible from the description of $\mathbf{R}$. $\mathcal{R}$ also outputs auxiliary information $\mathsf{aux}_{\mathbf{R}}$ that will be given to the honest parties and the adversary. As in [Gro16], $\mathsf{aux}_{\mathbf{R}}$ will be equal to $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda, n)$ for a well-defined n. Because of this, we will also give $\mathsf{aux}_{\mathbf{R}}$ as an input to the honest parties; if needed, one can include an additional auxiliary input as an input to the adversary. We recall that the choice of p and thus of the groups $\mathbb{G}_z$ depends on n. Let $\mathcal{L}_{\mathbf{R}} = \{\mathsf{x} : \exists \mathsf{inp}, (\mathsf{x}, \mathsf{inp}) \in \mathbf{R}\}$ be an **NP**-language.

We will define tag-based argument systems; in the non-tag-based case, the tag-space is $\mathsf{Tags} = \{\bot\}$ (empty string) and tags are ignored by all algorithms. A *non-interactive zero-knowledge (NIZK) argument system* Ψ for $\mathcal{R}$ consists of four PPT algorithms:

CRS generator: $\mathsf{K}_{\mathsf{crs}}$ is a probabilistic algorithm that, given $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$, outputs $(\mathsf{crs}, \mathsf{td})$ where crs is a CRS and td is a simulation trapdoor. Otherwise, it outputs a special symbol $\bot$. For the sake of efficiency and readability, we divide crs into $\mathsf{crs}_{\mathsf{P}}$ (the part needed by the prover) and $\mathsf{crs}_{\mathsf{V}}$ (the part needed by the verifier).

Prover: P is a probabilistic algorithm that, given $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \tau, \mathsf{x}, \mathsf{inp})$ for $\tau \in \mathsf{Tags}$ and $(\mathsf{x}, \mathsf{inp}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\bot$.

Verifier: V is a probabilistic algorithm that, given $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \tau, \mathsf{x}, \pi)$, returns either 0 (reject) or 1 (accept).

Simulator: Sim is a probabilistic algorithm that, given $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}, \mathsf{td}, \tau, \mathsf{x})$, outputs an argument π.

A NIZK argument system must satisfy completeness (an honest verifier accepts an honest verifier), knowledge-soundness (if a prover makes an honest verifier accept, then one can extract from the prover a witness inp), and zero-knowledge (there exists a simulator that, knowing 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). See Appendix A.1 for formal definitions. 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). SE [Sah99,DDO+01] is a stronger notion of knowledge-soundness, motivated by cryptographic applications like non-malleability and UC-security. An SE argument system remains knowledge-sound even if the soundness adversary has access to the simulation oracle. More precisely, one requires that there exists a universal extractor Ext, such that for each PPT soundness adversary $\mathcal{A}$ who has oracle access to the simulator, Ext can deduce the witness from $\mathcal{A}$.

Dodis *et al.* [DHLW10] 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. In the case of

strong any-simulation-extractability, the adversary wins even if she can come up with a new argument for a statement she has queried a simulation for, as long as she outputs it for a new tag not used in simulation queries.

Groth and Maller [GM17] introduced the notion of non-black-box simulation-extractability for SNARKs. In the case of non-black-box simulation-extractability, one requires that for each PPT soundness adversary $\mathcal{A}$ who has oracle access to the simulator, there exists a non-black-box extractor $\mathsf{Ext}_{\mathcal{A}}$ that can extract the witness from $\mathcal{A}$. The definition of SE from [GM17] corresponds to *non-black-box strong any-simulation extractability (SASE)*. Since we are interested in non-black-box SE, we will implicitly assume SE means non-black-box SE.Groth and Maller proved that for any SASE SNARK, the argument consists at least of three group elements and that there should be at least two verification equations. They also proposed one concrete SASE SNARK, based on the SAP (Square Arithmetic Program) language, that meets the lower bounds. We will design several SASE SNARKs based on different languages like QAP [GGPR13], SSP [DFGK14], SAP [Gro16], and QSP [GGPR13]. We will provide formal definitions of non-black-box (strong) any-simulation-extractability in Appendix A.2.

3 Subversion Algebraic Knowledge Assumptions

AGM is a new model [FKL18] that one can use to prove the security of a cryptographic assumption, protocol, or a primitive. Essentially, in AGM one assumes that each PPT algorithm (including the adversaries) is algebraic in the following sense: if the adversary's input includes $[\boldsymbol{x_\iota}]_\iota$ and no other elements from $\mathbb{G}_\iota$ ($\iota \in \{1,2\}$) and the adversary outputs group elements $[\boldsymbol{y_\iota}]_\iota$, then the adversary knows matrices $\boldsymbol{N}_\iota$, such that $\boldsymbol{y}_\iota = \boldsymbol{N}_\iota \boldsymbol{x}_\iota$. While [FKL18] defined AGM by requiring the adversaries in the security proof to output $[\boldsymbol{x}_\iota]_\iota$ together with $\boldsymbol{N}_\iota$, we find it more convenient to define AGM as a general knowledge assumption.

Let $\mathbb{G}$ be a cyclic group of prime order p. A PPT algorithm $\mathcal{A}$ is algebraic [BV98] if there exists an efficient extractor $\mathsf{Ext}_{\mathcal{A}}$ that for any PPT sampleable distribution $\mathcal{D}$, $\mathsf{Adv}^{\mathrm{ak}}_{\mathbb{G},\mathcal{D},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$, where $\mathsf{Adv}^{\mathrm{ak}}_{\mathbb{G},\mathcal{D},\mathcal{A}}(\lambda) :=$

$$\Pr\left[[\boldsymbol{x}]_\iota \leftarrow_\$ \mathcal{D}; r \leftarrow_\$ \mathsf{RND}(\mathcal{A}); [\boldsymbol{y}]_\iota \leftarrow_\$ \mathcal{A}([\boldsymbol{x}]_\iota; r); \boldsymbol{N} \leftarrow \mathsf{Ext}_{\mathcal{A}}([\boldsymbol{x}]_\iota; r) : \boldsymbol{y} \neq \boldsymbol{N}\boldsymbol{x}\right] .$$

A group $\mathbb{G}_\iota$ is *algebraic* if every PPT algorithm $\mathcal{A}$ that obtains inputs from $\mathbb{G}_\iota$ and outputs elements in $\mathbb{G}_\iota$ is algebraic.

The restriction that adversaries are algebraic is not valid in situations where the adversary can create new random group elements by say using elliptic curve hashing [Ica09]. We model this capability by allowing the adversary to create additional "input" group elements $[\boldsymbol{q}]_\iota$ for which she does not know discrete logarithms. We require that $[\boldsymbol{q}_\iota]_\iota$ can be extracted from the adversary, such that $\boldsymbol{y} = \boldsymbol{N} \cdot \binom{\boldsymbol{x}}{\boldsymbol{q}}$. Moreover, $\boldsymbol{q}$ must be sampled from a public distribution $\mathcal{D}'$. For example, if elliptic-curve hashing is used, one can assume that each coefficient of $\boldsymbol{q}$ is close to uniformly random. In the setting of a bilinear group p, we follow [BFS16,ABLZ17] and assume additionally that group elements created in $\mathbb{G}_1$ and $\mathbb{G}_2$ are independent. (See [BFS16,ABLZ17] for discussion.)

We say that a PPT algorithm $\mathcal{A}$ is *subversion-algebraic* if there exists an efficient extractor $\mathsf{Ext}_{\mathcal{A}}$, such that for any PPT distribution $\mathcal{D}$ and any sufficiently random (we will clarify this notion later) distribution $\mathcal{D}'$, $\mathsf{Adv}^{\mathrm{sak}}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{D}',\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l}[\boldsymbol{x}]_\iota \leftarrow_{\$} \mathcal{D}; r \leftarrow_{\$} \mathsf{RND}(\mathcal{A}); [\boldsymbol{y}]_\iota \leftarrow_{\$} \mathcal{A}([\boldsymbol{x}]_\iota; r); \\ (\boldsymbol{N}, [\boldsymbol{q}]_\iota) \leftarrow \mathsf{Ext}_{\mathcal{A}}([\boldsymbol{x}]_\iota; r) : (\boldsymbol{y} \neq \boldsymbol{N}\left(\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{q}\end{smallmatrix}\right)) \wedge ([\boldsymbol{q}]_\iota \sim \mathcal{D}')\end{array}\right] = \mathsf{negl}(\lambda) \quad .$$

A group $\mathbb{G}_\iota$ is *subversion-algebraic* if every PPT algorithm $\mathcal{A}$ that obtains inputs from $\mathbb{G}_\iota$ and outputs elements in $\mathbb{G}_\iota$ is subversion-algebraic. Clearly, a subversion-algebraic adversary has more power than an algebraic adversary.

The AGM (resp., Sub-AGM) is essentially the assumption that the given group is algebraic (resp., subversion-algebraic). Let us make this more precise. We think of the requirement that for fixed $(\mathcal{D}, \mathcal{D}')$, $\mathcal{A}$ is subversion-algebraic as a specialized $(\mathbb{G}, \mathcal{D}, \mathcal{D}', \mathcal{A})$-*subversion algebraic knowledge (SAK) assumption* stating that $\mathsf{Adv}^{\mathrm{sak}}_{\mathbb{G},\mathcal{D},\mathcal{D}',\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$. Analogously, the $(\mathbb{G}_\iota, \mathcal{D}, \mathcal{A})$-algebraic knowledge (AK) assumption states that $\mathsf{Adv}^{\mathrm{ak}}_{\mathbb{G},\mathcal{D},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

In AGM (resp., Sub-AGM), one assumes that $(\mathbb{G}_\iota, \mathcal{D}, \mathcal{A})$-AK (resp., $(\mathbb{G}_\iota, \mathcal{D}, \mathcal{D}', \mathcal{A})$-SAK) holds for all choices of $(\mathcal{D}, \mathcal{A})$ (resp., $(\mathcal{D}, \mathcal{D}', \mathcal{A})$). We thus call AGM (resp., Sub-AGM) the $\mathbb{G}_\iota$-*AK (resp.,* $\mathbb{G}_\iota$-*SAK) assumption*. While proving the security of a concrete protocol, it is sufficient to rely on the tautological $(\mathbb{G}_\iota, \mathcal{D}, \mathcal{D}')$-*SAK assumption* that there exists an extractor for each PPT adversary $\mathcal{A}$ that obtains inputs, distributed *according to the distribution* $\mathcal{D}$ *fixed by the security definition of that protocol* and then outputs $[\boldsymbol{q}]_\iota$ and $\boldsymbol{N}$ such that $[\boldsymbol{q}]_\iota \sim \mathcal{D}'$. A similar approach (but without mentioning AGM) was taken say in [ALSZ18] that defined KW-KE to be the assumption that there exists an extractor for each adversary that, given the CRS of the Kiltz-Wee QA-NIZK protocol Π'_{as} [KW15], outputs a purported argument.

Example 1. Let us demonstrate how a SAK assumption can be used. Consider the q-PCDH assumption [GJM03,Gro10], $\mathcal{D}$ outputs $\mathsf{x} = [1, x, x^1, \ldots, x^q]_1^\top$ for uniformly random x, and the adversary $\mathcal{A}$ is asked to output $[y]_1 = [x^{q+1}]_1$. In this case, the $(\mathbb{G}_\iota, \mathcal{D}, \mathcal{D}', \mathcal{A})$-SAK assumption states that one can efficiently extract $[\boldsymbol{q}]_\iota$ and integers N_i and N'_i such that $x^{q+1} = \sum_{i=0}^{q} N_i x^i + \sum N'_i q_i$. It means that either $f(X, \boldsymbol{Q}) = X^{q+1} - \sum_{i=0}^{q} N_i X^i - \sum N'_i Q_i = 0$ as a polynomial (which is impossible) or $\mathcal{A}$ has returned x, such that $(x, \boldsymbol{q})$ is a root of the non-zero polynomial $f(X, \boldsymbol{Q})$. If $\mathcal{A}$ created no new group elements then $f(X)$ is a univariate polynomial and the adversary has broken (a version of) the $(q, 0)$-PDL assumption. Otherwise, assuming $[\boldsymbol{q}]_\iota$ is "random enough" from the viewpoint of $\mathcal{A}$, the probability that $f(x, \boldsymbol{q}) = 0$ will be negligible.

Let us now analyze the notion of "random enough". Bellare *et al.* [BFS16,ABLZ17] modelled subversion security by using the subversion generic group model (Sub-GBGM) where the adversary can create random group elements $[\boldsymbol{q}_\iota]_1$ that are interpreted as new indeterminates $\boldsymbol{Q}_\iota$ in the security proof. Our approach clarifies what does it mean for the elements to be random. As we will see later (e.g., see the proof of Theorem 1), it will be needed that $\mathcal{D}'$ has

"high" min-entropy

$$H_\infty(\mathcal{D}') := -\log_2 \max_{\boldsymbol{y}} \Pr[\boldsymbol{q} \leftarrow\!\!\$\ \mathcal{D}' : \boldsymbol{q} = \boldsymbol{y}] = \omega(\log \lambda) \ .$$

Really, we need that for any $\boldsymbol{x}$, a randomly chosen $\boldsymbol{q}$ is a root of some "verification polynomial" $V^*(\boldsymbol{x}, \boldsymbol{q})$ with negligible probability. In the bilinear-group setting, V^* has degree one in any of the variables $Q_{\iota k}$. Thus, it follows from the Schwartz-Zippel lemma [Zip79,Sch80] that

$$\Pr[V^*(x, \boldsymbol{q}_1, \boldsymbol{q}_2) = 0] \leq 2^{-H_\infty(\mathcal{D}')} = 2^{-\omega(\log \lambda)} = \lambda^{-\omega(1)}$$

is negligible.[4] There exist simpler versions of elliptic curve hashing (so called encodings, [Ica09,BCI+10]) where the output is assumed to have high-entropy but is not close to uniform; importantly, such versions suffice for us.

Moreover, the assumption of high min-entropy is quite natural. Really, if $q_{\iota k}$ is equal to some $y_{\iota k}$ with a high probability then a non-uniform adversary that has $y_{\iota k}$ as an advise can compute the discrete logarithm $q_{\iota k}$ with a non-negligible probability. In such a case, $q_{\iota k}$ can be considered as an element with known discrete logarithm.

Importantly, when proving the securty under (S)AK assumptions, the adversary is allowed to make use of the group presentation as long as this does not contradict the concrete knowledge assumption. This is important in the tag-based setting where the adversary can choose her own tags (integers).

Finally, to simplify exposition, in the bilinear-group setting where we make both $(\mathbb{G}_1, \mathcal{D}_1, \mathcal{D}_1')$-SAK and $(\mathbb{G}_2, \mathcal{D}_2, \mathcal{D}_2')$-SAK assumptions, we will talk about a $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}, \mathcal{D}')$-SAK assumption where $\mathcal{D} = \mathcal{D}_1 \times \mathcal{D}_2$ and $\mathcal{D}' = \mathcal{D}_1' \times \mathcal{D}_2'$. We define the $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D})$-AK assumption analogously.

4 Knowledge-Sound SNARK for QAP

In this section, we will describe new SNARKs $\mathsf{S}_{\mathsf{qap}}$ (SNARK for QAP) and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ (SNARK for QAP, SE) for QAP. The template follows two objectives: (i) simple soundness proof under an SAK assumption, and (ii) efficiency. In fact, $\mathsf{S}_{\mathsf{qap}}$ is very similar to Groth's SNARK from EUROCRYPT 2016 [Gro16] with the main difference being the use of only two trapdoors instead of five. The second difference is an alternative, much simpler, knowledge-soundness proof in the case of asymmetric pairings; Groth on the other hand provided a very complex knowledge-soundness proof that is valid for both asymmetric and symmetric pairings. The new tag-based SASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is novel and we are not aware of any prior art tag-based SNARKs at all.

Let $u(X) = \sum_{j=1}^m a_j u_j(X)$, $v(X) = \sum_{j=1}^m a_j v_j(X)$, and $w(X) = \sum_{j=1}^m a_j w_j(X)$ as in Section 2. Recall from Eq. (2) that for $\chi(X) = u(X)v(X) - w(X) - h(X)\ell(X)$, the key equation of QAP states that $\chi(X) = 0$. That is, $h(X) := (u(X)v(X) - w(X))/\ell(X)$ is a polynomial iff the prover is honest.

[4] See [FKL18, Section 1.2] for a less concrete analysis of the Sub-AGM case.

The argument in the new template consists of three elements, $\pi = ([\mathsf{a},\mathsf{c}]_1,[\mathsf{b}]_2)$, where $\mathsf{a} = A(x,y)$, $\mathsf{b} = B(x,y)$, and $\mathsf{c} = C(x,y)$ for well-defined polynomials $A(X,Y)$, $B(X,Y)$, and $C(X,Y)$. Intuitively, $[\mathsf{a}]_1$ is a succinct commitment to $u(X)$, $[\mathsf{b}]_2$ is a succinct commitment to $v(X)$, and $[\mathsf{c}]_1$ is the "actual" argument that additionally commits to $w(X)$. More precisely, let α, β, γ, and δ be integers chosen later. Then

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

We now define

$$\begin{aligned} C(X,Y) &= (A(X,Y)+Y^\gamma)(B(X,Y)+Y^\delta) - Y^{\gamma+\delta} \\ &= A(X,Y)B(X,Y) + B(X,Y)Y^\gamma + A(X,Y)Y^\delta \\ &= u(X)Y^{\beta+\delta} + v(X)Y^{\beta+\gamma} + w(X)Y^{2\beta} + (u(X)v(X)-w(X))Y^{2\beta} + \\ &\quad r_b(r_aY^\alpha + u(X)Y^\beta + Y^\gamma)Y^\alpha + r_a(v(X)Y^\beta + Y^\delta)Y^\alpha \\ &= \textstyle\sum_{j=1}^m a_j(u_j(X)Y^{\beta+\delta} + v_j(X)Y^{\beta+\gamma} + w_j(X)Y^{2\beta}) + \\ &\quad (u(X)v(X)-w(X))Y^{2\beta} + r_b(A(X,Y)+Y^\gamma)Y^\alpha + \\ &\quad r_a(v(X)Y^\beta + Y^\delta)Y^\alpha \ . \end{aligned} \tag{4}$$

Since a SNARK also has a public input $(a_j)_{j=1}^{m_0}$, we define two polynomials $C_s(X,Y)$ and $C_p(X,Y)$, so that for another integer η,

$$C(X,Y) = C_p(X,Y)Y^\eta + C_s(X,Y)Y^\alpha \ ,$$

where $C_p(X,Y)$ depends only on a_j for $j \le m_0$, $C_s(X,Y)$ depends only on a_j for $j > m_0$, and $C_p(X,Y)$ only has m_0 addends (to minimize the computation, performed by the verifier):

$$\begin{aligned} C_p(X,Y) &= \textstyle\sum_{j=1}^{m_0} a_j \left(u_j(X)Y^{\beta-\eta+\delta} + v_j(X)Y^{\beta-\eta+\gamma} + w_j(X)Y^{2\beta-\eta}\right) \ , \\ C_s(X,Y) &= \textstyle\sum_{j=m_0+1}^{m} a_j \left(u_j(X)Y^{\beta-\alpha+\delta} + v_j(X)Y^{\beta-\alpha+\gamma} + w_j(X)Y^{2\beta-\alpha}\right) + \\ &\quad (u(X)v(X)-w(X))Y^{2\beta-\alpha} + r_b\left(A(X,Y)+Y^\gamma\right) + r_a v(X)Y^\beta + r_a Y^\delta \ . \end{aligned} \tag{5}$$

Here, we use the multiplicand Y^α for efficiency reasons, since $C(X,Y)$ has an addend $r_a A(X,Y)Y^\alpha$.

Hence, the argument consists of three elements, $\pi = ([\mathsf{a},\mathsf{c}_s]_1,[\mathsf{b}]_2)$, where $\mathsf{c}_s = C_s(x,y)$ and the verifier recomputes $[C(x,y)]_T = [\mathsf{c}_p]_1 \bullet [y^\eta]_2 + [\mathsf{c}_s]_1 \bullet [y^\alpha]_2$. Essentially, the verifier of the new SNARK checks that $[\mathsf{c}(x,y)]_T$ is computed correctly by checking that $[\mathsf{c}]_T = ([\mathsf{a}]_1 + [y^\gamma]_1) \bullet ([\mathsf{b}]_2 + [y^\delta]_2) - [y^{\gamma+\delta}]_T$.

We prove knowledge-soundness based on a SAK and a PDL assumption. Let $V(X,Y) = C(X,Y) - (A(X,Y)+Y^\gamma)(B(X,Y)+Y^\delta) + Y^{\gamma+\delta}$ be the verification polynomial. By the SAK assumption, all coefficients of V are known. Assume first that $V(X,Y) = 0$ is a zero polynomial. Then since the coefficient of $Y^{2\beta}$ is $u(X)v(X) - w(X)$, it divides by $\ell(X)$ only if $\chi(X) = 0$. Hence, by inserting to the CRS elements of type $[f(x)y^{2\beta}]_1$ only when $\ell(X) \mid f(X)$, we get that if

$V(X,Y) = 0$ iff the prover is honest. Second, if $V(X,Y) \neq 0$ but the verification succeeds, then $V(x,y) = 0$ and by the same strategy as in Section 3 (the example of PCDH), we can break the PDL assumption.

The full proof needs some more considerations. First, in the malicious case, the adversary can also create a number of new random group elements $[\boldsymbol{q}_1]_1$ and $[\boldsymbol{q}_2]_2$. Let $\boldsymbol{Q}_\iota$ be the vector of corresponding formal indeterminates in $\mathbb{G}_\iota$ for $\iota \in \{1,2\}$. Let $\boldsymbol{Q} = (\boldsymbol{Q}_1, \boldsymbol{Q}_2)$ and let $\boldsymbol{X} = (X, \boldsymbol{Q}, Y)$ be the vector of all indeterminates. Let $\boldsymbol{X}^* = (X, \boldsymbol{Q})$. Then, the polynomial corresponding to the verification equation is

$$V(\boldsymbol{X}) = (A(\boldsymbol{X}) + Y^\gamma)(B(\boldsymbol{X}) + Y^\delta) - Y^{\gamma+\delta} - C_p(\boldsymbol{X})Y^\eta - C_s(\boldsymbol{X})Y^\alpha \ , \qquad (6)$$

where $A(\boldsymbol{X})$, $B(\boldsymbol{X})$, and $C_s(\boldsymbol{X})$ are potentially maliciously computed polynomials that may depend on the indeterminates $\boldsymbol{Q}$.

Assume $V(\boldsymbol{X}) = 0$. Writing $V(\boldsymbol{X}) = \sum_i V_i(\boldsymbol{X}^*)Y^i$, we get that each $V_i(\boldsymbol{X}^*) = 0$. We aim to construct the SNARK so that for some small set Crit, if $V_i(\boldsymbol{X}^*) = 0$ for $i \in \mathsf{Crit}$ then $\chi(X) = 0$. We already know that $2\beta \in \mathsf{Crit}$.

To formalize this discussion, consider the malicious case where $A(\boldsymbol{X})$, $B(\boldsymbol{X})$, and $C_s(\boldsymbol{X})$ are any polynomials in the span of polynomials represented in the CRS in groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_1$, respectively. Since a subversion-algebraic adversary can only evaluate polynomials that are in the span of the polynomials in the CRS (of the same group) and of the newly generated group elements, then any maliciously computed polynomials $crs_1(a, \boldsymbol{X})$ and $crs_2(b, \boldsymbol{X})$, where a and b are symbolic, that represent group eements in $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, have to have the following shape. This shape follows from the description of crs in Fig. 2, which follows from the elements that either the honest prover or the honest verifier have to be able to compute during the argument.

$$\begin{aligned} crs_1(a, \boldsymbol{X}) = {} & \textstyle\sum_{j=1}^{m_0} a_j^*(u_j(X)Y^{\beta-\eta+\delta} + v_j(X)Y^{\beta-\eta+\gamma} + w_j(X)Y^{2\beta-\eta}) + \\ & \textstyle\sum_{i=m_0+1}^{m} a_i^*(u_i(X)Y^{\beta-\alpha+\delta} + v_i(X)Y^{\beta-\alpha+\gamma} + w_i(X)Y^{2\beta-\alpha}) + \\ & h_a(X)\ell(X)Y^{2\beta-\alpha} + r_aY^\alpha + u_a(X)Y^\beta + a_\gamma Y^\gamma + a_\delta Y^\delta + \textstyle\sum_k q_{ak}Q_{1k} \ , \\ crs_2(b, \boldsymbol{X}) = {} & r_bY^\alpha + v_b(X)Y^\beta + b_\delta Y^\delta + b_\eta Y^\eta + \textstyle\sum_k b_{qk}Q_{2k} \ , \end{aligned} \qquad (7)$$

where say $a_j^* \in \mathbb{Z}_p$, $u_a(X) \in \mathbb{Z}_p[X]$ is a degree-$\leq (n-1)$ polynomial, and $h_a(X) \in \mathbb{Z}_p[X]$ is a degree-$\leq (n-2)$ polynomial. Then, $A(\boldsymbol{X}) = crs_1(a, \boldsymbol{X}) = \cdots + r_aY^\alpha + u_a(X)Y^\beta + \cdots$, $C_s(\boldsymbol{X}) = crs_1(c, \boldsymbol{X}) = \cdots + r_cY^\alpha + u_c(X)Y^\beta + \cdots$, and $B(\boldsymbol{X}) = crs_2(b, \boldsymbol{X}) = r_bY^\alpha + \cdots$. Thus, $V(\boldsymbol{X})$ is defined as in Eq. (6) but with polynomials $A(\boldsymbol{X})$, $B(\boldsymbol{X})$, and $C_s(\boldsymbol{X})$ as defined in the current paragraph.

Second, for the described proof idea to go through, we need 2β to be different from all other exponents in maliciously computed $V^*(\boldsymbol{X})$. Moreover, the prover is honest iff $\chi(X) = 0$ iff $u(X)v(X) - w(X) = h(X)\ell(X)$ for some polynomial $h(X)$ iff the coefficient of $Y^{2\beta}$ in $C(X,Y)$ is divisible by $\ell(X)$. In addition, $C_s(X,Y)$ has addends $u_j(X)Y^{\beta-\alpha+\delta}$, $v_j(X)Y^{\beta-\alpha+\gamma}$, and $w_j(X)Y^{2\beta-\alpha}$; thus their sum can be written as $\sum_{j=1}^m a_j f_j(X,Y)$ for known polynomials $f_j(X,Y)$, as above. This and the shape of the coefficient of $Y^{2\beta-\alpha}$ are the main reasons why we

Y^i	Coeff. $V_i(\boldsymbol{X}^*)$ (KS and SASE)	$V_i^+(\boldsymbol{X}^*)$ (SASE only)
$Y^{2\beta}$	$u_a(X)v_b(X) - w(X) - h(X)\ell(X)$	$\sum_k(s_{c2k} - r_b s_{a2k})\sum_j \sigma_{kj} w_j(X)$
$Y^{\beta+\gamma}$	$(a_\gamma + 1)v_b(X) - v(X)$	$\sum_k(s_{c2k} - r_b s_{a2k})\sum_j \sigma_{kj} v_j(X)$
$Y^{\beta+\delta}$	$(b_\delta + 1)u_a(X) - u(X) + a_\delta v_b(X)$	$\sum_k(s_{c2k} - r_b s_{a2k})\sum_j \sigma_{kj} u_j(X)$
$Y^{\gamma+\delta}$	$b_\delta + a_\gamma(b_\delta + 1)$	
$Y^{\gamma+\eta}$	$(a_\gamma + 1)b_\eta$	
$Y^{2\delta}$	$(b_\delta + 1)\, a_\delta$	

Fig. 1. Critical coefficients in $\mathsf{S}_{\mathsf{qap}}$ (left) and addends to the same coefficients in the SASE case (right).

chose $C(X, Y)$ as in Eq. (4). Let $R = \{i : V_i(\boldsymbol{X}^*) \neq 0\}$, and

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

and $\overline{\mathsf{Crit}} = R \setminus \overline{\mathsf{Crit}}$ be the complement of Crit. Let $\boldsymbol{\Delta} = (\alpha, \beta, \gamma, \eta, \delta)$. To obtain knowledge-soundness, we will need to choose the values in $\boldsymbol{\Delta}$ so that Crit consists of mutually different integers ($|\mathsf{Crit}| = 6$) and $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$. The concrete value of the set Crit will become clear from the proof of Theorem 1 and from the following observation. Let $h(X) := h_c(X) - r_b h_a(X)$. Let

$$\tilde{a}_j = \begin{cases} a_j - b_\eta a_j^* \ , & j \leq m_0 \ , \\ c_j^* - r_b a_j^* \ , & j > m_0 \ . \end{cases}$$

Denote $u(X) = \sum_{j=1}^m \tilde{a}_j u_j(X)$, $v(X) = \sum_{j=1}^m \tilde{a}_j v_j(X)$, and $w(X) = \sum_{j=1}^m \tilde{a}_j w_j(X)$. In this case, the "critical" coefficients $V_i(\boldsymbol{X}^*)$, $i \in \mathsf{Crit}$, of $V(\boldsymbol{X})$ are depicted in Fig. 1. As argued in the the proof of Theorem 1, from $V_i(\boldsymbol{X}^*) = 0$ for $i \in \mathsf{Crit}$ it follows that $\chi(X) = 0$.

We are now ready to describe the SNARK $\mathsf{S}_{\mathsf{qap}}$, see Fig. 2, and prove its security. (Note that tags τ are only used in the SASE version.) Like [Gro16] but unlike say [GGPR13], $\mathsf{S}_{\mathsf{qap}}$ guarantees that $u(X)$, $v(X)$, and $w(X)$ use the same witness $\boldsymbol{a}$ without having to use a strong QAP [GGPR13].

Before stating the following theorem, we need to specify the $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}, \mathcal{D}')$-SAK assumption. More precisely, we need to define $\mathcal{D}$, since $\mathbb{G}_\iota$ is fixed by the protocol and it suffices if $\mathcal{D}'$ is any distribution of high min-entropy. Here, $\mathcal{D}_{qap}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})$ is the distribution of honestly generated CRS for the concrete QAP instance $\mathbf{R} = \mathsf{Inst}_{\mathsf{qap}}$ and for $\mathsf{aux}_{\mathbf{R}}$. Thus, for $\mathsf{K}_{\mathsf{crs}}$ depicted in Fig. 2,

$$\mathcal{D}_{qap}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) = \{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})\} \ .$$

Theorem 1. *Let $\mathbf{R} = \mathsf{Inst}_{\mathsf{qap}} = (\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=0}^m)$ be a QAP instance. Let $\mathsf{S}_{\mathsf{qap}}$ be the SNARK in Fig. 2.*
(1) Assume $\boldsymbol{\Delta}$ satisfies $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$ and $H_\infty(\mathcal{D}') = \omega(\log \lambda)$. Then, $\mathsf{S}_{\mathsf{qap}}$ is knowledge-sound under the $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{qap}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}), \mathcal{D}')$-SAK and $(2n-1, n-1)$-PDL assumptions.
(2) $\mathsf{S}_{\mathsf{qap}}$ is perfectly zero-knowledge.

$\mathsf{K_{crs}}(\mathbf{R}, \mathsf{aux_R})$: Sample $x, y, z \leftarrow_\$ \mathbb{Z}_p^*$, let $\mathsf{td} \leftarrow (x, y, z)$. Let

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

$$\mathsf{crs_V} \leftarrow \left(\begin{array}{l} \left[\{u_j(x)y^{\beta-\eta+\delta} + v_j(x)y^{\beta-\eta+\gamma} + w_j(x)y^{2\beta-\eta}\}_{j=1}^{m_0}, y^\gamma, z \right]_1, \\ \left[y^\alpha, y^\delta, y^\eta \right]_2, [y^{\gamma+\delta}]_T \end{array} \right);$$

$\mathsf{crs} \leftarrow (\mathsf{crs_P}, \mathsf{crs_V})$; return $(\mathsf{crs}, \mathsf{td})$;

$\mathsf{P}(\mathbf{R}, \mathsf{aux_R}, \mathsf{crs_P}, \tau, (a_j)_{j=1}^{m_0}, (a_j)_{j=m_0+1}^{m})$:

$u(X) \leftarrow \sum_{j=1}^{m} a_j u_j(X)$; $v(X) \leftarrow \sum_{j=1}^{m} a_j v_j(X)$; $w(X) \leftarrow \sum_{j=1}^{m} a_j w_j(X)$;
$h(X) \leftarrow (u(X)v(X) - w(X))/\ell(X)$;
$(r_a, r_b) \leftarrow_\$ \mathbb{Z}_p^2$;
$[\mathsf{a}]_1 \leftarrow r_a[y^\alpha]_1 + [u(x)y^\beta]_1$; $[\mathsf{b}]_2 \leftarrow r_b[y^\alpha]_2 + [v(x)y^\beta]_2$;
$[\mathsf{b_1}]_1 \leftarrow r_b(\tau[y^\alpha]_1 + [y^\alpha z]_1) + [\tau v(x)y^\beta]_1 + [v(x)y^\beta z]_1$;
$[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^{m} a_j[u_j(x)y^{\beta-\alpha+\delta} + v_j(x)y^{\beta-\alpha+\gamma} + w_j(x)y^{2\beta-\alpha}]_1 + [h(x)\ell(x)y^{2\beta-\alpha}]_1 + r_b([\mathsf{a}]_1 + [y^\gamma]_1) + r_a([y^\delta]_1 + [v(x)y^\beta]_1)$;
return $\pi \leftarrow \left(\left[\mathsf{a}, \mathsf{b_1}, \mathsf{c}_s \right]_1, [\mathsf{b}]_2 \right)$;

$\mathsf{V}(\mathbf{R}, \mathsf{aux_R}, \mathsf{crs_V}, \tau, (a_j)_{j=1}^{m_0}, \pi = (\left[\mathsf{a}, \mathsf{b_1}, \mathsf{c}_s \right]_1, [\mathsf{b}]_2))$:

$[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j[u_j(x)y^{\beta-\eta+\delta} + v_j(x)y^{\beta-\eta+\gamma} + w_j(x)y^{2\beta-\eta}]_1$;
Check that
1. $[\mathsf{c}_p]_1 \bullet [y^\eta]_2 + [\mathsf{c}_s]_1 \bullet [y^\alpha]_2 = [\mathsf{a} + y^\gamma]_1 \bullet [\mathsf{b} + y^\delta]_2 - [y^{\gamma+\delta}]_T$;
2. $[\mathsf{b_1}]_1 \bullet [1]_2 = [\tau + z]_1 \bullet [\mathsf{b}]_2$;

$\mathsf{Sim}(\mathbf{R}, \mathsf{aux_R}, \mathsf{crs}, \mathsf{td} = y, \tau, \mathsf{x} = (a_j)_{j=1}^{m_0})$:

$[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j[u_j(x)y^{\beta-\eta+\delta} + v_j(x)y^{\beta-\eta+\gamma} + w_j(x)y^{2\beta-\eta}]_1$;
$d \leftarrow_\$ \mathbb{Z}_p$; $e \leftarrow_\$ \mathbb{Z}_p$;
$[\mathsf{a}]_1 \leftarrow d[1]_1$; $[\mathsf{b_1}]_1 \leftarrow e(\tau[1]_1 + [z]_1)$; $[\mathsf{b}]_2 \leftarrow e[1]_2$;
$[\mathsf{c}_s]_1 \leftarrow y^{-\alpha}((de + dy^\delta + ey^\gamma)[1]_1 - y^\eta[\mathsf{c}_p]_1)$;
return $\pi \leftarrow \left(\left[\mathsf{a}, \mathsf{b_1}, \mathsf{c}_s \right]_1, [\mathsf{b}]_2 \right)$;

Fig. 2. The new SNARKs $\mathsf{S_{qap}}$ (*without* highlighted entries) and $\mathsf{S^{se}_{qap}}$ (*with* highlighted entries). Moreover, $\mathsf{S_{qsp}}$ is exactly like $\mathsf{S_{qap}}$ and $\mathsf{S^{se}_{qsp}}$ is exactly like $\mathsf{S^{se}_{qap}}$, except $w_j(X) = 0$

We emphasize that the following knowledge-soundness proof depends minimally on the concrete SNARK: the only SNARK-dependent part is the Step 1 in the knowledge-soundness proof. The rest of the knowledge-soundness proof can basically be copied to the knowledge-soundness proofs of all following SNARKs.

Proof. **(1: knowledge-soundness)** In all following knowledge-soundness/SASE proofs, we will use the following template. We use the $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{qap}(\mathbf{R}, \mathsf{aux_R}), \mathcal{D}')$-SAK assumption to extract all coefficients of $V(\boldsymbol{X})$. The fact that the verifier accepts means that $V(\boldsymbol{x}) = 0$, where $\boldsymbol{x}$ is the

vector of actual trapdoors. We will then move everything from group elements to indeterminates, and we argue that if the verification polynomial $V(\boldsymbol{X})$ satisfies $V(\boldsymbol{X}) = 0$, then the prover did not cheat. After that, we use the PDL assumption to derive a contradiction from $V(\boldsymbol{X}) \neq 0$ but $V(\boldsymbol{x}) = 0$; we also argue that due to the high min-entropy of $\mathcal{D}'$, creating random group elements does not benefit $\mathcal{A}$. These two steps together guarantee that the SNARK is knowledge-sound/SASE: if the verifier accepted and the PDL assumption holds, then $V(\boldsymbol{X}) = 0$ as a polynomial and thus the prover did not cheat.

Let $\mathcal{A}$ be a knowledge-soundness adversary that, given $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}; r)$ as an input succeeds in outputting (x, π) such that $\mathsf{x} \notin \mathcal{L}$ but V accepts. Denote $\mathsf{crs} = ([\boldsymbol{\Gamma}_1]_1, [\boldsymbol{\Gamma}_2]_2)$. ($[y^{\gamma+\delta}]_T$ is only included to the CRS as an optimization and has no influence on the security.) By the SAK assumption, there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ that on the same inputs returns $\boldsymbol{N}_1$, $\boldsymbol{N}_2$, $[\boldsymbol{q}_1]_1$ and $[\boldsymbol{q}_2]_1$, such that $(\mathsf{a}, \mathsf{b}_1, \mathsf{c}_s)^\top = \boldsymbol{N}_1 \left(\begin{smallmatrix} \boldsymbol{\Gamma}_1 \\ \boldsymbol{q}_1 \end{smallmatrix}\right)$ and $(\mathsf{a}, \mathsf{b}_1, \mathsf{c}_s)^\top = \boldsymbol{N}_2 \left(\begin{smallmatrix} \boldsymbol{\Gamma}_2 \\ \boldsymbol{q}_2 \end{smallmatrix}\right)$. As in above, write $\mathsf{a} = A(\boldsymbol{x})$, $\mathsf{b}_1 = B_1(\boldsymbol{x})$, $\mathsf{c} = C(\boldsymbol{x})$, and $\mathsf{b} = B(\boldsymbol{x})$. Taking into account that elements of $\boldsymbol{\Gamma}_1$ are known polynomials of x and y, we can efficiently the coefficients of the polynomials A, B_1, C, and B from $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$.

Step 1. Assume first that $V(\boldsymbol{X}) = 0$ and thus $V_i(\boldsymbol{X}^*) = 0$ for $i \in \mathsf{Crit}$. Consider Fig. 1. Since $V_{\gamma+\delta}(\boldsymbol{X}^*) = b_\delta + a_\gamma(b_\delta + 1) = 0$, we have $a_\gamma = -b_\delta/(b_\delta + 1)$. Thus, $a_\gamma, b_\delta \neq -1$ and $(a_\gamma + 1)(b_\delta + 1) = 1$. Moreover, $V_{\eta+\gamma}(\boldsymbol{X}^*) = (a_\gamma + 1)b_\eta = 0$ and thus $b_\eta = 0$. Next, $V_{2\delta}(\boldsymbol{X}^*) = a_\delta = 0$. Thus, $\tilde{a}_j = a_j$ for $j \leq m_0$. Finally, $(b_\delta + 1)u_a(X) = u(X)$, $(a_\gamma + 1)v_b(X) = v(X)$, and $\chi(X) = u(X)v(X) - w(X) - \ell(X)h(X) = 0$, thus the prover did not cheat.

Step 2. Assume that $V(\boldsymbol{X}) \neq 0$ but $V(\boldsymbol{x}) = 0$, then the extractor has produced a root of the non-zero multivariate polynomial $V(\boldsymbol{X})$. We will first consider the case when the adversary did not create any new group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, i.e., we are working in the "non-subversion" case. After that, we show how to reduce the subversion case to the non-subversion case.

In the non-subversion case, assume that $\mathcal{A}$ is a knowledge-soundness adversary. We construct the following PDL adversary $\mathcal{B}$. The challenger $\mathcal{C}$ samples $x \leftarrow_\$ \mathbb{Z}_p$ and then gives $\mathsf{x} = ([1, x, \ldots, x^{2n-1}]_1, [1, x, \ldots, x^{n-1}]_2)$ as an input to $\mathcal{B}$. $\mathcal{B}$ samples $y \leftarrow_\$ \mathbb{Z}_p$, and uses it together with $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})$ and x to create crs. $\mathcal{B}$ plays the challenger in the knowledge-soundness game with $\mathcal{A}$: after sending crs and $r \leftarrow_\$ \mathsf{RND}(\mathcal{A})$ to $\mathcal{A}$, $\mathcal{B}$ obtains a purported proof $([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2)$ from $\mathcal{A}$. $\mathcal{B}$ runs the extractor $\mathsf{Ext}_{\mathcal{A}}$, guaranteed by the SAK assumption, to obtain matrices $\boldsymbol{N}_\iota$ and $[\boldsymbol{q}_\iota]_\iota$ (the latter is empty in the non-subversion case). From $\boldsymbol{N}_\iota$, she computes the coefficients of various polynomials like $A(X, Y)$, $B(X, Y)$, $C_s(X, Y)$, and $V(X, Y)$. Now, $[\mathsf{a}]_1 = [A(x, y)]_1$, $[\mathsf{c}_s]_1 = [C_s(x, y)]_1$, and $[\mathsf{b}]_2 = [B(x, y)]_2$. Since the verifier accepts, $V(x, y) = 0$; however, $V(X, Y) \neq 0$ as a polynomial. For the fixed value of y, let $V^*(X) := V(X, y)$. Finally, $\mathcal{B}$ does the following:

1. Use an efficient polynomial factorization algorithm to obtain up to $2n - 1$ roots x_i of $V^*(X)$.
2. Return the root x_i that satisfies $[x_i y^\beta]_1 = [x y^\beta]_1$.

Clearly, $\mathcal{B}$ has broken the $(2n-1,n-1)$-PDL assumption with a probability statistically close to the success probability of $\mathcal{A}$, and $\mathcal{B}$'s running time is dominated by the running time of $\mathcal{A}$ and the time to perform polynomial factorization.

Consider now the subversion case when $\mathcal{A}$ has created at least one random group element $q_{\iota k}$. We will rely on the fact that the new group elements $q_{\iota k}$ are added additively to a polynomial of x in both groups, and moreover, the elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ are independent. Then, $V^*(X,\boldsymbol{Q})$ is a degree-1 polynomial in any indeterminate $Q_{\iota k}$. Thus, by the Schwartz-Zippel lemma and since $H_\infty(\mathcal{D}') = \omega(\log\lambda)$, for any x, the probability that $V^*(x,\boldsymbol{q}) = 0$ is negligible. Hence, the probability that an adversary, who created at least one (high min-entropy) group element $[q_{\iota k}]_1$, can make the verifier accept is negligible.

(2: zero-knowledge) To see that Sim makes the verifier accept, note that $(\mathsf{a}+y^\gamma)(\mathsf{b}+y^\delta) - \mathsf{c}_s y^\alpha - \mathsf{c}_p y^\eta - y^{\gamma+\delta} = de + dy^\delta + ey^\gamma - \mathsf{c}_p y^\eta - (de + dy^\delta + ey^\gamma - \mathsf{c}_p y^\eta) = 0$. 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. □

In Appendix B, we observe that if the verification consists of a single pairing execution then it is not even needed to assume that $\mathcal{D}'$ has high min-entropy.

Choice of $\boldsymbol{\Delta}$. Recall that we need to find values for $\boldsymbol{\Delta} = (\alpha,\ldots)$, such that $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$. For convenience sake, we require that the polynomials crs_1 and crs_2 both contain a non-zero monomial corresponding to Y^0 (then we can publish $[1]_1$ and $[1]_2$) and that the values i, such that j for which $f(X)Y^j$ belongs to the CRS for some $f(X)$, have as small absolute values as possible (although we do not consider subversion security, this potentially speeds up the CRS verification algorithm [ABLZ17]). Since there are too many coefficients that one can take into account, we used a computer search to find the following values for $\alpha,\beta,\ldots$:

$$\alpha = 3,\ \ \beta = 4,\ \ \gamma = 0,\ \ \delta = -5,\ \ \eta = 2. \tag{8}$$

In this case,

$$\mathsf{crs}_\mathsf{P} = \begin{pmatrix} \left[\{u_j(x)y^{-4} + v_j(x)y^1 + w_j(x)y^5\}_{j=m_0+1}^{m}, y^3, \{x^j y^4\}_{j=0}^{n-1}\right]_1, \\ \left[\{x^i\ell(x)y^5\}_{j=0}^{n-2}, y^0, y^{-5}, y^3 z, \{x^j y^4 z\}_{j=0}^{n-1}\right]_1, \left[y^3, \{x^i y^4\}_{i=0}^{n-1}\right]_2 \end{pmatrix}, \\ \mathsf{crs}_\mathsf{V} = \left([\{u_j(x)y^{-3} + v_j(x)y^2 + w_j(x)y^6\}_{j=1}^{m_0}, y^0]_1, \left[y^3, y^{-5}, y^2, z\right]_2, [y^2]_T\right) .$$

Efficiency. $\mathsf{S}_{\mathsf{qap}}$ has fewer trapdoors but otherwise the same complexity as Groth's knowledge-sound zk-SNARK [Gro16], see Table 1 for a comparison. E.g., 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 $n+1$ exponentiations, and it takes $\approx m+2n$ additional exponentiations to compute $[\mathsf{c}]_1$.

5 SASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ for QAP

Consider the case of simulation-extractability, where the adversary also can query the simulator. Let $\boldsymbol{\sigma}_k = (\sigma_{kj})_{j=1}^{m_0}$ be the (maliciously generated) simulator input used by the adversary during the kth query. Let $\boldsymbol{X} = (X, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{E}, Y)$ and $\boldsymbol{X}^* = (X, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{E})$, where D_k (resp., E_k) is the indeterminate corresponding to the random trapdoor d (resp., e) generated by the simulator during the kth query. In the case of SASE, in $\mathsf{S}_{\mathsf{qap}}$, $crs_1(a, \boldsymbol{X})$ and $crs_2(b, \boldsymbol{X})$ have additional addends (highlighted in what follows) that correspond to the indeterminates generated by the simulator oracle:

$$\begin{aligned}
\mathsf{crs}_1(a,\boldsymbol{X}) &= \ldots + \textstyle\sum_k s_{a1k} D_k + \sum_k s_{a2k} \left(Y^{\delta-\alpha} D_k + Y^{-\alpha} D_k E_k + Y^{\gamma-\alpha} E_k\right) + \\
&\quad \textstyle\sum_k s_{a2k} \sum_{j=1}^{m_0} \sigma_{kj} \left(u_j(X) Y^{\beta-\alpha+\delta} + v_j(X) Y^{\beta-\alpha+\gamma} + w_j(X) Y^{2\beta-\alpha}\right) \ , \\
crs_2(b,\boldsymbol{X}) &= \ldots + \textstyle\sum_k s_{bk} E_k \ .
\end{aligned}$$

In this case, due to the extra inputs from the simulator, the critical coefficients of $V_i(\boldsymbol{X}^*)$ of $V(\boldsymbol{X})$ will be changed by extra addends $V_i^+(\boldsymbol{X}^*)$, depicted in Fig. 1. For example, $V_{\beta+\delta}(\boldsymbol{X}^*) = (b_\delta + 1)u_a(X) - u(X) + a_\delta v_b(X) + \sum_k (s_{c2k} - r_b s_{a2k}) \sum_j \sigma_{kj} u_j(X)$.

First, assume that the first verification equation is used. Then the coefficient of $Y^{-\alpha+\delta+\gamma} E_k$ of $V(\boldsymbol{X})$ (namely, $(b_\delta + 1)s_{a2k}$) implies that $s_{a2k} = 0$. Moreover, the coefficients of $Y^\delta D_k$ (namely, $r_b s_{a2k} + (b_\delta + 1)s_{a1k} - s_{c2k}$), $Y^\gamma E_k$ (namely, $r_b s_{a2k} + s_{bk}/(b_\delta + 1) - s_{c2k}$), $Y^\alpha D_k$ (namely, $r_B s_{a1k} - s_{c1k}$), and $D_k E_k$ (namely, $r_b s_{a2k} + s_{a1k} s_{bk} - s_{c2k}$) in $V(\boldsymbol{X})$ imply that either

(i) $s_{a1k} = s_{bk} = 0$ and thus $s_{c1k} = r_b s_{a1k}$, for all k, or
(ii) $s_{a1k} = 1/(b_\delta + 1)$ and $s_{bk} = b_\delta + 1$ for at least one k.

In the first case, $s_{c1k} = r_b s_{a1k}$ for all k and thus we can eliminate the $V_i^+(\boldsymbol{X}^*)$ addends in Fig. 1, and thus get back to the (already solved) knowledge-soundness setting that guarantees us that $\chi(X) = 0$.

In the second case, for some k, $A(\boldsymbol{X}) = s_{a1k} D_k + \ldots$ and $B(\boldsymbol{X}) = s_{bk} E_k + \ldots$ for $s_{a1k} = 1/(b_\delta + 1) \neq 0$ and $s_{bk} = b_\delta + 1 \neq 0$. Now, for $k_1 \neq k_2$, the coefficient of $D_{k_1} E_{k_2}$ is $s_{a1k_1} s_{bk_2} = 0$. Since $s_{a1k} = 0$ iff $s_{bk} = 0$, we get that $s_{a1k}, s_{bk} \neq 0$ for at most one index $k := k_0$. Thus, the polynomials $V_i^+(\boldsymbol{X}^*)$ in Fig. 1 are equal to $\sum_j \sigma_{k_0 j} u_j(X)$, $\sum_j \sigma_{k_0 j} v_j(X)$, and $\sum_j \sigma_{k_0 j} w_j(X)$, respectively. Note also that $s_{a2k} = 0$ and thus $s_{c2k} = 1$.

To guarantee that the prover is honest, we must make it impossible for the prover to include a term $s_{a1k_0} D_{k_0}$, for non-zero s_{a1k_0}, to $A(\boldsymbol{X})$. The first idea how to achieve this is by asking the prover to additionally output $[\mathsf{b}]_1$ and then letting the verifier to check that $[\mathsf{b}]_1 \bullet [1]_2 = [1]_1 \bullet [\mathsf{b}]_2$. Since in the kth query, the simulator also outputs $[\mathsf{b}_k]_1 = [e_k]_1$, then now $A(\boldsymbol{X})$ also depends on E_k. That is, the polynomial $A(\boldsymbol{X})$ has an additional monomial $-\sum_k s_{a3k} E_k$ for each simulation query, and similarly for polynomial $B_1(\boldsymbol{X}) = \ldots - \sum_k s_{b_1 3k} E_k$. Thus, checking that $[\mathsf{b}]_1 \bullet [1]_2 = [1]_1 \bullet [\mathsf{b}]_2$ only guarantees that $B(\boldsymbol{X}) = r_b Y^\alpha + v_b(X) Y^\beta + \sum s_{bk_0} E_{k_0}$ for some r_b, $v_b(X)$, and (possibly non-zero) $s_{bk_0} = s_{b_1 3k}$.

The problem here is that the added step can be seen as a knowledge-sound QA-NIZK argument Π_{sub} [JR13] that $[\mathsf{b}]_2$ belongs to the "subspace"[5] generated by $[\boldsymbol{M}(X,Y)]_1 = [Y^\alpha, Y^\beta, XY^\beta, \ldots, X^{n-1}Y^\beta]_1$. Since we allow for simulation queries, we need a *simulation-extractable* QA-NIZK argument for the subspace language. While such QA-NIZK arguments are known, they are not very efficient, [KW15]. A saving grace for us is that it suffices for Π_{sub} to be one-time simulation-extractable (OTSE): that is, it suffices for Π_{sub} to be knowledge-sound after one malicious query to the simulator. Really, as we argued before, it is possible that $s_{a1k}, s_{bk} \neq 0$ for at most one index $k = k_0$. More precisely, assume now that $s_{a1k_0} = s_{bk_0} = 1$. Then, as seen from the coefficient of $Y^{-\alpha}D_{k_1}E_{k_2}^2$, $s_{a2k_1}s_{bk_2} = 0$ for any k_1, k_2. Thus, $s_{bk_0} = 0$ for any $k \neq k_0$.

Next, we use the main idea of the one-time simulation-sound (OTSS) QA-NIZK of Kiltz-Wee [KW15, Section 3.3] Π_{otss} by introducing a tag τ, and requiring that the simulation queries are made on tags τ_k that differ from the tag τ for which the malicious prover constructs a forgery attempt. We introduce for this a new indeterminate Z, and ask the prover to compute the QA-NIZK argument with respect to the sum $\tau + Z$. This can be interpreted as making use of the pairwise independent function $H_{Z_1,Z_2}(\tau) = \tau Z_1 + Z_2$ [WC81]. Note that also Π_{otss} is somewhat inefficient and therefore we do use it directly.

Let $\boldsymbol{X} = (X, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{E}, Y, Z)$ and $\boldsymbol{X}^* = (X, \boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{D}, \boldsymbol{E}, Z)$. Then,

$$
\begin{aligned}
crs_1(a,\boldsymbol{X}) = {} & \textstyle\sum_{j=1}^{m_0} a_j^*(u_j(X)Y^{\beta-\eta+\delta} + v_j(X)Y^{\beta-\eta+\gamma} + w_j(X)Y^{2\beta-\eta}) + \\
& \textstyle\sum_{i=m_0+1}^{m} a_i^*(u_i(X)Y^{\beta-\alpha+\delta} + v_i(X)Y^{\beta-\alpha+\gamma} + w_i(X)Y^{2\beta-\alpha}) + \\
& h_a(X)\ell(X)Y^{2\beta-\alpha} + r_aY^\alpha + u_a(X)Y^\beta + a_\gamma Y^\gamma + a_\delta Y^\delta + \textstyle\sum_k a_{qk}Q_{1k} - \\
& \textstyle\sum_k s_{a1k}D_k + \sum_k s_{a2k}\left(Y^{\delta-\alpha}D_k + Y^{-\alpha}D_kE_k + Y^{\gamma-\alpha}E_k\right) + \\
& \textstyle\sum_k s_{a2k}\sum_{j=1}^{m_0}\sigma_{kj}\left(u_j(X)Y^{\beta-\alpha+\delta} + v_j(X)Y^{\beta-\alpha+\gamma} + w_j(X)Y^{2\beta-\alpha}\right) + \\
& r_{za}Y^\alpha Z + u_{za}(X)Y^\beta Z - \textstyle\sum_k s_{a3k}E_k(\tau_k + Z) \ , \\
crs_2(b,\boldsymbol{X}) = {} & r_bY^\alpha + v_b(X)Y^\beta + b_\delta Y^\delta + b_\eta Y^\eta + \textstyle\sum_k b_{qk}Q_{2k} + \sum_k s_{bk}E_k \ .
\end{aligned}
$$

where $u_{za}(X) \in \mathbb{Z}_p^{(\leq n-1)}[X]$. Then, for example, $B_1(\boldsymbol{X}) = crs_1(b_1, \boldsymbol{X})$. Recall that the verifier's second verification guarantees that for fixed τ, the second verification polynomial $V^{se}(\boldsymbol{X})$ satisfies $V^{se}(\boldsymbol{x}) = 0$, where

$$V^{se}(\boldsymbol{X}) := B_1(\boldsymbol{X}) - (\tau + Z)B(\boldsymbol{X}) \ .$$

Consider again first the case $V^{se}(\boldsymbol{x}) = 0$ as a polynomial. Looking at the coefficient of E_k in $V^{se}(\boldsymbol{X})$, we get $\tau_k s_{b_1 3k} = -\tau s_{bk}$, while looking at the coefficient of E_kZ in $V^{se}(\boldsymbol{X})$, we get $s_{b_1 3k} = -s_{bk}$. Since $\tau_k \neq \tau$, we get $s_{b_1 3k} = s_{bk} = 0$. From the earlier discussion, we obtain that $s_{a1k} = 0$ and thus $s_{c2k} = r_b s_{a2k}$, which means that the polynomials $V_i^+(\boldsymbol{X}^*)$ in Fig. 1 are equal to 0 and thus SASE of $\mathsf{S}_{\mathsf{qap}}^{\mathsf{se}}$ follows from the knowledge-soundness of $\mathsf{S}_{\mathsf{qap}}$.

[5] Since this subspace is trivial (equal to the whole space), we need to rely on a knowledge assumption to achieve security. See [FLSZ17] that used a similar technique to combine QA-NIZK and SNARKs.

Due to the introduction of new indeterminate, we will have a larger set of critical coefficients. Let $\tilde{a}_j = a_j - \sum_k s_{c2k}\sigma_{kj}$ for $j \le m_0$ and $\tilde{a}_j = c_j^*$ for $j > m_0$. Let $u(X) = \sum_{j=1}^m \tilde{a}_j u_j(X)$, $v(X) = \sum_{j=1}^m \tilde{a}_j v_j(X)$, and $w(X) = \sum_{j=1}^m \tilde{a}_j w_j(X)$. Let $R' = \{i = Y^{j_0} D_{k_1}^{j_1} E_{k_1}^{j_2} E_{k_2}^{j_3} Z^{j_4} : V_i(\boldsymbol{X}^*) = 0\}$. Let $k = 2\beta$ and

$$\mathsf{Crit}' = \{2\beta, \beta+\gamma, \beta+\delta, \gamma+\delta, \gamma+\eta, 2\delta, Y^\gamma D_{k_1}, E_{k_1}, E_{k_1} Z\}$$

be the new set of critical coefficients. (Here, the first 6 coefficients are the same as in the case of $\mathsf{S}_{\mathsf{qap}}$.) Let $\overline{\mathsf{Crit}'} = R' \setminus \mathsf{Crit}'$.

Theorem 2. *Let $\mathbf{R} = \mathsf{Inst}_{\mathsf{qap}} = (\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=0}^m)$ be a QAP instance. Let $\mathsf{S}_{\mathsf{qap}}^{\mathsf{se}}$ be the SASE SNARK (with highlighted entries) in Fig. 2.*

(1) Assume $\boldsymbol{\Delta}$ is chosen so that $\mathsf{Crit}' \cap \overline{\mathsf{Crit}'} = \emptyset$ and that $H_\infty(\mathcal{D}') = \omega(\log \lambda)$. If the $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{qap}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}), \mathcal{D}')$-SAK and $(2n-1, n-1)$-PDL assumptions hold then $\mathsf{S}_{\mathsf{qap}}^{\mathsf{se}}$ in Fig. 2 is ASE. If, additionally, $\mathbf{R}_{\mathsf{Inst}_{\mathsf{qap}}}$ is a hard-membership language then $\mathsf{S}_{\mathsf{qap}}^{\mathsf{se}}$ is SASE.

(2) $\mathsf{S}_{\mathsf{qap}}^{\mathsf{se}}$ is perfectly zero-knowledge.

Proof. **(1: ASE and SASE)** First, let $\mathcal{A}$ be an ASE adversary that succeeded in outputting (x, π) such that $\mathsf{x} \notin \mathcal{L}$ but V accepts. Since we are proving ASE, another part of the input to $\mathcal{A}$ is the reply of the Sim oracle to each query. Due to the use of a SAK assumption, $\mathbb{G}_\iota$-outputs of $\mathcal{A}$ have to belong to the span of her inputs (one part of the input is the CRS and of the simulator replies in this group) and of new random group elements and moreover, one can extract the corresponding coordinates. When replying to jth query, Sim samples fresh random integers d_j and e_j. We model d_j and e_j as new indeterminates D_j and E_j. Let $\boldsymbol{X} = (X, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{E}, Y, Z)$ be the vector of all indeterminates.

Since the second verification equation holds, from the coefficients of E_k and $E_k Z$ of $V^{se}(\boldsymbol{X}) = 0$ (and from $\tau \neq \tau_k$) we get $s_{a1k} = s_{bk} = 0$. Then, from the coefficient of $Y^\gamma E_k$ of $V(\boldsymbol{X}) = 0$ we get that $s_{c2k} = r_b s_{a2k}$ for each k. This means that the polynomials $V_i^+(\boldsymbol{X}^*)$ in Fig. 1 are all equal to 0. The rest of the proof of ASE now follows from the knowledge-soundness proof from Theorem 1.

To see that also SASE holds, note that from the ASE security proof it follows that a SASE adversary $\mathcal{A}$ cannot use the answers of simulation queries (produced when using a different tag) while creating an argument. This means that $\mathcal{A}$, after seeing a simulated argument for a statement x, creates a new argument π for the same statement (but under a different tag τ), but without the help of the oracle. Thus, $\mathcal{A}$ can be used to decide membership in the QAP language, a contradiction. Note that the SASE case cannot be reduced to knowledge-soundness, since it also must be impossible to create new arguments for true statements.

(2: zero-knowledge) similar to Theorem 1. □

6 SAP-Based SNARKs

In the following sections, we will describe SNARKs for different languages SAP, SSP, and QSP. Since these SNARKs and their security proofs are modifications of $\mathsf{S}_{\mathsf{qap}}$, we will omit most of the details.

Groth [Gro16] and Groth and Maller [GM17] used SAP (Square Arithmetic Programs) instead of QAP. The only algebraic distinction here is that $v(X) = u(X)$ and thus $\mathbf{R} = \mathsf{Inst}_{\mathsf{sap}} = (\mathbb{Z}_p, m_0, \{u_j, w_j\}_{j=0}^m)$ is a SAP instance. $\mathbf{R}_{\mathsf{Inst}_{\mathsf{sap}}}$ is defined as $\mathbf{R}_{\mathsf{Inst}_{\mathsf{qap}}}$ in Eq. (1) except that $v(X) = u(X)$. Thus, each gate in the arithmetic circuit gets the same left and right inputs, or, putting it another way, the circuit consists of squaring gates only. Since each multiplication gate $c = ab$ can be implemented two squaring gates ($ab = (a/2 + b/2)^2 - (a/2 - b/2)^2$), one can verify the correctness of an arbitrary d-gate arithmetic circuit by transferring it to a circuit that has $m^* \leq 2d$ squaring gates and then constructing a SNARK for SAP for the resulting circuit. The main motivation behind introducing SAP is that one can construct SNARK where $A(X, Y) = B(X, Y)$, which potentially makes the SNARK more efficient.

We will next describe how to modify our approach to the case of SAP. Since $u(X) = w(X)$, the corresponding key equation is $\chi_{sap}(X) = 0$, where

$$\chi_{sap}(X) = u(X)^2 - w(X) - h(X)\ell(X) \ .$$

In this case, we simplify Eqs. (3) and (4) by setting $v(X) = u(X)$ and $r_a = r_b$. Then $A(X, Y) = B(X, Y) = r_a Y^\alpha + u(X)Y^\beta$.

Thus, Eqs. (3) to (5) simplify to

$$\begin{aligned}
A(X,Y) &= B(X,Y) = r_a Y^\alpha + u(X)Y^\beta \ , \\
C(X,Y) &= (A(X,Y) + Y^\gamma)(A(X,Y) + Y^\delta) - Y^{\gamma+\delta} \\
&= u(X)(Y^{\beta+\gamma} + Y^{\beta+\delta}) + u(X)^2 Y^{2\beta} + r_a \left(r_a Y^\alpha + 2u(X)Y^\beta + Y^\gamma + Y^\delta\right) Y^\alpha \ , \\
&= \left(u(X)(Y^{\beta+\gamma} + Y^{\beta+\delta}) + w(X)Y^{2\beta}\right) + (u(X)^2 - w(X))Y^{2\beta} + \\
&\quad r_a \left(r_a Y^\alpha + 2u(X)Y^\beta + Y^\gamma + Y^\delta\right) Y^\alpha \ , \\
C_p(X,Y) &= \textstyle\sum_{j=1}^{m_0} a_j \left(u_j(X)(Y^{\beta-\eta+\gamma} + Y^{\beta-\eta+\delta}) + w_j(X)Y^{2\beta-\eta}\right) \ , \\
C_s(X,Y) &= \textstyle\sum_{j=m_0+1}^{m} a_j \left(u_j(X)(Y^{\beta-\alpha+\gamma} + Y^{\beta-\alpha+\delta}) + w_j(X)Y^{2\beta-\alpha}\right) + \\
&\quad h(X)\ell(X)Y^{2\beta-\alpha} + r_a(r_a Y^\alpha + 2u(X)Y^\beta + Y^\gamma + Y^\delta) \ .
\end{aligned}$$

We construct the SNARK $\mathsf{S}_{\mathsf{sap}}$ by correspondingly simplifying Fig. 2, see Fig. 3. We can find a suitable $\boldsymbol{\Delta}$ as in the case of QAP in Section 4, see Eq. (8).

Knowledge-soundness. Since $\mathsf{S}_{\mathsf{sap}}$ is an optimized version of $\mathsf{S}_{\mathsf{qap}}$, its knowledge-soundness can be proven by using the same approach. That is, one can follow the proof of Theorem 1. Let $h(X) := h_c(X) - r_a h_a(X)$. Let

$$\tilde{a}_j = \begin{cases} a_j - b_\eta a_j^* \ , & j \leq m_0 \ , \\ c_j^* - r_a a_j^* \ , & j > m_0 \ . \end{cases}$$

Denote $u(X) = \sum_{j=1}^m \tilde{a}_j u_j(X)$ and $w(X) = \sum_{j=1}^m \tilde{a}_j w_j(X)$. In this case, the "significant" coefficients $V_i(X, \boldsymbol{Q})$, $i \in \mathsf{Crit}$, of $V(\boldsymbol{X})$ are depicted in Fig. 4. The differences compared to Fig. 1 are solely due to the setting $v(X) = u(X)$.

$\mathsf{K}_{\mathsf{crs}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})$: Sample $x, y, z \leftarrow_{\$} \mathbb{Z}_p^*$, let $\mathsf{td} \leftarrow (x, y, z)$. Let

$$\mathsf{crs}_{\mathsf{P}} \leftarrow \begin{pmatrix} \left[\{u_j(x)y^{\beta-\alpha+\delta} + u_j(x)y^{\beta-\alpha+\gamma} + w_j(x)y^{2\beta-\alpha}\}_{j=m_0+1}^{m}, y^{\alpha}\right]_1, \\ \left[\{x^j y^{\beta}\}_{j=0}^{n-1}, \{x^i \ell(x) y^{2\beta-\alpha}\}_{j=0}^{n-2}, y^{\gamma}, y^{\delta}, y^{\alpha}z, \{x^j y^{\beta} z\}_{j=0}^{n-1}\right]_1, \\ \left[y^{\alpha}, \{x^i y^{\beta}\}_{i=0}^{n-1}\right]_2 \end{pmatrix};$$

$$\mathsf{crs}_{\mathsf{V}} \leftarrow \begin{pmatrix} \left[\{u_j(x)y^{\beta-\eta+\delta} + u_j(x)y^{\beta-\eta+\gamma} + w_j(x)y^{2\beta-\eta}\}_{j=1}^{m_0}, y^{\gamma}, z, y^{\gamma}z\right]_1, \\ \left[y^{\alpha}, y^{\delta}, y^{\eta}, y^{\eta}z, y^{\alpha}z\right]_2, [y^{\gamma+\delta}]_T \end{pmatrix};$$

$\mathsf{crs} \leftarrow (\mathsf{crs}_{\mathsf{P}}, \mathsf{crs}_{\mathsf{V}})$; return $(\mathsf{crs}, \mathsf{td})$;

$\mathsf{P}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \tau, (a_j)_{j=1}^{m_0}, (a_j)_{j=m_0+1}^{m})$:

$u(X) \leftarrow \sum_{j=1}^{m} a_j u_j(X)$; $w(X) \leftarrow \sum_{j=1}^{m} a_j w_j(X)$; $h(X) \leftarrow (u(X)^2 - w(X))/\ell(X)$;
$r_a \leftarrow_{\$} \mathbb{Z}_p$;
$[u']_1 \leftarrow r_a[y^{\alpha}]_1$; $[u'']_1 \leftarrow [u(x)y^{\beta}]_1$;
$[\mathsf{a}]_1 \leftarrow \tau\cdot([u']_1 + [u'']_1) + r_a[y^{\alpha}z]_1 + [u(x)y^{\beta}z]_1$; $[\mathsf{b}]_2 \leftarrow r_a[y^{\alpha}]_2 + [u(x)y^{\beta}]_2$;
$[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^{m} a_j[u_j(x)y^{\beta-\alpha+\delta} + u_j(x)y^{\beta-\alpha+\gamma} + w_j(x)y^{2\beta-\alpha}]_1 + [h(x)\ell(x)y^{2\beta-\alpha}]_1 + r_a\left([u']_1 + 2[u'']_1 + [y^{\gamma}]_1 + [y^{\delta}]_1\right)$;
return $\pi \leftarrow \left([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2\right)$;

$\mathsf{V}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \tau, (a_j)_{j=1}^{m_0}, \pi = ([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2))$:

$[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j[u_j(x)y^{\beta-\eta+\delta} + u_j(x)y^{\beta-\eta+\gamma} + w_j(x)y^{2\beta-\eta}]_1$;
Check that

1. $[\mathsf{c}_p]_1 \bullet [(\tau+z)y^{\eta}]_2 + [\mathsf{c}_s]_1 \bullet [(\tau+z)y^{\alpha}]_2 = [\mathsf{a} + (\tau+z)y^{\gamma}]_1 \bullet [\mathsf{b} + y^{\delta}]_2 - [(\tau+z)y^{\gamma+\delta}]_T$;
2. $[\mathsf{a}]_1 \bullet [1]_2 = [\tau+z]_1 \bullet [\mathsf{b}]_2$;

$\mathsf{Sim}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}, \mathsf{td} = y, \tau, \mathsf{x} = (a_j)_{j=1}^{m_0})$:

$[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j[u_j(x)y^{\beta-\eta+\delta} + u_j(x)y^{\beta-\eta+\gamma} + w_j(x)y^{2\beta-\eta}]_1$;
$d \leftarrow_{\$} \mathbb{Z}_p$; $[\mathsf{a}]_1 \leftarrow \tau\cdot d[1]_1 + d[z]_1$; $[\mathsf{b}]_2 \leftarrow e[1]_2$;
$[\mathsf{c}_s]_1 \leftarrow y^{-\alpha}((d^2 + d(y^{\delta} + y^{\gamma}))[1]_1 - y^{\eta}[\mathsf{c}_p]_1)$;
return $\pi \leftarrow \left([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2\right)$;

Fig. 3. The new SNARKs for SAP and SSP: knowledge-sound $\mathsf{S}_{\mathsf{sap}}$ (*without* highlighted entries) and SASE $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ (*with* highlighted entries). $\mathsf{S}_{\mathsf{ssp}}$ is like $\mathsf{S}_{\mathsf{sap}}$ and $\mathsf{S}_{\mathsf{ssp}}^{\mathsf{se}}$ is like $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$, except that then also $w_j(X) = u_j(X)$.

Let $\mathbf{R} = (\mathbb{Z}_p, m_0, \{u_j, w_j\}_{j=0}^{m})$ be a SAP instance. Here, we need to use the knowledge assumption for $\mathcal{D}_{sap}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})$, where for $\mathsf{K}_{\mathsf{crs}}$ depicted in Fig. 3,

$$\mathcal{D}_{sap}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) = \{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})\} \ .$$

Theorem 3. *Let* $\mathbf{R} = \mathsf{Inst}_{\mathsf{sap}} = (\mathbb{Z}_p, m_0, \{u_j, w_j\}_{j=0}^{m})$ *be a SAP instance.*
(1) Assume $\boldsymbol{\Delta}$ *is chosen so that* $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$ *and that* $H_{\infty}(\mathcal{D}') = \omega(\log \lambda)$. *If the* $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{sap}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}), \mathcal{D}')$*-SAK and* $(2n-1, n-1)$*-PDL assumptions hold*

Y^i	Coeff. $V_i(\boldsymbol{X}^*)$ (KS and SASE)	$V_i^+(\boldsymbol{X}^*)$ (SASE only)
$Y^{2\beta}$	$u_a(X)v_b(X) - w(X) - h(X)\ell(X)$	$\sum_k(\tau s_{c2k} - r_a s_{a2k})\sum_j \sigma_{kj} w_j(X)$
$Y^{\beta+\gamma}$	$(a_\gamma + \tau)v_b(X) - v(X)$	$\sum_k(\tau s_{c2k} - r_a s_{a2k})\sum_j \sigma_{kj} v_j(X)$
$Y^{\beta+\delta}$	$(b_\delta + 1)u_a(X) - u(X) + a_\delta v_b(X)$	$\sum_k(\tau s_{c2k} - r_a s_{a2k})\sum_j \sigma_{kj} u_j(X)$
$Y^{\gamma+\delta}$	$b_\delta(a_\gamma + \tau) + a_\gamma$	
$Y^{2\delta}$	$(b_\delta + 1)\, a_\delta$	
$Y^{\eta+\gamma}$	$(a_\gamma + \tau)b_\eta$	

Fig. 4. Critical coefficients in $\mathsf{S}_{\mathsf{qap}}$ (left, $\tau = 1$) and addends to the same coefficients in the SASE case (right).

then $\mathsf{S}_{\mathsf{sap}}$ *in Fig. 2 is knowledge-sound.*
(2) $\mathsf{S}_{\mathsf{sap}}$ *is perfectly zero-knowledge.*

Proof (Sketch). **(1: knowledge-soundness)** Since the rest of the knowledge-sound is essentially the same as in the proof of Theorem 1, we only reprove the Step 1 from that proof.

Consider the polynomials in Fig. 4. Since $b_\delta + a_\gamma(b_\delta + 1) = 0$, we get $a_\gamma = -b_\delta/(b_\delta + 1)$ and $a_\gamma, b_\delta \neq -1$ and $(a_\gamma + 1)(b_\delta + 1) = 1$. Thus $a_\delta = b_\eta = 0$, which means that $\tilde{a}_j = a_j$ for $j \leq m_0$. Thus, $u_a(X)v_b(X) = u(X)^2$ and $u(X)^2 - w(X) = h(X)\ell(X)$, which means that $\chi_{sap}(X) = 0$.

(2: zero-knowledge) as in Theorem 1, except that we use only one new trapdoor d due to the fact that $\mathsf{a} = \mathsf{b}$. □

Efficiency. Clearly, in $\mathsf{S}_{\mathsf{sap}}$, the CRS has $(n) + (n-1) + m + 2 = 2n + m + 1$ elements from $\mathbb{G}_1$ and $n + 3$ elements from $\mathbb{G}_2$. The prover's computation is $n + 1$ exponentiations to compute $[\mathsf{a}]_1$, $n + 1$ exponentiations to compute $[\mathsf{b}]_2$, and $1 + (m - m_0) + (n - 1) = n + m - m_0$ exponentiations to compute $[\mathsf{c}_s]_1$.

SASE SNARK $\mathbf{S}^{\mathbf{se}}_{\mathsf{sap}}$. Consider the case of SASE with $\mathsf{S}_{\mathsf{sap}}$. Taking into account answers from simulation queries, the polynomials crs_1 and crs_2 have the following new addends:

$$crs_1(a,\boldsymbol{X}) = \ldots + \sum_k s_{a1k} D_k + \sum_k s_{a2k} \left(Y^{\delta-\alpha} + Y^{-\alpha} D_k + Y^{\gamma-\alpha}\right) D_k + \sum_k s_{a2k} \sum_{j=1}^{m_0} \sigma_{kj} \left(u_j(X) Y^{\beta-\alpha+\delta} + u_j(X) Y^{\beta-\alpha+\gamma} + w_j(X) Y^{2\beta-\alpha}\right) ,$$

$$crs_2(b,\boldsymbol{X}) = \ldots + \sum_k s_{bk} D_k \ .$$

(Compared to $\mathsf{S}_{\mathsf{qap}}$, we just changed $v_j(X)$ to $u_j(X)$ and E_k to D_k.) In the honest case, $[\mathsf{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\mathsf{b}]_2$. However, if the verifier additionally checks that $[\mathsf{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\mathsf{b}]_2$, one obtains the guarantee that $A(\boldsymbol{X}) = B(\boldsymbol{X}) = r_a Y^\alpha + u_a(X) Y^\beta - \sum s_{a1k} D_k$. Again, this does not guarantee simulation-extractability.

We proceed similarly to the case of $\mathsf{S}_{\mathsf{qap}}$ but take into account the fact that $\mathsf{a} = \mathsf{b}$ in the honest case. Recall that in $\mathsf{S}_{\mathsf{qap}}$, one constructed b_1, such that

$\mathsf{b}_1 = \mathsf{b}(\tau + z)$ (where τ is a tag and Z is a new indeterminate). What we do next is to define $\mathsf{a} = \mathsf{b}_1$, and then modify the verification equations to take that into account, as in Fig. 3. This defines two verification polynomials,

$$\begin{aligned} V(\boldsymbol{X}) =& (A(\boldsymbol{X}) + (\tau + Z)Y^{\gamma})(B(\boldsymbol{X}) + Y^{\delta}) - (\tau + Z)C_p(\boldsymbol{X})Y^{\eta} - \\ & (\tau + Z)C_s(\boldsymbol{X})Y^{\alpha} - (\tau + Z)Y^{\gamma+\delta} \ , \\ V^{se}(\boldsymbol{X}) =& A(\boldsymbol{X}) - (\tau + Z)B(\boldsymbol{X}) \ . \end{aligned}$$

After this, note that the CRS has to additionally include some new elements (highlighted in Fig. 3). This changes the polynomials crs_1 and crs_2 to

$$\begin{aligned} crs_1(a,\boldsymbol{X}) = & \textstyle\sum_{j=1}^{m_0} a_j^*(u_j(X)Y^{\beta-\eta+\delta} + u_j(X)Y^{\beta-\eta+\gamma} + w_j(X)Y^{2\beta-\eta}) + \\ & \textstyle\sum_{i=m_0+1}^{m} a_i^*(u_i(X)Y^{\beta-\alpha+\delta} + u_i(X)Y^{\beta-\alpha+\gamma} + w_i(X)Y^{2\beta-\alpha}) + \\ & h_a(X)\ell(X)Y^{2\beta-\alpha} + r_aY^{\alpha} + u_a(X)Y^{\beta} + a_{\gamma}Y^{\gamma} + a_{\delta}Y^{\delta} + \textstyle\sum_k a_{qk}Q_{1k} - \\ & \textstyle\sum_k s_{a1k}(\tau_k + Z)D_k + \sum_k s_{a2k}\left(Y^{\delta-\alpha}D_k + Y^{-\alpha}D_k^2 + Y^{\gamma-\alpha}D_k\right) + \\ & \textstyle\sum_k s_{a2k}\sum_{j=1}^{m_0}\sigma_{kj}\left(u_j(X)Y^{\beta-\alpha+\delta} + u_j(X)Y^{\beta-\alpha+\gamma} + w_j(X)Y^{2\beta-\alpha}\right) + \\ & a_{Y^{\alpha}Z}Y^{\alpha}Z + u_{za}(X)Y^{\beta}Z + a_ZZ + a_{\gamma z}Y^{\gamma}Z \ , \\ crs_2(b,\boldsymbol{X}) = & r_bY^{\alpha} + u_b(X)Y^{\beta} + b_{\delta}Y^{\delta} + b_{\eta}Y^{\eta} + \textstyle\sum_k b_{qk}Q_{2k} + \\ & \textstyle\sum_k s_{bk}D_k + b_{\alpha z}Y^{\alpha}Z + b_{\eta z}Y^{\eta}Z \ . \end{aligned}$$

Since the second verification accepts holds, then $V^{se}(\boldsymbol{x}) = 0$. If $V^{se}(\boldsymbol{X}) = 0$ as a polynomial then $A(\boldsymbol{X}) = (\tau + Z)B(\boldsymbol{X})$, and from the coefficients of D_k and D_kZ of $V^{se}(\boldsymbol{X})$ we get $\tau_k s_{a1k} = \tau s_{bk}$ and $s_{a1k} = s_{bk}$. Since $\tau \neq \tau_k$, this means $s_{a1k} = s_{bk} = 0$. From the coefficient of $Y^{\gamma}D_k$ of $V(\boldsymbol{X})$, we get $r_b s_{a2k_1} - \tau s_{c2k_1} + s_{bk_2}(A_{\gamma} + \tau) = 0$. and thus $\tau s_{c2k_1} = r_b s_{a2k_1}$. Thus means that $V_i^+(\boldsymbol{X}^*) = 0$ in Fig. 4 and thus, as in the case of $\mathsf{S}_{\mathsf{qap}}^{\mathsf{se}}$, SASE of $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ can be reduced to the knowledge-soundness of $\mathsf{S}_{\mathsf{sap}}$ and the hardness of SAP.

Theorem 4. *Let* $\mathbf{R} = \mathsf{Inst}_{\mathsf{sap}} = (\mathbb{Z}_p, m_0, \{u_j, w_j\}_{j=0}^{m})$ *be a SAP instance.*
(1) Assume $\boldsymbol{\Delta}$ *is chosen so that* $\mathsf{Crit}' \cap \overline{\mathsf{Crit}'} = \emptyset$ *and that* $H_{\infty}(\mathcal{D}') = \omega(\log \lambda)$. *If the* $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{sap}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}), \mathcal{D}')$*-SAK and* $(2n-1, n-1)$*-PDL assumptions hold then* $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ *in Fig. 3 is ASE. If, additionally,* $\mathbf{R}_{\mathsf{Inst}_{\mathsf{sap}}}$ *is a hard-membership language then* $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ *is SASE.*
(2) $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ *is perfectly zero-knowledge.*

Efficiency. Clearly, in $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$, the CRS has $(n)+(n-1)+m+2+(n+3) = 3n+m+4$ elements from $\mathbb{G}_1$ and $n+5$ elements from $\mathbb{G}_2$. The prover's computation is $n+1+1 = n+2$ exponentiations to compute $[\mathsf{a}]_1$, $n+1$ exponentiations to compute $[\mathsf{b}]_2$, and $1 + (m - m_0) + (n-1) = n + m - m_0$ exponentiations to compute $[\mathsf{c}_s]_1$. The verifier executes 5 pairings and $m_0 + 4$ exponentiations.

7 SSP-Based SNARKs

In this section, we will construct a knowledge-sound SNARK $\mathsf{S}_{\mathsf{ssp}}$ and a SASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ for SSP (Square Span Programs, [DFGK14]). We recall that by using SSP, one can prove that different linear combinations of witness coefficients are simultaneously Boolean. As shown in [DFGK14], this is sufficient to show that a Boolean circuit has been correctly evaluated on (secret or public) inputs:

- For each wire, one checks that the wire value is Boolean.
- For each gate, one can check that it has implemented its Boolean function correctly by checking that certain linear combination of its input and output wire values is Boolean. For example, $a \bar{\wedge} b = c$ iff $a + b + 2c - 2 \in \{0, 1\}$ and $a \oplus b = c$ iff $(a + b + c)/2 \in \{0, 1\}$ [DFGK14].

Thus, one can implement SSP by using a QAP-type approach, by checking $n = d + m$ constraints of type $(\sum_{j=1}^{m} U_{ij} a_j)^2 = \sum_{j=1}^{m} U_{ij} a_j$, $i \in [1\,..\,n]$, where d is the number of the gates and m is the number of the wires. (In a QAP-based approach for arithmetic circuits, $n = d$.) Based on this observation, we design $\mathsf{S}_{\mathsf{ssp}}$ around the verification equation as in Section 4. The only difference in the language is that $u(X) = v(X) = w(X)$, and thus the key equation is $\chi_{ssp}(X) = 0$, where

$$\chi_{ssp}(X) = u(X)(u(X) - 1) - h(X)\ell(X) \ .$$

Thus, $h(X) = u(X)(u(X) - 1)/\ell(X)$ is a polynomial iff the prover is honest. The new SNARK $\mathsf{S}_{\mathsf{ssp}}$ for SSP in Fig. 3 is like $\mathsf{S}_{\mathsf{sap}}$, except that now we have $u_j(X) = v_j(X) = w_j(X)$ instead of just $u_j(X) = v_j(X)$.

Let $\mathbf{R} = (\mathbb{Z}_p, m_0, \{u_j\}_{j=0}^m)$ be a SSP instance. $\mathbf{R}_{\mathsf{Inst}_{\mathsf{ssp}}}$ is defined as $\mathbf{R}_{\mathsf{Inst}_{\mathsf{qap}}}$ in Eq. (1) except that $u(X) = v(X) = w(X)$. Here, we need to use the knowledge assumption for $\mathcal{D}_{ssp}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})$, where for $\mathsf{K}_{\mathsf{crs}}$ depicted in Fig. 3,

$$\mathcal{D}_{ssp}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) = \{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})\} \ .$$

Theorem 5. *Let* $\mathbf{R} = \mathsf{Inst}_{\mathsf{ssp}} = (\mathbb{Z}_p, m_0, \{u_j\}_{j=0}^m)$ *be a SSP instance.*
(1) Assume $\boldsymbol{\Delta}$ *is chosen so that* $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$ *and that* $H_\infty(\mathcal{D}') = \omega(\log \lambda)$. *If the* $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{ssp}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}), \mathcal{D}')$*-SAK and* $(2n-1, n-1)$*-PDL assumptions hold then* $\mathsf{S}_{\mathsf{ssp}}$ *in Fig. 3 is knowledge-sound.*
(2) $\mathsf{S}_{\mathsf{ssp}}$ *is perfectly zero-knowledge.*

Proof. Follows directly from Theorem 3. □

Importantly, since a_j are Boolean, it is cheaper to compute say $[u(X)u^\beta]_1 \leftarrow \sum_{j=1}^m a_j [u_j(X) y^\beta]_1$: this requires m multiplications compared to n exponentiations in the case of QAP and SAP. (Here, and in the next section, we count the number of multiplications in the worst case. In the average case, it will be reduce by a factor of two.) Moreover, setting $w_j(X) = u_j(X)$ allows for additional minor optimizations. For example, to compute $[\mathsf{a}]_1$ and $[\mathsf{c}_s]_1$, the prover can first set $[u']_1 \leftarrow r_a [y^\alpha]_1$; $[u'']_1 \leftarrow \sum_{j=1}^m a_j [u_j(x) y^\beta]_1$,

and then $[\mathsf{a}]_1 \leftarrow \tau\cdot([u']_1 + [u'']_1) + r_a[y^\alpha z]_1 + \sum_{j=1}^m a_j[u_j(x)y^\beta z]_1$ and $[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^m a_j[u_j(x)y^{\beta-\alpha+\delta} + u_j(x)y^{\beta-\alpha+\gamma} + w_j(x)y^{2\beta-\alpha}]_1 + [h(x)\ell(x)y^{2\beta-\alpha}]_1 + r_a\left([u']_1 + 2[u'']_1 + [y^\gamma]_1 + [y^\delta]_1\right)$. Thus, the prover spends one exponentiation and m multiplications in $\mathbb{G}_1$ to compute $[u']_1$ and $[u'']_1$, and additional $m - m_0$ multiplications and $(n-1)+1 = n$ exponentiations in $\mathbb{G}_1$ to compute $[\mathsf{c}_s]_1$. She also spends 1 exponentiation and m multiplications in $\mathbb{G}_2$ to compute $[\mathsf{b}]_2$.

SASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$. $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ is defined as $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$, setting $w_j(X) = u_j(X)$.

Theorem 6. *Let $\mathbf{R} = \mathsf{Inst}_{\mathsf{ssp}} = (\mathbb{Z}_p, m_0, \{u_j\}_{j=0}^m)$ be a SSP instance.*
(1) Assume $\boldsymbol{\Delta}$ is chosen so that $\mathsf{Crit}' \cap \overline{\mathsf{Crit}'} = \emptyset$ and that $H_\infty(\mathcal{D}') = \omega(\log\lambda)$. If the $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{ssp}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}), \mathcal{D}')$-SAK and $(2n-1, n-1)$-PDL assumptions hold then $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ in Fig. 3 is ASE. If, additionally, $\mathbf{R}_{\mathsf{Inst}_{\mathsf{ssp}}}$ is a hard-membership language then $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ is SASE.
(2) $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ is perfectly zero-knowledge.

Efficiency-wise, $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ is like $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$, except that the prover needs to compute $3m - m_0$ multiplications and $n+2$ exponentiations in $\mathbb{G}_1$ and 1 exponentiation and m multiplications in $\mathbb{G}_2$.

8 QSP-Based SNARKs

In addition to QAP, Gennaro *et al.* [GGPR13] proposed another formalism called QSP (Quadratic Span Program). This approach was further optimized by Lipmaa [Lip13]. Without going to full details, we mention that there exists a reduction from Boolean circuit satisfiability to QSPs. The reduction itself is not as efficient than the reduction to SSPs, and in particular, the size of the QSP, given the same circuit, is considerably larger than that of the SSP. (According to [DFGK14], if the circuit has m wires and n gates, SSP matrices have size $\approx m \times (m+n)$ while QSP matrices have size $\approx 14n \times 11n$.) However, QSP-based solutions like the SSP-based solutions have a short argument and CRS. They also result in 2-query linear PCPs for Circuit-SAT, [BCI+13,Lip13].

In this section, we assume that one has already constructed a reduction to the QSP. Given now a concrete QSP instance, we construct a knowledge-sound and a SASE SNARK fo QSP. We also assume that the QSP matrix size is $n \times m$ (thus, n and m do not correspond to the circuit size anymore.)

In the case of QSP [GGPR13,Lip13], $w(X) = 0$ and thus the key equation is

$$\chi_{qsp}(X) = u(X)v(X) - h(X)\ell(X) = 0 \ .$$

We now construct the SNARK $\mathsf{S}_{\mathsf{qsp}}$, see Fig. 2 (the case $w_j(X) = u_j(X)$). Thus, each cost parameter is the same as in the case of $\mathsf{S}_{\mathsf{qap}}$ except that there are significantly more constraints (that are hidden in the reduction from circuits to QSP, [Lip13]) and thus n is larger.

Let $\mathbf{R} = (\mathbb{Z}_p, m_0, \{u_j, v_j\}_{j=0}^m)$ be a QSP instance. $\mathbf{R}_{\mathsf{lnst}_{\mathsf{qsp}}}$ is defined as $\mathbf{R}_{\mathsf{lnst}_{\mathsf{qap}}}$ in Eq. (1) except that $w(X) = 0$. Here, we need to use the knowledge assumption for $\mathcal{D}_{qsp}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})$, where for $\mathsf{K}_{\mathsf{crs}}$ depicted in Fig. 2,

$$\mathcal{D}_{qsp}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) = \{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}})\} \ .$$

Theorem 7. *Let $\mathbf{R} = (\mathbb{Z}_p, m_0, \{u_j, v_j\}_{j=0}^m)$ be a QSP instance.*
(1) Assume $\boldsymbol{\Delta}$ is chosen so that $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$ and that $H_\infty(\mathcal{D}') = \omega(\log \lambda)$. If the $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{qsp}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}), \mathcal{D}')$-SAK and $(2n-1, n-1)$-PDL assumptions hold then $\mathsf{S}_{\mathsf{qsp}}$ in Fig. 2 is knowledge-sound.
(2) $\mathsf{S}_{\mathsf{qsp}}$ is perfectly zero-knowledge.

As in the case of SSP, since the witness is Boolean, we can significantly speed up the prover's computation. Really, the prover computes $[\mathsf{a}]_1 \leftarrow r_a[y^\alpha]_1 + \sum_{j=1}^m a_j[u_j(x)y^\beta]_1$, $[\mathsf{b}]_2 \leftarrow r_b[y^\alpha]_2 + \sum_{j=1}^m a_j[v_j(x)y^\beta]_2$, and $[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^m a_j[u_j(x)y^{\beta-\alpha+\delta} + v_j(x)y^{\beta-\alpha+\gamma}]_1 + [h(x)\ell(x)y^{2\beta-\alpha}]_1 + r_b([\mathsf{a}]_1 + [y^\gamma]_1) + r_a([y^\delta]_1 + \sum_{j=1}^m a_j[v_j(x)y^\beta]_1)$. Thus, the prover executes $1+1+((n-1)+1) = n+2$ exponentiations and $m + m + ((m - m_0) + m) = 4m - m_0$ multiplications in $\mathbb{G}_1$ and 1 exponentiation and m multiplications in $\mathbb{G}_2$.

SASE SNARK $\mathbf{S}^{\mathsf{se}}_{\mathsf{qsp}}$. One obtains a SASE version of $\mathsf{S}_{\mathsf{qsp}}$ exactly as in the case of $\mathsf{S}_{\mathsf{qap}}$ in Section 4. Thus, $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ is like $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$, except that $w_j(X) = 0$.

Theorem 8. *Let $\mathbf{R} = (\mathbb{Z}_p, m_0, \{u_j, v_j\}_{j=0}^m)$ be a QSP instance.*
(1) Assume $\boldsymbol{\Delta}$ is chosen so that $\mathsf{Crit}' \cap \overline{\mathsf{Crit}'} = \emptyset$ and that $H_\infty(\mathcal{D}') = \omega(\log \lambda)$. If the $(\mathbb{G}_1, \mathbb{G}_2, \mathcal{D}_{qsp}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}), \mathcal{D}')$-SAK and $(2n-1, n-1)$-PDL assumptions hold then $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ in Fig. 2 is ASE. If, additionally, $\mathbf{R}_{\mathsf{lnst}_{\mathsf{qsp}}}$ is a hard-membership language then $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ is SASE.
(2) $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ is perfectly zero-knowledge.

Efficiency-wise, compared to $\mathsf{S}_{\mathsf{qsp}}$, the prover additionally computes $[\mathsf{b}_1]_1 \leftarrow r_b(\tau[y^\alpha]_1 + [y^\alpha z]_1) + \tau \sum_{j=1}^m a_j[v_j(x)y^\beta]_1 + \sum_{j=1}^m a_j[v_j(x)y^\beta z]_1$, which takes 3 exponentiations and m additional multiplications (since $\sum_{j=1}^m a_j[v_j(x)y^\beta]_1$ is already computed) in $\mathbb{G}_1$.

Acknowledgment. Helger Lipmaa was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780477 (project PRIViLEDGE), and by the Estonian Research Council grant (PRG49).

References

ABLZ17. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017. 1, 1, 3, 3, 4

ALSZ18. Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michał Zając. On QA-NIZK in the BPK Model. Technical Report 2018/877, IACR, September 18, 2018. Available from `https://eprint.iacr.org/2018/877`, last checked version from 16 May 2019. 3

Bag19. Karim Baghery. On the efficiency of privacy-preserving smart contract systems. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 118–136. Springer, Heidelberg, July 2019. 1, 1

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, And Back Again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349, Cambridge, MA, USA, January 8–10, 2012. ACM Press. 1

BCCT13. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC 2013*, pages 241–250, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 1

BCG+13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013. 2

BCG+14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014. 1

BCI+10. Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 237–254. Springer, Heidelberg, August 2010. 3

BCI+13. Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013. 3, 1, 8

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the Existence of Extractable One-Way Functions. In David Shmoys, editor, *STOC 2014*, pages 505–514, New York, NY, USA, May 31 – Jun 3, 2014. ACM Press. A.1

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. 1

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016. 1, 1, 2, 3, 3, A.1

BG18. Sean Bowe and Ariel Gabizon. 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`. 1, 1

BLS04. Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In Mitsuru Matsui and Robert J. Zuccherato, edi-

tors, *SAC 2003*, volume 3006 of *LNCS*, pages 17–25. Springer, Heidelberg, August 2004. 2

Bow17. Sean Bowe. BLS12-381: New zk-SNARK Elliptic Curve Construction. Blog post, `https://blog.z.cash/new-snark-curve/`, last accessed in July, 2018, March 11, 2017. 2

BV98. Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71. Springer, Heidelberg, May / June 1998. 3

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. 1

CGGN17. Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 229–243. ACM Press, October / November 2017. 1

Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992. 1, 1

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001. 1, 2

Den02. Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, Heidelberg, December 2002. 1, 1

DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014. 1, 1, 2, 7, 8

DFKP13. George Danezis, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. pages 27–30, Berlin, Germany, November 4, 2013. ACM. 1

DGP+19. Vanesa Daza, Alonso González, Zaira Pindado, Carla Ràfols, and Javier Silva. Shorter quadratic QA-NIZK proofs. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 314–343. Springer, Heidelberg, April 2019. 1

DHLW10. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Heidelberg, December 2010. 2, 4, 5

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013. 1, 2

Fis00. Marc Fischlin. A note on security proofs in the generic model. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 458–469. Springer, Heidelberg, December 2000. 1, 1

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. 1, 3, 4

FLSZ17. Prastudy Fauzi, Helger Lipmaa, Janno Siim, and Michal Zajac. An efficient pairing-based shuffle argument. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 97–127. Springer, Heidelberg, December 2017. 1, 5

FLZ16. Prastudy Fauzi, Helger Lipmaa, and Michal Zajac. A shuffle argument secure in the generic model. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 841–872. Springer, Heidelberg, December 2016. 1

Fuc18. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018. 1, 1

Gab19. Ariel Gabizon. On the security of the BCTV Pinocchio zk-SNARK variant. Technical Report 2019/199, IACR, February 5, 2019. Available from `https://eprint.iacr.org/2019/199`. 1

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. 1, 1, 2, 2, 4, 8

GJM03. Philippe Golle, Stanislaw Jarecki, and Ilya Mironov. Cryptographic primitives enforcing communication and storage complexity. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 120–135. Springer, Heidelberg, March 2003. 1

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017. 1, 1, 2, 6, 4, 5

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985. 1

Gol93. Oded Goldreich. A Uniform-Complexity Treatment of Encryption and Zero-Knowledge. *J. Cryptology*, 6(1):21–53, 1993. A.1

GPS08. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for Cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. 2

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006. 1, 3

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. 1, 1

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. 1, 1, 1, 1, 2, 2, 4, 4, 4, 6, A.1

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. 1

Ica09. Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316. Springer, Heidelberg, August 2009. 1, 3, 3

JR10. Tibor Jager and Andy Rupp. The semi-generic group model and applications to pairing-based cryptography. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 539–556. Springer, Heidelberg, December 2010. 2

JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013. 1, 5

KW15. Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, April 2015. 1, 3, 5, 4, 5

KZM+15. Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. How to use SNARKs in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093, 2015. `http://eprint.iacr.org/2015/1093`. 1, 1

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012. 1, 2

Lip13. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013. 1, 3, 1, 8

Nec94. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994. 1

Par15. Bryan Parno. A note on the unsoundness of vnTinyRAM's SNARK. Cryptology ePrint Archive, Report 2015/437, 2015. `http://eprint.iacr.org/2015/437`. 1

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. 1

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. 1, 2

Sch80. Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980. 3

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. 1

SPMS02. Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung,

editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 93–110. Springer, Heidelberg, August 2002. 1

Sta08. Grzegorz Stachowiak. Proofs of knowledge with several challenge values. Cryptology ePrint Archive, Report 2008/181, 2008. `http://eprint.iacr.org/2008/181`. 2

THS+09. Pairat Thorncharoensri, Qiong Huang, Willy Susilo, Man Ho Au, Yi Mu, and Duncan S. Wong. Escrowed Deniable Identification Schemes. In Dominik Slezak, Tai-Hoon Kim, Wai-Chi Fang, and Kirk P. Arnett, editors, *FGIT-SecTech 2009*, volume 58 of *Communications in Computer and Information Science*, pages 234–241, Jeju Island, Korea, December 10–12, 2009. Springer. 2

WC81. Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981. 5

Zip79. Richard Zippel. Probabilistic Algorithms for Sparse Polynomials. In Edward W. Ng, editor, *EUROSM 1979*, volume 72 of *LNCS*, pages 216–226, Marseille, France, June 1979. Springer, Heidelberg. 3

A Formal Security Definitions

A.1 Zero-Knowledge

As in [Gro16], we define all security notions against a non-uniform adversary. However, since our security reductions are uniform, it is a simple matter to consider only uniform adversaries, as it was done by Bellare *et al.* [BFS16] (see also [Gol93]).

Definition 1 (Perfect Completeness). *A non-interactive argument Ψ is* perfectly complete for $\mathcal{R}$*, if for all λ, all $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$, tag $\tau \in \mathsf{Tags}$, and $(\mathsf{x}, \mathsf{inp}) \in \mathbf{R}$,*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) : \\ \mathsf{V}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \tau, \mathsf{x}, \mathsf{P}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \tau, \mathsf{x}, \mathsf{inp})) = 1\end{array}\right] = 1 \ .$$

Definition 2 (Computational Knowledge-Soundness). *Ψ is computationally (adaptively)* knowledge-sound for $\mathcal{R}$*, if for every non-uniform PPT $\mathcal{A}$, there exists a non-uniform PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. for all λ,*

$$\Pr\left[\begin{array}{l}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) \leftarrow \mathcal{R}(1^\lambda); (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}); r \leftarrow_r \mathsf{RND}(\mathcal{A}); \\ (\tau, \mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}; r); \mathsf{inp} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}; r) : \\ (\mathsf{x}, \mathsf{inp}) \notin \mathbf{R} \wedge \mathsf{V}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \tau, \mathsf{x}, \pi) = 1\end{array}\right] \approx_\lambda 0 \ .$$

Here, $\mathsf{aux}_{\mathbf{R}}$ can be seen as a common auxiliary input to $\mathcal{A}$ and $\mathsf{Ext}_{\mathcal{A}}$ that is generated by using a benign [BCPR14] relation generator; we recall that we just think of $\mathsf{aux}_{\mathbf{R}}$ as being the description of a secure bilinear group. A knowledge-sound argument system is called an *argument of knowledge*.

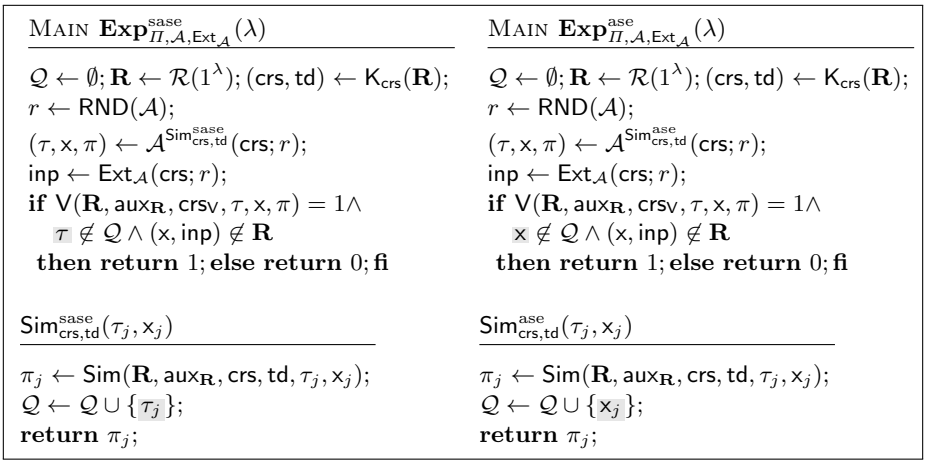

Fig. 5. Simulation-extractability experiments: strong any-simulation (SASE, left) and any-simulation (ASE, right). Differences are highlighted

Definition 3 (Statistically Unbounded ZK [Gro06]). *Ψ is* statistically unbounded Sub-ZK for $\mathcal{R}$*, if for all λ, all $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$, all $\tau \in \mathsf{Tags}$, and all computationally unbounded $\mathcal{A}$, $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$, where*

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

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

A.2 Simulation-Extractability

ASE (any simulation-extractability) and SASE (strong any simulation-extractability) guarantee knowledge-soundness even if one can obtain simulation answers to arbitrary statement, with the following difference: an ASE SNARK does not use tags and explicitly forbids simulation queries to the same statement that the adversary outputs. On the other hand, an SASE SNARK uses tags and allows to ask simulation queries even to the statement that the adversary outputs; however, it forbids to ask simulation queries for the same tag.

Definition 4 (SASE SNARK [DHLW10,KW15,GM17]). *Let $\Pi = (\mathsf{K}_{\mathsf{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a SNARK for relation $\mathbf{R}$. Define*

$$\mathsf{Adv}^{\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)] \ ,$$

where the experiment $\mathbf{Exp}^{\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ is depicted in Fig. 5. Π is non-black-box strong any-simulation-extractable (SASE) *if for any PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$ such that $\mathsf{Adv}^{\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \approx_\lambda 0$.*

Definition 5 (ASE SNARK [DHLW10,KW15,GM17]). *Let* $\Pi = (\mathsf{K}_{\mathsf{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a SNARK for relation* $\mathbf{R}$*. Define*

$$\mathsf{Adv}^{\mathrm{ase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{ase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)] \ ,$$

where the experiment $\mathbf{Exp}^{\mathrm{ase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is depicted in Fig. 5.* Π *is* non-black-box any-simulation-extractable (ASE) *if for any PPT adversary* $\mathcal{A}$ *there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that* $\mathsf{Adv}^{\mathrm{ase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \approx_\lambda 0$*.*

B Discussion: Verification Equation with One Pairing

Consider the case the verification equation V consists of only one pairing evaluation. That is,

$$V^*(X, \boldsymbol{Q}_1, \boldsymbol{Q}_2) = (f_1(X) + \textstyle\sum_k c_{1k} Q_{1k})\,(f_2(X) + \sum_k c_{2k} Q_{2k})$$

for polynomials $f_1(X)$ and $f_2(X)$ and coefficients $c_{\iota k}$. (As in the knowledge-soundness proof of Theorem 1, Y is not an indeterminate.) In this case, under a PDL assumption, the creation of new random group elements — independently of their distribution — does not help the adversary at all.

Really, $V^*(x, \boldsymbol{q}_1, \boldsymbol{q}_2) = 0$ is only possible if $\sum c_{\iota k} q_{\iota k} = -f_\iota(x)$ for at least one ι. W.l.o.g., assume it holds for $\iota = 1$. This means that $[q_{1k}]_1$ (or at least their weighted sum, that is the only thing that matters) are not created as random new group elements but as evaluations of some polynomials in x. Thus, one can assume that the adversary created no random group elements in $\mathbb{G}_1$. Hence,

$$V^*(X, \boldsymbol{Q}_1, \boldsymbol{Q}_2) = f_1(X) \left(f_2(X) + \sum c_{2k} q_{2k}\right)$$

for some polynomials $f_1(X)$ and $f_2(X)$. Next, either $f_1(x) = 0$ or $\sum c_{2k} q_{2k} = -f_2(x)$. In the first case, $f_1(X) \neq 0$ (otherwise also $V^*(X, \boldsymbol{Q}_1, \boldsymbol{Q}_2) = 0$ as a polynomial) but $f_1(x) = 0$ and thus one has broken the $(2n-1, n-1)$-PDL assumption. In the second case, the adversary created no random group elements in $\mathbb{G}_1$ and thus

$$V^*(X, \boldsymbol{Q}_1, \boldsymbol{Q}_2) = f_1(X) f_2(X) \ .$$

Again, since $f_1(x) f_2(x) = 0$ but $f_1(X) f_2(X) \neq 0$, the adversary has broken the $(2n-1, n-1)$-PDL assumption. Hence, creating new group elements does not increase the adversarial power.