Simple Yet Efficient Knowledge-Sound and Non-Black-Box Any-Simulation-Extractable ZK-SNARKs

Helger Lipmaa

University of Tartu, Tartu, Estonia helger.lipmaa@gmail.com

Abstract. Motivated by applications like verifiable computation and privacy-preserving cryptocurrencies, many efficient pairing-based SNARKs were recently proposed. However, the most efficient SNARKs like the one by Groth (EUROCRYPT 2016) have a very brittle and difficult-to-verify knowledge-soundness proof in the generic model. Due to that, it is difficult to modify such SNARKs to, e.g., satisfy simulation-extractability or to implement some other language instead of QAP (Quadratic Arithmetic Program).

We propose a template for constructing knowledge-sound and non-black-box any-simulation-extractable (NBBASE) SNARKs for QAP. This template is designed so that the knowledge-soundness and even NBBASE proofs of the new SNARKs are quite simple. The new knowledge-sound SNARK for QAP is very similar to the mentioned SNARK of Groth, except it has fewer trapdoors. To achieve NBBASE, we add to the knowledge-sound SNARK a few well-motivated extra steps, while its security proof is even simpler due to the use of a second verification equation. Moreover, we give a simple characterization of languages like SAP, SSP, and QSP in the terms of QAP and show how to modify the SNARK for QAP correspondingly. The only prior published efficient simulation-extractable SNARK was for SAP.

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

1 Introduction

Zero-knowledge proof systems [GMR85] are fundamental for the theory and applications of cryptography. In particular, a zero-knowledge proof system guarantees that participants of a protocol follow the protocol correctly. For zero-knowledge proof systems to actually be used in practice, one however has to use an "efficient" zero-knowledge proof system that satisfies "reasonable" security definitions under "reasonable" cryptographic and trust assumptions. Due to their performance and versatility, zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs, [Gro10,BCCT12,Lip12,GGPR13,PHGR13,BCCT13,BCTV14,Gro16,GM17]) 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]. All mentioned zk-SNARKs are knowledge-sound in the CRS model [BFM88].

Nonetheless, it is quite difficult to obtain expertise to design new zk-SNARKs, and it is easy for even well-established research groups to err in such an endeavor [CGGN17,Gab19]. One explanation for this is that for the proof system to be secure, the constant number of proof elements and verification equations need to be carefully designed to simultaneously satisfy a number of properties:

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

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

Third, throughout this process, one needs to assure that the SNARK remains 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, most efficient zk-SNARKs are proven to be knowledge-sound in the generic model.

Fourth, most of the existing pairing-based zk-SNARKs are defined in the CRS model assuming the existence of a trusted third party that samples a CRS from the correct distribution and does not leak any trapdoors. The existence of such a trusted third party is often a too strong assumption. Hence, the size and structure of the CRS and of the trapdoor is thus another important concern.

Fifth, sometimes, knowledge-soundness is not enough and one desires to achieve simulation-extractability (SE, [GM17]). Intuitively, SE SNARKs guarantee that knowledge-soundness holds even if the prover has seen many simulated proofs, a property that is needed in many applications [GM17] including UC-security.

Related to the latter, it has been studied how to achieve UC-security for SNARKs. Kosba *et al.* [KZM+15] constructed a black-box simulation-extractable (BBSE) version of SNARKs; BBSE is sufficient to obtain UC-security, [Gro06]. However, their transformation results in quite a large overhead and, in particular, results in a linear-size commitment. Alternatively, Groth and Maller [GM17] proposed a non-black-box strong any-simulation-extractable (NBBSASE) SNARK for SAP 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 approximately two times larger circuits than the ones underlying the QAP languages. They also proved that their construction achieved the lower bound for the argument length for NBBSASE SNARKs. While NBBSASE is not sufficient to obtain UC-security, it is clearly a stronger security notion than knowledge-soundness.

No other simulation-extractable SNARKs are known at this moment (except [BG18] that works in the random-oracle model), not even ones that are just NBBASE (non-black-box any-simulation-extractable, allows an adversary after seeing simulation queries to modify a valid argument to a different valid argument for the same statement) or NBBTSE (non-black-box any-simulation-extractable, allows an adversary after seeing simulation queries to *true statement* to modify a valid argument to a different valid argument for the same statement). While NBBASE and NBBTSE are weaker properties than NBBSASE, they are sufficient in many applications. Moreover, in some applications, one desires to have rerandomization properties of NBBASE or NBBTSE SNARKs. (See [DHLW10] for discussion.)

This brings us to the main question of this paper:

> *Is it possible to construct SNARKs for a multitude of languages (like QAP, SAP, QSP, and SSP) that would simultaneously (i) satisfy a strong soundness definition (like some sort of SE), (ii) have a simple soundness proof in the generic model, and (iii) be as or almost as efficient as the most efficient known knowledge-sound SNARKs.*

Our Contributions. We answer positively to the main question. We will propose a template for knowledge-sound and non-black-box any-simulation-extractable (NBBASE) zk-SNARKs for QAP. The knowledge-sound version $\mathsf{S}_{\mathsf{qap}}$ of the new SNARK for QAP is similar to Groth's SNARK [Gro16] and the NBBASE version is obtained it by well-motivated (minimal) modifications. We provide a different (and very simple) knowledge-soundness proof for this version. NBBASE is weaker than NBBSASE by allowing one to maul an acceptable argument for a statement to a different acceptable argument for the *same* statement. Both NBBSASE and NBBASE SNARKs allow the adversary to ask simulations of false statements. After that, we modify both the knowledge-sound and the NBBASE version of $\mathsf{S}_{\mathsf{qap}}$ to cover SAP, QSP, and SSP languages. This is based on a simple observation about algebraic relations (summarized in Table 2) between these languages and QAP. See Table 1 for efficiency comparison with the previous work. We emphasize that it is only fair to compare SNARKs for the same language; to compare SNARKs for different languages, one also has to take into account the complexity of the reduction from circuits to these languages. (Note that [Lip13] only described a reduction from Boolean circuits to QSP and a linear PCP [BCI+13] for QSP, leaving the cryptographic details of constructing a SNARK.)

Table 1. Efficiency comparison: of QAP/SAP/SSP/QSP-based SNARKs. m, n (or n^*), and m^* denote the number of wires, gates, and constraints in the solutions. In the case of $|\mathsf{crs}|$ and P's computation we have omitted constant (or m_0-dependent) addends like $+(m_0+3)|\mathbb{G}_1|$. "eι" ("mι") denotes exponentiation (multiplication) in group $\mathbb{G}_\iota$, "p" denotes pairing, and $\mathbb{G}_\iota$ denotes the representation length of a $\mathbb{G}_\iota$ element in bits.

Π	security	$\lvert\mathsf{crs}\rvert$	P computation	$\lvert\pi\rvert$	V computation
QAP-based (arithmetic circuit, with n gates), $m^* = m$					
[BCTV14]	KS	$(6m+n)\lvert\mathbb{G}_1\rvert + m\lvert\mathbb{G}_2\rvert$	$(6m+n)$ e1 + m e2	$7\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	12 p + m_0 e1
[Gro16]	KS	$(m+2n)\lvert\mathbb{G}_1\rvert + n\lvert\mathbb{G}_2\rvert$	$(m+3n)$ e1 + n e2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	3 p + m_0 e1
$\mathsf{S}_{\mathsf{qap}}$ § 3	KS	$(m+2n)\lvert\mathbb{G}_1\rvert + n\lvert\mathbb{G}_2\rvert$	$(m+2n)$ e1 + n e2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	3 p + m_0 e1
$\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ § 3	NBBASE	$(m+3n)\lvert\mathbb{G}_1\rvert + n\lvert\mathbb{G}_2\rvert$	$(m+3n)$ e1 + n e2	$3\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	5 p + m_0 e1
SAP-based (arithmetic circuit, with n^* squaring gates): $u = v$, $n^* \approx 2n$, $m^* \approx m$					
[GM17]	NBBSASE	$(m^*+4n^*)\lvert\mathbb{G}_1\rvert + 2n^*\lvert\mathbb{G}_2\rvert$	(m^*+4n^*) e1 + $2n^*$ e2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	5 p + m_0 e1
$\mathsf{S}_{\mathsf{sap}}$ § 4	KS	$(m^*+2n^*)\lvert\mathbb{G}_1\rvert + n^*\lvert\mathbb{G}_2\rvert$	(m^*+2n^*) e1 + n^* e2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	3 p + m_0 e1
$\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$ § 4	NBBASE	$(m^*+2n^*)\lvert\mathbb{G}_1\rvert + n^*\lvert\mathbb{G}_2\rvert$	(m^*+2n^*) e1 + n^* e2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	5 p + m_0 e1
SSP-based (Boolean circuit with n gates): $u = v = w$, $m^* = m + n$					
[DFGK14]	KS	$(m^*+n)\lvert\mathbb{G}_1\rvert + n\lvert\mathbb{G}_2\rvert$	$2m^*$ m1 + n e1 + m^* m2	$3\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	6 p + m_0 m1
$\mathsf{S}_{\mathsf{ssp}}$ § 5	KS	$(m^*+2n)\lvert\mathbb{G}_1\rvert + n\lvert\mathbb{G}_2\rvert$	$2m^*$ m1 + n e1 + m^* m2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	3 p + m_0 m1
$\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ § 5	NBBASE	$(m^*+2n)\lvert\mathbb{G}_1\rvert + n\lvert\mathbb{G}_2\rvert$	$2m^*$ m1 + n e1 + m^* m2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	5 p + m_0 m1
QSP-based (Boolean circuit with n gates): $w = 0$, $n^* \approx 14n$ [Lip13]					
[Lip13]	KS	–	–	–	–
$\mathsf{S}_{\mathsf{qsp}}$ § 6	KS	$(m^*+2n^*)\lvert\mathbb{G}_1\rvert + n^*\lvert\mathbb{G}_2\rvert$	$2m^*$ m1 + n^* e1 + n^* e2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	3 p + m_0^* m1
$\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ § 6	NBBASE	$(m^*+3n^*)\lvert\mathbb{G}_1\rvert + n^*\lvert\mathbb{G}_2\rvert$	$3m^*$ m1 + n^* e1 + n^* e2	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	5 p + m_0^* m1

In Section 3, we propose a knowledge-sound SNARK $\mathsf{S}_{\mathsf{qap}}$ for QAP. Recall that to show that the prover is honest, one needs to show that $\chi(X) := u(X)v(X) - w(X) - h(X)\ell(X) = 0$ (see [GGPR13]), where $\ell(X)$ is a public fixed polynomial, the polynomials $u(X)$, $v(X)$, and $w(X)$ depend on the concrete circuit and the witness the prover is using, and $h(X) = (u(X)v(X) - w(X))/\ell(X)$ is a polynomial iff the prover is honest. Note that $\chi(X)$ consists of a linear combination of 3 terms, each of us being a product of two polynomials, with polynomials $\ell(X)$ and 1 being publicly known.

We now 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) 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 generic prover [Sho97] by inserting to the CRS elements of type $f(X)Y^\kappa$ only for polynomials $f(X)$ that divide by $\ell(X)$. The resulting zk-SNARK $\mathsf{S}_{\mathsf{qap}}$ guarantees that (i) $u(X)$, $v(X)$, and $w(X)$ use the same witness, and (ii) the public input encoded into $u(X)$ is correct. It will be somewhat more complicated (but not less efficient!).

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

The way we choose these powers of Y is interesting by itself. Let $\boldsymbol{X}^* = (X, \ldots)$ be the vector of all indeterminates, except Y, that are available in the knowledge-soundness (or NBBASE) proof. This includes indeterminates created by the adversary by using elliptic curve hashing and (in the case of NBBASE proof) indeterminates created by simulator queries. Then, $V(\boldsymbol{X}^*, Y) = \sum V_i(\boldsymbol{X}^*)Y^i$ for some polynomials $V_i(\boldsymbol{X}^*)$ where i is a linear combination of some initially undetermined values $\alpha, \beta, \gamma, \ldots$. We identify that the prover is honest ($\chi(X) = 0$, the public inputs are correctly encoded, and one uses the

same witness in $u(X), v(X), w(X)$) if and only $V_i(\boldsymbol{X}^*) = 0$ for six (in the case of $\mathsf{S}_{\mathsf{qap}}$) so-called *critical* values i. We then choose $\alpha, \beta, \ldots$ so that the corresponding six linear combinations i are distinct from each other and all other "non-critical" linear combinations j. Moreover, we choose $\alpha, \beta, \ldots$ so that the SNARK is relatively efficient. E.g., we require 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 used an exhaustive search at this point. Due to this, exponents in the resulting SNARKs can look somewhat obscure, e.g., $A(X, Y) = r_a Y^0 + u(X)Y^{-1}$ and $B(X, Y) = r_b Y^4 + v(X)Y^3$.

We modify $\mathsf{S}_{\mathsf{qap}}$ minimally to also make it NBBASE. More precisely, to achieve NBBASE, we establish that for any k a malicious prover has an attack vector by setting $A(X, Y) = s_{a1k}\mathsf{D}_k + \ldots$ for non-zero s_{a1k} and an indeterminate D_k generated during the kth simulation query. We eliminate this vector by additionally requiring the prover to compute polynomial $A(X, Y)Z$ for a new indeterminate Z (as often done in knowledge-assumption-based security proofs). This increases the complexity of the SNARK slightly, but the resulting NBBASE SNARK is still significantly more efficient than the Groth-Maller NBBSASE zk-SNARK from CRYPTO 2017. Moreover, the latter was only given for the language SAP that has an efficient reduction from arithmetic circuits that only have squaring gates (and are thus usually twice larger than circuits that have generic multiplication gates, [GM17]).

Importantly, $\mathsf{S}_{\mathsf{qap}}$ has a simple Sub-GBGM knowledge-soundness proof where only the value of the six critical coefficients of V matter. The NBBASE proof of the NBBASE SNARK has an arguably even simpler proof due to the effective use of a knowledge assumption. This should be compared to Groth's SNARK from EUROCRYPT 2016 [Gro16] (resp., the Groth-Maller SNARK from CRYPTO 2017 [GM17]) that has a very complicated knowledge-soundness (resp., NBBSASE) proof.

As we mentioned before, $\mathsf{S}_{\mathsf{qap}}$ is very similar to Groth's SNARK. We obtain a simpler knowledge-soundness proof by assuming that the pairing is asymmetric. (Asymmetric pairings are much more efficient than symmetric pairings and thus strongly preferred in practice.) On the other hand, Groth proved knowledge-soundness in the case of symmetric pairing, which results in both malicious $A(\boldsymbol{X}^*, Y)$ and $B(\boldsymbol{X}^*, Y)$ having more terms and thus $V(\boldsymbol{X}^*, Y) = \sum V_i(\boldsymbol{X}^*)Y^i$ having more critical coefficients. Thus, one corollary of our knowledge-sound proof is the (up to our knowledge, novel) observation that Groth's SNARK has a very simple knowledge-soundness proof given that one uses asymmetric pairings. Our goal was *not* to duplicate Groth's SNARK but to construct an efficient SNARK that has a simple knowledge-soundness proof. Thus, our exposition of the derivation of $\mathsf{S}_{\mathsf{qap}}$ can also be seen as an intuitive pedagogical re-derivation of (a slight variant) the most efficient existing pairing-based SNARK. We emphasize that, on the other hand, $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is novel.

After that, we consider languages SAP [Gro16,GM17], SSP [DFGK14], and QSP [GGPR13,Lip13] that have been used in the pairing-based SNARK literature. For each of them, we explain its algebraic relation to QAP, which helps us to lift both $\mathsf{S}_{\mathsf{qap}}$ and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ to the setting of the corresponding languages. In the SNARK literature, all four languages are usually handled separately and our (simple) relation seems to be novel. In some of the cases, we improve on the efficiency of previous known SNARKs for the same language. In the case of NBBASE, we propose the first known SNARK for QAP, SSP, and SAP. In particular, we propose the first known (efficient) NBBASE SNARK for SSP and QSP and, therefore, 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 4, we describe a SNARK $\mathsf{S}_{\mathsf{sap}}$ for the language SAP (Square Arithmetic Program, [GM17]). SAP has an efficient reduction to arithmetic circuits that use squaring gates instead of multiplication gates. Algebraically, SAP is a variant of QAP with $v(X) = u(X)$; thus, $\chi(X) = u(X)^2 - w(X) - h(X)\ell(X)$. $\mathsf{S}_{\mathsf{sap}}$ is as efficient than $\mathsf{S}_{\mathsf{qap}}$. Since the argument contains $([\mathsf{a}]_1, [\mathsf{b}]_2)$ with $\mathsf{a} = \mathsf{b}$ in the honest case, to obtain NBBASE it is sufficient to check that $[\mathsf{a}]_1 \bullet [1]_2 = [1]_1 \bullet [1]_1$; that is, one does not have to introduce the new variable Z. However, one has to take into account that such a circuit has usually 2 times more gates; this is since in general one needs two squaring gates to implement a multiplication gate; however

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	$= u(X)$	$= u(X)$	$= u(X)$
QSP	general	general	$= 0$

this is a difference in the reduction overhead between circuits and the corresponding language, not in the cryptographic construction of the SNARK.

In Section 5, we describe a SNARK $\mathsf{S}_{\mathsf{ssp}}$ for the SSP language [DFGK14] that has efficient reduction from Boolean circuit. Algebraically, SSP is a variant of QAP, where one sets $u(X) = v(X) = w(X)$. In this case, $\chi(X) = u(X)(u(X) - 1) - h(X)\ell(X)$. $\mathsf{S}_{\mathsf{ssp}}$ is approximately as efficient as the SSP-based SNARK of Danezis *et al* but it has a shorter argument with more efficient verification (only one verification equation instead of two). The new NBBASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ for SSP uses the same idea as $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$, by just adding a single verification to check that $A(X, Y) = B(X, Y)$. No previous NBBASE SNARKs for SSP were known. Moreover, 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 for applications where one needs to use Boolean circuits. They are also useful in applications like shuffle arguments [FLZ16,FLSZ17], and SSP has been used as the basis for falsifiable SNARKs with long commitments [DGP+19].

Finally, in Section 6, we design a SNARK for QSP (Quadratic Span Programs, [GGPR13,Lip13]). Algebraically, QSP is a variant of QAP, where one sets $w(X) = 0$. However, the reduction from Boolean circuits to QSP is relatively complex, with the need to implement span-program-based gate checkers and error-correcting-code-based wire checkers [GGPR13,Lip13]. We construct an NBBASE version $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ of this SNARK similarly to how we constructed $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$. $\mathsf{S}_{\mathsf{qsp}}$ is again more efficient than previously known knowledge-sound SNARKs for QSP, while there was no previously known NBBASE SNARK for QSP.

To construct eight different SNARKs and verify their sets of critical coefficients and also soundness in the generic model, we used computer algebra and exhaustive search. We believe that the soundness of the SNARKs is obvious, assuming that the variables $\alpha, \beta, \gamma, \ldots$ have been chosen so that exponents of Y corresponding to the critical coefficients are different from all other exponents. However, finding *small* values of these variables seems to require exhaustive search — the number of non-zero coefficients of $V_i(X)$ (even in the knowledge-soundness proof without allowing the generic adversary to create new indeterminates) is approximately 30, depending on the SNARK. This issue can be solved by using more trapdoors as in [Gro16].

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

Another open question is the applicability of NBBASE SNARKs. One obvious application (further studying of which we will leave as an open problem) is the construction of UC-secure SNARKs as in [KZM+15]. Kosba *et al.* used a knowledge-sound SNARK to construct a UC-secure SNARK with non-succinct commitment. This construction used a generic transformation from non-black-box knowledge-sound argument systems (with perfectly binding and extractable commitments) to black-box simulation-extractable argument systems, implemented as a very large arithmetic circuit. We conjecture that there exists a more efficient transformation from NBBASE argument systems to black-box simulation-extractable argument systems.

Finally, we conjecture that at least the NBBASE security of $\mathsf{S}^{\mathsf{se}}_{\mathsf{sap}}$ and $\mathsf{S}^{\mathsf{se}}_{\mathsf{ssp}}$ should be provable under reasonable knowledge assumptions.

2 Preliminaries

Let PPT denote probabilistic polynomial-time and let $\lambda \in \mathbb{N}$ be the security parameter. All adversaries will be stateful. For an algorithm $\mathcal{A}$, range($\mathcal{A}$) is the range of $\mathcal{A}$, i.e., the set of of valid outputs of $\mathcal{A}$, $\mathsf{RND}(\mathcal{A})$ denotes the random tape of $\mathcal{A}$, and $r \leftarrow\!\!\$\, \mathsf{RND}(\mathcal{A})$ denotes the random choice of the randomizer r from $\mathsf{RND}(\mathcal{A})$. By $y \leftarrow \mathcal{A}(\mathsf{x}; r)$ we denote the fact that $\mathcal{A}$, given an input x and a randomizer r, outputs y. When we use this notation then r represents the full random tape of $\mathcal{A}$. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function, and by $\mathsf{poly}(\lambda)$ an arbitrary polynomial function. We write $a \approx_\lambda b$ if $|a - b| \leq \mathsf{negl}(\lambda)$. For a matrix $\boldsymbol{A}$, $\boldsymbol{A}_i$ denotes its ith row and $\boldsymbol{A}^{(j)}$ denotes its jth column.

Assume n is a power of two, and let ω be the n-th primitive root of unity modulo p. Such ω exists, given that $n \mid (p-1)$. Then, $\ell(X) := \prod_{i=1}^{n}(X - \omega^{i-1})$ is the unique degree n monic polynomial such that $\ell(\omega^{i-1}) = 0$ for all $i \in [1\,..\,n]$. For $i \in [1\,..\,n]$, let $\ell_i(X)$ be the *ith Lagrange basis polynomial*, i.e., the unique degree $n-1$ polynomial s.t. $\ell_i(\omega^{i-1}) = 1$ and $\ell_i(\omega^{j-1}) = 0$ for $i \neq j$. Given any $\chi \in \mathbb{Z}_p$, 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 $(\ell(\omega^{i-1}))_{i=1}^{n} = \boldsymbol{0}_n$.

A bilinear group generator $\mathsf{Pgen}(1^\lambda)$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ 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 require the bilinear pairing to be Type-3 [GPS08], i.e., we assume that there is no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. 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 $\hat{e}([a]_1, [b]_2)$ as $[a]_1 \bullet [b]_2$. We use freely the bracket notation together with matrix notation, e.g., if $\boldsymbol{AB} = \boldsymbol{C}$ then $[\boldsymbol{A}]_1 \bullet [\boldsymbol{B}]_2 = [\boldsymbol{C}]_T$.

QAP. Quadratic Arithmetic Program (QAP) was introduced in [GGPR13] as a language where for an input x and witness w, $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$ can be verified by using a parallel quadratic check, and that has an efficient reduction from the well-known language (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 and m to the number of wires in the circuit. Here, we consider arithmetic circuits that consist only of fan-in-2 multiplication gates, but either input of each multiplication gate can be a weighted sum of some wire values, [GGPR13].

Let $\mathbb{F} = \mathbb{Z}_p$, such that ω is the n-th primitive root of unity modulo p. (This requirement is needed for the sake of efficiency, and we will make it implicitly throughout the paper. However, it is not needed for the new SNARKs to work.) A QAP is characterized by n constraints $(\sum_{j=1}^{m} U_{ij} u_j(X))(\sum_{j=1}^{m} V_{ij} u_j(X)) = \sum_{j=1}^{m} W_{ij} w_j(X)$, where $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ are instant-dependent matrices. For $j \in [0\,..\,m]$, define $u_j(X) := L_{\boldsymbol{U}^{(j)}}(X)$, $v_j(X) := L_{\boldsymbol{V}^{(j)}}(X)$, and $w_j(X) := L_{\boldsymbol{W}^{(j)}}(X)$ to be interpolating polynomials of the jth column of the corresponding matrix. Thus, $u_j, v_j, w_j \in \mathbb{Z}_p^{(\leq n-1)}[X]$.

An QAP instance $\mathcal{Q}_p$ is specified by the so defined $(\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=0}^{m})$. This instance defines the following relation, where we assume that $a_0 = 1$:

$$\mathbf{R}_{\mathcal{Q}_p} = \left\{(\mathsf{x}, \mathsf{w}) \colon \mathsf{x} = (a_1, \ldots, a_{m_0})^\top \wedge \mathsf{w} = (a_{m_0+1}, \ldots, a_m)^\top \quad u(X)v(X) \equiv w(X) \pmod{\ell(X)}\right\}$$

where $u(X) = \sum_{j=0}^{m} a_j u_j(X)$, $v(X) = \sum_{j=0}^{m} a_j v_j(X)$, and $w(X) = \sum_{j=0}^{m} a_j w_j(X)$. Alternatively, $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$ if there exists a (degree $\leq n-2$) polynomial $h(X)$, s.t.

$$\chi(X) := u(X)v(X) - w(X) \cdot 1 - h(X)\ell(X) = 0 \ , \tag{1}$$

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

A *non-interactive zero-knowledge argument system* Ψ for $\mathcal{R}$ consists of four PPT algorithms:

CRS generator: $\mathsf{K}_{\mathsf{crs}}$ is a probabilistic algorithm that, given $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$, outputs $(\mathsf{crs}, \mathsf{tc})$ where crs is a CRS and tc is a CRS 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 $(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{x}, \mathsf{w})$ for $(\mathsf{x}, \mathsf{w}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\perp$.

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

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

A SNARK is required to satisfy completeness (honest verifier accepts honest verifier), knowledge-soundness (if a prover makes honest verifier accept, then we can extract from the prover the witness w), and zero-knowledge (there exists a simulator that, knowing CRS trapdoor but not knowing the witness, can produce accepting statements with the verifier's view being indistinguishable from the view when interacting with a honest prover). See Appendix B.1 for formal definitions.

Simulation-Extractability (SE). SE [Sah99,DDO+01] is a stronger notion of knowledge-soundness, motivated by cryptographic applications like non-malleability and UC-security. An SE NIZK 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, that can deduce the witness from $\mathcal{A}$.

Dodis *et al.* [DHLW10] differentiated between several different favors of SE. In the case of any-simulation-extractability (ASE), the simulator can be queried with any (potentially false) statement while in the case of true-simulation-extractability (TSE), the simulator can only be queried with true statements. Moreover, in the case of strong (any or true)-simulation-extractability, the adversary wins even if she can come up with a new argument for a statement she has queried a simulation for.

Groth and Maller [GM17] introduced the notion of non-black-box simulation-extractability (NBBSE) for SNARKs. In NBBSE, 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 deduce the witness from $\mathcal{A}$. See [KZM+15] for a treatment of SE, with an universal extractor,, for SNARKs.

The definition of SE from [GM17] corresponds to a *non-black-box strong any-simulation extractability (NBBSASE).* Groth and Maller proved that for any NBBSASE, the argument consists at least of three group elements and that there should be at least two verification equations. They also proposed one concrete NBBSASE SNARK, based on the SAP (Square Arithmetic Program) language, that meets the lower bounds. In the current paper, we design several non-black-box any-simulation-extractable (NBBASE) SNARKs based on different languages like QAP [GGPR13], SSP [DFGK14], SAP [Gro16], and QSP [GGPR13]. As argued by Dodis *et al.* [DHLW10], TSE (and thus also ASE) is sufficient in many

applications. We will provide formal definitions of non-black-box simulation-extractability (NBBSASE and NBBASE) in Appendix B.2.

Generic Model. In the *Generic Bilinear Group Model* (GBGM) [Nec94,Sho97,Mau05,BBG05], one assumes that the adversary has only access to group elements via generic bilinear-group operations (group operations and the bilinear map) together with an equality test. In the *subversion GBGM* (Sub-GBGM, [SPMS02,BFS16,ABLZ17]; named *generic group model with hashing into the group* in [BFS16]), the adversary has an additional power of creating new indeterminates in bilinear group. The Sub-GBGM is motivated by elliptic curve hashing algorithms [Ica09,BCI+10,TK17] that allow one to efficiently create elliptic-curve group elements without knowing their discrete logarithms.

Thus, Sub-GBGM is a weaker model than GBGM. As an important example, knowledge assumptions that state that the output group element must belong to the span of input group elements hold in the GBGM but not in the Sub-GBGM. This is since in the Sub-GBGM, the adversary can create new group elements without knowing their discrete logarithms; indeed the output element might be equal to one such created group elements. Hence, an Sub-GBGM adversary is less restricted than a GBGM adversary. Moreover (see, e.g., [ALSZ18]), some knowledge assumptions that have a trivial security proof in the GBGM have quite a complicated proof in the Sub-GBGM.

All knowledge-soundness and NBBASE proofs in this paper are in the Sub-GBGM. Intuitively, it means that we will operate with polynomials in several indeterminates by using constraints given by the pairing-based setting. In the actual SNARK, we will instead have representations of said polynomials in one of the three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, where the representation of $f(\boldsymbol{X})$ in group $\mathbb{G}_\iota$ is $[f(\boldsymbol{x})]_\iota$ for $\boldsymbol{x}$ being a vector of secret trapdoors. In the pairing-based setting, the adversary can only compute new polynomials "in $\mathbb{G}_\iota$" from the span of the polynomials "in $\mathbb{G}_\iota$" in her input (that importantly includes the CRS) and new indeterminates she has created in $\mathbb{G}_\iota$. For example, if the CRS is $([x^2y^3, y^2, 1]_1, [1, xy]_2)$ then any output of the adversary has to be a known linear combination of the form $[a_1x^2y^3 + a_2y^2 + a_3 + \sum_k q_kQ_{\iota k}]_1$ or $[a_1 + a_2xy + \sum q_kQ_k]_2$ for known constants a_i, where $Q_{\iota k}$ are indeterminates are created by the adversary in group $\mathbb{G}_\iota$. In addition, given polynomials $A(X, Y)$ and $B(X, Y)$ in the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, one can compute the polynomial $A(X, Y)B(X, Y)$ in the target group $\mathbb{G}_T$, by using bilinear pairing. Due to this, we will use the language of polynomials in the rest of this paper, switching back to their representation in some group only at the end of each section.

See Appendix A for a long introduction to GBGM and Sub-GBGM.

3 QAP-Based SNARKs

In this section, we will describe a generic template for SNARKs for QAP and then give details of concrete new SNARKs $\mathsf{S}_{\mathsf{qap}}$ (SNARK for QAP) and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ (SNARK for QAP, SE). The template follows two objectives: (i) simple soundness proof in the generic model, and (ii) efficiency. In fact, $\mathsf{S}_{\mathsf{qap}}$ will be 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 for the case of asymmetric pairings; Groth on the other hand provided a very complex knowledge-soundness proof that is valid for both asymmetric and symmetric pairings. The NBBASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is novel, and has arguably an even simpler security proof.

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

The main intuition behind our approach is that $\chi(X)$ can be seen as a linear combination of three products of two polynomials. Thus, if we represent one multiplicand polynomial in each addend as a member of $\mathbb{G}_1$ and another multiplicand as a member of $\mathbb{G}_2$, we can use pairings and group operations to check if $\chi(X) = 0$, i.e., that the prover was honest. (More precisely, we represent an evaluation of the

polynomial at a random point in the corresponding group; however, for a moment we will just use the terminology of polynomials.)

Our very first idea is to commit to $u(X)$, $v(X)$, and $w(X)$ as $A(X,Y) = u(X)Y + w(X)Y^2 + h(X)Y^3$ and $B(X,Y) = v(X)Y^3 - Y^2 - \ell(X)Y$. In this case, the coefficient of Y^4 in $C(X,Y) := A(X,Y)B(X,Y)$ is $\chi(X)$. Thus, if an adversary can compute $C(X,Y)$ (i.e., compute its coefficients), he can also compute $\chi(X)$. One can construct a knowledge-sound SNARK so that the CRS does not contain any elements of the shape $f(X)Y^4$ where $f(X)$ is a non-zero polynomial; this means that a generic adversary will not be able to compute a representation of $C(X,Y)$ unless the input belongs to QAP. We construct a verification equation that accepts iff the coefficient of Y^4 in $C(X,Y)$ is 0 iff the prover is honest. To obtain zero-knowledge, one can define $A(X,Y) = r_a + u(X)Y + w(X)Y^2 + h(X)Y^3$ and $B(X,Y) = r_b + v(X)Y^3 - Y^2 - \ell(X)Y$ for uniformly random r_a and r_b.

We optimize this by noting that if $A(X,Y) = u(X)Y + w(X)Y^2$ and $B(X,Y) = v(X)Y^3 - Y^2$ then the coefficient of Y^4 in $C(X,Y) = A(X,Y)B(X,Y)$ is $u(X)v(X) - w(X) = h(X)\ell(X)$ for some $h(X)$ iff the prover is honest. One obtains knowledge-soundness by including to the CRS elements of type $f(X)Y^4$ only if $\ell(X) \mid f(X)$. Since $A(X,Y)$ and $B(X,Y)$ now have one fewer addend, $C(X,Y)$ has fewer addends and thus both the prover's computation and the CRS length are smaller.

Unfortunately, this general approach is too simplistic: on top of $\chi(X) = 0$, we need to additionally guarantee that the public input is correct and that the same coefficients a_j are used in the computation of each $u(X) = \sum a_j u_j(X)$, $v(X) = \sum a_j v_j(X)$, and $w(X) = \sum a_j w_j(X)$. Moreover, the described SNARK is less efficient (especially in the CRS size) than the state-of-the-art SNARKs like [Gro16]. Thus, we will use a somewhat more complex approach.

The argument in the new template consists of three elements, $\pi = ([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2)$, where $\mathsf{a} = A(x,y)$, $\mathsf{b} = B(x,y)$, and $\mathsf{c}_s = C_s(x,y)$ for well-defined polynomials $A(X,Y)$, $B(X,Y)$, and $C_s(X,Y)$. Intuitively, $[\mathsf{a}]_1$ is a succinct commitment to $u(X)$ while $[\mathsf{b}]_2$ is a succinct a commitment to $v(X)$. Moreover, $[\mathsf{c}_s]_1$ is the actual argument. More precisely, we set $\mathsf{c} = C(x,y)$ where

$$C(X,Y) = (A(X,Y) + Y^\epsilon)(B(X,Y) + Y^\zeta) - Y^{\epsilon+\zeta} = A(X,Y)B(X,Y) + B(X,Y)Y^\epsilon + A(X,Y)Y^\zeta \quad (2)$$

for integers ϵ, ζ chosen later. We define $A(X,Y)$, $B(X,Y)$, and $C(X,Y)$ (chosen as per Eq. (2)) as follows, where also $\alpha, \beta, \gamma, \kappa$ are chosen later:

$$\begin{aligned}
A(X,Y) =& r_a Y^\alpha + u(X)Y^\gamma \ , \\
B(X,Y) =& r_b Y^\beta + v(X)Y^{\kappa-\gamma} \ , \\
C(X,Y) =& r_a r_b Y^{\alpha+\beta} + r_b u(X) Y^{\beta+\gamma} + r_a v(X) Y^{\alpha-\gamma+\kappa} + r_a Y^{\alpha+\zeta} + r_b Y^{\beta+\epsilon} + u(X)v(X)Y^\kappa + \\
& u(X)Y^{\gamma+\zeta} + v(X)Y^{-\gamma+\kappa+\epsilon} \\
=& r_b(A(X,Y) + Y^\epsilon)Y^\beta + r_a v(X) Y^{\alpha-\gamma+\kappa} + r_a Y^{\alpha+\zeta} + \\
& \textstyle\sum_{j=1}^m a_j (u_j(X)Y^{\gamma+\zeta} + v_j(X)Y^{-\gamma+\kappa+\epsilon} + w_j(X)Y^\kappa) + (u(X)v(X) - w(X))Y^\kappa \ .
\end{aligned} \quad (3)$$

Since a SNARK also has a public input $(a_j)_{j=1}^{m_0}$, we define two polynomials $C_s(X,Y)$ and $C_p(X,Y)$, so that[1] $C_p(X,Y)Y^\delta + C_s(X,Y)Y^\beta = C(X,Y)$, 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):

$$\begin{aligned}
C_p(X,Y) =& \textstyle\sum_{j=1}^{m_0} a_j \left(u_j(X)Y^{-\delta+\gamma+\zeta} + v_j(X)Y^{-\delta-\gamma+\kappa+\epsilon} + w_j(X)Y^{-\delta+\kappa}\right) \ , \\
C_s(X,Y) =& \textstyle\sum_{j=m_0+1}^{m} a_j \left(u_j(X)Y^{-\beta+\gamma+\zeta} + v_j(X)Y^{\beta+-\gamma+\kappa+\epsilon} + w_j(X)Y^{-\beta+\kappa}\right) + \\
& (u(X)v(X) - w(X))Y^{-\beta+\kappa} + r_b\left(A(X,Y) + Y^\epsilon\right) + r_a v(X) Y^{\alpha-\beta-\gamma+\kappa} + r_a Y^{\alpha-\beta+\zeta} \ .
\end{aligned} \quad (4)$$

[1] We use the multiplicand Y^β for efficiency reasons, since $C(X,Y)$ has an addend $r_a A(X,Y)Y^\beta$.

To minimize the length of the CRS, we make one more optimization, setting $\beta = \alpha - 2\gamma + \kappa$. Without this, the CRS contains $[\{x^i y^\gamma\}_{i=0}^{n-1}]_1$ and $[\{x^i y^{\alpha-\beta-\gamma+\kappa}\}_{i=0}^{n-1}]_1$ separately. Thus, Eq. (4) simplifies to

$$\begin{aligned}
A(X,Y) =& r_a Y^\alpha + u(X)Y^\gamma \ , \\
B(X,Y) =& r_b Y^{\alpha-2\gamma+\kappa} + v(X)Y^{\kappa-\gamma} \ , \\
C_p(X,Y) =& \textstyle\sum_{j=1}^{m_0} a_j \left(u_j(X)Y^{-\delta+\gamma+\zeta} + v_j(X)Y^{-\delta-\gamma+\kappa+\epsilon} + w_j(X)Y^{-\delta+\kappa}\right) \ , \\
C_s(X,Y) =& \textstyle\sum_{j=m_0+1}^{m} a_j \left(u_j(X)Y^{-\alpha+3\gamma+\zeta-\kappa} + v_j(X)Y^{-\alpha+\gamma+\epsilon} + w_j(X)Y^{2\gamma-\alpha}\right) + \\
& (u(X)v(X) - w(X))Y^{-\alpha+2\gamma} + r_b\left(A(X,Y) + Y^\epsilon\right) + r_a v(X)Y^\gamma + r_a Y^{2\gamma-\kappa+\zeta} \ .
\end{aligned} \tag{5}$$

Note that if $-\alpha+2\gamma$ (now, considering α and γ to be integers) is different from all other exponents in $C(X,Y)$, then the coefficient of $Y^{-\alpha+2\gamma}$ in $C(X,Y)$ is $u(X)v(X) - w(X)$. 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 $Y^{-\alpha+2\gamma}$ in $C(X,Y)$ is divisible by $\ell(X)$. In addition, $C_s(X,Y)$ has addends $u_j(X)Y^{-\alpha+3\gamma+\zeta-\kappa}$, $v_j(X)Y^{-\alpha+\gamma+\epsilon}$, and $w_j(X)Y^{2\gamma-\alpha}$; 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 $Y^{-\alpha+2\gamma}$ is the main reason why we chose $C(X,Y)$ as in Eq. (2).

In the resulting SNARK, the verifier V checks that $([\mathsf{a}]_1 + [y^\epsilon]_1) \bullet ([\mathsf{b}]_2 + [y^\zeta]_2) - [y^{\eta+\zeta}]_T = [\mathsf{c}_p]_1 \bullet [y^\delta]_2 + [\mathsf{c}_s]_1 \bullet [y^\beta]_2$, where $[\mathsf{c}_p]_1$ is recomputed by V. If this holds, then in the generic model, also Eq. (2) holds. In $\mathbb{G}_1$, the CRS contains (representations of) polynomials, needed to compute $A(X,Y) + Y^\epsilon$, $C_p(X,Y)$, and $C_s(X,Y)$. In $\mathbb{G}_2$, the CRS contains polynomials, needed the compute $B(X,Y) + Y^\epsilon$, $Y^{\alpha-2\gamma+\kappa}$, and Y^δ. See the description of the CRS in Fig. 1. (Some of the computation is done by the prover only and some of it by the verifier only; this explains the division of the CRS to crs_P and crs_V.)

In the malicious case, the Sub-GBGM adversary can also create a number of new indeterminates. Let $\boldsymbol{Q}_1$ be the vector of new indeterminates in $\mathbb{G}_1$ and $\boldsymbol{Q}_2$ be the corresponding vector in $\mathbb{G}_2$. Let $\boldsymbol{Q} = (\boldsymbol{Q}_1, \boldsymbol{Q}_2)$ and $\boldsymbol{X} = (X, \boldsymbol{Q}, Y)$. Then, the polynomial corresponding to the verification equation is

$$V(\boldsymbol{X}) = (A(X,\boldsymbol{Q}_1,Y)+Y^\epsilon)(B(X,\boldsymbol{Q}_2,Y)+Y^\zeta) - Y^{\zeta+\epsilon} - C_p(X,\boldsymbol{Q}_1,Y)Y^\delta - C_s(X,\boldsymbol{Q}_1,Y)Y^{\alpha-2\gamma+\kappa} \ , \tag{6}$$

for potentially maliciously computed polynomials $A(X,\boldsymbol{Q}_1,Y)$, $B(X,\boldsymbol{Q}_2,Y)$, and $C_s(X,\boldsymbol{Q}_1,Y)$ (that can also depend on the new indeterminates). Thus, if the verifier accepts then (since we are working in the generic model) $V(\boldsymbol{X}) = 0$. Writing $V(\boldsymbol{X}) = \sum_i V_i(X,\boldsymbol{Q})Y^i$, it follows from $V(\boldsymbol{X}) = 0$ that each $V_i(X,\boldsymbol{Q}) = 0$. We now aim to construct the SNARK so that from $V_i(X,\boldsymbol{Q}) = 0$, for $i \in S$ (with $\kappa \in S$) and $|S|$ being small, it follows that the prover is honest.

To formalize the previous discussion, consider the malicious case where $A(X,\boldsymbol{Q}_1,Y)$, $B(X,\boldsymbol{Q}_2,Y)$, and $C_s(X,\boldsymbol{Q}_1,Y)$ can be arbitrary polynomials in the span of polynomials represented in the CRS in groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_1$, respectively. Since a generic adversary can only compute polynomials that are in the span of the polynomials in the CRS (of the same group), then we know that maliciously computed $A(X,\boldsymbol{Q}_1,Y)$ and $B(X,\boldsymbol{Q}_2,Y)$ have to have the following shape.

$$\begin{aligned}
A(X,\boldsymbol{Q}_1,Y) =& \textstyle\sum_{j=1}^{m_0} a_j^*(u_j(X)Y^{-\delta+\gamma+\zeta} + v_j(X)Y^{-\delta-\gamma+\kappa+\epsilon} + Y^{-\delta+\kappa}w_j(X)) + \\
& \textstyle\sum_{i=m_0+1}^{m} a_i^*(u_i(X)Y^{-\alpha+3\gamma+\zeta-\kappa} + v_i(X)Y^{-\alpha+\gamma+\epsilon} + w_i(X)Y^{2\gamma-\alpha}) + \\
& \ell(X)h_a(X)Y^{-\alpha+2\gamma} + r_a Y^\alpha + u_a(X)Y^\gamma + a_\epsilon Y^\epsilon + a_{2\gamma+\zeta-\kappa}Y^{2\gamma+\zeta-\kappa} + \textstyle\sum_k q_{ak}Q_{1k} \ , \\
B(X,\boldsymbol{Q}_2,Y) =& r_b Y^{\alpha-2\gamma+\kappa} + v_b(X)Y^{\kappa-\gamma} + b_\zeta Y^\zeta + b_\delta Y^\delta + \textstyle\sum_k q_{bk}Q_{2k} \ .
\end{aligned} \tag{7}$$

$C(X,\boldsymbol{Q}_1,Y)$ is defined like $A(X,\boldsymbol{Q}_1,Y)$, except that one has to replace the letter a with the letter c in variable names (e.g., r_a is replaced with r_c).

Let $R = \{i : V_i(X,\boldsymbol{Q}) \neq 0\}$, where $\boldsymbol{\Delta} = (\alpha,\gamma,\epsilon,\zeta,\delta,\kappa)$. Let

$$S = \{\gamma+\zeta, -\gamma+\varepsilon+\kappa, \kappa, \epsilon+\zeta, \delta+\epsilon, 2\gamma+2\zeta-\kappa\}$$

$\mathsf{K}_{\mathsf{crs}}(\mathsf{p})$: Sample $x, y, z \leftarrow\!\$\, \mathbb{Z}_p^*$, return $\mathsf{tc} \leftarrow (x, y, z)$. Let

$$\mathsf{crs}_{\mathsf{P}} \leftarrow \left(\begin{bmatrix} y^\alpha, \{x^j y^\gamma\}_{j=0}^{n-1}, y^\alpha z, \{x^j y^\gamma z\}_{j=0}^{n-1}, \{x^i \ell(x) y^{-\alpha+2\gamma}\}_{j=0}^{n-2}, y^\epsilon, y^{2\gamma+\zeta-\kappa}, \\ \{u_j(x) Y^{-\alpha+3\gamma+\zeta-\kappa} + v_j(x) y^{-\alpha+\gamma+\epsilon} + w_j(x) y^{2\gamma-\alpha}\}_{j=m_0+1}^{m} \end{bmatrix}_1, \left[y^{\alpha-2\gamma+\kappa}, \{x^i y^{\kappa-\gamma}\}_{i=0}^{n-1} \right]_2 \right) \;;$$

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

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

$\mathsf{P}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, (a_j)_{j=m_0+1}^{m})$: $(r_a, r_b) \leftarrow\!\$\, \mathbb{Z}_p^2$; $u(X) \leftarrow \sum_{j=1}^m a_j u_j(X)$; $v(X) \leftarrow \sum_{j=1}^m a_j v_j(X)$; $w(X) \leftarrow \sum_{j=1}^m a_j w_j(X)$; $h(X) \leftarrow (u(X)v(X) - w(X))/\ell(X)$; $[\mathsf{a}]_1 \leftarrow r_a[y^\alpha]_1 + [u(x)y^\gamma]_1$; $[\mathsf{a}_z]_1 \leftarrow r_a[y^\alpha z]_1 + [u(x)y^\gamma z]_1$; $[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^m a_j [u_j(x) y^{-\alpha+3\gamma+\zeta-\kappa} + v_j(x) y^{-\alpha+\gamma+\epsilon} + w_j(x) y^{2\gamma-\alpha}]_1 + [h(x)\ell(x) y^{-\alpha+2\gamma}]_1 + r_b([A(x,Y)]_1 + [Y^\epsilon]_1) + r_a([y^{2\gamma-\kappa+\zeta}]_1 + [v(x)y^\gamma]_1)$; $[\mathsf{b}]_2 \leftarrow r_b[y^{\alpha-2\gamma+\kappa}]_2 + [v(x)y^{\kappa-\gamma}]_2$; return $\pi \leftarrow \left([\mathsf{a}, \mathsf{a}_z, \mathsf{c}_s]_1, [\mathsf{b}]_2\right)$;

$\mathsf{V}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, \pi = \left([\mathsf{a}, \mathsf{a}_z, \mathsf{c}_s]_1, [\mathsf{b}]_2\right))$: Set $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j [u_j(x) Y^{-\delta+\gamma+\zeta} + v_j(x) Y^{-\delta-\gamma+\kappa+\epsilon} + w_j(x) Y^{-\delta+\kappa}]_1$. Check that $[\mathsf{a} + y^\epsilon]_1 \bullet [\mathsf{b} + y^\zeta]_2 = [\mathsf{c}_p]_1 \bullet [y^\delta]_2 + [\mathsf{c}_s]_1 \bullet [y^{\alpha-2\gamma+\kappa}]_2 + [y^{\zeta+\epsilon}]_T$ and $[\mathsf{a}]_1 \bullet [z]_2 = [\mathsf{a}_z]_1 \bullet [1]_2$;

$\mathsf{Sim}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, \mathsf{tc} = y)$: $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j [u_j(x) y^{-\delta+\gamma+\zeta} + v_j(x) y^{-\delta-\gamma+\kappa+\epsilon} + w_j(x) y^{-\delta+\kappa}]_1$; $d \leftarrow\!\$\, \mathbb{Z}_p$; $e \leftarrow\!\$\, \mathbb{Z}_p$; $[\mathsf{a}]_1 \leftarrow d[1]_1$; $[\mathsf{a}_z]_1 \leftarrow dy^{-\alpha}[y^\alpha z]_1$; $[\mathsf{b}]_2 \leftarrow e[1]_2$; $[\mathsf{c}_s]_1 \leftarrow y^{-\beta}((de + dy^\zeta + ey^\epsilon)[1]_1 - y^\delta [\mathsf{c}_p]_1)$; return $\pi \leftarrow \left([\mathsf{a}, \mathsf{a}_z, \mathsf{c}_s]_1, [\mathsf{b}]_2\right)$;

Fig. 1. The new SNARKs for QAP: $\mathsf{S}_{\mathsf{qap}}$ (*without* gray entries) and $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ (*with* gray entries)

and $\bar{S} = R \setminus \bar{S}$ be the complement of S. As we will see shortly, to obtain knowledge-soundness, we will need to choose the values in $\boldsymbol{\Delta}$ so that S consists of mutually different integers ($|S| = 6$) and $S \cap \bar{S} = \emptyset$. Let $h(X) := h_c(X) - r_b h_a(X)$. Let $\tilde{a}_j = a_j - b_\delta a_j^*$ for $j \le m_0$ and $\tilde{a}_i = c_j^* - r_b a_j^*$ for $j > m_0$. 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 "significant" coefficients $V_i(X, \boldsymbol{Q})$, $i \in S$, of $V(\boldsymbol{X})$ are depicted in Fig. 2.

We are now ready to describe the full SNARK $\mathsf{S}_{\mathsf{qap}}$, see Fig. 1, and prove its knowledge-soundness. 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]. Unlike [Gro16], we will not prove knowledge-soundness in the case of symmetric pairings, although Groth's proof can be easily adapted to our case.

Theorem 1. *(1) Assume $\boldsymbol{\Delta}$ is chosen so that $S \cap \bar{S} = \emptyset$. Then $\mathsf{S}_{\mathsf{qap}}$ in Fig. 1 is knowledge-sound in the Sub-GBGM.*
(2) $\mathsf{S}_{\mathsf{qap}}$ is perfectly zero-knowledge.

Proof (Sketch). **(1: knowledge-soundness)** Let $\mathcal{A}$ be a knowledge-soundness adversary that has succeeded in outputting (x, π) such that $\mathsf{x} \notin \mathcal{L}$ but V accepts. Since the proof is in the generic model, the $\mathbb{G}_\iota$-outputs of $\mathcal{A}$ have to belong to the span of her inputs (one part of the input is the CRS) and moreover, one can extract the corresponding coordinates. Let $\boldsymbol{X} = (X, \boldsymbol{Q}, Y)$ be a vector of all indeterminates. Recall that since we work the Sub-GBGM model, we give $\mathcal{A}$ the power to create new indeterminates $Q_{\iota k}$ in $\mathbb{G}_\iota$ (see Section 2 for the posed restrictions on the latter).

In the generic model, we can extract all coefficients of $V(\boldsymbol{X})$. Then, from the verification equation (since the adversary is generic) it follows that $V(\boldsymbol{X}) = 0$ and thus $V_i(X, \boldsymbol{Q}) = 0$ for $i \in S$. Since $V_{\epsilon+\zeta}(X, \boldsymbol{Q}) = b_\zeta + a_\epsilon(b_\zeta + 1) = 0$, we have $a_\epsilon = -b_\zeta/(b_\zeta + 1)$. In particular, $a_\epsilon, b_\zeta \neq -1$ and $(a_\epsilon + 1)(b_\zeta + 1) = 1$. Moreover, $V_{\delta+\epsilon}(X, \boldsymbol{Q}) = (a_\epsilon + 1)b_\delta = 0$ and thus $b_\delta = 0$. Next, $V_{2\gamma+2\zeta-\kappa}(X, \boldsymbol{Q}) = a_{2\gamma+\zeta-\kappa} = 0$. Thus, $\tilde{a}_j = a_j$ for $j \le m_0$. Finally, $(b_\zeta + 1)u_a(X) = u(X)$, $(a_\epsilon + 1)v_b(X) = v(X)$, and $\chi(X) = u(X)v(X) - w(X) - \ell(X)h(X) = 0$, thus $\mathsf{S}_{\mathsf{qap}}$ is knowledge-sound.

(2: zero-knowledge) To see that simulator makes the verifier accept, note that $(\mathsf{a} + y^\epsilon)(\mathsf{b} + y^\zeta) - \mathsf{c}_s y^{\alpha-2\gamma+\kappa} - \mathsf{c}_p y^\delta - y^{\epsilon+\zeta} = de + dy^\zeta + ey^\epsilon - \mathsf{c}_p y^\delta - (de + dy^\zeta + ey^\epsilon - \mathsf{c}_p y^\delta) = 0$. The simulator's output comes from the correct distribution since both a and b are individually uniform in $\mathbb{Z}_p$, and c is chosen so that the verification accepts. $\square$

Y^i	Coefficient $V_i(X, \boldsymbol{Q})$ (KS and NBBASE)	$V_i^+(X, \boldsymbol{Q})$ (NBBASE only)
$Y^{\gamma+\zeta}$	$-u(X) + (b_\zeta + 1)u_a(X) + v_b(X)a_{2\gamma+\zeta-\kappa}$	$\sum_k (s_{c2k} \sum_j c^s_{kj} u_j(X) - r_b s_{a2k} \sum_j a^s_{kj} u_j(X))$
$Y^{-\gamma+\epsilon+\kappa}$	$-v(X) + (a_\epsilon + 1)v_b(X)$	$\sum_k (s_{c2k} \sum_j c^s_{kj} v_j(X) - r_b s_{a2k} \sum_j a^s_{kj} v_j(X))$
Y^κ	$u_a(X)v_b(X) - w(X) - h(X)\ell(X)$	$\sum_k (s_{c2k} \sum_j c^s_{kj} w_j(X) - r_b s_{a2k} \sum_j a^s_{kj} w_j(X))$
$Y^{\epsilon+\zeta}$	$b_\zeta + a_\epsilon(b_\zeta + 1)$	
$Y^{\delta+\epsilon}$	$(a_\epsilon + 1)b_\delta$	
$Y^{2\gamma+2\zeta-\kappa}$	$(b_\zeta + 1)\, a_{2\gamma+\zeta-\kappa}$	

Fig. 2. Critical coefficients in $\mathsf{S}_{\mathsf{qap}}$ (left) and addends to the same coefficients in the NBBASE case (right).

The choice of $\boldsymbol{\Delta}$. Recall that we need to find values for $\boldsymbol{\Delta} = (\alpha, \ldots)$, such that $S \cap \bar{S} = \emptyset$. For convenience sake, it makes sense to require that crs_1 and crs_2 both have a non-zero monomial corresponding to Y^0 (then one can publish $[1]_1$ and $[2]_1$) and that the values i, such that j for which $f(X)Y^j$ belongs to the CRS for some $f(X)$, have as small absolute values as possible (although we do not consider subversion security, this potentially speeds up the CRS verification algorithm [ABLZ17]). Since there are too many coefficients that one can take into account, we used a computer search to the following values for $\alpha, \gamma, \ldots$:

$$\alpha = 0, \gamma = -1, \epsilon = 6, \zeta = 0, \delta = 5, \kappa = 2 \ .$$

In this case,

$$\mathsf{crs}_\mathsf{P} = \left(\begin{bmatrix} y^0, \{x^j y^{-1}\}_{j=0}^{n-1}, y^0 z, \{x^j y^{-1} z\}_{j=0}^{n-1}, \{x^i \ell(x) y^{-2}\}_{j=0}^{n-2}, \\ y^6, y^{-4}, \{u_j(x) y^{-5} + v_j(x) y^5 + w_j(x) y^{-2}\}_{j=m_0+1}^{m} \end{bmatrix}_1 , \left[y^4, \{x^i y^3\}_{i=0}^{n-1}\right]_2 \right) \ ,$$
$$\mathsf{crs}_\mathsf{V} = \left([\{u_j(x) y^{-6} + v_j(x) y^4 + w_j(x) y^{-3}\}_{j=1}^{m_0}, y^6]_1, \left[y^4, y^0, y^5, z\right]_2, [y^{11}]_T \right) \ .$$

Efficiency. $\mathsf{S}_{\mathsf{qap}}$ has fewer trapdoors but otherwise essentially the same complexity as Groth's (knowledge-sound) SNARK [Gro16]. (See Table 1 for an efficiency comparison.) For example, crs_P has $(m - m_0) + 1 + n + (n-1) + 1 = m + 2n - m_0 + 1$ elements from $\mathbb{G}_1$ and $(n+2)$ elements from $\mathbb{G}_2$. Moreover, crs_V has $m_0 + 1$ elements from $\mathbb{G}_1$, 3 elements from $\mathbb{G}_2$, and one elements from $\mathbb{G}_T$. Hence, $|\mathsf{crs}| = (m + 2n + 2)|\mathbb{G}_1| + (n+4)|\mathbb{G}_2| + |\mathbb{G}_T|$ since crs_P and crs_V have one common element in $\mathbb{G}_1$. Note that $[\mathsf{a}]_1$ can be computed from $[y^\alpha]_1$ and $[x^i y^\gamma]_1$ by using $n+1$ exponentiations.

NBBASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$. Let now $\boldsymbol{X} = (X, \boldsymbol{Q}, \mathbf{D}, \mathbf{E}, Y)$ and $\boldsymbol{X}^* = (X, \boldsymbol{Q}, \mathbf{D}, \mathbf{E})$. Here, 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 NBBASE, maliciously generated $A^*(\boldsymbol{X})$, $B^*(\boldsymbol{X})$, and $C^*(\boldsymbol{X})$ have addends that correspond to the indeterminates generated by the simulator oracle:

$$\begin{aligned} A(X, \boldsymbol{Q}_1, Y) = & \textstyle\sum_{j=1}^{m_0} a_j^* (u_j(X) Y^{-\delta+\gamma+\zeta} + v_j(X) Y^{-\delta-\gamma+\kappa+\epsilon} + Y^{-\delta+\kappa} w_j(X)) + \\ & \textstyle\sum_{i=m_0+1}^{m} a_i^* (u_i(X) Y^{-\alpha+3\gamma+\zeta-\kappa} + v_i(X) Y^{-\alpha+\gamma+\epsilon} + w_i(X) Y^{2\gamma-\alpha}) + \\ & \ell(X) h_a(X) Y^{-\alpha+2\gamma} + r_a Y^\alpha + u_a(X) Y^\gamma + a_\epsilon Y^\epsilon + a_{2\gamma+\zeta-\kappa} Y^{2\gamma+\zeta-\kappa} + \textstyle\sum_k q_{ak} Q_{1k} - \\ & \textstyle\sum_k s_{a1k} \mathsf{D}_k + \sum_k s_{a2k} \sum_j \left(Y^{-\alpha+2\gamma+\zeta-\kappa} \mathsf{D}_k + Y^{-\alpha+2\gamma-\kappa} \mathsf{D}_k \mathsf{E}_k + Y^{-\alpha+2\gamma-\kappa+\epsilon} \mathsf{E}_k \right) + \\ & \textstyle\sum_k s_{a2k} \sum_j a^s_{kj} \left(u_j(X) Y^{-\alpha+3\gamma+\zeta-\kappa} + v_j(X) Y^{-\alpha+\gamma+\epsilon} + w_j(X) Y^{2\gamma-\alpha} \right) \ , \\ B(X, \boldsymbol{Q}_2, Y) = & r_b Y^{\alpha-2\gamma+\kappa} + v_b(X) Y^{\kappa-\gamma} + b_\zeta Y^\zeta + b_\delta Y^\delta + \textstyle\sum_k q_{bk} Q_{2k} + \sum_k s_{bk} \mathsf{E}_k \ . \end{aligned}$$

In this case, due to the extra inputs from the simulator, the critical coefficients of $V_i(\boldsymbol{X})$ will be changed by extra addend $V_i^+(\boldsymbol{X})$ in Fig. 2.

First, assume that only the first verification equation is used. Then the coefficients of $\mathsf{D}_{k_1}\mathsf{E}_{k_2}$ ($r_b s_{a2k_1} + s_{ak_1} s_{bk_2} - s_{c2k_1}$), $Y^\zeta \mathsf{D}_{k_1}$ ($r_b s_{a2k_1} + (B_\zeta + 1) s_{ak_1} - s_{c2k_1}$), and $Y^\epsilon \mathsf{E}_{k_2}$ ($r_b s_{a2k_1} + s_{bk_2}/(B_\zeta + 1) - s_{2Ck_1}$) in

$i = Y^{j_1}\mathsf{D}_k^{j_2}\mathsf{E}_k^{j_3}$	Coefficient $V_i(\boldsymbol{X}^*, Z)$
$Y^{\gamma+\zeta}$	$(b_\zeta+1)u_a^z(X) - u(X)$
$Y^{-\gamma+\epsilon+\kappa}$	$v_b(X) - v(X)$
Y^{κ}	$u_a^z(X)v_b(X) - w(X) - \ell(X)h_c(X)$
$Y^{\epsilon+\zeta}$	b_ζ
$Y^{\zeta}\mathsf{D}_{k_1}$	$-s_{c2k_1}$

Fig. 3. Critical coefficients in $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ after taking into account the verification equation $A_z(\boldsymbol{X}, Z) = A(\boldsymbol{X}, Z)Z$.

$V(\boldsymbol{X})$ imply that either (i) $s_{ak_1}s_{bk_2} = 0$, for all k_1 and k_2, or (ii) $s_{a1k_1} = 1/(b_\zeta+1)$ and $s_{bk_1} = b_\zeta + 1$ for at least one $k_1 = k_0$. In the first case, we can eliminate the $V_i^+(\boldsymbol{X}^*)$ addends in Fig. 2, and thus get back to the (already solved) knowledge-soundness setting.

In the second case, $A(\boldsymbol{X}) = s_{ak_1}\mathsf{D}_{k_1} + \ldots$ for $s_{ak_1} \neq 0$. To guarantee that the prover is honest, we must make it impossible for the prover to include a term $s_{ak_1}\mathsf{D}_{k_1}$, for non-zero s_{ak_1}, to $A(\boldsymbol{X})$. We achieve this by introducing a new knowledge variable Z, adding elements $Y^\alpha Z$ and $X^i Y^\gamma Z$ (in $\mathbb{G}_1$) and Z (in $\mathbb{G}_2$) to the CRS, asking the prover to compute $A_z(\boldsymbol{X}, Z) := A(\boldsymbol{X}, Z)Z$ and then letting the verifier additionally verify that $A_z(\boldsymbol{X}, Z) = A(\boldsymbol{X}, Z)Z$. After adding these elements to the CRS,

$$
\begin{aligned}
A(\boldsymbol{X}, Z) =& r_{zb}Y^\alpha Z + u_a^z(X)Y^\gamma Z + \textstyle\sum_{j=1}^{m_0} a_j^*(u_j(X)Y^{-\delta+\gamma+\zeta} + v_j(X)Y^{-\delta-\gamma+\kappa+\epsilon} + Y^{-\delta+\kappa}w_j(X)) + \\
& \textstyle\sum_{i=m_0+1}^{m} a_i^*(u_i(X)Y^{-\alpha+3\gamma+\zeta-\kappa} + v_i(X)Y^{-\alpha+\gamma+\epsilon} + w_i(X)Y^{2\gamma-\alpha}) + \\
& \ell(X)h_a(X)Y^{-\alpha+2\gamma} + r_aY^\alpha + u_a(X)Y^\gamma + a_\epsilon Y^\epsilon + a_{2\gamma+\zeta-\kappa}Y^{2\gamma+\zeta-\kappa} + \textstyle\sum_k q_{ak}Q_{1k} - \\
& \textstyle\sum_k s_{a1k}\mathsf{D}_k + \sum_k s_{a2k}\sum_j \left(Y^{-\alpha+2\gamma+\zeta-\kappa}\mathsf{D}_k + Y^{-\alpha+2\gamma-\kappa}\mathsf{D}_k\mathsf{E}_k + Y^{-\alpha+2\gamma-\kappa+\epsilon}\mathsf{E}_k\right) + \\
& \textstyle\sum_k s_{a2k}\sum_j a_{kj}^s\left(u_j(X)Y^{-\alpha+3\gamma+\zeta-\kappa} + v_j(X)Y^{-\alpha+\gamma+\epsilon} + w_j(X)Y^{2\gamma-\alpha}\right) \ , \\
B(\boldsymbol{X}, Z) =& b_zZ + r_bY^{\alpha-2\gamma+\kappa} + v_b(X)Y^{\kappa-\gamma} + b_\zeta Y^\zeta + b_\delta Y^\delta + \textstyle\sum_k q_{bk}Q_{2k} + \sum_k s_{bk}\mathsf{E}_k \ ,
\end{aligned}
$$

where $v_b(X) = \sum_{j=1}^m b_j^* v_j(X)$ and $u_a^z(X) = \sum_{j=1}^m u_j^z u_j(X)$. (Again, $C(\boldsymbol{X}, Z)$ is defined like $A(\boldsymbol{X}, Z)$ by the substitution $a \to c$.) Since the prover is generic, the verifier's second verification guarantees that $A(\boldsymbol{X}, Z) = r_aY^\alpha + u_a(X)Y^\gamma$ for some r_a and $u_a(X) = \sum a_j^* u_j(X)$.

At this point, since $A(\boldsymbol{X}, Z)$, $B(\boldsymbol{X}, Z)$, and $C(\boldsymbol{X}, Z)$ are different from the case of knowledge-soundness proof, we will have a new set of critical coefficients. Let $\tilde{a}_j = a_j - \sum_k s_{c2k}c_{kj}^s$ for $j \leq m_0$ and $\tilde{a}_j = c_j^*$ for $j > m_0$. Let $u(X) = \sum_{j=1}^m \tilde{a}_j u_j(X)$, $v(X) = \sum_{j=1}^m \tilde{a}_j v_j(X)$, and $w(X) = \sum_{j=1}^m \tilde{a}_j w_j(X)$. Let $R' = \{i = Y^{j_0}\mathsf{D}_{k_1}^{j_1}\mathsf{E}_{k_1}^{j_2}\mathsf{E}_{k_2}^{j_3} : V_i(\boldsymbol{X}^*) = 0\}$, where one takes into account that the first verification equation accepted and thus $A(\boldsymbol{X}) = r_aY^\alpha + u_a(X)Y^\gamma$. Let

$$S' = \{Y^{\gamma+\zeta}, Y^{-\gamma+\epsilon+\kappa}, Y^\kappa, Y^{\epsilon+\gamma}, Y^\zeta\mathsf{D}_{k_1}\}$$

be the new set of critical coefficients. We depict the critical coefficients $V_i(\boldsymbol{X}^*, Z)$ for $i \in S'$ of the resulting $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ in Fig. 3. Let $\bar{S}' = R' \setminus S'$.

Theorem 2. *(1) Assume $\boldsymbol{\Delta} = (\alpha, \gamma, \ldots)$ is such that $S' \cap \bar{S}' = \emptyset$. Then $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ in Fig. 1 is NBBASE in the Sub-GBGM.*
(2) $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is perfectly zero-knowledge.

Proof (Sketch). **(1: knowledge-soundness)** Let $\mathcal{A}$ be a NBBASE adversary that has succeeded in outputting (x, π) such that $\mathsf{x} \notin \mathcal{L}$ but V accepts. Since the proof is in the generic model, the $\mathbb{G}_\iota$-outputs of $\mathcal{A}$ have to belong to the span of her inputs (one part of the input is the CRS) and moreover, one can extract the corresponding coordinates. . Since we are proving NBBASE, another part of the input to $\mathcal{A}$ is the reply of the Sim oracle for each query. 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. Since we have the Sub-GBGM model, we

give $\mathcal{A}$ the power to create new indeterminates $Q_{\iota i}$ in $\mathbb{G}_\iota$ (see Section 2 for the posed restrictions on the latter). Let $\boldsymbol{X} = (X, \boldsymbol{Q}, \mathbf{D}, \mathbf{E}, Y, Z)$ be a vector of all indeterminates.

Since the second verification equation holds, we get $A(\boldsymbol{X}, Z) = r_a Y^\alpha + u_a(X) Y^\delta$ for known r_a and $u_a(X)$. Consider now the verification polynomial $V(\boldsymbol{X}^*, Z)$ corresponding to the first verification equation, but assuming that $A(\boldsymbol{X}, Z) = r_a Y^\alpha + u_a(X) Y^\delta$. We now equate all five polynomials $V_i(\boldsymbol{X}^*, Z)$ in Fig. 3, $i \in S'$, to zero. First, $V_{Y^{\epsilon+\zeta}}(\boldsymbol{X}^*, Z) = b_\zeta = 0$. Second, for any k_1, $V_{Y^\zeta \mathsf{D}_{k_1}}(\boldsymbol{X}^*, Z) = -s_{c2k_1} = 0$. Thus $\tilde{a}_j = a_j$ for $j \le m_0$. Finally, $u_a^z(X) = u(X)$ (since $V_{Y^{\gamma+\zeta}} = 0$), $v_b(X) = v(X)$ (since $V_{Y^{-\gamma+\epsilon+\kappa}} = 0$), and $\chi(X) = u(X)v(X) - w(X) - \ell(X)h(X) = 0$ (since $V_{Y^\kappa} = 0$) and thus $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is NBBASE.

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

$\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is not NBBSASE-secure. It is easy to show $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ is not NBBASE-secure by showing how one can easily modify an accepting argument to another argument for the same statement. Given an accepting argument $\pi = ([\mathsf{a}, \mathsf{a}_z, \mathsf{c}_s]_1, [\mathsf{b}]_2)$, one can generate another accepting argument $\pi' = ([\mathsf{a}', \mathsf{a}'_z, \mathsf{c}'_s]_1, [\mathsf{b}']_2)$ by using at least one of the two following strategies [GM17]:

1. for random r, $[\mathsf{a}']_1 \leftarrow r[\mathsf{a}]_1 + (r-1)[y^\epsilon]_1$, $[\mathsf{a}'_z]_1 \leftarrow r[\mathsf{a}_z]_1 + (r-1)[y^\epsilon z]_1$, $[\mathsf{b}']_2 \leftarrow 1/r[\mathsf{b}]_2 + (1/r-1)[y^\zeta]_2$, $[\mathsf{c}'_s]_1 \leftarrow [\mathsf{c}_s]_1$. Note that this is an attack against NBBASE, not only NBBSASE. Thus, as expected, this attack does not work in the case of $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ since due to the second verification accepting, y^ϵ has a 0 coefficient in $[\mathsf{a}]_1$.
2. for random r, $[\mathsf{a}']_1 \leftarrow [\mathsf{a}]_1$, $[\mathsf{a}'_z]_1 \leftarrow [\mathsf{a}_z]_1$, $[\mathsf{b}']_2 \leftarrow [\mathsf{b}]_2 + r[y^{\alpha-2\gamma+\kappa}]_2$, $[\mathsf{c}'_s]_1 \leftarrow [\mathsf{c}_s]_1 + r([\mathsf{a}]_1 + [y^\epsilon]_1)$.

Clearly, the second attack is feasible also in the case of $\mathsf{S}^{\mathsf{se}}_{\mathsf{qap}}$ and thus $\mathsf{S}_{\mathsf{qap}}$ is not NBBSASE. However, in many applications, NBBASE (or even NBBTSE) is sufficient or even desirable [DHLW10]. In particular, (non-black-box) strong any-simulation-extractability is not required to achieve UC-security: any-simulation-extractability and even true-simulation-extractability.

4 SAP-Based SNARKs

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

Groth [Gro16] and Groth and Maller [GM17] used SAP (Square Arithmetic Programs) instead of QAP. The only distinction here is that it is assumed that $u(X) = v(X)$. This means that each gate in the arithmetic circuit gets the same left and right inputs, or, putting it another way, the circuit consists of squaring gates only. Since each multiplication gate $c = ab$ can be implemented two squaring gates ($ab = (a/2+b/2)^2 - (a/2-b/2)^2$), one can verify the correctness of an arbitrary d-gate arithmetic circuit by transferring it to a circuit that has $m^* \le 2d$ squaring gates and then constructing a SNARK for SAP for the resulting circuit. The main motivation behind introducing SAP is that one can construct SNARK in a way that $A(X,Y) = B(X,Y)$, 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 Eq. (3) by setting $v(X) = u(X)$. Then $A(X,Y) = r_a Y^\alpha + u(X)Y^\gamma$ and $B(X,Y) = r_b Y^{\alpha-2\gamma+\kappa} + v(X)Y^{\kappa-\gamma}$. It makes sense to do additional simplifications so that $A(X,Y) = B(X,Y)$, by assuming that $r_a = r_b$ and $\kappa = 2\gamma$.

Thus, Eq. (5) simplifies to

$$\begin{aligned} A(X,Y) =& B(X,Y) = r_a Y^\alpha + u(X)Y^\gamma \ , \\ C(X,Y) =& (A(X,Y) + Y^\epsilon)(A(X,Y) + Y^\zeta) - Y^{\epsilon+\zeta} \end{aligned}$$

$\mathsf{K}_{\mathsf{crs}}(\mathsf{p})$: Sample $x, y \leftarrow\!\$\, \mathbb{Z}_p^*$, return $\mathsf{tc} \leftarrow (x, y)$. Let

$$\mathsf{crs}_{\mathsf{P}} = \left(\left[\begin{array}{l} y^{\alpha}, \{x^j y^{\gamma}\}_{j=0}^{n-1}, \{x^i \ell(x) y^{-\alpha+2\gamma}\}_{j=0}^{n-2}, y^{\epsilon}, \boxed{y^{\zeta}}, \\ \left\{ u_j(x) y^{-\alpha+\gamma+\epsilon} + \boxed{u_j(x) y^{-\alpha+\gamma+\zeta}} + w_j(x) y^{-\alpha+2\gamma} \right\}_{j=m_0+1}^{m} \end{array} \right]_1 , \left[y^{\alpha}, \{x^i y^{\gamma}\}_{i=0}^{n-1} \right]_2 \right) ,$$

$$\mathsf{crs}_{\mathsf{V}} = \left(\left[\left\{ u_j(x) y^{-\delta+\gamma+\epsilon} + \boxed{u_j(x) y^{-\delta+\gamma+\zeta}} + w_j(x) y^{-\delta+2\gamma} \right\}_{j=1}^{m_0}, y^{\epsilon} \right]_1 , \left[y^{\alpha}, \boxed{y^{\zeta}}, y^{\delta} \right]_2 , \boxed{[y^{\epsilon+\zeta}]_T} \right) ;$$

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

$\mathsf{P}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, (a_j)_{j=m_0+1}^{m})$: $(r_a, r_b) \leftarrow\!\$\, \mathbb{Z}_p^2$; $u(X) \leftarrow \sum_{j=1}^{m} a_j u_j(X)$; $w(X) \leftarrow \sum_{j=1}^{m} a_j w_j(X)$; $h(X) \leftarrow (u(X)^2 - w(X))/\ell(X)$; $[\mathsf{a}]_1 \leftarrow r_a[y^{\alpha}]_1 + [u(x)y^{\gamma}]_1$; $[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^{m} a_j \left[u_j(x) y^{-\alpha+\gamma+\epsilon} + \boxed{u_j(x) y^{-\alpha+\gamma+\zeta}} + w_j(X) y^{-\alpha+2\gamma} \right]_1 + [h(x)\ell(x)y^{-\alpha+2\gamma}]_1 + r_a \left(r_a[y^{\alpha}]_1 + 2[u(x)y^{\gamma}]_1 + [y^{\epsilon}]_1 + \boxed{[y^{\zeta}]_1} \right)$; $[\mathsf{b}]_2 \leftarrow r_a[y^{\alpha}]_2 + [u(x)y^{\gamma}]_2$; return $\pi \leftarrow \left([\mathsf{a}, \mathsf{c}_s]_1 , [\mathsf{b}]_2 \right)$;

$\mathsf{V}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, \pi = \left([\mathsf{a}, \mathsf{c}_s]_1 , [\mathsf{b}]_2 \right))$: Set $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j \left[\boxed{2} u_j(x) Y^{-\delta+\gamma+\epsilon} + w_j(x) Y^{-\delta+\kappa} \right]_1$. Check that $[\mathsf{a} + y^{\epsilon}]_1 \bullet \left[\mathsf{b} \boxed{+ y^{\zeta}} \right]_2 = [\mathsf{c}_p]_1 \bullet [y^{\delta}]_2 + [\mathsf{c}_s]_1 \bullet [y^{\alpha}]_2 \boxed{+ [y^{\epsilon+\zeta}]_T}$ and $[\mathsf{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\mathsf{b}]_2$;

$\mathsf{Sim}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, \mathsf{tc} = y)$: $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j \left[u_j(x) y^{-\delta+\gamma+\epsilon} + \boxed{u_j(x) y^{-\delta+\gamma+\zeta}} + w_j(x) y^{-\delta+2\gamma} \right]_1$; $d \leftarrow\!\$\, \mathbb{Z}_p$; $[\mathsf{a}]_1 \leftarrow d[1]_1$; $[\mathsf{b}]_2 \leftarrow d[1]_2$; $[\mathsf{c}_s]_1 \leftarrow y^{-\alpha} \left(\left(d^2 + \boxed{dy^{\zeta} +} dy^{\epsilon} \right) [1]_1 - y^{\delta} [\mathsf{c}_p]_1 \right)$; return $\pi \leftarrow \left([\mathsf{a}, \mathsf{c}_s]_1 , [\mathsf{b}]_2 \right)$;

Fig. 4. The new SNARKs for SAP: knowledge-sound $\mathsf{S}_{\mathsf{sap}}$ (*with* boxed entries and *without* gray entries) and NBBASE $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ (*with* gray entries and *without* boxed entries)

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

We now construct the SNARK $\mathsf{S}_{\mathsf{sap}}$ by doing corresponding simplifications in Fig. 1, see Fig. 4. 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 $1 + (m - m_0) + (n-1) = n + m - m_0$ additional exponentiations to compute $[\mathsf{c}_s]_1$.

We can find a suitable $\boldsymbol{\Delta}$ by using the same approach as in the case of QAP in Section 3, but limiting the exhaustive search by setting $\kappa = 2\gamma$. In particular, one can set

$$\alpha = 3, \;\; \gamma = 4, \;\; \delta = 2, \;\; \epsilon = 0, \;\; \zeta = -5, \;\; \kappa = 8 \;\; .$$

In this case,

$$\mathsf{crs}_{\mathsf{P}} = \left(\left[\begin{array}{l} y^3, \{x^j y^4\}_{j=0}^{n-1}, \{x^i \ell(x) y^5\}_{j=0}^{n-2}, y^0, \boxed{y^{-5}}, \\ \left\{ u_j(x) y^1 + \boxed{u_j(x) y^{-4}} + w_j(x) y^5 \right\}_{j=m_0+1}^{m} \end{array} \right]_1 , \left[y^3, \{x^i y^4\}_{i=0}^{n-1} \right]_2 \right) ,$$

$$\mathsf{crs}_{\mathsf{V}} = \left(\left[\left\{ u_j(x) y^2 + \boxed{u_j(x) y^{-3}} + w_j(x) y^6 \right\}_{j=1}^{m_0}, y^0 \right]_1 , \left[y^3, \boxed{y^{-5}}, y^2 \right]_2 , \boxed{[y^{-5}]_T} \right) .$$

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, taking into account that

Y^i	Coefficient $V_i(X, \boldsymbol{Q})$ (KS and NBBASE)
$Y^{\gamma+\zeta}$	$-u(X) + (b_\zeta + 1)u_a(X) + a_\zeta u_b(X)$
$Y^{-\gamma+\epsilon+\kappa}$	$-u(X) + (a_\epsilon + 1)v_b(X)$
Y^{κ}	$u_a(X)u_b(X) - w(X) - h(X)\ell(X)$
$Y^{\epsilon+\zeta}$	$b_\zeta + a_\epsilon(b_\zeta + 1)$
$Y^{\delta+\epsilon}$	$(a_\epsilon + 1)b_\delta$
$Y^{2\zeta}$	$a_\zeta(b_\zeta + 1)$

Fig. 5. Critical coefficients in $\mathsf{S}_{\mathsf{sap}}$.

$\kappa = 2\gamma$. This will slightly change the sets R, S, and $\bar{S}$. Let $h(X) := h_c(X) - r_b h_a(X)$. Let $\tilde{a}_j = a_j - b_\delta a_j^*$ for $j \le m_0$ and $\tilde{a}_i = c_j^* - r_b a_j^*$ for $j > m_0$. Denote $u(X) = \sum_{j=1}^m \tilde{a}_j u_j(X)$ and $w(X) = \sum_{j=1}^m \tilde{a}_j w_j(X)$. In this case, the "significant" coefficients $V_i(X, \boldsymbol{Q})$, $i \in S$, of $V(\boldsymbol{X})$ are depicted in Fig. 5. The differences compared to Fig. 2 are solely due to the settings $\kappa = 2\gamma$ and $v(X) = u(X)$.

Corollary 1. *(1) Assume $\boldsymbol{\Delta}$ is chosen so that $S \cap \bar{S} = \emptyset$. Then $\mathsf{S}_{\mathsf{sap}}$ in Fig. 1 is knowledge-sound in the Sub-GBGM.*
(2) $\mathsf{S}_{\mathsf{sap}}$ is perfectly zero-knowledge.

Proof (Sketch). **(1: knowledge-soundness)** Consider the polynomials in Fig. 5. Since $b_\zeta + a_\epsilon(b_\zeta + 1) = 0$, we get $a_\epsilon = -b_\zeta/(b_\zeta + 1)$ and $a_\epsilon, b_\zeta \neq -1$ and $(a_\epsilon + 1)(b_\zeta + 1) = 1$. Thus $a_\zeta = b_\delta = 0$, which means that $\tilde{a}_j = a_j$ for $j \le m_0$. 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 e due to the fact that $\mathsf{a} = \mathsf{b}$. □

NBBASE SNARK $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$. Obtaining NBBASE in this case is simpler than in Section 3: to guarantee that $A(\boldsymbol{X})$ has no dependency on D_k, one can peruse the fact that $A(\boldsymbol{X}) = B(\boldsymbol{X})$ in the honest case. Thus, to achieve NBBASE, one will not have to rely on a new variable Z: the main change compared to the knowledge-sound version $\mathsf{S}_{\mathsf{sap}}$ is that the verifier now additionally checks that $[\mathsf{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\mathsf{b}]_2$. Adding this verification equation (that efficiently establishes that $A(\boldsymbol{X}) = B(\boldsymbol{X}) = r_a Y^\alpha + u(X)Y^\delta + a_\zeta Y^\zeta$ use the same witness) means that we can now omit the term Y^ζ from the definition of $C(\boldsymbol{X})$, slightly simplifying the verifier's work.[2] That is, $C(\boldsymbol{X}) = (A(\boldsymbol{X}) + Y^\epsilon)A(\boldsymbol{X})$. This also means that there is no element $[y^\zeta]_1$ in the CRS, meaning that $A(X, Y) = B(X, Y) = r_a Y^\alpha + u(X)Y^\gamma$. See Fig. 4 for a complete description of $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$. The exclusion of $\boxed{\text{boxed}}$ entries (corresponding to the described optimization) cut down a bit from the cost of the $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ as compared to $\mathsf{S}_{\mathsf{sap}}$.

The CRS has $(n) + (n-1) + m + 2 = 2n + m + 1$ elements from $\mathbb{G}_1$ and $n + 2$ 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 $1 + (m - m_0) + (n-1) = n + m - m_0$ additional exponentiations to compute $[\mathsf{c}]_1$.

5 SSP-Based SNARKs

In this section, we will build two SNARKs for SSP (Square Span Programs, [DFGK14]), one ($\mathsf{S}_{\mathsf{ssp}}$) being knowledge-sound and another one ($\mathsf{S}_{\mathsf{ssp}}^{\mathsf{se}}$) being NBBASE. 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:

– For each wire, one checks that the wire value is Boolean.

[2] This term Y^ζ was used to make $C(\boldsymbol{X})$ to linearly depend on $A(\boldsymbol{X})Y^\zeta$ and $B(\boldsymbol{X})Y^\epsilon$, which was then used to verify that $A(\boldsymbol{X})$ and $B(\boldsymbol{X})$ use the same witness $\boldsymbol{a}$. Here, $A(\boldsymbol{X}) = B(\boldsymbol{X})$ and thus such a verification is unnecessary.

– 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 3. In the case of SSP, the only difference in the language is that $u(X) = v(X) = w(X)$, and thus the key equation become $\chi_{ssp}(X) = 0$, 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.

Modifying $u(X) = v(X) = w(X)$ in Eq. (3), we get $A(X,Y) = r_a Y^{\alpha} + u(X)Y^{\gamma}$ and $B(X,Y) = r_b Y^{\beta} + v(X)Y^{\kappa-\gamma} = r_b Y^{\alpha-2\gamma+\kappa} + u(X)Y^{\kappa-\gamma}$. We can simplify the SNARK construction by assuming that $A(X,Y) = B(X,Y)$, that is, $r_a = r_b$ and $\kappa = 2\gamma$. Thus, Eq. (5) simplifies to

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

We now construct the SNARK $\mathsf{S}_{\mathsf{ssp}}$ by doing corresponding simplifications in Fig. 1, see Fig. 6. Thus the CRS length is $(m+2n+1)|\mathbb{G}_1|+(n+2)|\mathbb{G}_2|$. The prover's computation is m *multiplications* (since a_j are Boolean!) and 1 exponentiation to compute $[\mathsf{a}]_1$, m *multiplications* and 1 exponentiation to compute $[\mathsf{b}]_2$, and $m-m_0$ *multiplications* and $1+(n-1) = n$ exponentiations to compute $[\mathsf{c}_s]_1$. However, as explained in [DFGK14], one can speed up the computation of $[h(x)\ell(x)y^{-\alpha+2\gamma}]_1$ by using FFT. The verifier executes $\approx m_0$ multiplications in $\mathbb{G}_1$ and 3 pairings.

We can find a suitable $\boldsymbol{\Delta}$ by using the same approach as in the case of QAP in Section 3, but limiting the exhaustive search by setting $\kappa = 2\gamma$. In particular, one can set (as in Section 4)

$$\alpha = 3, \ \ \gamma = 4, \ \ \delta = 2, \ \ \epsilon = 0, \ \ \zeta = -5, \ \ \kappa = 8 \ .$$

Knowledge-soundness. Since the resulting SNARK $\mathsf{S}_{\mathsf{ssp}}$ is an optimized version of $\mathsf{S}_{\mathsf{qap}}$, it can proven knowledge-sound by using the same basic idea. That is, one can follow the approach of Section 3 but by taking into account that $\kappa = 2\gamma$. This will slightly change the sets R, S, and $\bar{S}$ but the rest of the analysis stays the same. Let $h(X) := h_c(X) - r_b h_a(X)$. Let $\tilde{a}_j = a_j - b_\delta a_j^*$ for $j \le m_0$ and $\tilde{a}_i = c_j^* - r_b a_j^*$ for $j > m_0$. Denote $u(X) = \sum_{j=1}^{m} \tilde{a}_j u_j(X)$. In this case, the "significant" coefficients $V_i(X, \boldsymbol{Q})$, $i \in S$, of $V(\boldsymbol{X})$ are depicted in Fig. 7. The differences compared to Fig. 2 are solely due to the settings $\kappa = 2\gamma$ and $w(X) = v(X) = u(X)$.

Corollary 2. *(1) Assume $\boldsymbol{\Delta}$ is chosen so that $S \cap \bar{S} = \emptyset$. Then $\mathsf{S}_{\mathsf{ssp}}$ in Fig. 1 is knowledge-sound in the Sub-GBGM.*
(2) $\mathsf{S}_{\mathsf{ssp}}$ is perfectly zero-knowledge.

$\mathsf{K}_{\mathsf{crs}}(\mathsf{p})$: Sample $x, y \leftarrow\!\$\, \mathbb{Z}_p^*$, return $\mathsf{tc} \leftarrow (x, y)$. Let

$$\mathsf{crs}_{\mathsf{P}} = \left(\begin{bmatrix} y^\alpha, \{x^j y^\gamma\}_{j=0}^{n-1}, \{x^i \ell(x) y^{-\alpha+2\gamma}\}_{i=0}^{n-2}, y^\epsilon, \boxed{y^\zeta}, \\ \left\{ u_j(x) y^{-\alpha+\gamma+\epsilon} + \boxed{u_j(x) y^{-\alpha+\gamma+\zeta}} + u_j(x) y^{\alpha-2\gamma} \right\}_{j=m_0+1}^{m} \end{bmatrix}_1, \left[y^\alpha, \{x^i y^\gamma\}_{i=0}^{n-1} \right]_2 \right),$$

$$\mathsf{crs}_{\mathsf{V}} = \left(\left[\left\{ u_j(x) y^{-\delta+\gamma+\epsilon} + \boxed{u_j(x) y^{-\delta+\gamma+\zeta}} + u_j(x) y^{-\delta+2\gamma} \right\}_{j=1}^{m_0}, y^\epsilon \right]_1, \left[y^\alpha, \boxed{y^\zeta}, y^\delta \right]_2, \boxed{[y^{\epsilon+\zeta}]_T} \right);$$

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

$\mathsf{P}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, (a_j)_{j=m_0+1}^{m})$: $(r_a, r_b) \leftarrow\!\$\, \mathbb{Z}_p^2$; $u(X) \leftarrow \sum_{j=1}^m a_j u_j(X)$; $h(X) \leftarrow (u(X)^2 - u(X))/\ell(X)$; $[\mathsf{a}]_1 \leftarrow r_a [y^\alpha]_1 + \sum_{j=1}^m a_j [u_j(x) y^\gamma]_1$; $[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^m a_j \left[u_j(x) y^{-\alpha+\gamma+\epsilon} + \boxed{u_j(x) y^{-\alpha+\gamma+\zeta}} + u_j(X) y^{-\alpha+2\gamma} \right]_1 + [h(x)\ell(x) y^{-\alpha+2\gamma}]_1 + r_a \left(r_a [y^\alpha]_1 + 2 \sum_{j=1}^m a_j [u_j(x) y^\gamma]_1 + [y^\epsilon]_1 + \boxed{[y^\zeta]_1} \right)$; $[\mathsf{b}]_2 \leftarrow r_a [y^\alpha]_2 + \sum_{j=1}^m a_j [u_j(x) y^\gamma]_2$; return $\pi \leftarrow \left([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2 \right)$;

$\mathsf{V}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, \pi = \left([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2 \right))$: Set $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j \left[u_j(x) Y^{-\delta+\gamma+\epsilon} + \boxed{u_j(x) Y^{-\delta+\gamma+\zeta}} + u_j(x) Y^{-\delta+\kappa} \right]_1$. Check that $[\mathsf{a} + y^\epsilon]_1 \bullet \left[\mathsf{b} \boxed{+ y^\zeta} \right]_2 = [\mathsf{c}_p]_1 \bullet [y^\delta]_2 + [\mathsf{c}_s]_1 \bullet [y^\alpha]_2 \boxed{+ [y^{\epsilon+\zeta}]_T}$ and $[\mathsf{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\mathsf{b}]_2$;

$\mathsf{Sim}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, \mathsf{tc} = y)$: $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j \left[u_j(x) y^{-\delta+\gamma+\epsilon} + \boxed{u_j(x) y^{-\delta+\gamma+\zeta}} + u_j(x) y^{-\delta+2\gamma} \right]_1$; $d \leftarrow\!\$\, \mathbb{Z}_p$; $[\mathsf{a}]_1 \leftarrow d[1]_1$; $[\mathsf{b}]_2 \leftarrow d[1]_2$; $[\mathsf{c}_s]_1 \leftarrow y^{-\alpha} \left(\left(d^2 + \boxed{dy^\zeta +} dy^\epsilon \right) [1]_1 - y^\delta [\mathsf{c}_p]_1 \right)$; return $\pi \leftarrow \left([\mathsf{a}, \mathsf{c}_s]_1, [\mathsf{b}]_2 \right)$;

Fig. 6. The new SNARKs for SSP: knowledge-sound $\mathsf{S}_{\mathsf{ssp}}$ (*with* boxed entries and *without* gray entries) and NBBASE $\mathsf{S}_{\mathsf{ssp}}^{\mathsf{se}}$ (*with* gray entries and *without* boxed entries)

Y^i	Coefficient $V_i(X, \boldsymbol{Q})$ (KS and NBBASE)
$Y^{\gamma+\zeta}$	$-u(X) + (b_\zeta + 1) u_a(X) + a_\zeta u_b(X)$
$Y^{-\gamma+\epsilon+\kappa}$	$-u(X) + (a_\epsilon + 1) v_b(X)$
Y^κ	$u_a(X) u_b(X) - u(X) - h(X)\ell(X)$
$Y^{\epsilon+\zeta}$	$b_\zeta + a_\epsilon (b_\zeta + 1)$
$Y^{\delta+\epsilon}$	$(a_\epsilon + 1) b_\delta$
$Y^{2\zeta}$	$a_\zeta (b_\zeta + 1)$

Fig. 7. Critical coefficients in $\mathsf{S}_{\mathsf{sap}}$.

Proof (Sketch). **(1: knowledge-soundness)** Consider the polynomials in Fig. 7. Since $b_\zeta + a_\epsilon (b_\zeta + 1) = 0$, we get $a_\epsilon = -b_\zeta/(b_\zeta + 1)$ and $a_\epsilon, b_\zeta \neq -1$ and $(a_\epsilon + 1)(b_\zeta + 1) = 1$. Thus $a_\zeta = b_\delta = 0$, which means that $\tilde{a}_j = a_j$ for $j \leq m_0$. Thus, $u_a(X) v_b(X) = u(X)^2$ and $u(X)^2 - u(X) = h(X)\ell(X)$, which means that $\chi_{ssp}(X) = 0$.

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

NBBASE SNARK $\mathsf{S}_{\mathsf{ssp}}^{\mathsf{se}}$. To guarantee that $A(X, Y)$ has no dependency on D_k, one can peruse the fact that $A(X, Y) = B(X, Y)$ in the honest case. Thus, to achieve NBBASE, one will not have to rely on a new variable Z: the only changes compared to $\mathsf{S}_{\mathsf{ssp}}$ are that (i) the verifier additionally checks that $[\mathsf{a}]_1 \bullet [1]_2 = [1]_1 \bullet [\mathsf{b}]_2$, and (ii) there is no term y^ζ in the verification equation. Alternatively, $\mathsf{S}_{\mathsf{ssp}}^{\mathsf{se}}$ is as $\mathsf{S}_{\mathsf{sap}}^{\mathsf{se}}$ in Section 4 except $w_j(X)$ is always replaced by $u_j(X)$.

6 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

$\mathsf{K}_{\mathsf{crs}}(\mathsf{p})$: Sample $x, y, z \leftarrow_\$ \mathbb{Z}_p^*$, return $\mathsf{tc} \leftarrow (x, y, z)$. Let

$$\mathsf{crs}_\mathsf{P} \leftarrow \left(\begin{bmatrix} y^\alpha, \{x^j y^\gamma\}_{j=0}^{n-1}, y^\alpha z, \{x^j y^\gamma z\}_{j=0}^{n-1}, \{x^i \ell(x) y^{-\alpha+2\gamma}\}_{j=0}^{n-2}, y^\epsilon, y^{2\gamma+\zeta-\kappa}, \\ \{u_j(x) y^{-\alpha+3\gamma+\zeta-\kappa} + v_j(x) y^{-\alpha+\gamma+\epsilon}\}_{j=m_0+1}^{m} \end{bmatrix}_1, \left[y^{\alpha-2\gamma+\kappa}, \{x^i y^{\kappa-\gamma}\}_{i=0}^{n-1} \right]_2 \right) ;$$

$$\mathsf{crs}_\mathsf{V} \leftarrow \left([\{u_j(x) y^{-\delta+\gamma+\zeta} + v_j(x) y^{-\delta-\gamma+\kappa+\epsilon}\}_{j=1}^{m_0}, y^\epsilon]_1, \left[y^{\alpha-2\gamma+\kappa}, y^\zeta, y^\delta, z \right]_2, [y^{\epsilon+\zeta}]_T \right) ;$$

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

$\mathsf{P}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, (a_j)_{j=m_0+1}^{m})$: $(r_a, r_b) \leftarrow_\$ \mathbb{Z}_p^2$; $u(X) \leftarrow \sum_{j=1}^m a_j u_j(X)$; $v(X) \leftarrow \sum_{j=1}^m a_j v_j(X)$; $h(X) \leftarrow u(X)v(X)/\ell(X)$; $[\mathsf{a}]_1 \leftarrow r_a[y^\alpha]_1 + \sum_{j=1}^m a_j[u_j(x)y^\gamma]_1$; $[\mathsf{a}_z]_1 \leftarrow r_a[y^\alpha z]_1 + \sum_{j=1}^m a_j[u_j(x)y^\gamma z]_1$; $[\mathsf{c}_s]_1 \leftarrow \sum_{j=m_0+1}^m a_j[u_j(x)y^{-\alpha+3\gamma+\zeta-\kappa} + v_j(x)y^{-\alpha+\gamma+\epsilon}]_1 + [h(x)\ell(x)y^{-\alpha+2\gamma}]_1 + r_b\left(r_a[y^\alpha]_1 + \sum_{j=1}^m a_j[u_j(x)y^\gamma]_1 + [y^\epsilon]_1\right) + r_a([y^{2\gamma-\kappa+\zeta}]_1 + \sum_{j=1}^m a_j[v_j(x)y^\gamma]_1)$; $[\mathsf{b}]_2 \leftarrow r_b[y^{\alpha-2\gamma+\kappa}]_2 + [v(x)y^{\kappa-\gamma}]_2$; return $\pi \leftarrow \left([\mathsf{a}, \mathsf{a}_z, \mathsf{c}_s]_1, [\mathsf{b}]_2\right)$;

$\mathsf{V}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, \pi = \left([\mathsf{a}, \mathsf{a}_z, \mathsf{c}_s]_1, [\mathsf{b}]_2\right))$: Set $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j[u_j(x)y^{-\delta+\gamma+\zeta} + v_j(x)y^{-\delta-\gamma+\kappa+\epsilon}]_1$. Check that $[\mathsf{a} + y^\epsilon]_1 \bullet [\mathsf{b} + y^\zeta]_2 = [\mathsf{c}_p]_1 \bullet [y^\delta]_2 + [\mathsf{c}_s]_1 \bullet [y^{\alpha-2\gamma+\kappa}]_2 + [y^{\zeta+\epsilon}]_T$ and $[\mathsf{a}]_1 \bullet [z]_2 = [\mathsf{a}_z]_1 \bullet [1]_2$;

$\mathsf{Sim}(\mathsf{p}, \mathsf{crs}; (a_j)_{j=1}^{m_0}, \mathsf{tc} = y)$: $[\mathsf{c}_p]_1 \leftarrow \sum_{j=1}^{m_0} a_j[u_j(x)y^{-\delta+\gamma+\zeta} + v_j(x)y^{-\delta-\gamma+\kappa+\epsilon}]_1$; $d \leftarrow_\$ \mathbb{Z}_p$; $e \leftarrow_\$ \mathbb{Z}_p$; $[\mathsf{a}]_1 \leftarrow d[1]_1$; $[\mathsf{a}_z]_1 \leftarrow dy^{-\alpha}[y^\alpha z]_1$; $[\mathsf{b}]_2 \leftarrow e[1]_2$; $[\mathsf{c}_s]_1 \leftarrow y^{-\beta}((de + dy^\zeta + ey^\epsilon)[1]_1 - y^\delta[\mathsf{c}_p]_1)$; return $\pi \leftarrow \left([\mathsf{a}, \mathsf{a}_z, \mathsf{c}_s]_1, [\mathsf{b}]_2\right)$;

Fig. 8. The new SNARKs for QSP: knowledge-sound $\mathsf{S}_{\mathsf{qsp}}$ (*without* gray entries) and NBBASE $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$ (*with* gray entries)

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, an SSP has size $\approx m \times (m+n)$ while a QSP has size $\approx 14n \times 11n$.) However, QSP-based solutions like the SSP-based solutions have a short argument and CRS.

In this section, we assume that one has already constructed a reduction to the QSP, and given that, we propose several updatable QSP-based SNARKs. 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

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

Thus, here, Eq. (4) simplifies to

$$\begin{aligned} A(X,Y) =& r_a Y^\alpha + u(X)Y^\gamma \ , \\ B(X,Y) =& r_b Y^{\alpha-2\gamma+\kappa} + v(X)Y^{\kappa-\gamma} \ , \\ C_p(X,Y) =& \textstyle\sum_{j=1}^{m_0} a_j \left(u_j(X)Y^{-\delta+\gamma+\zeta} + v_j(X)Y^{-\delta-\gamma+\kappa+\epsilon}\right) \ , \\ C_s(X,Y) =& \textstyle\sum_{j=m_0+1}^{m} a_j \left(u_j(X)Y^{-\alpha+3\gamma+\zeta-\kappa} + v_j(X)Y^{-\alpha+\gamma+\epsilon}\right) + u(X)v(X)Y^{-\alpha+2\gamma} + \\ & r_b\left(A(X,Y) + Y^\epsilon\right) + r_a(v(X)Y^\gamma + Y^{2\gamma-\kappa+\zeta}) \ . \end{aligned} \tag{8}$$

Clearly, $C_s(X,Y)Y^{\alpha-2\gamma+\kappa} + C_p(X,Y)Y^\zeta = C(X,Y)$.

We now construct the SNARK $\mathsf{S}_{\mathsf{qsp}}$ by doing corresponding simplifications in Fig. 1, see Fig. 8. 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]).

The prover's computation is m *multiplications* (since a_j are Boolean!) and 1 exponentiation to compute $[\mathsf{a}]_1$, m *multiplications* and 1 exponentiation to compute $[\mathsf{b}]_2$, and $2m - m_0$ *multiplications* and $(n-1)+3 = n+2$ exponentiations to compute $[\mathsf{c}_s]_1$.

We can find a suitable $\boldsymbol{\Delta}$ by using the same approach as in the case of QAP in Section 3. In particular, one can set (as in Section 3)

$$\alpha = 0, \gamma = -1, \epsilon = 6, \zeta = 0, \delta = 5, \kappa = 2 \ .$$

NBBASE SNARK $\mathsf{S}^{\mathsf{se}}_{\mathsf{qsp}}$. One obtains a NBBASE version of $\mathsf{S}_{\mathsf{qsp}}$ exactly as in the case of $\mathsf{S}_{\mathsf{qap}}$ in Section 3, by introducing a new indeterminate Z. (See Fig. 8.)

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

References

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

ALSZ18. Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michał Zając. QANIZK in the BPK Model. Technical Report 2018/877, IACR, September 18, 2018. Available from `https://eprint.iacr.org/2018/877`, updated version from 19 Feb 2019.

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

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

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

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.

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.

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

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.

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

BCTV14. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, Heidelberg, August 2014.

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

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.

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`.

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.

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.

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

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.

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.

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.

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.

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.

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.

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.

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

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

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

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

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.

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.

SPMS02. Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 93–110. Springer, Heidelberg, August 2002.

TK17. Mehdi Tibouchi and Taechan Kim. Improved elliptic curve hashing and point representation. *Des. Codes Cryptography*, 82(1–2):161–177, 2017.

A GBGM and Sub-GBGM

Generic Bilinear Group Model. Next, we will introduce the Generic Bilinear Group Model (GBGM) [Nec94,Sho97,Mau05,BBG05], by following the exposition in [ABLZ17].

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

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

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

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

Sub-GBGM. By following [SPMS02,BFS16,ABLZ17], we enhance the power of generic bilinear group model. Since the power of the generic adversary will increase, security proofs in the resulting *Sub-GBGM* are more realistic than in the GBGM, see Section 2.

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

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

The semantics of create dictates that the actual value of the indeterminate Y_i is uniformly random in $\mathbb{Z}_p$, that is, the adversary cannot create indeterminates for which she does not know the discrete logarithm and that yet are not random.

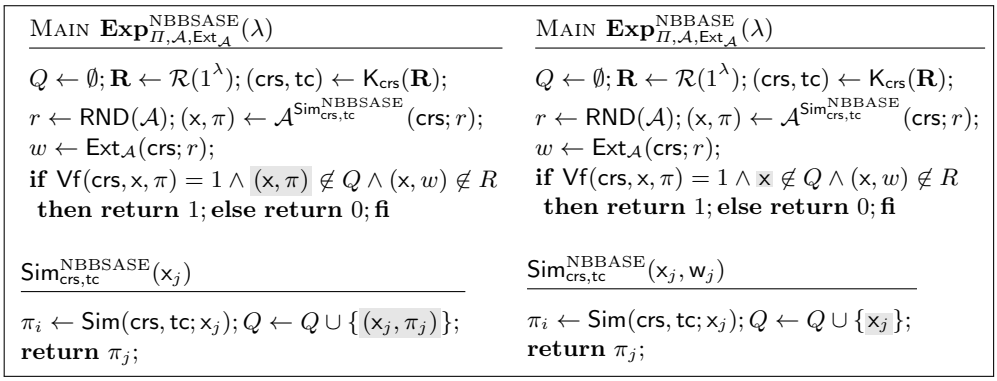

Fig. 9. NBBSE experiment: strong any-simulation (NBBSASE, left) and any-simulation (NBBASE, right). Differences are outlined in grey

B Formal Security Definitions

B.1 Zero-Knowledge

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

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

$$\Pr\left[(\mathsf{crs}, \mathsf{tc}) \leftarrow \mathsf{K}_{\mathsf{crs}}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{tc}) : \mathsf{V}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{x}, \mathsf{P}(\mathbf{R}, \mathsf{aux}_{\mathbf{R}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{x}, \mathsf{w})) = 1\right] = 1 \ .$$

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

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

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

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

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

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

B.2 Simulation-Extractability

Definition 4 (NBBSASE SNARK [GM17]). *Let $\Pi = (\mathsf{K}_{\mathsf{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be a SNARK for relation $\mathbf{R}$. Define $\mathsf{Adv}^{nbbsase}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) = \Pr[\mathbf{Exp}^{NBBSASE}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)]$, where the experiment $\mathbf{Exp}^{NBBSASE}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ is depicted in Fig. 9. Π is* non-black-box strong any-simulation-extractable (NBBSASE) *if for any PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$ such that $\mathsf{Adv}^{nbbsase}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \approx_\lambda 0$.*

Definition 5 (NBBASE SNARK). *Let* $\Pi = (\mathsf{K}_{\mathsf{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be a SNARK for relation* $\mathbf{R}$*. Define* $\mathsf{Adv}^{nbbase}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) = \Pr[\mathbf{Exp}^{NBBASE}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)]$*, where the experiment* $\mathbf{Exp}^{NBBASE}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda)$ *is depicted in Fig. 9.* Π *is* non-black-box any-simulation-extractable (NBBASE) *if for any PPT adversary* $\mathcal{A}$ *there exists a PPT extractor* $\mathsf{Ext}_{\mathcal{A}}$ *such that* $\mathsf{Adv}^{nbbase}_{\Pi,\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) \approx_\lambda 0$*.*