Simulation-Extractable SNARKs Revisited

February 8, 2020

Helger Lipmaa[1,2]

[1] Simula UiB, Bergen, Norway
[2] University of Tartu, Tartu, Estonia
firstname.lastname@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, which 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 tag-based strong any-simulation-extractable (tagSASE) subversion-zero knowledge SNARKs for QAP that by design have a relatively simple security proof. The knowledge-sound SNARK is similar to Groth's SNARK, except having fewer trapdoors. To achieve tagSASE, we add to it a one-time simulation-extractable QA-NIZK for a subspace language. We give a simple characterization of languages like SAP, SSP, and QSP in terms of QAP and show how to modify the SNARKs for QAP correspondingly. The only prior published efficient simulation-extractable SNARK was for the impractical SAP language. We prove soundness and tagSASE under hash-algebraic knowledge (HAK) assumptions that are a concrete version of the hash-algebraic group model. The framework of HAK assumptions is another major contribution of this paper. We also show that one can achieve tagless SASE by using an efficient transformation.

Keywords: Algebraic group model, NIZK, non-black-box, QAP, QSP, SNARK, SAP, SSP, simulation-extractability, subversion 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, [DL08,Gro10,Lip12,GGPR13,PHGR13,Lip13,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].

It is challenging 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,Fuc19] for related cryptanalysis). One explanation for this is that for the proof system to be secure, one needs to carefully design the small 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 QAP (quadratic arithmetic program, [GGPR13]). Other related languages are SAP (square arithmetic program, [Gro16,GM17]), QSP (quadratic span program, [GGPR13,Lip13]), and SSP (square span program, [DFGK14]). QSP and SSP (resp., SAP and QAP) are convenient when 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 lowest 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 well-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 needed in many applications, including UC-security [Can01].

Kosba *et al.* [KZM$^+$15] studied how to achieve UC-security for SNARKs. They 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 significant 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 no-

ticed 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). 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]. We construct a tag-based [CHK04,Kil06] SASE (tagSASE) version $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ of it by using well-motivated modifications. Based on an 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] Tag-based tagSASE is sufficient in many applications of SE, like UC-security; see Section 2. Nevertheless, in Appendix B, we use a version of the transformation of [CHK04,Kil06] to achieve tagless SASE. This will increase the computation of both the prover and the verifier by only two exponentiations.

While the current trend is to construct efficient updatable zk-SNARKs [GKM+18], non-updatable zk-SNARKs like the ones of the current paper still have their applications, especially since most efficient zk-SNARKs [MBKM19] rely on the random oracle model and are yet not as efficient as existing non-updatable zk-SNARKs like [Gro16]. We are not aware of any work up to now on updatable SE zk-SNARKs. We hope advances of the current paper will help one to construct the latter.

Since the new SASE SNARKs are tag-based and maliciously chosen tags can depend on the group elements, we use (tautological) knowledge assumptions, instead of the generic group model (GGM, [Nec94,Sho97]) or algebraic group model (AGM, [FKL18]), to prove their security. 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 the bracket notation of [EHK+13]), there exists an efficient extractor $\mathsf{Ext}_{\mathcal{A}}$ that outputs a matrix $\boldsymbol{N}$, s.t. $\boldsymbol{y} = \boldsymbol{N}\boldsymbol{x}$. An *algebraic knowledge (AK) assumption* states that the same

[3] We emphasize that it is only fair to compare SNARKs for the same language (e.g., QAP); to compare SNARKs for different languages (e.g., QAP versus SAP), one 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 (or $\tilde{m}$) and n (or $\tilde{n}$) denote the number of wires and gates (or 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	$\|\mathsf{crs}\|$	P computation	$\|\pi\|$	V computation
QAP-based (arithmetic circuit, with n gates), $\tilde{m}=m$					
[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	tagSASE	$(m+3n)\mathfrak{g}_1+n\mathfrak{g}_2$	$(m+4n)\mathfrak{e}_1+n\mathfrak{e}_2$	$3\mathfrak{g}_1+1\mathfrak{g}_2$	$5\mathfrak{p}+(m_0+1)\mathfrak{e}_1$
SAP-based (arithmetic circuit, with $\tilde{n}$ squaring gates): $u=v$, $\tilde{n}\approx 2n$, $\tilde{m}\approx 2m$					
[GM17]	SASE	$(\tilde{m}+2\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$(\tilde{m}+2\tilde{n})\mathfrak{e}_1+\tilde{n}\mathfrak{e}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$5\mathfrak{p}+m_0\mathfrak{e}_1$
$\mathsf{S}_{\mathsf{sap}}$ § 6	KS	$(\tilde{m}+2\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$(\tilde{m}+2\tilde{n})\mathfrak{e}_1+\tilde{n}\mathfrak{e}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$3\mathfrak{p}+m_0\mathfrak{e}_1$
$\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$ § 6	tagSASE	$(\tilde{m}+3\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$(\tilde{m}+2\tilde{n})\mathfrak{e}_1+\tilde{n}\mathfrak{e}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$5\mathfrak{p}+(m_0+4)\mathfrak{e}_1$
SSP-based (Boolean circuit with n gates): $u=v=w$, $\tilde{n}=m+n$					
[DFGK14]	KS	$(m+\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$2m\mathfrak{m}_1+\tilde{n}\mathfrak{e}_1+m\mathfrak{m}_2$	$3\mathfrak{g}_1+1\mathfrak{g}_2$	$6\mathfrak{p}+m_0\mathfrak{m}_1$
$\mathsf{S}_{\mathsf{ssp}}$ § 7	KS	$(m+2\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$2m\mathfrak{m}_1+\tilde{n}\mathfrak{e}_1+m\mathfrak{m}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$3\mathfrak{p}+m_0\mathfrak{m}_1$
$\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ § 7	tagSASE	$(m+3\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$3m\mathfrak{m}_1+\tilde{n}\mathfrak{e}_1+m\mathfrak{m}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$5\mathfrak{p}+(m_0+4)\mathfrak{m}_1$
QSP-based (Boolean circuit with n gates): $w=0$, $\tilde{n}\approx 14n$ [Lip13]					
[Lip13]	KS	–	–	–	–
$\mathsf{S}_{\mathsf{qsp}}$ § D	KS	$(\tilde{m}+2\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$4\tilde{m}\mathfrak{m}_1+\tilde{n}\mathfrak{e}_1+\tilde{m}\mathfrak{m}_2$	$2\mathfrak{g}_1+1\mathfrak{g}_2$	$3\mathfrak{p}+m_0\mathfrak{m}_1$
$\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ § D	tagSASE	$(\tilde{m}+3\tilde{n})\mathfrak{g}_1+\tilde{n}\mathfrak{g}_2$	$5\tilde{m}\mathfrak{m}_1+\tilde{n}\mathfrak{e}_1+\tilde{m}\mathfrak{m}_2$	$3\mathfrak{g}_1+1\mathfrak{g}_2$	$5\mathfrak{p}+(m_0+1)\mathfrak{m}_1$

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 hashing version HAK of the AK assumption. HAK is essentially a concrete AGM version of the generic group model with hashing (GGMH, [Bro01,BFS16,ABLZ17]) that models the ability of an adversary to create elliptic-curve group elements by using elliptic-curve hashing without knowing their discrete logarithm, [Ica09]. When using a HAK assumption, we require the extractor to output a matrix $\boldsymbol{N}$ and a vector of group elements $[\boldsymbol{q}]_\iota$, such that $\boldsymbol{y}=\boldsymbol{N}\left(\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{q}\end{smallmatrix}\right)$. Here, the adversary does not know the discrete logarithm of $[q_k]_\iota$. In the new SNARKs, to prove that their knowledge-soundness holds under a HAK assumption, we need each $[q_k]_\iota$ to come from a distribution of high min-entropy.

Similarly to GGMH and algebraic group model with hashing (AGMH), a HAK 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 assumptions. On the other hand, using a HAK assumption as compared to the generic model enables one to handle a wider 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]. Arguably, HAK assumptions hit a sweet spot, being weakest (known) non-falsifiable assumptions that allow to prove the security of maximally efficient protocols. Note that some of the very recent SNARKs [CHM+19] have two versions: a more efficient one proven secure in the AGM and a less efficient one proven secure under knowledge-of-exponent assumptions. The use of HAK assumptions eliminates the need for two separate versions.[4] Hence, the introduction of the HAK framework can be seen as one of the major contributions of the current paper.

In Section 4, we propose a knowledge-sound zk-SNARK $\mathsf{S}_{\mathsf{qap}}$ for QAP. Recall 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 used in simulation; thus, it is logical or at least aesthetic to drop the other trapdoors. In particular, in the case of subversion-security [ABLZ17,Fuc18], one needs to extract all trapdoors; the fewer trapdoors there are, the simpler is this step. 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. (A similar technique to use one trapdoor to align "interesting" monomials together was used in the context of updatable SNARKs in [GKM+18]. However, while their SNARK has the desirable updatability property, it is quite inefficient.)

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 tagSASE) proof. It includes X, Y, indeterminates created by the adversary by using elliptic curve hashing [Ica09], and (in the case of tagSASE) indeterminates created by simulator queries. Then, $V(\boldsymbol{X}^*, Y) = \sum V_{Y^i}(\boldsymbol{X}^*)Y^i$ for known polynomials $V_{Y^i}(\boldsymbol{X}^*)$, where i is a linear combination of an initially

[4] Very recently, Kastner and Pan [KP19] proposed an instantiation of AGM based on knowledge-of-exponent assumptions. However, in their instantiation, every group is always at least doubled and, sometimes, more than doubled. Thus, most of the protocols will suffer an efficiency penalty when relying on their instantiations.

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 $V_{Y^i}(\boldsymbol{X}^*) = 0$ 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 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.

In Section 5, we modify $\mathsf{S}_{\mathsf{qap}}$ to make it tagSASE. We establish that for any k, a tagSASE 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 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 tagSASE SNARK is more efficient.

Importantly, $\mathsf{S}_{\mathsf{qap}}$ has a simple knowledge-soundness proof where only the value of the six critical coefficients of V matters. The tagSASE proof $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ relies upon only a few more additional 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 more straightforward 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 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. (This implies knowledge-soundness also in the case of asymmetric pairing.) 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 prior simulation-extractable SNARKs [GM17,BG18] use tags.

After that, we consider languages SAP [Gro16,GM17], SSP [DFGK14], and QSP [GGPR13,Lip13] that have also been used in the 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. Previous research handled all four languages 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. We propose the first known tagSASE SNARKs for QAP, SSP, and QSP. In fact, we propose the first known efficient tagSASE 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 usually has 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 tagSASE 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, $\chi(X) = u(X)(u(X) - 1) - h(X)\ell(X)$. $\mathsf{S}_{\mathsf{ssp}}$ 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 tagSASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ for SSP uses the same transformation as $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$; no previous efficient SE 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 Appendix D, we design a SNARK for QSP (Quadratic Span Programs, [GGPR13,Lip13]). (This part is postponed to the appendix since QSP is the least popular of the four languages.) 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

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	$= 0$

checkers [GGPR13,Lip13]. The new tagSASE 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 formerly known SE 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 evident, 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 $V_{Y^i}(\boldsymbol{X}^*)$ (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.

In Appendix F, we show that $\mathsf{S}_{\mathsf{qap}}$ and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ can be made subversion-zero knowledge (Sub-ZK, [BFS16,ABLZ17,Fuc18]). Recall that a Sub-ZK SNARK remains zero knowledge even if the CRS creator was malicious. According to the template of [ABLZ17], one can deal with it by constructing a public CRS verification algorithm that checks that the CRS corresponds to *some* trapdoor, and then use a knowledge assumption to recover the trapdoor and simulate the argument. As explained in [ALSZ20], Sub-ZK is equivalent to no-auxiliary-string non-black-box zero knowledge in the weak bare public key (BPK, [CGGM00,MR01]) model. Hence, $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is a simulation-extractable (no-auxiliary-string non-black-box) zk-SNARK in the BPK model.

2 Preliminaries

For a matrix $\boldsymbol{A}$, $\boldsymbol{A}_i$ denotes its ith row and $\boldsymbol{A}^{(j)}$ denotes its jth column. A random variable X has min-entropy k, $H_\infty(X) = k$, if $\max_x \Pr[X = x] = 2^{-k}$.

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 $(\ell(\omega^{i-1}))_{i=1}^n = \mathbf{0}_n$.

PPT denotes probabilistic polynomial-time; $\lambda \in \mathbb{N}$ is 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 $\mathcal{A}$, $\mathsf{RND}_\lambda(\mathcal{A})$ denotes the random tape of $\mathcal{A}$ (for given λ), and $r \leftarrow_\$ \mathsf{RND}_\lambda(\mathcal{A})$ denotes the uniformly random choice of the randomizer r from $\mathsf{RND}_\lambda(\mathcal{A})$. By $y \leftarrow \mathcal{A}(\mathsf{inp}; r)$ we denote the fact that $\mathcal{A}$, given an input inp 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 $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{A}}(\lambda) \approx_\lambda 0$, where $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{A}}(\lambda) := \Pr[\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda, n), x \leftarrow_\$ \mathbb{Z}_p^* : \mathcal{A}(\mathsf{p}; [(x^i)_{i=0}^{d_1(n)}]_1, [(x^i)_{i=0}^{d_2(n)}]_2) = x]$. The q-PDL assumption in $\mathbb{G}_1$ (resp., $\mathbb{G}_2$) is equal to the $(q, 0)$-PDL (resp., $(0, q)$-PDL) assumption.

QAP. Quadratic Arithmetic Program (QAP) was introduced in [GGPR13] as a language where for an input inp and witness wit, $(\mathsf{inp}, \mathsf{wit}) \in \mathbf{R}$ 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 $\mathbb{F} = \mathbb{Z}_p$. For the sake of efficiency, we require the existence of the n-th primitive root of unity modulo p, denoted by ω. (However, this is not needed for the new SNARKs to work.) Let U, V, and W be instance-dependent matrices and let a be a witness. A QAP is characterized by the constraint $Ua \circ Va = Wa$. For $j \in [1\,..\,m]$, define $u_j(X) := L_{U^{(j)}}(X)$, $v_j(X) := L_{V^{(j)}}(X)$, and $w_j(X) := L_{W^{(j)}}(X)$ to be interpolating polynomials of the jth column of the corresponding matrix. Thus, $u_j, v_j, w_j \in \mathbb{Z}_p^{(\leq n-1)}[X]$. Let $u(X) = \sum a_j u_j(X)$, $v(X) = \sum a_j v_j(X)$, and $w(X) = \sum a_j w_j(X)$. Then $Ua \circ Va = Wa$ iff

$\ell(X) \mid u(X)v(X) - w(X)$ iff $u(X)v(X) \equiv w(X) \pmod{\ell(X)}$ iff there exists a polynomial $h(X)$ such that $u(X)v(X) - w(X) = h(X)\ell(X)$.

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:

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

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, $(\mathsf{inp}, \mathsf{wit}) \in \mathbf{R} = \mathbf{R}_{\mathsf{Inst}_{\mathsf{qap}}}$ if there exists a (degree $\leq n-2$) polynomial $h(X)$, such that the following key equation holds:

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

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 $j \leq m$.

SNARKs. Let RelGen be a relation generator, such that $\mathsf{RelGen}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\mathsf{inp}, \mathsf{wit})\}$. Here, inp is a statement, and wit is a witness. We assume that λ is explicitly deductible from the description of $\mathbf{R}$. RelGen also outputs auxiliary information $\mathsf{aux}_{\mathbf{R}}$ that will be given to the honest parties and the adversary. Intuitively, $\tilde{\mathsf{z}}_{\mathbf{R}} := (\mathbf{R}, \mathsf{aux}_{\mathbf{R}})$ is the common auxiliary input to an adversary and the corresponding extractor. 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 $\mathcal{L}_{\mathbf{R}} = \{\mathsf{inp} : \exists \mathsf{wit}, (\mathsf{inp}, \mathsf{wit}) \in \mathbf{R}\}$ be an NP-language.

We will define tag-based [MY04] argument systems; in the non-tag-based case, the tag-space is $\mathsf{Tags} = \{\epsilon\}$ (empty string) and tags are ignored by all algorithms. A *non-interactive zero-knowledge (NIZK) argument system* $\Psi = (\mathsf{K}_{\mathsf{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for RelGen consists of four PPT algorithms:

CRS generator: $\mathsf{K}_{\mathsf{crs}}$ is a probabilistic algorithm that, given $\tilde{\mathsf{z}}_{\mathbf{R}} \in \mathrm{range}(\mathsf{RelGen}(1^\lambda))$, outputs $(\mathsf{crs}, \mathsf{td})$ where crs is a CRS and td is a simulation trapdoor. Otherwise, it outputs a special symbol $\perp$. For the sake of efficiency and readability, we divide crs into $\mathsf{crs}_{\mathsf{P}}$ (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 $(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \tau, \mathsf{inp}, \mathsf{wit})$ for $\tau \in \mathsf{Tags}$ and $(\mathsf{inp}, \mathsf{wit}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\perp$.

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

Simulator: Sim is a probabilistic algorithm that, given $(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}, \mathsf{td}, \tau, \mathsf{inp})$, 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 wit), 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 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 $\mathcal{A}$.

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)* according to the definition of [DHLW10]. 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 of at least 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 tag-based SASE SNARKs based on different languages like QAP [GGPR13], SSP [DFGK14], SAP [Gro16], and QSP [GGPR13]. The following definition of tagless SASE SNARKs corresponds to the definition of SE SNARKs in [GM17, Definition 2.10].

Definition 1 (Tagless SASE SNARK [DHLW10,GM17]). *Let* $\Pi = (\mathsf{K_{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a SNARK for the relation* $\mathbf{R}$. *Define* $\mathsf{Adv}^{\overline{\tau}\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) := \Pr[\mathbf{Exp}^{\overline{\tau}\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)]$, *where the experiment* $\mathbf{Exp}^{\overline{\tau}\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is depicted in Fig. 1.* Π *is* tagless non-black-box strong any-simulation-extractable (tagless SASE) *if for any PPT adversary* $\mathcal{A}$ *there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that* $\mathsf{Adv}^{\overline{\tau}\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \approx_\lambda 0$.

Tag-Based Simulation-Extractability. Tag-based primitives [CHK04,Kil06,KW15] are commonly used in settings that are needed for say the UC security [Can01]. They guarantee security if the simulation queries are made on tags, differing from tags output by the adversary. In the case of UC security, a tag corresponds to a triple $(sid, sender, receiver)$, and simulation queries are made in the case the sender is honest while the adversary corresponds to corrupted senders. Thus, simulation queries are guaranteed by default to use different tags compared to adversaries.

Definition 2 (tagSASE SNARK [DHLW10,KW15]). *Let* $\Pi = (\mathsf{K_{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a SNARK for the relation* $\mathbf{R}$. *Define* $\mathsf{Adv}^{\tau\mathrm{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) :=$

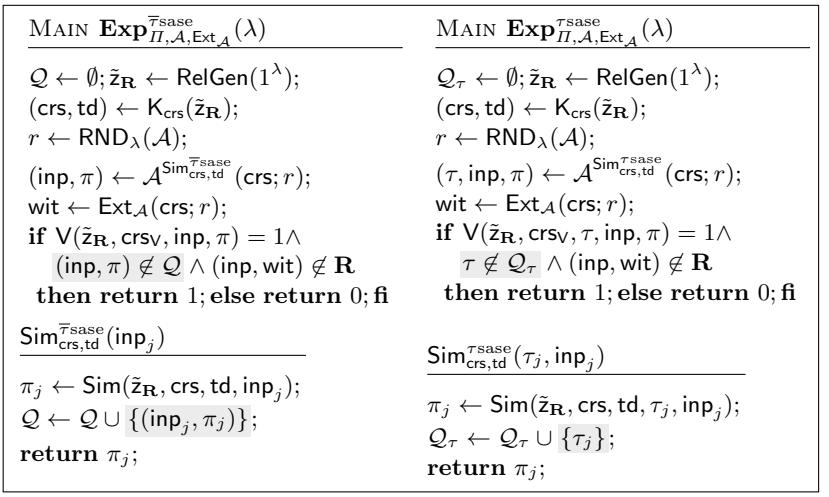

Fig. 1. Simulation-extractability experiments: tagless strong any-simulation (tagSASE, left) and tag-based strong any-simulation (SASE, right). Differences are highlighted

$\Pr[\mathbf{Exp}^{\tau\text{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)]$*, where the experiment* $\mathbf{Exp}^{\tau\text{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is depicted in Fig. 1.* Π *is* non-black-box tag-based strong any-simulation-extractable (tagSASE) *if for any PPT adversary* $\mathcal{A}$ *there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that* $\mathsf{Adv}^{\tau\text{sase}}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \approx_\lambda 0$.

In Appendix B, we use a version of the transformation of [CHK04,Kil06] from tag-based tagSASE SNARKs to tagless SASE SNARKs. This will increase the computation of both P and V by only two exponentiations if one is willing to assume the non-falsifiable one-more discrete logarithm assumption [BNPS03]. A slightly less efficient transformation is secure under the standard falsifiable discrete logarithm assumption; see Appendix B.

3 Hash-Algebraic Knowledge Assumptions

Algebraic Group Model (AGM, [FKL18]) is a new model that one can use to prove the security of cryptographic assumptions, protocols, and primitives. Essentially, in AGM one assumes that each PPT algorithm (including adversaries) is algebraic in the following sense: if the adversary $\mathcal{A}$'s input includes $[\boldsymbol{x}_\iota]_\iota$ and no other elements from the group $\mathbb{G}_\iota$ and $\mathcal{A}$ outputs group elements $[\boldsymbol{y}_\iota]_\iota$, then $\mathcal{A}$ 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 $\mathbb{G}_\iota$ be a cyclic group of prime order p. A PPT algorithm $\mathcal{A}$ is *algebraic [BV98] (in* $\mathbb{G}_\iota$*)* if there exists an efficient extractor $\mathsf{Ext}_{\mathcal{A}}$, such that for any PPT sampleable distribution $\mathcal{D}$, $\mathsf{Adv}^{\text{ak}}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{A}}(\lambda) \approx_\lambda 0$, where $\mathsf{Adv}^{\text{ak}}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{A}}(\lambda) :=$

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

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 [Bro01,BFS16,ABLZ17] in situations where the adversary can create new random group elements by using elliptic curve hashing; in this case, she provably does not know their discrete logarithms [Ica09]. We model this capability by allowing the adversary to create additional group elements $[\boldsymbol{q}]_\iota$ for which she does not know discrete logarithms. We require that $[\boldsymbol{q}_\iota]_\iota$ (but not necessarily $\boldsymbol{q}_\iota$) can be extracted from the adversary, such that $[\boldsymbol{y}]_\iota = \boldsymbol{N} \cdot [\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{q}\end{smallmatrix}]_\iota$. Moreover, the random variable $[\boldsymbol{q}]_\iota$ must have high min-entropy (we will analyze the reason for high min-entropy after Lemma 2). For example, if elliptic-curve hashing [Ica09] is used, one can assume that $[\boldsymbol{q}]_\iota$ is close to uniformly random in a large subset of $\mathbb{G}_\iota$.

A PPT algorithm $\mathcal{A}$ is *hash-algebraic (in $\mathbb{G}_\iota$)* if there exists an efficient extractor $\mathsf{Ext}_\mathcal{A}$, s.t. for any PPT sampleable distribution $\mathcal{D}$, $\mathsf{Adv}^{\mathrm{hak}}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l}[\boldsymbol{x}]_\iota \leftarrow_\$ \mathcal{D}; r \leftarrow_\$ \mathsf{RND}_\lambda(\mathcal{A}); [\boldsymbol{y}]_\iota \leftarrow_\$ \mathcal{A}([\boldsymbol{x}]_\iota; r); (\boldsymbol{N}, [\boldsymbol{q}]_\iota) \leftarrow \mathsf{Ext}_\mathcal{A}([\boldsymbol{x}]_\iota; r) : \\ (\boldsymbol{y} \neq \boldsymbol{N}(\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{q}\end{smallmatrix})) \vee (\exists i : H_\infty([q_i]_\iota) = O(\log \lambda))\end{array}\right] \approx_\lambda 0 \ .$$

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

The AGM (resp., AGMH) is essentially the assumption that the given group is algebraic (resp., hash-algebraic). We make this more precise, by formalizing the requirement that for fixed $\mathcal{D}$, $\mathcal{A}$ is hash-algebraic as the $(\mathcal{D},\mathcal{A})$-*hash-algebraic knowledge (HAK) assumption in* $\mathbb{G}_\iota$ stating that $\mathsf{Adv}^{\mathrm{hak}}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{A}}(\lambda) \approx_\lambda 0$. Analogously, the $(\mathcal{D},\mathcal{A})$-*algebraic knowledge (AK) assumption in* $\mathbb{G}_\iota$ states that $\mathsf{Adv}^{\mathrm{ak}}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{A}}(\lambda) \approx_\lambda 0$. In *AGM for* $\mathbb{G}_\iota$, one assumes that $(\mathcal{D},\mathcal{A})$-AK holds in $\mathbb{G}_\iota$ for all choices of $(\mathcal{D},\mathcal{A})$. In *AGMH for* $\mathbb{G}_\iota$, one assumes that $(\mathcal{D},\mathcal{A})$-HAK holds in $\mathbb{G}_\iota$ for all choices of $(\mathcal{D},\mathcal{A})$. We equate AGM (resp., AGMH) the $\mathbb{G}_\iota$-*AK (resp.,* $\mathbb{G}_\iota$-*HAK) assumption.* On the other hand, when proving the security of a concrete protocol, it suffices to rely on the following tautological $\mathcal{D}$-HAK assumption.

Definition 3 ($\mathcal{D}$-HAK assumption in $\mathbb{G}_\iota$). *For each PPT $\mathcal{A}$ that obtains inputs, distributed according to the distribution $\mathcal{D}$, there exists an extractor that outputs $[\boldsymbol{q}]_\iota$ and $\boldsymbol{N}$ such that $[q_i]_\iota$ has high min-entropy. More precisely, $\mathsf{Adv}^{\mathrm{hak}}_{\mathbb{G}_\iota,\mathcal{D},\mathcal{A}}(\lambda) \approx_\lambda 0$ for each PPT adversary $\mathcal{A}$.*

Let us next demonstrate how HAK assumptions can be used. Consider the q-PCDH assumption used in some earlier SNARKs [GJM03,Gro10]. Let $\mathcal{D}^{pdl}_\iota = \{[1, x, x^2, \ldots, x^q]^\top_\iota : x \leftarrow_\$ \mathbb{Z}^*_p\}$. A q-PCDH adversary $\mathcal{A}$ is asked, given an input from $\mathcal{D}^{pdl}_\iota$, to output $[y]_\iota = [x^{q+1}]_\iota$. (See Appendix C.1 for a proof.)

Lemma 1. *The q-PCDH assumption is secure under the q-PDL and the $\mathcal{D}^{pdl}_\iota$-HAK assumptions (all in $\mathbb{G}_\iota$).*

As another example, consider Damgård's original Knowledge-of-Exponent (KE, [Dam92]) assumption in $\mathbb{G}_\iota$. Here, let $\mathcal{D}^{ke}_\iota = \{[1, x]^\top_\iota : x \leftarrow_\$ \mathbb{Z}^*_p\}$. Damgård's

KE states that given $[1, x]_\iota^\top \sim \mathcal{D}_\iota^{ke}$, if the adversary outputs $[y, z]_\iota$ such that $z = xy$, then there exists an extractor that extracts y. (See Appendix C.2 for a proof.)

Lemma 2. *The KE assumption is secure under the DL and the $\mathcal{D}_\iota^{ke}$-HAK assumptions (all in $\mathbb{G}_\iota$).*

We note that the opposite does not always hold: KE assumption (and its generalizations) cannot be used to extract unless each input group element $[z]_1$ is accompanied with a "knowledge" input $[xz]_1$ for random x.

Let us now analyze the need for high min-entropy. Following [Bro01], Bellare *et al.* [BFS16,ABLZ17] proved subversion-security in the generic bilinear group model with elliptic curve hashing (GBGMH) where the adversary can create random group elements $[\boldsymbol{q}_\iota]_\iota$ that are interpreted as new indeterminates $\boldsymbol{Q}_\iota$ in the security proof. Our approach will be more precise. Motivated by the upcoming security proofs (e.g., see the proof of Theorem 1), we require that each $[q_i]_1$ has min-entropy

$$H_\infty([q_i]_1) := -\log_2 \max_{q^*} \Pr[q_i : q_i = q^*] = \omega(\log \lambda) \ .$$

Really, we will need that for any $\boldsymbol{x}$, a random $\boldsymbol{q}$ is a root of a verification polynomial $V^*(\boldsymbol{X}, \boldsymbol{Q})$ with negligible probability. We assume that group elements created in $\mathbb{G}_1$ and $\mathbb{G}_2$ are independent; see [BFS16,ABLZ17] for a discussion. Thus, V^* has degree one in any indeterminate $Q_{\iota i}$, and it follows from the Schwartz-Zippel lemma [Zip79,Sch80] that $\Pr[V^*(x, \boldsymbol{q}) = 0] \leq \max_{\iota,i} 2^{-H_\infty([q_{\iota i}]_\iota)} = 2^{-\omega(\log \lambda)} = \lambda^{-\omega(1)}$.[5] Intuitively, we will use this as follows: since $V^*(x, \boldsymbol{q}) = 0$ and each $q_{\iota i}$ has high min-entropy, then due to the Schwartz-Zippel lemma, with high probability, $V^*(x, \boldsymbol{Q}) = 0$ as a polynomial. Hence, in particular, $V^*(x, \boldsymbol{0}) = 0$, and thus one only needs to analyze the case when the adversary does not create new group elements. For this proof technique to work, however, it is needed that all $\mathcal{A}$'s outputs are involved in some verification equation.

There exist simpler versions of elliptic curve hashing (so-called encodings, [Ica09]) where the output is assumed to have high-entropy but not close to uniform; importantly, we obtain security also when using such hashing. The assumption of high min-entropy is quite natural: if $q_{\iota k}$ is equal to some $y_{\iota k}$ with a non-negligible probability, then a non-uniform adversary that has $y_{\iota k}$ as the advice can compute the discrete logarithm $q_{\iota k}$ with a non-negligible probability. In such a case, one can consider $q_{\iota k}$ to be an element with a known discrete logarithm. Thus, instead of a usual knowledge assumption that states that $\mathcal{A}$ knows some value, we now have an intuitively weaker assumption that states that either $\mathcal{A}$ knows some value or is reasonably uncertain about it. That is, there exists an extractor $\mathsf{Ext}_\mathcal{A}$ that, given non-black-box access to $\mathcal{A}$ and her inputs and random tape, either returns this value or is sufficiently confident that $\mathcal{A}$ does not know it (and returns $[q_{\iota k}]_\iota$).

Importantly, when proving the security under (H)AK assumptions, the adversary is allowed to make use of the group presentation as long as this does

[5] See [FKL18, Section 1.2] for a less concrete analysis of the AGMH case.

not contradict the concrete knowledge assumption. This is important in the tag-based setting where the adversary can choose her tags (integers).

Finally, in the bilinear-group setting we make both $\mathcal{D}_1$-HAK (in $\mathbb{G}_1$) and $\mathcal{D}_2$-HAK (in $\mathbb{G}_2$) assumptions, with $\mathcal{D}_1$ and $\mathcal{D}_2$ being possibly correlated (e.g., the input could contain both $[x]_1$ and $[x]_2$ for random x). In this case, we say that a PPT algorithm $\mathcal{A}$ is *hash-algebraic (in* p*)* if there exists an efficient extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. for any PPT sampleable distribution $\mathcal{D}$, $\mathsf{Adv}^{\mathrm{hak}}_{\mathsf{p},\mathcal{D},\mathcal{A}}(\lambda) :=$

$$\Pr\begin{bmatrix}\mathsf{inp} = ([\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2) \leftarrow_{\$} \mathcal{D}; r \leftarrow_{\$} \mathsf{RND}_\lambda(\mathcal{A}); ([\boldsymbol{y}_1]_1, [\boldsymbol{y}_2]_2) \leftarrow_{\$} \mathcal{A}(\mathsf{inp}; r);\\ (\boldsymbol{N}_1, \boldsymbol{N}_2, [\boldsymbol{q}_1]_1, [\boldsymbol{q}_2]_2) \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{inp}; r) :\\ (\boldsymbol{y}_1 \neq \boldsymbol{N}_1(\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{q}_1\end{smallmatrix}) \vee \boldsymbol{y}_2 \neq \boldsymbol{N}_2(\begin{smallmatrix}\boldsymbol{x}_2\\ \boldsymbol{q}_2\end{smallmatrix})) \vee (\exists \iota, k : H_\infty([q_{\iota k}]_\iota) = O(\log \lambda))\end{bmatrix} \approx_\lambda 0 \ .$$

A $(\mathcal{D}, \mathcal{A})$-HAK assumption in p states that $\mathsf{Adv}^{\mathrm{hak}}_{\mathsf{p},\mathcal{D},\mathcal{A}}(\lambda) \approx_\lambda 0$. We define the $\mathcal{D}$-AK assumption in p analogously.

4 Knowledge-Sound SNARK for QAP

In this section, we will describe the new knowledge-sound SNARK $\mathsf{S}_{\mathsf{qap}}$ (SNARK for QAP). It follows a template that emphasizes two objectives: (i) simple soundness proof under a HAK assumption, and (ii) efficiency. $\mathsf{S}_{\mathsf{qap}}$ 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.

We will be using bivariate polynomials, where the indeterminate X is related to the definition of QAP, and the indeterminate Y is used to group together correct monomials in the security proof. (Such an approach was also used in say [GKM+18].) 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.

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

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

Since the 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):

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

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

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 PDL and a HAK assumption. Recall that the adversary can create several 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. Then, the verification guarantees that $V(\boldsymbol{x}) = 0$, where

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

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

By the HAK assumption, all coefficients of V are known. Assume first that $V(\boldsymbol{X}) = 0$ is a zero polynomial and let $\boldsymbol{X}^* = (X,\boldsymbol{Q})$. Writing $V(\boldsymbol{X}) = \sum_i V_{Y^i}(\boldsymbol{X}^*)Y^i$, we get that each $V_{Y^i}(\boldsymbol{X}^*) = 0$. To simplify the knowledge-soundness proof, we construct the SNARK so that for some *small* set Crit, $\chi(X) = 0$ follows from $V_{Y^i}(\boldsymbol{X}^*) = 0$ for $Y^i \in \mathsf{Crit}$. Second, if $V(\boldsymbol{X}) \neq 0$ as a polynomial but the verification succeeds, then $V(\boldsymbol{x}) = 0$ and by a modification of the same strategy of Lemmas 1 and 2, one can break the PDL assumption.

To formalize this discussion, in the malicious case $A(\boldsymbol{X})$, $B(\boldsymbol{X})$, and $C_s(\boldsymbol{X})$ can be any polynomials in the span of polynomials represented in the CRS and of monomials consisting of the new indeterminates $Q_{\iota k}$ in groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_1$, respectively. More precisely, any maliciously computed polynomials $crs_1(a, \boldsymbol{X})$ and $crs_2(b, \boldsymbol{X})$, where a and b are symbolic ("string") variables, that represent group elements 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. 3. The latter, in its turn, follows from the elements that either the honest prover or the honest verifier have to be able to compute during the argument. It also takes into account indeterminates $Q_{\iota k}$ that correspond to new group elements $[q_{\iota k}]_\iota$.

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

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 Eq. (7).

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 $C(\boldsymbol{X})$ is divisible by $\ell(X)$. In addition, $C_s(\boldsymbol{X})$ 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 defined $C(X, Y)$ as in Eq. (4). Let $\mathsf{Coeff} = \{Y^i : V_{Y^i}(\boldsymbol{X}^*) \neq 0\}$,

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

and $\overline{\mathsf{Crit}} = \mathsf{Coeff} \setminus \mathsf{Crit}$ 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 powers of Y ($|\mathsf{Crit}| = 6$) and $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$. The reason for this definition of 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 $V_{Y^i}(\boldsymbol{X}^*)$, $Y^i \in \mathsf{Crit}$, of $V(\boldsymbol{X})$ are depicted in Fig. 2. (The last row in Fig. 2 only applies in the case of tagSASE SNARK in Section 5.) As argued in the the proof of Theorem 1, from $V_{Y^i}(\boldsymbol{X}^*) = 0$ for $Y^i \in \mathsf{Crit}$ it follows that $\chi(X) = 0$. We will give a concrete suggestion for the value of $\boldsymbol{\Delta}$ in Eq. (8).

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

Fig. 2. Critical coefficients in $\mathsf{S}_{\mathsf{qap}}$ (left) and addends to the same coefficients in the tagSASE case (right). The coefficient of $Y^\gamma E_k$ is 0 in the case of $\mathsf{S}_{\mathsf{qap}}$.

$\mathsf{K}_{\mathsf{crs}}(\tilde{\mathsf{z}}_{\mathbf{R}})$**:** Sample $x, y, z \leftarrow_{\$} \mathbb{Z}_p^*$ s.t. $x^n \neq 1$, let $\mathsf{td} \leftarrow (x, y, z)$. Let

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

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

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

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

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

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

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

$\mathsf{Sim}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}, \mathsf{td} = y, \tau, \mathsf{inp} = (a_j)_{j=1}^{m_0})$**:**

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

Fig. 3. The new SNARKs $\mathsf{S}_{\mathsf{qap}}$ (*without* highlighted entries) and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ (*with* highlighted entries). Moreover, $\mathsf{S}_{\mathsf{qsp}}$ is exactly like $\mathsf{S}_{\mathsf{qap}}$ and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ is exactly like $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$, except $w_j(X) = 0$. Tags τ are only used in the tagSASE version.

We are now ready to describe the SNARK $\mathsf{S}_{\mathsf{qap}}$, see Fig. 3, and prove its security. 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 $\mathcal{D}$-HAK assumption. More precisely, we need to define $\mathcal{D}$, since $\mathbb{G}_\iota$ is fixed by the protocol. Here and in the rest of the paper, $\mathcal{D}^{\mathsf{crs}}_{\tilde{\mathsf{z}}_{\mathbf{R}}}(\Psi)$ is the distribution of honestly generated CRS of SNARK $\Psi = (\mathsf{K}_{\mathsf{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for the concrete QAP instance $\mathbf{R} = \mathsf{Inst}_{\mathsf{qap}}$ and for $\mathsf{aux}_{\mathbf{R}}$. That is,

$$\mathcal{D}^{\mathsf{crs}}_{\tilde{\mathsf{z}}_{\mathbf{R}}}(\Psi) = \{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\tilde{\mathsf{z}}_{\mathbf{R}})\} \ .$$

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. 3.*
(1) Assume $\boldsymbol{\Delta}$ satisfies $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$. (An example recommendation for $\boldsymbol{\Delta}$ is given in Eq. (8).) Then, $\mathsf{S}_{\mathsf{qap}}$ is knowledge-sound under the $(2n-2, n-1)$-PDL and $\mathcal{D}^{\mathsf{crs}}_{\tilde{\mathsf{z}}_{\mathbf{R}}}(\mathsf{S}_{\mathsf{qap}})$-HAK assumptions.
(2) $\mathsf{S}_{\mathsf{qap}}$ is perfectly zero-knowledge.

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

Proof. **(1: knowledge-soundness)** In all following knowledge-soundness and tagSASE proofs, we will use the following template. We use the $\mathcal{D}^{\mathsf{crs}}_{\tilde{\mathsf{z}}_{\mathbf{R}}}(\mathsf{S}_{\mathsf{qap}})$-HAK assumption to extract all coefficients of $V(\boldsymbol{X})$. Since the verifier accepts, $V(\boldsymbol{x}) = 0$ where $\boldsymbol{x}$ is the vector of actual trapdoors. We will move everything from group elements to indeterminates, and we argue that if the verification polynomial $V(\boldsymbol{X})$ is a zero polynomial ($V(\boldsymbol{X}) = 0$) 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 the distribution of $([\boldsymbol{q}_1]_1, [\boldsymbol{q}_2]_2)$, creating random group elements does not benefit $\mathcal{A}$. These two steps together guarantee that the SNARK is knowledge-sound/tagSASE: 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 $(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}; r)$ as an input outputs (inp, π), such that V accepts. Denote $\mathsf{crs} = ([\boldsymbol{\Gamma}_1]_1, [\boldsymbol{\Gamma}_2]_2)$ (here, $[y^{\gamma+\delta}]_T$ is only included to the CRS as an optimization and has no influence on the security) and $\mathcal{D} = \mathcal{D}^{\mathsf{crs}}_{\tilde{\mathsf{z}}_{\mathbf{R}}}(\mathsf{S}_{\mathsf{qap}})$. By the HAK assumption, there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ that on the same inputs, with probability $1 - \mathsf{Adv}^{\mathrm{hak}}_{\mathsf{p},\mathcal{D},\mathcal{A}}(\lambda) = 1 - \mathsf{negl}(\lambda)$, returns $\boldsymbol{N}_1$, $\boldsymbol{N}_2$, $[\boldsymbol{q}_1]_1$ and $[\boldsymbol{q}_2]_2$, such that $(\mathsf{a}, \mathsf{c}_s)^\top = \boldsymbol{N}_1 \left(\begin{smallmatrix} \boldsymbol{\Gamma}_1 \\ \boldsymbol{q}_1 \end{smallmatrix}\right)$ and $\mathsf{b} = \boldsymbol{N}_2 \left(\begin{smallmatrix} \boldsymbol{\Gamma}_2 \\ \boldsymbol{q}_2 \end{smallmatrix}\right)$. We abort if the extractor fails. As above, write $\mathsf{a} = A(\boldsymbol{x})$, $\mathsf{c} = C(\boldsymbol{x})$, and $\mathsf{b} = B(\boldsymbol{x})$. Taking into account that elements of $\boldsymbol{\Gamma}_\iota$ are known polynomials of X and Y, we can efficiently extract the coefficients of the polynomials $A(\boldsymbol{X})$, $C(\boldsymbol{X})$, and $B(\boldsymbol{X})$ from $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$.

Step 1. Assume first that $V(\boldsymbol{X}) = 0$ and thus $V_{Y^i}(\boldsymbol{X}^*) = 0$ for $Y^i \in \mathsf{Crit}$. Consider Fig. 2. Since $V_{Y^{\gamma+\delta}}(\boldsymbol{X}^*) = b_\delta + a_\gamma(b_\delta + 1) = 0$, 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, $V_{Y^{\eta+\gamma}}(\boldsymbol{X}^*) = (a_\gamma + 1)b_\eta = 0$ and thus $b_\eta = 0$. Next, $V_{Y^{2\delta}}(\boldsymbol{X}^*) = a_\delta = 0$. Thus, $\tilde{a}_j = a_j$ for

$j \le m_0$. 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 "non-hashing" case when the adversary did not create any new group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, i.e., where we do not consider elliptic-curve hashing. After that, we show how to reduce the "hashing" case to the "non-hashing" case.

In the non-hashing case, assume that the HAK assumption holds and that $\mathcal{A}$ is a knowledge-soundness adversary that succeeds with probability $\varepsilon_{snd} := \mathsf{Adv}^{\mathrm{snd}}_{\mathsf{RelGen},\mathcal{A}}(\lambda)$ where $\varepsilon_{snd} > \varepsilon_{hak} := \mathsf{Adv}^{\mathrm{hak}}_{\mathsf{p},\mathcal{D},\mathcal{A}}(\lambda)$. Note that due to the HAK assumption, $\varepsilon_{hak} \approx_\lambda 0$. We construct the following PDL adversary $\mathcal{B}$. The PDL challenger $\mathcal{C}$ samples $x \leftarrow_\$ \mathbb{Z}_p^*$ and gives $\mathsf{inp} = ([1, x, \ldots, x^{2n-2}]_1, [1, x, \ldots, x^{n-1}]_2)$ as an input to $\mathcal{B}$. $\mathcal{B}$ samples $y \leftarrow_\$ \mathbb{Z}_p^*$, and uses it together with $\tilde{\mathsf{z}}_{\mathbf{R}}$ and inp to create correctly distributed crs. $\mathcal{B}$ plays the challenger in the knowledge-soundness game with $\mathcal{A}$: after sending crs and $r \leftarrow_\$ \mathsf{RND}_\lambda(\mathcal{A})$ 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 HAK assumption to succeed with probability $1 - \mathsf{negl}(\lambda)$, to obtain matrices $\boldsymbol{N}_\iota$ and $[\boldsymbol{q}_\iota]_\iota$ (the latter is empty in the non-hashing 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 $y \in \mathbb{Z}_p^*$, let $V^*(X) := V(X,y)$. Finally, $\mathcal{B}$ does the following:

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

Clearly, $\mathcal{B}$ has broken the $(2n-2, n-1)$-PDL assumption with probability $\varepsilon_{pdl} = \mathsf{Adv}^{\mathrm{pdl}}_{2n-2,n-1,\mathsf{Pgen},\mathcal{B}}(\lambda) \ge \varepsilon_{snd} - \varepsilon_{hak}$. Thus,

$$\varepsilon_{snd} \le \mathsf{Adv}^{\mathrm{pdl}}_{2n-2,n-1,\mathsf{Pgen},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathrm{hak}}_{\mathsf{p},\mathcal{D},\mathcal{A}}(\lambda) \ .$$

Moreover, $\mathcal{B}$'s running time is dominated by the running time of $\mathcal{A}$ and the time to perform polynomial factorization.

Consider now the "hashing" case when $\mathcal{A}$ has created at least one random group element $q_{\iota i}$. We will rely on the fact that the new group elements $q_{\iota i}$ are added additively to a polynomial of x in both groups; 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 $Q_{\iota i}$. Thus, by the Schwartz-Zippel lemma and since $H_\infty([q_{\iota i}]_\iota) = \omega(\log \lambda)$, 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 $[q_{\iota i}]_1$, 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 E, we observe that if the verification consists of a single pairing execution, then it is not even needed to assume that $[q_{\iota i}]_\iota$ 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 polynomial sets crs_1 and crs_2 both contain a non-zero monomial corresponding to $Y^0 = 1$ (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 (this potentially speeds up the CRS verification algorithm [ABLZ17], see Appendix F). Since there are too many coefficients that one has to take into account, we used a computer search to find the following values for $\alpha, \beta, \ldots$:

$$\alpha = 0, \;\; \beta = 1, \;\; \gamma = -6, \;\; \delta = 4, \;\; \eta = -1. \tag{8}$$

In this case, $\mathsf{Crit} = \{Y^2, Y^{-5}, Y^5, Y^{-2}, Y^{-7}, Y^8\}$ and

$$\begin{aligned} \mathsf{crs}_\mathsf{P} &= \begin{pmatrix} [\{u_j(x)y^5 + v_j(x)y^{-5} + w_j(x)y^2\}_{j=m_0+1}^{m}, y^0, \{x^j y^1\}_{j=0}^{n-1}]_1, \\ \left[\{x^i \ell(x) y^2\}_{j=0}^{n-2}, y^{-6}, y^4, y^0 z, \{x^j y^1 z\}_{j=0}^{n-1}\right]_1, [y^0, \{x^j y^1\}_{j=0}^{n-1}]_2 \end{pmatrix}, \\ \mathsf{crs}_\mathsf{V} &= \left([\{u_j(x)y^6 + v_j(x)y^{-4} + w_j(x)y^3\}_{j=1}^{m_0}, y^{-6}]_1, \left[y^0, y^4, y^{-1}, z\right]_2, [y^{-2}]_T\right) \;. \end{aligned} \tag{9}$$

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

In this section, we will describe the new tagSASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ (SE SNARK for QAP). Recall that in the case of simulation-extractability, the adversary 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 tagSASE, 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 $V_{Y^i}(\boldsymbol{X}^*)$ of $V(\boldsymbol{X})$ will be changed by extra addends $V^+_{Y^i}(\boldsymbol{X}^*)$, depicted in Fig. 2. For example, $V_{Y^{\beta+\delta}}(\boldsymbol{X}^*) = (b_\delta+1)u_a(X) - u(X) + a_\delta v_b(X) + \sum_k (s_{c2k} - r_b s_{a2k}) \sum_j \sigma_{kj} u_j(X)$.

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 $Y^{\gamma+\delta}$ (namely, $b_\delta + a_\gamma(b_\delta+1)$), $Y^\delta D_k$ (namely, $r_b s_{a2k} - s_{c2k} + (b_\delta+1)s_{a1k}$), $Y^\gamma E_k$ (namely, $r_b s_{a2k} - s_{c2k} + (a_\gamma+1)s_{bk}$), $Y^\alpha D_k$ (namely, $r_b s_{a1k} - s_{c1k}$), and $D_k E_k$ (namely, $r_b s_{a2k} - s_{c2k} + s_{a1k} s_{bk}$) in $V(\boldsymbol{X})$ imply that either

(i) $s_{a1k} = s_{bk} = 0$ and thus $s_{c2k} = r_b s_{a2k} = 0$, for all k, or
(ii) $s_{a1k} = 1/(b_\delta+1)$ and $s_{bk} = b_\delta + 1$ for at least one k.

(We note that we will not use all these coefficients in the actual tagSASE proof.)

In the first case, $s_{c2k} = r_b s_{a2k}$ for all k and thus we can eliminate the $V^+_{Y^i}(\boldsymbol{X}^*)$ addends in Fig. 2, 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 $V^+_{Y^i}(\boldsymbol{X}^*)$ in Fig. 2 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"[6] 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; see [KW15, Section 4]. 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 only possible that $s_{a1k}, s_{bk} \neq 0$ for at most one index $k = k_0$. More precisely, assume that $s_{a1k_0} = s_{bk_0} = 1$. Then, as seen from the coefficient of $Y^{-\alpha} D_{k_1} E_{k_1} E_{k_2}$, $s_{a2k_1} s_{bk_2} = 0$ for any k_1, k_2. Thus, $s_{bk} = 0$ for any $k \neq k_0$.

[6] Since this subspace is trivial (equal to the whole space), we need to rely on a knowledge assumption to achieve security. See [FLSZ17,CFQ19] that used a similar technique to combine QA-NIZK and SNARKs.

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. (However, our construction is more efficient than Π_{otss}.) 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 not 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 $u_{za}(X) \in \mathbb{Z}_p^{(\le n-1)}[X]$. 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 $V_{Y^i}^{+}(\boldsymbol{X}^*)$ in Fig. 2 are equal to 0 and thus tagSASE of $\mathsf{S}_{\mathsf{qap}}^{\mathsf{se}}$ follows from the knowledge-soundness of $\mathsf{S}_{\mathsf{qap}}$.

Due to the introduction of the new indeterminate Z, we will have a bigger set of critical coefficients. Let $\mathsf{Coeff}' = \{i = Y^{j_0}D_{k_1}^{j_1}E_{k_1}^{j_2}E_{k_2}^{j_3}Z^{j_4} : V_i(\boldsymbol{X}^*) \neq 0\}$, $\mathsf{Coeff}'^{se} = \{i = Y^{j_0}D_{k_1}^{j_1}E_{k_1}^{j_2}E_{k_2}^{j_3}Z^{j_4} : V_i^{se}(\boldsymbol{X}^*) \neq 0\}$,

$$\mathsf{Crit}' = \mathsf{Crit} \cup \{Y^{\gamma}D_k\}_k = \{Y^{2\beta}, Y^{\beta+\gamma}, Y^{\beta+\delta}, Y^{\gamma+\delta}, Y^{\gamma+\eta}, Y^{2\delta}\} \cup \{Y^{\gamma}D_k\}_k$$

be the set of critical coefficients of $V(\boldsymbol{X})$ and $\mathsf{Crit}'^{se} = \{E_k, E_kZ\}_k$ be the set of critical coefficients of $V^{se}(\boldsymbol{X})$. Let $\overline{\mathsf{Crit}'} = \mathsf{Coeff}' \setminus \mathsf{Crit}'$ and $\overline{\mathsf{Crit}'^{se}} = \mathsf{Coeff}'^{se} \setminus \mathsf{Crit}'^{se}$. Another difference with Section 4 is that in the current section, $\tilde{a}_j = a_j - \sum_k s_{c2k}\sigma_{kj}$ for $j \le m_0$ and $\tilde{a}_j = c_j^*$ for $j > m_0$.

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 tagSASE SNARK (with highlighted entries) in Fig. 3.*

(1) Assume $\boldsymbol{\Delta}$ *is chosen so that* $\mathsf{Crit}' \cap \overline{\mathsf{Crit}'} = \emptyset$. *(The setting of Eq.* (8) *is*

suitable.) If the $(2n-2, n-1)$*-PDL and* $\mathcal{D}^{\mathsf{crs}}_{\bar{z}_{\mathbf{R}}}(\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}})$*-HAK assumptions hold then* $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ *in Fig. 3 is tagSASE.*
(2) $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ *is perfectly zero-knowledge.*

Proof. **(1: tagSASE)** We use a proof template, very similar to the proof of Theorem 1. Let $\mathcal{A}$ be a tagSASE adversary that outputs (inp, π), s.t. V accepts. Since we are proving tagSASE, another part of the input to $\mathcal{A}$ is the reply of the Sim oracle to each query. Due to the use of a HAK assumption, $\mathbb{G}_\iota$-outputs of $\mathcal{A}$ belong to the span of her inputs (the elements of CRS, Sim replies in $\mathbb{G}_\iota$, and $[\boldsymbol{q}_\iota]_1$ extracted from $\mathcal{A}$) 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 and let $\boldsymbol{X}^* = (X, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{E}, Z)$ be the vector of all indeterminates but Y.

Since the second verification equation holds, from the coefficients of E_k (namely, $\tau_k s_{b_1 3k} - \tau s_{bk}$) and $E_k Z$ (namely, $s_{b_1 3k} - s_{bk}$) of $V^{se}(\boldsymbol{X}) = 0$, we get $M\left(\begin{smallmatrix} s_{b_1 3k} \\ s_{bk} \end{smallmatrix}\right) = \mathbf{0}_2$ where $M := \left(\begin{smallmatrix} \tau_k & -\tau \\ 1 & -1 \end{smallmatrix}\right)$. Since $\tau \neq \tau_k$, M is invertible and thus $s_{b_1 3k} = s_{bk} = 0$. From the coefficient of $Y^\gamma E_k$ (namely, $r_b s_{a2k} + (a_\gamma + 1) s_{bk} - s_{c2k}$) of $V(\boldsymbol{X}) = 0$ we get that $s_{c2k} = r_b s_{a2k}$ for each k. Thus, the first three polynomials $V^+_{Y^i}(\boldsymbol{X}^*)$ in Fig. 2 are all equal to 0. The rest of the proof of tagSASE follows from the knowledge-soundness proof of Theorem 1.

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

We emphasize that in the proof of tagSASE, the reduction to knowledge-soundness lays crucially on the fact that M is invertible. This is not the case when $\tau_k = \tau$, in this case one only obtains $s_{bk} = s_{b_1 3k}$, and one will not be able to show that the polynomials $V^+_{Y^i}(\boldsymbol{X}^*)$ are equal to 0. Thus, $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is not SASE.

6 SAP-Based SNARKs

In the following sections, we will describe SNARKs for three 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 *et al.* [Gro16,GM17] used SAP (Square Arithmetic Programs) instead of QAP. The only algebraic distinction here is that $v(X) = u(X)$ and thus a SAP instance is of form $\mathbf{R} = \mathsf{Inst}_{\mathsf{sap}} = (\mathbb{Z}_p, m_0, \{u_j, w_j\}_{j=0}^m)$. $\mathbf{R}_{\mathsf{Inst}_{\mathsf{sap}}}$ 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 by using 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 $\tilde{m} \leq 2d$ squaring gates and then constructing a SNARK for SAP for the resulting circuit. The primary motivation behind introducing SAP is that one can construct a zk-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

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

We construct the SNARK $\mathsf{S}_{\mathsf{sap}}$ by correspondingly simplifying Fig. 3, see Fig. 4. 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

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

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 $V_{Y^i}(X, \boldsymbol{Q})$, $Y^i \in \mathsf{Crit}$, of $V(\boldsymbol{X})$ are depicted in Fig. 5. (The last row is only relevant in the tagSASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$.) The differences compared to Fig. 2 are solely due to the setting $v(X) = u(X)$.

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* $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$. *If the* $(2n-2, n-1)$*-PDL and* $\mathcal{D}^{\mathsf{crs}}_{\tilde{\mathbf{z}}_{\mathbf{R}}}(\mathsf{S}_{\mathsf{sap}})$*-HAK assumptions hold then* $\mathsf{S}_{\mathsf{sap}}$ *in Fig. 3 is knowledge-sound.*
(2) $\mathsf{S}_{\mathsf{sap}}$ *is perfectly zero-knowledge.*

(See Appendix C.3 for the proof sketch.)

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 $[\mathsf{c}_s]_1$.

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

$$crs_1(a,\boldsymbol{X}) = \ldots + \textstyle\sum_k s_{a1k} D_k + \sum_k s_{a2k} \left(Y^{\delta-\alpha} + Y^{-\alpha} D_k + Y^{\gamma-\alpha}\right) D_k +$$

$\mathsf{K}_{\mathsf{crs}}(\tilde{\mathsf{z}}_{\mathbf{R}})$: Sample $x, y, z \leftarrow_{\$} \mathbb{Z}_p^*$ s.t. $x^n \neq 1$, let $\mathsf{td} \leftarrow (x, y, z)$. Let

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

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

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

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

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

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

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

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

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

Fig. 4. The new SNARKs for SAP and SSP: knowledge-sound $\mathsf{S}_{\mathsf{sap}}$ (*without* highlighted entries) and tagSASE $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ (*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 $w_j(X) = u_j(X)$.

$Y^i D_k^j$	Coeff. $V_{Y^i D_k^j}(\boldsymbol{X}^*)$ (KS and tagSASE)	$V^+_{Y^i D_k^j}(\boldsymbol{X}^*)$ (tagSASE only)
$Y^{2\beta}$	$u_a(X)u_b(X) - w(X) - h(X)\ell(X)$	$\sum_k(\tau s_{c2k} - r_a s_{a2k})\sum_j \sigma_{kj} w_j(X)$
$Y^{\beta+\gamma}$	$(a_\gamma + \tau)u_b(X) - u(X)$	$\sum_k(\tau s_{c2k} - r_a s_{a2k})\sum_j \sigma_{kj} u_j(X)$
$Y^{\beta+\delta}$	$(b_\delta + 1)u_a(X) - u(X) + a_\delta u_b(X)$	$\sum_k(\tau s_{c2k} - r_a s_{a2k})\sum_j \sigma_{kj} u_j(X)$
$Y^{\gamma+\delta}$	$b_\delta(a_\gamma + \tau) + a_\gamma$	
$Y^{2\delta}$	$(b_\delta + 1)\, a_\delta$	
$Y^{\eta+\gamma}$	$(a_\gamma + \tau)b_\eta$	
$Y^{\gamma} D_k$		$r_a s_{a2k} - \tau s_{c2k} + s_{bk}(a_\gamma + \tau)$

Fig. 5. Critical coefficients in $\mathsf{S}_{\mathsf{sap}}$ (left, $\tau = 1$) and addends to the same coefficients in the tagSASE case (right).

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

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

(Compared to $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$, 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 $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ but take advantage of the fact that $\mathsf{a} = \mathsf{b}$ in the honest case. Recall that in $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$, 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. 4. That is, we define 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, the CRS has to additionally include some new elements (highlighted in Fig. 4). This changes the polynomials crs_1 and crs_2 to

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

Since the second verification accepts, 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 $D_k Z$ 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 $\tau s_{c2k_1} = r_b s_{a2k_1}$. This means that $V_i^+(\boldsymbol{X}^*) = 0$ in Fig. 5 and thus, analogously to the case of $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$, tagSASE of $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$ 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* $\mathsf{Crit}' \cap \overline{\mathsf{Crit}'} = \emptyset$. *If the* $(2n-2, n-1)$*-PDL and* $\mathcal{D}^{\mathsf{crs}}_{\tilde{z}_{\mathbf{R}}}(\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}})$*-HAK assumptions hold then* $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$ *in Fig. 4 is tagSASE.*
(2) $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$ *is perfectly zero-knowledge.*

Efficiency. Clearly, in $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$, 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 tagSASE 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. 4 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 $u(X) = v(X) = w(X)$.

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 $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$. If the $(2n-2, n-1)$-PDL and $\mathcal{D}^{\mathsf{crs}}_{\mathsf{z}_{\mathbf{R}}}(\mathsf{S}_{\mathsf{ssp}})$-HAK assumptions hold then $\mathsf{S}_{\mathsf{ssp}}$ in Fig. 4 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 $[\mathsf{b}]_2$.

tagSASE 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 $\mathsf{Crit}' \cap \overline{\mathsf{Crit}'} = \emptyset$. If the $(2n-2, n-1)$-PDL and $\mathcal{D}^{\mathsf{crs}}_{\mathsf{zR}}(\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}})$-HAK assumptions hold then $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ in Fig. 4 is tagSASE.
(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 Discussion

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 black-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. We did not also consider the important question of updatable SNARKs [GKM+18] since in that case there exists even no efficient knowledge-sound (random-oracle-less) solutions.

Historic Remark. The second eprint 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 (AK and SAK) assumptions.

The third eprint version (from July 23, 2019) adds subversion-security and many typo fixes.

The fourth eprint version (from Feb 8, 2020) has a better explanation of the (H)AK assumption framework. We renamed SAK assumptions to HAK assumptions. We differentiated clearly between the tag-based (as used in most of this paper) and tagless SASE. We clarified why tagSASE is sufficient in applications like UC-security. Nevertheless, we described an efficient transformation from tagSASE SNARKs to tagless SASE SNARKs. We simplified the proof of tagSASE (it does not rely on the hardness of the language anymore.) The new

SNARKs are however the same as in the July 23, 2019 version. We also corrected a lot of small typos (and small but non-essential mistakes).

Acknowledgment. We thank Mikhail Volkhov for helpful comments. The author was partially supported by the Estonian Research Council grant (PRG49).

References

ABLZ17. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70700-6_1.

ALSZ20. Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michał Zając. On QA-NIZK in the BPK Model. In Aggelos Kiayias, editor, *PKC 2020*, volume ? of *LNCS*, pages ?–?, Edinburgh, UK, May 4–7, 2020. Springer, Cham. doi:?

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. doi:10.1007/978-3-030-23696-0_7.

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. doi:10.1007/978-3-642-40084-1_6.

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. doi:10.1109/SP.2014.36.

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. doi:10.1007/978-3-642-36594-2_18.

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014. doi:10.1145/2591796.2591859.

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6_26.

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.

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, editors, *SAC 2003*, volume 3006 of *LNCS*, pages 17–25. Springer, Heidelberg, August 2004. doi:10.1007/978-3-540-24654-1_2.

BNPS03. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003. doi:10.1007/s00145-002-0120-1.

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

Bro01. Daniel R. L. Brown. The exact security of ECDSA. Contributions to IEEE P1363a, January 2001. http://grouper.ieee.org/groups/1363/.

BS07. Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 201–216. Springer, Heidelberg, April 2007. doi:10.1007/978-3-540-71677-8_14.

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. doi:10.1007/BFb0054117.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959888.

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019. doi:10.1145/3319535.3339820.

CGGM00. Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *32nd ACM STOC*, pages 235–244. ACM Press, May 2000. doi:10.1145/335305.335334.

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. doi:10.1145/3133956.3134060.

CHK04. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_13.

CHM+19. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. Technical Report 2019/1047, IACR, September 13, 2019. https://eprint.iacr.org/2019/1047, last retrieved version from Sep 19, 2019.

Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_36.

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kil-

ian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_33.

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. doi:10.1007/3-540-36178-2_6.

DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8_28.

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

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. doi:10.1007/978-3-030-17253-4_11.

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. doi:10.1007/978-3-642-17373-8_35.

DL08. Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP Proofs from an Extractability Assumption. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *Computability in Europe, CIE 2008*, volume 5028 of *LNCS*, pages 175–185, Athens, Greece, June 15–20, 2008. Springer, Heidelberg.

EHK+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. doi:10.1007/978-3-642-40084-1_8.

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. doi:10.1007/3-540-44448-3_35.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96881-0_2.

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. doi:10.1007/978-3-319-70697-9_4.

FLZ16. Prastudy Fauzi, Helger Lipmaa, and Michal Zajac. A shuffle argument secure in the generic model. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 841–872. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6_28.

Fuc18. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_11.

Fuc19. Georg Fuchsbauer. WI Is Not Enough: Zero-Knowledge Contingent (Service) Payments Revisited. Technical Report 2019/964, IACR, August 24, 2019. https://eprint.iacr.org/2019/964, last retrieved version from August 25, 2019.

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

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_37.

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.

GKM+18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_24.

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. doi:10.1007/978-3-319-63715-0_20.

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. doi:10.1145/22145.22178.

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

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

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006. doi:10.1007/11935230_29.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_19.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_11.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. doi:10.1145/1993636.1993651.

Ica09. Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316. Springer, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_18.

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. doi:10.1007/978-3-642-17373-8_31.

JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42033-7_1.

Kil06. Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer, Heidelberg, March 2006. doi:10.1007/11681878_30.

KP19. Julia Kastner and Jiaxin Pan. Towards Instantiating the Algebraic Group Model. Technical Report 2019/1018, IACR, September 10, 2019. https://eprint.iacr.org/2019/1018.

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. doi:10.1007/978-3-662-46803-6_4.

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.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_10.

Lip13. Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42033-7_3.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019. doi:10.1145/3319535.3339817.

MR01. Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 542–565. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_32.

MY04. Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 382–400. Springer, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_23.

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

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`.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. `doi:10.1109/SP.2013.47`.

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. `doi:10.1109/SFFCS.1999.814628`.

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

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. `doi:10.1007/3-540-69053-0_18`.

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

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.

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.

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

A 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 4 (Perfect Completeness). *A non-interactive argument Ψ is* perfectly complete for RelGen*, if for all λ, all $\tilde{\mathsf{z}}_{\mathbf{R}} \in \mathrm{range}(\mathsf{RelGen}(1^\lambda))$, tag $\tau \in \mathsf{Tags}$, and $(\mathsf{inp}, \mathsf{wit}) \in \mathbf{R}$,*

$$\Pr\left[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\tilde{\mathsf{z}}_{\mathbf{R}}) : \mathsf{V}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \tau, \mathsf{inp}, \mathsf{P}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \tau, \mathsf{inp}, \mathsf{wit})) = 1\right] = 1 \ .$$

Definition 5 (Computational Knowledge-Soundness). *Ψ is computationally (adaptively)* knowledge-sound for RelGen*, if for every PPT $\mathcal{A}$, there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. for all λ, $\mathsf{Adv}^{\mathrm{snd}}_{\mathsf{RelGen},\mathcal{A}}(\lambda) :=$*

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

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*.

Definition 6 (Statistically Unbounded ZK [Gro06]). *Ψ is* statistically unbounded Sub-ZK for RelGen*, if for all λ, all $\tilde{\mathsf{z}}_{\mathbf{R}} \in \mathrm{range}(\mathsf{RelGen}(1^\lambda))$, all $\tau \in \mathsf{Tags}$, and all computationally unbounded $\mathcal{A}$, $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$, where*

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

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

B CHK Tag-Elimination Transformation

Canetti, Halevi, and Katz showed in [CHK04] (in another context) how to eliminate the use of tags. We will use the CHK transformation in the context of SASE SNARKs. The resulting (tagless) SNARK will be based on a tagSASE SNARK and a one-time signature scheme. To avoid confusion, in this section only, we use the adjective "tag-based" to refer to tagSASE SNARKs that satisfy Definition 2.

Preliminaries. A one-time signature scheme $\Sigma = (\Sigma.\mathsf{KGen}, \Sigma.\mathsf{Sig}, \Sigma.\mathsf{V})$ consists of three polynomial-time algorithms:

Key-generation algorithm $\Sigma.\mathsf{KGen}(1^\lambda)$: randomized algorithm that returns $(\mathsf{vk}, \mathsf{sk})$ (a verification key and a signing key).

Signing algorithm $\Sigma.\mathsf{Sig}(\mathsf{sk}, M)$: an algorithm that allows a user to sign a message M by using signing key sk.

Verification algorithm $\Sigma.\mathsf{V}(\mathsf{vk}, M, \sigma)$: an algorithm that allows a user to verify whether the signature σ on the message M is correct, given verification key vk. It outputs either 0 or 1.

Σ is *perfectly correct* if for all $\lambda \in \mathbb{N}$ and all messages M from the message space,

$$\Pr[(\mathsf{vk}, \mathsf{sk}) \leftarrow_{\$} \Sigma.\mathsf{KGen}(1^\lambda) : \Sigma.\mathsf{V}(\mathsf{vk}, M, \Sigma.\mathsf{Sig}(\mathsf{sk}, M)) = 1] \ .$$

A one-time signature scheme Σ is *strongly unforgeable*, if

$$\Pr\left[(\mathsf{vk},\mathsf{sk}) \leftarrow_{\$} \Sigma.\mathsf{KGen}(1^\lambda); (M^*,\sigma^*) \leftarrow_{\$} \mathcal{A}^{\Sigma.\mathsf{Sig}}(\mathsf{vk}) : \Sigma.\mathsf{V}(\mathsf{sk},M^*,\sigma^*)=1\right] \approx_\lambda 0$$

for all PPT adversaries $\mathcal{A}$. Here, $\mathcal{A}$ is allowed to make only a single query to the oracle $\Sigma.\mathsf{Sig}$, and the target pair (M^*,σ^*) output by $\mathcal{A}$ must be different from the message/signature pair (M,σ) obtained from the oracle query. Bellare and Shoup [BS07] proposed a general construction of a one-time signature scheme based on a canonical identification protocol ID (i.e., Σ protocols where the prover has a public key) and collision-resistant hash functions H. They proved that the resulting scheme is strongly unforgeable if ID is specially sound under concurrent attacks and H is collision-resistant. Based on Schnorr's identification protocol that has the required security property under the non-falsifiable one-more discrete logarithm (OMDL, [BNPS03]) problem, one can obtain a one-time signature scheme where the key length is two group elements and the key of H, the signature consists of one integer from $\mathbb{Z}_p$, the signing is dominated by one multiplication, and the verification is dominated by two exponentiations. In group $\mathbb{G}_1$, this scheme Σ is defined as follows:

$\Sigma.\mathsf{KGen}(1^\lambda)$**:** generate $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$, a new key K for H, $x,y \leftarrow_{\$} \mathbb{Z}_p$. Set $\mathsf{vk} \leftarrow (K,[x,y]_1)$ and $\mathsf{sk} \leftarrow (x,y)$.
$\Sigma.\mathsf{Sig}(\mathsf{sk},M)$**:** $e \leftarrow H_K([y]_1,M)$; return $\sigma \leftarrow ex+y$;
$\Sigma.\mathsf{V}(\mathsf{vk},M,\sigma)$**:** $e \leftarrow H_K([y]_1,M)$; Output 1 iff $\sigma[1]_1 = e[x]_1 + [y]_1$.

Based on Okamoto's identification protocol that has the required security property under the falsifiable discrete logarithm problem, one can obtain a slightly less efficient one-time signature scheme where the key length is three group elements and the key of H, the signature consists of two integers from $\mathbb{Z}_p$, the signing is dominated by two multiplications, and the verification is dominated by three exponentiations.

Transformation to SASE SNARKs. Canetti, Halevi, and Katz [CHK04] and Kiltz [Kil06] proposed a transformation from a tag-based to a tagless encryption scheme. We use essentially the same transformation to transfer a tagSASE SNARK $\Pi = (\mathsf{K}_{\mathsf{crs}},\mathsf{P},\mathsf{V},\mathsf{Sim})$, by using a strongly unforgeable one-time signature scheme Σ, to a tagless SASE SNARK $\Pi' = (\mathsf{K}'_{\mathsf{crs}},\mathsf{P}',\mathsf{V}',\mathsf{Sim}')$; see Fig. 6. This transformation applies to any of $\Pi \in \{\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}},\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}},\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}},\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}\}$. Note that all algorithms in Π use vk as the tag τ.

Theorem 7. *Let Σ be a one-time signature scheme and let Π be a tag-based SNARK. Let Π' the result of the transformation in Fig. 6.*
(1) If Σ is perfectly correct and Π is perfectly complete then Π' is perfectly complete.
(2) If Σ is strongly unforgeable and Π is computationally tagSASE then Π' is tagless computationally SASE.
(3) If Π is perfect composable zero-knowledge then Π' is perfect composable zero-knowledge.

$\mathsf{K}'_{\mathsf{crs}}(\tilde{\mathsf{z}}_{\mathbf{R}})$: return $\mathsf{K}_{\mathsf{crs}}(\tilde{\mathsf{z}}_{\mathbf{R}})$;

$\mathsf{P}'(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{inp}, \mathsf{wit})$: $(\mathsf{vk}, \mathsf{sk}) \leftarrow_{\$} \Sigma.\mathsf{KGen}(1^\lambda)$; $\pi \leftarrow \mathsf{P}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{vk}, \mathsf{inp}, \mathsf{wit})$; $\sigma \leftarrow \Sigma.\mathsf{Sig}(\mathsf{sk}, \pi)$; return $\pi' \leftarrow (\pi, \mathsf{vk}, \sigma)$;

$\mathsf{V}'(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{inp}, \pi' = (\pi, \mathsf{vk}, \sigma))$: if $\Sigma.\mathsf{V}(\mathsf{vk}, \pi, \sigma) = 0$ then return 0; else return $\mathsf{V}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{vk}, \mathsf{inp}, \pi)$;

$\mathsf{Sim}'(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{td}, \mathsf{inp})$: $(\mathsf{vk}, \mathsf{sk}) \leftarrow_{\$} \Sigma.\mathsf{KGen}(1^\lambda)$; $\pi \leftarrow \mathsf{Sim}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{td}, \mathsf{vk}, \mathsf{inp})$; $\sigma \leftarrow \Sigma.\mathsf{Sig}(\mathsf{sk}, \pi)$; return $\pi' \leftarrow (\pi, \mathsf{vk}, \sigma)$;

Fig. 6. The CHK transformation on SASE SNARKs.

Proof. In this case, (1) and (3) are obvious.

(2: SASE): Assume that $\mathcal{A}'$ is a SASE adversary for Π'. We construct the following tagSASE adversary $\mathcal{A}$ for Π. In the game $\mathbf{Exp}^{\tau\text{sase}}$, after being called by the challenger with the CRS crs, $\mathcal{A}(\mathsf{crs})$ calls $\mathcal{A}'(\mathsf{crs})$. $\mathcal{A}$ plays challenger to and answers the simulations queries of $\mathcal{A}'$ in the game $\mathbf{Exp}^{\overline{\tau}\text{sase}}$. When $\mathcal{A}'$ queries the simulator oracle $\mathsf{Sim}^{\overline{\tau}\text{sase}}_{\mathsf{crs},\mathsf{td}}(\mathsf{inp}_j)$, $\mathcal{A}$ does the following:

1. sample $(\mathsf{vk}_j, \mathsf{sk}_j) \leftarrow_{\$} \Sigma.\mathsf{KGen}(1^\lambda)$;
2. call her own tag-based simulation oracle $\pi_j \leftarrow \mathsf{Sim}^{\tau\text{sase}}_{\mathsf{crs},\mathsf{td}}(\mathsf{vk}_j, \mathsf{inp}_j)$;
3. set $\sigma_j \leftarrow \Sigma.\mathsf{Sig}(\mathsf{sk}_j, \pi_j)$; set $\pi'_j \leftarrow (\pi_j, \mathsf{vk}_j, \sigma_j)$; add (inp_j, π'_j) to $\mathcal{Q}$;
4. return π'_j as the simulation answer to $\mathcal{A}'$.

Additionally, vk_j is added to $\mathcal{Q}_\tau$. Clearly, this is a perfect simulation. $\mathcal{A}$ makes no additional simulation queries. When the tagless adversary $\mathcal{A}'$ finally returns $(\mathsf{inp}, \pi' = (\pi, \mathsf{vk}, \sigma))$, $\mathcal{A}$ returns $(\mathsf{vk}, \mathsf{inp}, \pi)$. The extractor $\mathsf{Ext}_{\mathcal{A}'}(\mathsf{crs}; r)$ returns $\mathsf{wit} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{crs}; r)$ and thus wit is the same in both $\mathsf{Sim}^{\overline{\tau}\text{sase}}$ and $\mathsf{Sim}^{\tau\text{sase}}$.

Let us now analyze the case $\mathcal{A}'$ wins the tagless game, but $\mathcal{A}$ does not win the tag-based game. If $\mathcal{A}'$ wins, then $\mathsf{V}'(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{inp}, \pi') = 1$, $(\mathsf{inp}, \mathsf{wit}) \notin \mathbf{R}$, and no $\mathsf{Sim}^{\overline{\tau}\text{sase}}_{\mathsf{crs},\mathsf{td}}(\mathsf{inp})$ query returned $\pi' = (\pi, \mathsf{vk}, \sigma)$. According to Fig. 6 and the execution of $\mathsf{Sim}^{\overline{\tau}\text{sase}}_{\mathsf{crs},\mathsf{td}}$ by $\mathcal{A}$, hence $\Sigma.\mathsf{V}(\mathsf{vk}, \pi, \sigma) = 1$, $\mathsf{V}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{vk}, \mathsf{inp}, \pi) = 1$, $(\mathsf{inp}, \mathsf{wit}) \notin \mathbf{R}$, and no $\mathsf{Sim}^{\tau\text{sase}}_{\mathsf{crs},\mathsf{td}}(\mathsf{vk}, \mathsf{inp})$ query returned π.

Since $\mathcal{A}$ did not win in the tag-based game, $\mathcal{A}$ made the (say, jth) $\mathsf{Sim}^{\tau\text{sase}}_{\mathsf{crs},\mathsf{td}}(\mathsf{vk}, \mathsf{inp})$ query, obtaining $\pi_j \leftarrow \mathsf{Sim}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{td}, \mathsf{vk}_j, \mathsf{inp}_j)$ as the answer. Thus, $\mathsf{vk} = \mathsf{vk}_j$ (for a newly generated $(\mathsf{vk}_j, \mathsf{sk}_j)$) and $\mathsf{inp} = \mathsf{inp}_j$. Hence, $\mathcal{A}'$ made the simulation query $\mathsf{Sim}^{\overline{\tau}\text{sase}}_{\mathsf{crs},\mathsf{td}}(\mathsf{inp}_j)$ that returned $\pi'_j = (\pi_j, \mathsf{vk}_j = \mathsf{vk}, \sigma_j)$, such that $(\mathsf{vk}_j, \mathsf{sk}_j) \leftarrow_{\$} \Sigma.\mathsf{KGen}(1^\lambda)$, $\sigma_j \leftarrow \Sigma.\mathsf{Sig}(\mathsf{sk}_j, \pi_j)$, and thus also $\Sigma.\mathsf{V}(\mathsf{vk}_j, \pi_j, \sigma_j) = 1$. Since no query returned $\pi' = (\pi, \mathsf{vk}, \sigma)$, we have $\pi' \neq \pi'_j$ and thus $(\pi, \sigma) \neq (\pi_j, \sigma_j)$. In particular, since vk was created during the jth query, $\mathcal{A}'$ does not know the corresponding sk and cannot herself sign messages with sk. Since $(\pi, \sigma) \neq (\pi_j, \sigma_j)$, then $\Sigma.\mathsf{Vf}(\mathsf{vk}, \pi, \sigma) = 0$; otherwise, $\mathcal{A}'$ would have forged a new valid signature σ on π. Contradiction with the assumption that $\mathcal{A}'$ won the tagless game. □

C Postponed Proofs

C.1 Proof of Lemma 1

Proof. Here, the $\mathcal{D}_\iota^{pdl}$-HAK assumption in $\mathbb{G}_\iota$ states that one can efficiently extract $[\boldsymbol{q}]_\iota$ (that has a high min-entropy) 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

$$V(X, \boldsymbol{Q}) = X^{q+1} - \textstyle\sum_{i=0}^{q} N_i X^i - \sum N'_i Q_i = 0$$

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 $V(X, \boldsymbol{Q})$. If $\mathcal{A}$ created no new group elements then $V(X)$ is a univariate polynomial and the adversary has broken the q-PDL assumption in $\mathbb{G}_\iota$. Otherwise, since $[q_i]_\iota$ has min-entropy $\omega(\log \lambda)$ from the viewpoint of $\mathcal{A}$, the probability that $V(x, \boldsymbol{q}) = 0$ is negligible. □

C.2 Proof of Lemma 2

Proof. By the HAK assumption, there exist an extractor that can extract $\boldsymbol{N}$ and $[\boldsymbol{q}]_1$ (that has a high min-entropy), such that $\left(\begin{smallmatrix} y \\ z \end{smallmatrix}\right) = \boldsymbol{N}\left(\begin{smallmatrix} 1 \\ x \\ \boldsymbol{q} \end{smallmatrix}\right)$, thus $y = Y(x, \boldsymbol{q})$ for $Y(X, \boldsymbol{Q}) = N_{11} + N_{12}X + \sum N_{1,k+2}Q_k$ and $z = Z(x, \boldsymbol{q})$ for $Z(X, \boldsymbol{Q}) = N_{21} + N_{22}X + \sum N_{2,k+2}Q_k$. Moreover, we know $z = xy$ and thus for

$$\begin{aligned} V(X, \boldsymbol{Q}) :=& Z(X, \boldsymbol{Q}) - XY(X, \boldsymbol{Q}) \\ =& N_{21} + N_{22}X + \textstyle\sum N_{2,k+2}Q_k - X(N_{11} + N_{12}X + \sum N_{1,k+2}Q_k) \ , \end{aligned}$$

it holds $V(x, \boldsymbol{q}) = 0$. If $V(X, \boldsymbol{Q}) = 0$ as a polynomial then $N_{21} = N_{12} = N_{2,k+2} = N_{1,k+2} = 0$ and $N_{22} = N_{11}$. Thus $Y(X, \boldsymbol{Q}) = N_{11}$ and $Z(X, \boldsymbol{Q}) = N_{22}X = XN_{11}$. Thus, we have extracted $y = N_{11}$ that satisfies $z = yx$. Now, consider the case $V(X, \boldsymbol{Q}) \neq 0$ but $V(x, \boldsymbol{q}) = 0$. If $\mathcal{A}$ created no new group elements then $V(X)$ is univariate and the adversary has broken the DL assumption (i.e., the 1-PDL assumption) in $\mathbb{G}_\iota$. Otherwise, since $[q_{\iota k}]_\iota$ has min-entropy $\omega(\log \lambda)$ from the viewpoint of $\mathcal{A}$, the probability that $V(x, \boldsymbol{q}) = 0$ is negligible. □

C.3 Proof of Theorem 3

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. 5. 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 $\mathsf{a} = \mathsf{b}$. □

D 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 tagSASE 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. 3 (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. $\mathbf{R}_{\mathsf{Inst}_{\mathsf{qsp}}}$ is defined as $\mathbf{R}_{\mathsf{Inst}_{\mathsf{qap}}}$ in Eq. (1) except that $w(X) = 0$.

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 $\mathsf{Crit} \cap \overline{\mathsf{Crit}} = \emptyset$. If the $(2n-2, n-1)$-PDL and $\mathcal{D}_{\mathsf{z}_\mathbf{R}}^{\mathsf{crs}}(\mathsf{S}_{\mathsf{qsp}})$-HAK assumptions hold then $\mathsf{S}_{\mathsf{qsp}}$ in Fig. 3 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 $\mathbb{G}_2$.

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

Theorem 9. *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 $\mathsf{Crit}' \cap \overline{\mathsf{Crit}'} = \emptyset$. If the $(2n-2, n-1)$-PDL and $\mathcal{D}_{\mathsf{z}_\mathbf{R}}^{\mathsf{crs}}(\mathsf{S}_{\mathsf{qsp}}^{\mathsf{se}})$-HAK assumptions hold then $\mathsf{S}_{\mathsf{qsp}}^{\mathsf{se}}$ in Fig. 3 is tagSASE.
(2) $\mathsf{S}_{\mathsf{qsp}}^{\mathsf{se}}$ 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 $\mathbb{G}_1$.

E 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, which 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 $(2n-2, n-1)$-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 $(2n-2, n-1)$-PDL assumption. Hence, creating new group elements does not increase the adversarial power.

F Subversion-Zero Knowledge

In a subversion zero-knowledge (Sub-ZK) SNARK [BFS16,ABLZ17,Fuc18], one wants to obtain zero-knowledge even if the CRS creator cannot be trusted. As proven in [ALSZ20], Sub-ZK is equivalent to non-auxiliary-string non-black-box zero knowledge in the Bare Public Key (BPK, [CGGM00,MR01]) model.

Let us show that under the setting in Eq. (8) with CRS as in Eq. (9), the correctness (i.e., that it corresponds to *some* choice of trapdoors) of the CRS of $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ and thus also of $\mathsf{S}_{\mathsf{qap}}$ can be verified by using a public CV algorithm. Note that here we assume $u_j(X) = \sum u_{ji}X^i$, $v_j(X) = \sum v_{ji}X^i$, and $w_j(X) =$

$\mathsf{CV}(\tilde{\mathsf{z}}_{\mathbf{R}}, \mathsf{crs}, \mathsf{crs}_{\mathsf{CV}})$:

Check that the following holds:
// Trapdoors are not 0 and $x^n \neq 1$:
$[xy]_1 \neq [0]_1; [\ell(x)y^2]_1 \neq [0]_1; [t]_1 \neq [0]_1;$ $[z]_1 \neq [0]_1;$
// The bracketed element in $\mathbb{G}_1$ and $\mathbb{G}_2$ are consistent:
$[y^4]_1 \bullet [1]_2 = [1]_1 \bullet [y^4]_2;$ $[z]_1 \bullet [1]_2 = [1]_1 \bullet [z]_2;$
for $j = 0$ **to** $n-1$ **do** $[x^j y]_1 \bullet [1]_2 = [1]_1 \bullet [x^j y]_2$; **endfor**
// Degrees of y^i are consistent:
$[y]_1 \bullet [y^{-1}]_2 = [1]_1 \bullet [1]_2; [y^2 t]_1 \bullet [y^{-1}]_2 = [t]_1 \bullet [y]_2;$
$[y^4 t]_1 \bullet [y^{-1}]_2 = [y^2 t]_1 \bullet [y]_2; [t]_1 \bullet [y^4]_2 = [y^4 t]_1 \bullet [1]_2;$
$[y^{-6}]_1 \bullet [y^4]_2 = [y]_1 \bullet [y]_2;$
// Degrees of $x^j y$ are consistent:
for $j = 1$ **to** $n-2$ **do** $[x^{j+1} y]_1 \bullet [y]_2 = [x^j y]_1 \bullet [xy]_2$; **endfor**
// $x^j yz$ are consistent:
for $j = 1$ **to** $n-1$ **do** $[x^j yz]_1 \bullet [1]_2 = [x^j y]_1 \bullet [z]_1$; **endfor**
// $x^j \ell(x) y$ are consistent:
$[\ell(x)y^2]_1 \bullet [1]_2 = [x^{n-1} y]_1 \bullet [xy]_2 - [y]_1 \bullet [y]_2;$
for $j = 0$ **to** $n-3$ **do** $[x^{j+1} \ell(x) y^2]_1 \bullet [y]_2 = [x^j \ell(x) y^2]_1 \bullet [xy]_2$; **endfor**
// Polynomials of type $u_j(x)y^6 + v_j(x)y^{-4} + w_j(x)y^3$ are consistent:
for $j = 1$ **to** m_0 **do**
$[u_j(x)y^6 + v_j(x)y^{-4} + w_j(x)y^3]_1 \bullet [y^{-1}]_2 =$
$\sum_{i=0}^{n-1} u_{ji}[x^i y]_1 \bullet [y^4]_2 + [y^{-6}]_1 \bullet \sum_{i=0}^{n-1} v_{ji}[x^i y]_2 + \sum_{i=0}^{n-1} w_{ji}[x^i y]_1 \bullet [y]_2;$
endfor
for $j = m_0 + 1$ **to** m **do**
$[u_j(x)y^5 + v_j(x)y^{-5} + w_j(x)y^2]_1 \bullet [1]_2 =$
$\sum_{i=0}^{n-1} u_{ji}[x^i y]_1 \bullet [y^4]_2 + [y^{-6}]_1 \bullet \sum_{i=0}^{n-1} v_{ji}[x^i y]_2 + \sum_{i=0}^{n-1} w_{ji}[x^i y]_1 \bullet [y]_2;$
endfor

Fig. 7. The CRS verification algorithm CV in $\mathsf{S}_{\mathsf{qap}}$ and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ (*with* highlighted entries)

$\sum w_{ji} X^i$. The only problem one encounters is the check that $y^\delta = y^4$ is correctly computed. We deal with it by introducing a random trapdoor t, adding $\mathsf{crs}_{\mathsf{CV}} := [t, y^2 t, y^4 t]_1$ to the CRS, and then additionally verifying the correctness of $\mathsf{crs}_{\mathsf{CV}}$. Clearly, inclusion of such elements does not change the critical coefficients from Crit. Alternatively, one can define new trapdoors s and t instead of $y^\gamma = y^{-6}$ and $y^\delta = y^4$, but then (in the case of $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$) one would have the setting of 5 independent trapdoors (x, y, z, s, t) again.

The existence of public CV can be used to make $\mathsf{S}_{\mathsf{qap}}$ and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ subversion-resistant, by using the techniques of [ABLZ17]. In particular, if CV accepts crs, one can use the BDH-KE assumption [ABLZ17] to extract y and then use extracted y to simulate the argument. Alternatively, one can use the potentially weaker assumption that if an adversary on input $\tilde{\mathsf{z}}_{\mathbf{R}}$ succeeds in creating $([y^\alpha, y^\gamma, y^\delta]_1, [y^\alpha, y^\delta, y^\eta]_2)$, such that at least two of $\{\alpha, \gamma, \delta\}$ (or $\{\alpha, \delta, \eta\}$) elements are distinct (and non-zero), then he knows y. The assumption on $\alpha, \ldots, \eta$ holds for the setting discussed in this paper, see Eq. (8). Let us call this the

$(\{\alpha,\gamma,\delta\},\{\alpha,\delta,\eta\})$-PKE assumption. Both assumptions hold under a suitable HAK assumption.

Lemma 3. *Assume $\alpha,\gamma,\delta,\eta = \mathsf{poly}(\lambda)$. Let ϵ be the empty string. If the ϵ-HAK assumption holds then the $(\{\alpha,\gamma,\delta\},\{\alpha,\delta,\eta\})$-PKE assumption holds.*

Proof (Sketch). Assume $\mathcal{A}$ is an adversary that, on input $\tilde{\mathbf{z}}_{\mathbf{R}}$, outputs $([y^\alpha, y^\gamma, y^\delta]_1, [y^\alpha, y^\delta, y^\eta]_2)$. Under the ϵ-HAK assumption, there exists an extractor that outputs $\boldsymbol{N}_\iota$ and $\boldsymbol{q}_\iota$ such that

$$\begin{bmatrix} y^\alpha \\ y^\gamma \\ Y^\delta \end{bmatrix}_1 = \boldsymbol{N}_1 \begin{bmatrix} 1 \\ \boldsymbol{q}_1 \end{bmatrix}_1 \text{ and } \begin{bmatrix} y^\alpha \\ y^\delta \\ y^\eta \end{bmatrix}_1 = \boldsymbol{N}_2 \begin{bmatrix} 1 \\ \boldsymbol{q}_2 \end{bmatrix}_1 \ .$$

Assume, w.l.o.g., that $\alpha \neq \gamma$ with $\alpha, \beta \neq 0$. Thus, $y^\alpha = N_{111} + \sum N_{11,k+1} q_{1k}$ and $y^\gamma = N_{121} + \sum N_{12,k+1} q_{1k}$. Let $V(\boldsymbol{Q}_1) = (N_{111} + \sum N_{11,k+1} Q_{1k})^\gamma - (N_{121} + \sum N_{12,k+1} q_{1k})^\alpha$. By the HAK assumption, $V(\boldsymbol{q}_1) = 0$. If $V = 0$ as a polynomial then, since $\gamma \neq \alpha$, $N_{11,k+1} = N_{12,k+1} = 0$, and thus $y^\alpha = N_{111}$, from which one can extract y. If $V \neq 0$ as a polynomial then, since $[q_{1k}]_1$ has high min-entropy $O(\log \lambda)$, then by the Schwartz-Zippel lemma, the probability that $V(\boldsymbol{q}_1) = 0$ is negligible. □

Note that CV can be sped up significantly by using batching techniques as explained in [ABLZ17].

CV for other new SNARKs is very similar: it is essentially the same for knowledge-sound versions while the added CRS elements in the tagSASE versions are trivially verifiable due to $[1]_1$ and $[z]_2$ being a part of the CRS.