Succinct Functional Commitment for a Large Class of Arithmetic Circuits

Helger Lipmaa and Kateryna Pavlyk

Simula UiB, Bergen, Norway {helger,kateryna}@simula.no

Abstract. A succinct functional commitment (SFC) scheme for a circuit class $\mathbf{CC}$ enables, for any circuit $\mathcal{C} \in \mathbf{CC}$, the committer to first succinctly commit to a vector $\boldsymbol{\alpha}$, and later succinctly open the commitment to $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$, where the verifier chooses $\boldsymbol{\beta}$ at the time of opening. Unfortunately, SFC commitment schemes are known only for severely limited function classes like the class of inner products. By making non-black-box use of SNARK-construction techniques, we propose a SFC scheme for the large class of semi-sparse polynomials. The new SFC scheme can be used to, say, efficiently (1) implement sparse polynomials, and (2) aggregate various interesting SFC (e.g., vector commitment and polynomial commitment) schemes. The new scheme is evaluation-binding under a new instantiation of the computational uber-assumption. We provide a thorough analysis of the new assumption.

Keywords: aggregated functional commitment, Dejà Q, functional commitment, SNARK, uber-assumption, vector commitment

1 Introduction

A succinct functional commitment (SFC) scheme [29] for a circuit class $\mathbf{CC}$ enables the committer, for any $\mathcal{C} \in \mathbf{CC}$, to first commit to a vector $\boldsymbol{\alpha}$, and later open the commitment to $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$, where the verifier chooses $\boldsymbol{\beta}$ at the time of opening. An SFC scheme must be evaluation-binding (given a commitment, it is intractable to open it to $\boldsymbol{\xi} = \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $\boldsymbol{\xi}' = \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ for $\boldsymbol{\xi} \neq \boldsymbol{\xi}'$) and hiding (a commitment and possibly many openings should not reveal any additional information about $\boldsymbol{\alpha}$). Succinctness means that both the commitment and the opening have length $\mathsf{polylog}(|\boldsymbol{\alpha}|, |\boldsymbol{\beta}|)$.

In particular, an SFC scheme for inner products (SIPFC) assumes that, $\mathcal{C}$ computes the inner product $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \to \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$ [30,25,29]. As explained in [29], one can use an SIPFC scheme to construct succinct vector commitment schemes [12], polynomial commitment schemes [27], and accumulators [5]. Each of these primitives has a large number of independent applications. Succinct polynomial commitment schemes have recently become very popular since they can be used to construct (updatable) SNARKs [41,40,35,15] (a direction somewhat opposite to the one we will pursue in the current paper). Since, in several applications (e.g., in cryptocurrencies, [38]), one has to run many instances of SFC in parallel,

there is a recent surge of interest in aggregatable SFC schemes, [8,28,9,22,38]. All mentioned papers propose *succinct* FC schemes for limited functionalities.

Since there are no prior SFC schemes for broader classes of functions, there is a large gap between function classes for which an SFC scheme is known and the class of all efficiently (e.g., poly-size arithmetic circuits) verifiable functions. Filling a similar gap is notoriously hard in the case of related primitives like functional encryption, homomorphic encryption, and NIZK. A natural question to ask is whether something similar holds in the case of functional commitment.

It is easy to construct an SFC for all poly-size circuits under non-falsifiable assumptions: given a commitment to $\boldsymbol{\alpha}$, the opening consists of a SNARK argument [23,31,20] that $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \boldsymbol{\xi}$. However, while non-falsifiable assumptions are required to construct SNARKs [21], they are not needed in the case of SFC schemes. Thus, just using SNARK as a black-box is not a satisfactory solution.

Moreover, since one can construct non-succinct NIZK from falsifiable assumptions for NP, one can construct a non-succinct FC (nSFC) from a non-succinct NIZK. Bitansky [6] pursued this approach, proposing an nSFC, for all circuits, that uses NIZK as a black-box. By using NIWIs in a non-black-box manner, Bitansky proposed another, non-trivial, nSFC scheme that does not achieve zero-knowledge but does not require the CRS model. Alternatively, consider the FC scheme where the commitment consists of fully-homomorphic encryptions C_i of individual coefficients α_i, and the opening is the randomizer R of the evaluation of the circuit $\mathcal{C}$ on them. The verifier can re-evaluate the circuit on C_i and her input, and then check that the result is equal to $\mathsf{Enc}(\boldsymbol{\xi}; R)$. However, the resulting FC is not succinct since one has to encrypt all α_i individually.

Thus, the main question is to construct *succinct* FC schemes, under falsifiable assumptions, for a wide variety of functionalities.

Our Contributions. We propose a falsifiable SFC scheme $\mathsf{FC}_{\mathsf{sn}}$ for the class of *semi-sparse polynomials* $\mathbf{CC} = \mathbf{CC}_{\Sigma\Pi\forall}$ whose correct computation can be verified by using an arbitrary polynomial-size arithmetic circuit that is "compilable" according to the definition, given in a few paragraphs. Notably, $\mathsf{FC}_{\mathsf{sn}}$ allows efficiently aggregate various SFC schemes, e.g., vector commitments with inner-product commitments and polynomial commitments. We analyze the power of $\mathbf{CC}_{\Sigma\Pi\forall}$ by using techniques from algebraic complexity theory; the name of the class will be explain in Section 4.

We prove that $\mathsf{FC}_{\mathsf{sn}}$ is secure under a new falsifiable assumption (computational span-uber-assumption in a group $\mathbb{G}_1$) that is reminiscent of the well-known computational uber-assumption in $\mathbb{G}_1$. We then thoroughly analyze the security of the new assumption.

Our Techniques. Next, we provide a high-level overview of our technical contributions. The construction of $\mathsf{FC}_{\mathsf{sn}}$ consists of the following steps.

1. Compilation of the original circuit $\mathcal{C}$ computing the fixed function $\mathcal{F} \in \mathbf{CC}$ to a circuit $\mathcal{C}^*$ consisting of four public subcircuits.
2. Representation of $\mathcal{C}^*$ in the QAP language which SNARKs usually use.

3. Construction of SFC for the QAP representation, by using SNARK techniques in a non-black-box way.

Next we describe these steps in detail.

Circuit Compilation. Let $\mathcal{C} : \mathbb{Z}_p^{\mu_\alpha} \times \mathbb{Z}_p^{\mu_\beta} \to \mathbb{Z}_p^{\kappa}$ be a polynomial-size arithmetic circuit that, on input $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, outputs $\boldsymbol{\xi} = \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}))_{i=1}^{\kappa}$. Here, the committer's input $\boldsymbol{\alpha}$ is secret, and the verifier's input is public. We modify the circuit $\mathcal{C}$ to a compiled circuit $\mathcal{C}^*$, see Fig. 1, that consists of the subcircuits $\mathcal{C}_\phi$, $\mathcal{C}_\psi$, $\mathcal{C}_\chi$, and $\mathcal{C}_\xi$. In the commitment phase, the committer uses the circuit $\mathcal{C}_\phi$ to compute several polynomials $\phi_i(\boldsymbol{\alpha})$ depending on only 1 (this allows the output polynomials to have a non-zero constant term) and $\boldsymbol{\alpha}$. In the opening phase, the verifier sends $\boldsymbol{\beta}$ to the committer, who uses the circuit $\mathcal{C}_\psi$ to compute several polynomials $\psi_i(\boldsymbol{\beta})$ depending on 1 and $\boldsymbol{\beta}$. The verifier can redo this part of the computation. After that, the committer uses the circuit $\mathcal{C}_\chi$ to compute several polynomials $\chi_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$ from the inputs and outputs of $\mathcal{C}_\phi$ and $\mathcal{C}_\psi$. Finally, the committer uses $\mathcal{C}_\xi$ to compute the outputs $\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of $\mathcal{C}^*$. We will explain more thoroughly this compilation in Section 3.

Intuitively, the compilation restricts the class of circuits in two ways. First, we add a small circuit $\mathcal{C}_\xi$ at the top of the compiled circuit to guarantee that the R1CS representation of $\mathcal{C}^*$ has several all-zero columns and rows, which helps us in the security reduction. This does, however, not restrict the circuit class for which the SFC is defined and it only increases the number of gates by κ. Second, $\mathcal{C}_\chi$ is restricted to have multiplicative depth 1, i.e., it sums up products of polynomials in $\boldsymbol{\alpha}$ with polynomials in $\boldsymbol{\beta}$. This guarantees that in a collision, the two accepted openings have a linear relation that does not depend on secret data $\boldsymbol{\alpha}$. The latter makes it possible for the reduction to break the underlying falsifiable assumption. Thus, we are restricted to the class $\mathbf{CC}_{\Sigma\Pi\forall}$ of circuits where each output can be written as $\sum_{i,j} \phi_i(\boldsymbol{\alpha})\psi_j(\boldsymbol{\beta})$, for efficiently computable polynomials ϕ_i and ψ_j, and the sum is taken over number $\mathsf{poly}(\lambda)$ products.

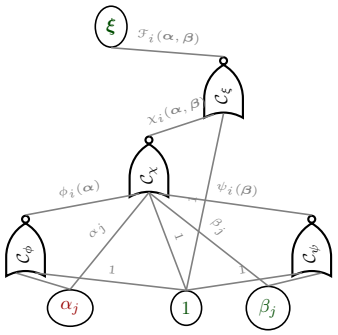


Fig. 1. The compiled circuit $\mathcal{C}^*$.

By employing tools from the algebraic complexity theory, in Section 4, we study the class $\mathbf{CC}_{\Sigma\Pi\forall}$ of "compilable" (according to the given definition) arithmetic circuits. We say that a polynomial $f \in \mathbf{CC}_{\Sigma\Pi\forall}$ if f has a circuit that belongs to $\mathbf{CC}_{\Sigma\Pi\forall}$. The new SFC scheme can implement f iff $f \in \mathbf{CC}_{\Sigma\Pi\forall}$. First, we show that any sparse polynomial (over indeterminates, chosen by both the committer and the verifier) f belongs to $\mathbf{CC}_{\Sigma\Pi\forall}$. Second, we construct a non-sparse polynomial $f \in \mathbf{CC}_{\Sigma\Pi\forall}$. This relies on a result of Ben-Or who constructed an $O(n^2)$-size arithmetic circuit that simultaneously computes the dth symmetric polynomial $\sigma_d(X_1, \ldots, X_n)$, for $d \in [1\,..\,n]$. Third, we construct a

polynomial $f \in \mathbf{VP}$ such that $f \notin \mathbf{CC}_{\Sigma\Pi\forall}$, where $\mathbf{VP}$ is the class of poly-degree polynomials that have poly-size circuits, [39].

R1CS/QAP Representation. Let $\mathcal{C}$ be an arithmetic circuit, and $\mathcal{C}^*$ be its compilation. A circuit evaluation can be verified by verifying a matrix equation, where matrices define the circuit uniquely and reflect all the circuit constraints. SNARKs usually use QAP (Quadratic Arithmetic Program, [20]), a polynomial version of R1CS, which allows for better efficiency.

Constructing the Underlying SNARK. Intuitively, we start constructing a SNARK for $\mathcal{C}^*$ by following the approach of Groth [24] who proposed the most efficient known zk-SNARK, or more precisely, its recent modification by Lipmaa [33]. However, we modify this approach whenever it suits our goals. The new SFC inherits the efficiency of Groth's SNARK; this is the main reason why we chose Groth's SNARK; it may be the case that SFCs constructed from less efficient SNARKs have other desirable properties, but this is out of the scope of the current paper. We chose the modified version of [33] due to its versatility: [33] explains sufficiently well how to construct a SNARK for QAP so that it is feasible to modify its approach to suit the current paper.

The New SFC Scheme. In the SNARKs of [24,33], the argument consists of three group elements, $\pi = ([\mathsf{A}]_1, [\mathsf{B}]_2, [\mathsf{C}]_1)$. (We use the bracket additive notation, see Section 2.) Due to our restrictions on $\mathcal{C}^*$, both $[\mathsf{A}]_1$ and $[\mathsf{B}]_2$ can be written as sums of a non-functional commitment that depends on the secret data only and a non-functional commitment that depends on public data only. By the public data we mean $(\boldsymbol{\beta}, \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta}))$; any other function of $\boldsymbol{\alpha}$ is a part of the secret data. E.g., $[\mathsf{A}]_1 = [\mathsf{A}_s]_1 + [\mathsf{A}_p]_1$, where $[\mathsf{A}_s]_1$ is computed by the committer before $\boldsymbol{\beta}$ becomes available, and $[\mathsf{A}_p]_1$ can be recomputed by the verifier since it only depends on the public data. However, $[\mathsf{C}]_1 = [\mathsf{C}_{sp}]_1 + [\mathsf{C}_p]_1$, where $[\mathsf{C}_p]_1$ depends only on public data but $[\mathsf{C}_{sp}]_1$ depends both on public and private data.

In the new SFC commitment scheme, the functional commitment is $C = ([\mathsf{A}_s]_1, [\mathsf{B}_s]_2)$ and the opening is $[\mathsf{C}_{sp}]_1$. After receiving the opening, the verifier recomputes $[\mathsf{A}_p]_1$, $[\mathsf{B}_p]_2$, and $[\mathsf{C}_p]_1$, and then runs the SNARK verifier on the argument $\pi = ([\mathsf{A}_s]_1 + [\mathsf{A}_p]_1, [\mathsf{B}_s]_2 + [\mathsf{B}_p]_2, [\mathsf{C}_{sp}]_1 + [\mathsf{C}_p]_1)$. However, as we will see later, the commitment also includes auxiliary elements $[\mathsf{B}_i^{\mathsf{aux}}]_1$ needed to obtain an efficient security reduction.

We will denote the new SFC commitment scheme by $\mathsf{FC}_{\mathsf{sn}}$. We denote by $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ its specialization to the circuit $\mathcal{C}$.

Applications. To demonstrate the usefulness of $\mathsf{FC}_{\mathsf{sn}}$, we will give several applications: some of them are well-known, and some are new. In all cases, the function of interest can be rewritten as a semi-sparse polynomial in $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Some of these examples are closely related to but still sufficiently different from IPFC. In particular, [29] showed how to use an efficient IPFC to construct SFC for polynomial commitments [27], accumulators [5], and vector commitments [12] (See the full

version [34].). We use $\mathsf{FC}_{\mathsf{sn}}$ to construct subvector commitments [28], aggregated polynomial commitment [15,9] (one can commit to multiple polynomials at once, each of which can be opened at a different point), and multivariate polynomial commitments [10]. Also, we outline a few seemingly new applications like the aggregated inner product (that, in particular, can be used to implement subvector commitment) and evaluation-point commitment schemes. (See the full version [34].) All described commitment schemes are succinct.

Importantly, $\mathsf{FC}_{\mathsf{sn}}$ achieves easy aggregation in a more general sense. Let $\mathcal{C}_i$ be some circuits for which efficient SFC schemes exist. We can then construct an efficient SFC for the circuit that consists of the sequential composition of $\mathcal{C}_i$-s. In particular, we can aggregate multiple polynomial commitment schemes, some vector commitment schemes, and say an evaluation-point commitment scheme. Some of the referred papers [8,28,9,38,22] construct aggregated commitment schemes for a concrete circuit (e.g., an aggregated polynomial commitment scheme). Importantly, $\mathsf{FC}_{\mathsf{sn}}$ allows one to aggregate different SFC schemes.

Security. The correctness and perfect hiding proofs are straightforward. The main thing worthy of note here is that we have three definitions of hiding (com-hiding, open-hiding, and zero-knowledge, see Section 2). For the sake of completeness, we also give three different hiding proofs. The SFC schemes must work in the CRS model to obtain zero-knowledge. However, since zero-knowledge is stronger than the other two definitions, the proof of zero-knowledge, that follows roughly from the zero-knowledge of the related SNARK, suffices. Note that say [29] only considered the weakest hiding notion (com-hiding).

The evaluation-binding proof differs significantly from the knowledge-soundness proofs of SNARKs. The knowledge-soundness of SNARKs can only be proven under non-falsifiable assumptions [21]. In particular, Groth proved the knowledge-soundness of the SNARK from [24] in the generic group model while Lipmaa [33] proved it under HAK (hash-algebraic knowledge assumption, a tautological knowledge assumption) and a known computational assumption (namely, q-PDL [31]). Such assumptions have very little in common with assumptions we use. As expected, a knowledge-soundness proof that uses non-falsifiable assumptions has a very different flavor compared to an evaluation-binding proof that only uses falsifiable assumptions. We emphasize it is not clear a priori that an SFC constructed from SNARKs could rely on falsifiable assumptions.

We prove the evaluation-binding of $\mathsf{FC}_{\mathsf{sn}}$ under the new $(\mathcal{R}, \mathcal{S}, \{f_i\})$-*computational span-uber-assumption* in source group $\mathbb{G}_1$, where $\mathcal{R}, \mathcal{S} \subset \mathbb{Z}_p[X, Y]$ and $f_i \in \mathbb{Z}_p[X, Y]$ with $f_i \notin \operatorname{span}(\mathcal{R})$. This assumption states that given a commitment key $\mathsf{ck} = ([\varrho(\chi, y) : \varrho \in \mathcal{R}]_1, [\sigma(\chi, y) : \sigma \in \mathcal{S}]_2)$, where χ, y are random trapdoors, it is difficult to compute $(\boldsymbol{\Delta} \neq \mathbf{0}, \sum_{i=1}^{\kappa} \Delta_i [f_i(\chi, y)]_1)$, where $\boldsymbol{\Delta}$ is adversarially chosen. (See Definition 6 for a formal definition.) Importantly, if $\kappa = 1$ then we just have an uber-assumption in $\mathbb{G}_1$. We show that (see Theorem 2), for concrete $\mathcal{R}$ and f_i, $f_i(X, Y) \notin \operatorname{span}(\mathcal{R})$.

The full evaluation-binding proof is quite tricky and relies significantly on the structure of matrices U, V, W, and of the commitment key. Given a collision, we

"almost" compute $(\boldsymbol{\Delta}, \sum \Delta_i[f_i(\chi, y)]_1)$, where $\boldsymbol{\Delta}$ is the componentwise difference between two claimed values of $\mathcal{F}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. To eliminate "almost" in the previous sentence, the committer outputs κ additional "helper" elements $[\mathsf{B}_i^{\mathsf{aux}}]_1$, where extra care has to be used to guarantee that the helper elements can be computed given the commitment key. In both cases, to succeed, we need to assume that the matrices (U, V, W) satisfy some natural restrictions stated in individual theorems. These restrictions are collected together in Theorem 1.

Analysis of the Span-Uber-Assumption. The span-uber-assumption is falsifiable and, thus, significantly more realistic than non-falsifiable (knowledge) assumptions needed to prove the adaptive soundness of SNARKs. Still, it is a new assumption and thus we have written down *three* different proofs that it follows from already known assumptions. (See Lemma 2 and Theorem 4, and another theorem in the full version.)

In the full version [34], we prove that the span-uber-assumption in $\mathbb{G}_1$ holds under the known $(\mathcal{R}, \mathcal{S}, f_i')$-computational uber-assumption in the target group $\mathbb{G}_T$ [7]. Here, f_i' are different from but related to f_i. We also prove that $f_i' \notin \operatorname{span}(\mathcal{RS})$. Since $f_i(X, Y) \notin \operatorname{span}(\mathcal{R})$ and $f_i'(X, Y) \notin \operatorname{span}(\mathcal{RS})$ (in the case of the uber-assumption in $\mathbb{G}_T$), we have an instantiation of the computational uber-assumption, known to be secure [7] in the generic group model.

Since the generic group model is very restrictive and has known weaknesses [17,16] not shared by well-chosen knowledge assumptions, we will use the newer methodology of [33]. In the full version [34], we prove that if $f_i \notin \operatorname{span}(\mathcal{R})$ then the $(\mathcal{R}, \mathcal{S}, \{f_i\})$-computational span-uber-assumption in $\mathbb{G}_1$ holds under a HAK and a PDL assumption. Since uber-assumption in $\mathbb{G}_T$ is not secure under a HAK assumption (the latter only handles the case the adversary outputs elements in source groups since the target group is non-generic), this result is orthogonal to the previous result. As a corollary of independent interest, we get that if $f_i(X, Y) \notin \operatorname{span}(\mathcal{R})$ then uber-assumption in $\mathbb{G}_1$ holds under a HAK and a PDL assumption.

In composite-order bilinear groups, the computational uber-assumption in $\mathbb{G}_T$ holds under a subgroup hiding assumption [13]. Thus, due to Lemma 2, a composite-order group span-uber-assumption (and also the new SFC) is secure under a subgroup hiding assumption. In Theorem 4, we use the Déjà Q approach of [14] to prove that the span-uber-assumption in $\mathbb{G}_\iota$, $\iota \in \{1, 2\}$, is secure under a subgroup hiding assumption. This proof is more direct than the reduction through an uber-assumption in $\mathbb{G}_T$. Moreover, the Déjà Q approach is more applicable if one is working in the source group. Whether a similar reduction holds in the case of prime-order groups is an interesting open question.

Efficiency. It is difficult to provide a detailed efficiency comparison of our newly constructed scheme to all the abundant existing work in all applications. $\mathsf{FC}_{\mathsf{sn}}$ is *generic*, works for a large class of circuits, and can tackle scenarios, not possible with previous work, but at the same time, it can also be used to solve the much simpler case of, e.g., inner product. We stress that $\mathsf{FC}_{\mathsf{sn}}$, when straightforwardly

specialized to the IPFC case, is nearly as efficient as the most efficient known prior IPFC, losing ground only in the CRS length. On the other hand, we are not aware of *any* previous aggregated IPFC schemes (See the full version [34].).

This paper uses heavily a yet unpublished paper [33] of the first author.

2 Preliminaries

If $\mathcal{R} = (\varrho_1(\boldsymbol{X}), \dots, \varrho_n(\boldsymbol{X}))$ is a tuple of polynomials over $\mathbb{Z}_p[\boldsymbol{X}]$ and $\boldsymbol{x}$ is a vector of integers then $\mathcal{R}(\boldsymbol{x}) := (\varrho_1(\boldsymbol{x}), \dots, \varrho_n(\boldsymbol{x}))$. Let $\mathbb{Z}_p^{(\leq d)}[X]$ be the set of degree-$\leq d$ polynomials over $\mathbb{Z}_p$. For a matrix U, let $\boldsymbol{U}_i$ be its ith row, $\boldsymbol{U}^{(j)}$ be its jth column. Let $\boldsymbol{a} \circ \boldsymbol{b}$ denote the component-wise product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $(\boldsymbol{a} \circ \boldsymbol{b})_i = a_i b_i$. Let $\boldsymbol{a}_1 // \dots // \boldsymbol{a}_n = \begin{pmatrix} \boldsymbol{a}_1 \\ \dots \\ \boldsymbol{a}_n \end{pmatrix}$ denote the vertical concatenation of vectors $\boldsymbol{a}_i$. λ is the security parameter, and 1^λ denotes its unary representation. PPT denotes probabilistic polynomial-time. For an algorithm $\mathcal{A}$, range($\mathcal{A}$) is the range of $\mathcal{A}$, i.e., the set of valid outputs of $\mathcal{A}$, $\mathsf{RND}_\lambda(\mathcal{A})$ denotes the random tape of $\mathcal{A}$ (assuming the given value of λ), and $r \leftarrow_\$ \mathcal{S}$ denotes the uniformly random choice of a randomizer r from the set/distribution $\mathcal{S}$.

Interpolation. Assume ν is a power of two, and let ω be the νth primitive root of unity modulo p. Such ω exists, given that $\nu \mid (p-1)$. Then,

- $\ell(X) := \prod_{i=1}^{\nu} (X - \omega^{i-1}) = X^\nu - 1$ is the unique degree ν monic polynomial such that $\ell(\omega^{i-1}) = 0$ for all $i \in [1 \,..\, \nu]$.
- For $i \in [1 \,..\, \nu]$, $\ell_i(X)$ is the *ith Lagrange basis polynomial*, i.e., the unique degree $\nu - 1$ polynomial s.t. $\ell_i(\omega^{i-1}) = 1$ and $\ell_i(\omega^{j-1}) = 0$ for $i \neq j$. Clearly, $\ell_i(X) := \ell(X)/(\ell'(\omega^{i-1})(X - \omega^{i-1})) = (X^\nu - 1)\omega^{i-1}/(\nu(X - \omega^{i-1}))$.

Moreover, $(\ell_j(\omega^{i-1}))_{i=1}^{\nu} = \boldsymbol{e}_j$ (the jth unit vector) and $(\ell(\omega^{i-1}))_{i=1}^{\nu} = \boldsymbol{0}_\nu$.

Bilinear Pairings. Let ν be an integer parameter (the circuit size in our application). A bilinear group generator $\mathsf{Pgen}(1^\lambda, \nu)$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, \mathsf{P}_1, \mathsf{P}_2)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are three additive cyclic groups of prime order p, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear pairing, and P_ι is a fixed generator of $\mathbb{G}_\iota$. We assume $\mathsf{P}_T = \hat{e}(\mathsf{P}_1, \mathsf{P}_2)$. We require the bilinear pairing to be Type-3, i.e., there is no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. For efficient interpolation, we assume that p is such that $\nu \mid (p-1)$. When emphasizing efficiency is not important, we drop the parameter ν and just write $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$. We use additive notation together with the standard elliptic-curve "bracket" notation. Namely, we write $[a]_\iota$ to denote $a\mathsf{P}_\iota$, and $[a]_1 \bullet [b]_2$ to denote $\hat{e}([a]_1, [b]_2)$ as . We use freely the bracket notation together with matrix notation, e.g., if $AB = C$ as matrices then $[A]_1 \bullet [B]_2 = [C]_T$.

Uber-Assumption. The following assumption is a special case of the more general uber-assumption of [7,11].

Definition 1 ([7,11]). *Let* $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$. *Let* $\mathcal{R}$, $\mathcal{S}$, *and* $\mathcal{T}$ *be three tuples of bivariate polynomials from* $\mathbb{Z}_p[X, Y]$. *Let* f *be a bivariate polynomial from* $\mathbb{Z}_p[X, Y]$. *The* $(\mathcal{R}, \mathcal{S}, \mathcal{T}, f)$-computational uber-assumption *for* Pgen

in group $\mathbb{G}_\iota$*, where* $\iota \in \{1,2,T\}$*, states that for any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathrm{uber}}_{\mathsf{Pgen},\mathcal{R},\mathcal{S},\mathcal{T},f,\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$*, where* $\mathsf{Adv}^{\mathrm{uber}}_{\mathsf{Pgen},\mathcal{R},\mathcal{S},\mathcal{T},f,\mathcal{A}}(\lambda) :=$

$$\Pr\begin{bmatrix}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \chi, y \leftarrow_{\$} \mathbb{Z}_p^*; \mathsf{ck} \leftarrow ([\mathcal{R}(\chi,y)]_1, [\mathcal{S}(\chi,y)]_2, [\mathcal{T}(\chi,y)]_T) : \\ \mathcal{A}(\mathsf{ck}) = [f(\chi,y)]_\iota\end{bmatrix} .$$

[7,11] considered the general case of c-variate polynomials for any c. In our case, $\mathcal{T} = \emptyset$; then, we have an $(\mathcal{R},\mathcal{S},f)$-computational uber-assumption in $\mathbb{G}_\iota$.

Importantly [7,11], (i) if $f(X,Y)$ is not in the span of $\{\varrho(X,Y)\}$ then the $(\mathcal{R},\mathcal{S},\mathcal{T},f)$-computational uber-assumption for $\mathbb{G}_1$ holds in the generic group model, and (ii) if $f(X,Y)$ is not in the span of $\{\varrho(X,Y)\sigma(X,Y)+\tau(X,Y)\}$ then the $(\mathcal{R},\mathcal{S},\mathcal{T},f)$-computational uber-assumption for $\mathbb{G}_T$ is difficult in the generic group model. We will only invoke the uber-assumption in the case $f(X,Y)$ is not in the span of $\{\varrho(X,Y)\}$.

QAP. Let $\mathbf{R} = \{(\mathsf{z},\mathsf{wit})\}$ be a relation between statements and witnesses. Quadratic Arithmetic Program (QAP) was introduced in [20] as a language where for an input z and witness wit, $(\mathsf{z},\mathsf{wit}) \in \mathbf{R}$ can be verified by using a parallel quadratic check. QAP has an efficient reduction from the (either Boolean or Arithmetic) Circuit-SAT. Thus, an efficient zk-SNARK for QAP results in an efficient zk-SNARK for Circuit-SAT.

We consider arithmetic circuits that consist only of fan-in-2 multiplication gates, but either input of each multiplication gate can be any weighted sum of wire values, [20]. Let $\mu_0 < \mu$ be a non-negative integer. In the case of arithmetic circuits, ν is the number of multiplication gates, μ is the number of wires, and μ_0 is the number of public inputs.

Let $\mathbb{F} = \mathbb{Z}_p$, such that ω is the ν-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 SFC to work. Let U, V, and W be instance-dependent matrices and let $\mathbf{a}$ be a witness. A QAP is characterized by the constraint $U\mathbf{a} \circ V\mathbf{a} = W\mathbf{a}$. Let $L_{\boldsymbol{a}}(X) := \sum_{i=1}^{\nu} a_i \ell_i(X)$ be the interpolating polynomial of $\boldsymbol{a} = (a_1,\ldots,a_\nu)^\top$ at points ω^{i-1}, with $L_{\boldsymbol{a}}(\omega^{i-1}) = a_i$. For $j \in [1\,..\,\mu]$, define $u_j(X) := L_{\boldsymbol{U}^{(j)}}(X)$, $v_j(X) := L_{\boldsymbol{V}^{(j)}}(X)$, and $w_j(X) := L_{\boldsymbol{W}^{(j)}}(X)$ to be interpolating polynomials of the jth column of the corresponding matrix. Thus, $u_j, v_j, w_j \in \mathbb{Z}_p^{(\le \nu-1)}[X]$. Let $u(X) = \sum_{j=1}^{\mu} \mathsf{a}_j u_j(X)$, $v(X) = \sum_{j=1}^{\mu} \mathsf{a}_j v_j(X)$, and $w(X) = \sum_{j=1}^{\mu} \mathsf{a}_j w_j(X)$. Then $U\mathbf{a} \circ V\mathbf{a} = W\mathbf{a}$ iff $\ell(X) \mid u(X)v(X) - w(X)$ iff $u(X)v(X) \equiv w(X) \pmod{\ell(X)}$ iff there exists a polynomial $\mathcal{H}(X)$ such that $u(X)v(X) - w(X) = \mathcal{H}(X)\ell(X)$.

A QAP instance $\mathcal{I}_{\mathsf{qap}}$ is equal to $(\mathbb{Z}_p, \mu_0, \{u_j, v_j, w_j\}_{j=1}^{\mu})$. $\mathcal{I}_{\mathsf{qap}}$ defines the following relation:

$$\mathbf{R}_{\mathcal{I}_{\mathsf{qap}}} = \left\{ \begin{array}{l} (\mathsf{z},\mathsf{wit}) \colon \mathsf{z} = (\mathsf{a}_1,\ldots,\mathsf{a}_{\mu_0})^\top \wedge \mathsf{wit} = (\mathsf{a}_{\mu_0+1},\ldots,\mathsf{a}_\mu)^\top \wedge \\ u(X)v(X) \equiv w(X) \pmod{\ell(X)} \end{array} \right\} , \quad (1)$$

where $u(X)$, $v(X)$, and $w(X)$ are as above. Alternatively, $(\mathsf{z},\mathsf{wit}) \in \mathbf{R}$ if there exists a (degree $\le \nu-2$) polynomial $\mathcal{H}(X)$, s.t. the following key equation holds:

$$\chi(X) := u(X)v(X) - w(X) - \mathcal{H}(X)\ell(X) = 0 . \quad (2)$$

On top of checking Eq. (2), the verifier also needs to check that $u(X)$, $v(X)$, and $w(X)$ are correctly computed: that is, (i) the first μ_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 $j \leq \mu$.

Since both the committer and the verifier have inputs, we will use a variation of QAP that handles public inputs differently (see Section 3). In particular, we will use different parameters instead of μ_0.

SNARKs. Let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ returns a polynomial-time decidable binary relation $\mathbf{R} = \{(\mathsf{z}, \mathsf{wit})\}$. Here, z is a statement, and wit is a witness. $\mathcal{R}$ also outputs the system parameters p that will be given to the honest parties and the adversary. A *non-interactive zero-knowledge (NIZK) argument system* $\Psi = (\mathsf{K_{crs}}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for $\mathcal{R}$ consists of four PPT algorithms:

CRS generator: $\mathsf{K_{crs}}$ is a probabilistic algorithm that, given $(\mathbf{R}, \mathsf{p}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$, outputs $(\mathsf{crs}, \mathsf{td})$ where crs is a CRS and td is a simulation trapdoor. Otherwise, it outputs a special symbol $\perp$.

Prover: P is a probabilistic algorithm that, given $(\mathbf{R}, \mathsf{p}, \mathsf{crs}, \mathsf{z}, \mathsf{wit})$ for $(\mathsf{z}, \mathsf{wit}) \in \mathbf{R}$, outputs an argument π. Otherwise, it outputs $\perp$.

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

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

A NIZK argument system must satisfy completeness (an honest verifier accepts an honest prover), knowledge-soundness (if a prover makes an honest verifier accept, then one can extract from the prover a witness wit), and zero-knowledge (there exists a simulator that, knowing CRS trapdoor but not the witness, can produce accepting statements with the verifier's view being indistinguishable from the view when interacting with an honest prover). See the full version [34] for formal definitions. A *SNARK (succinct non-interactive argument of knowledge, [23,31,20,32,24,33])* is a NIZK argument system where the argument is sublinear in the input size.

Functional Commitment Schemes. Let $\mathcal{D}$ be some domain. In a functional commitment scheme for a circuit $\mathcal{C} : \mathcal{D}^{\mu_\alpha} \times \mathcal{D}^{\mu_\beta} \to \mathcal{D}^{\kappa}$, one first commits to a vector $\boldsymbol{\alpha} \in \mathcal{D}^{\mu_\alpha}$, obtaining a functional commitment C. The goal is to allow the committer to later open C to $\boldsymbol{\xi} = \mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathcal{D}^{\kappa}$, where $\boldsymbol{\beta} \in \mathcal{D}^{\mu_\beta}$ is a public input that is chosen by the verifier before the opening. We generalize the notion of functional commitment, given in [29], from inner products to arbitrary circuits. Compared to [29], we also provide a stronger hiding definition.

Let $\mathbf{CC}$ be a class of circuits $\mathcal{C} : \mathcal{D}^{\mu_\alpha} \times \mathcal{D}^{\mu_\beta} \to \mathcal{D}^{\kappa}$. A *functional commitment scheme* FC *for* $\mathbf{CC}$ is a tuple of four (possibly probabilistic) polynomial time algorithms $(\mathsf{KC}, \mathsf{com}, \mathsf{open}, \mathsf{V})$, where

Commitment-key generator: $\mathsf{KC}(1^\lambda, \mathcal{C})$ is a probabilistic algorithm that, given a security parameter $\lambda \in \mathbb{N}$ and a circuit $\mathcal{C} \in \mathbf{CC}$, outputs a commitment key ck and a trapdoor key tk. We implicitly assume 1^λ and $\mathcal{C}$ are described by ck.

Commitment: $\mathsf{com}(\mathsf{ck}, \boldsymbol{\alpha}; r)$ is a probabilistic algorithm that takes as input the commitment key ck, a message vector $\boldsymbol{\alpha} \in \mathcal{D}^{\mu_\alpha}$ and some randomizer r. It outputs (C, D), where C is a commitment to $\boldsymbol{\alpha}$ and D is a decommitment information. We denote the first output C of $\mathsf{com}(\mathsf{ck}; \boldsymbol{\alpha}; r)$ by $\mathsf{com}_1(\mathsf{ck}; \boldsymbol{\alpha}; r)$.

Opening: $\mathsf{open}(\mathsf{ck}, C, D, \boldsymbol{\beta})$ is a deterministic algorithm that takes as input the commitment key ck, a commitment C (to $\boldsymbol{\alpha}$), a decommitment information D, and a vector $\boldsymbol{\beta} \in \mathcal{D}^{\mu_\beta}$. Assume that the ith output value of the circuit $\mathcal{C}$ is $\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$, where $\mathcal{F}_i$ is a public function. It computes an opening $\mathrm{op}_{\boldsymbol{\xi}}$ to $\boldsymbol{\xi} = \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) := (\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}))_{i=1}^{\kappa}$.

Verification: $\mathsf{V}(\mathsf{ck}, C, \mathrm{op}_{\boldsymbol{\xi}}, \boldsymbol{\beta}, \boldsymbol{\xi})$ is a deterministic algorithm that takes as input the commitment key ck, a commitment C, an opening $\mathrm{op}_{\boldsymbol{\xi}}$, a vector $\boldsymbol{\beta} \in \mathcal{D}^{\mu_\beta}$, and $\boldsymbol{\xi} \in \mathcal{D}^{\kappa}$. It outputs 1 if $\mathrm{op}_{\boldsymbol{\xi}}$ is a valid opening for C being a commitment to some $\boldsymbol{\alpha} \in \mathcal{D}^{\mu_\alpha}$ such that $\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \boldsymbol{\xi}$ and outputs 0 otherwise.

Security of FC. Next, we give three definitions of the hiding property for FC schemes of increasing strength. The first definition corresponds to the definition of hiding given in [29] and essentially states that commitments do not reveal any information about $\boldsymbol{\alpha}$. The other two definitions seem to be novel at least in the context of general FC. We provide all three definitions, since in some applications, a weaker definition might be sufficient. Moreover, the third definition (zero-knowledge) makes only sense in the CRS model; in a CRS-less model, one can rely on the open-hiding property.

Definition 2 (Perfect com-hiding). *A functional commitment scheme* $\mathsf{FC} = (\mathsf{KC}, \mathsf{com}, \mathsf{open}, \mathsf{V})$ *for circuit class* $\mathbf{CC}$ *is* perfectly hiding *if for any* λ, $\mathcal{C} \in \mathbf{CC}$, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KC}(1^\lambda, \mathcal{C})$, *for all* $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathcal{D}^{\mu_\alpha}$ *with* $\boldsymbol{\alpha}_1 \neq \boldsymbol{\alpha}_2$, *the two distributions* δ_1 *and* δ_2 *are identical, where*

$$\delta_b := \{(\mathsf{ck}, C_b) : r \leftarrow\!\!\$\, \mathsf{RND}_\lambda(\mathsf{com}); (C_b, D_b) \leftarrow \mathsf{com}(\mathsf{ck}, \boldsymbol{\alpha}_b; r)\} \ .$$

The open-hiding property is considerably stronger, stating that the commitment and the openings together do not reveal more information on $\boldsymbol{\alpha}$ than the values $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}_i)$ on queried values $\boldsymbol{\beta}_i$. Trivial non-succinct FC schemes, where one uses a perfectly-hiding commitment scheme to commit to $\boldsymbol{\beta}$, and then in the opening phase, opens the whole database, are com-hiding but not open-hiding.

Definition 3 (Perfect open-hiding). *A functional commitment scheme* $\mathsf{FC} = (\mathsf{KC}, \mathsf{com}, \mathsf{open}, \mathsf{V})$ *for circuit class* $\mathbf{CC}$ *is* perfectly open-hiding *if for any* λ, $\mathcal{C} \in \mathbf{CC}$, $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KC}(1^\lambda, \mathcal{C})$, *for all* $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathcal{D}^{\mu_\alpha}$ *with* $\boldsymbol{\alpha}_1 \neq \boldsymbol{\alpha}_2$, *and* $Q = \mathsf{poly}(\lambda)$ *of* $\boldsymbol{\beta}_i$ *such that* $\mathcal{C}(\boldsymbol{\alpha}_1, \boldsymbol{\beta}_i) = \mathcal{C}(\boldsymbol{\alpha}_2, \boldsymbol{\beta}_i)$ *for all* $i \leq Q$, *the two distributions* δ_1 *and* δ_2 *are identical, where* $\delta_b :=$

$$\{(\mathsf{ck}, C_b, \{\mathsf{open}(\mathsf{ck}, C_b, D_b, \boldsymbol{\beta}_i)\}) : r \leftarrow\!\!\$\, \mathsf{RND}_\lambda(\mathsf{com}); (C_b, D_b) \leftarrow \mathsf{com}(\mathsf{ck}, \boldsymbol{\alpha}_b; r)\} \ .$$

Finally, zero-knowledge FC schemes have simulation-based hiding. While simulation-based security is a gold standard in cryptography, it is usually more complicated to achieve than game-based security. In particular, one needs to have a trusted ck (and its trapdoor) to achieve zero-knowledge. We will leave it

as an open problem whether one can use instead the much weaker bare public key (BPK) model, by using the techniques of [4,1,18,2]. Note that [33] showed that their SNARKs are all secure in the BPK model.

Definition 4 (Perfect zero-knowledge). *An FC scheme* $\mathsf{FC} = (\mathsf{KC}, \mathsf{com}, \mathsf{open}, \mathsf{V})$ *for* $\mathbf{CC}$ *is* perfectly zero-knowledge *if there exists a PPT simulator* Sim*, such that for all* λ*, all* $\mathcal{C} \in \mathbf{CC}$*,* $(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KC}(1^\lambda, \mathcal{C})$*, for all* $\boldsymbol{\alpha} \in \mathcal{D}^{\mu_\alpha}$*, for any poly-size set of* $\boldsymbol{\beta}_i$*,* δ_0 *and* δ_1 *are identical, where*

$$\delta_0 := \{(\mathsf{ck}, C, \{\mathsf{open}(\mathsf{ck}, C, D, \boldsymbol{\beta}_i)\}) : r \leftarrow\!\!\$\, \mathsf{RND}_\lambda(\mathsf{com}); (C, D) \leftarrow \mathsf{com}(\mathsf{ck}, \boldsymbol{\alpha}; r)\} \ ,$$
$$\delta_1 := \{(\mathsf{ck}, \mathsf{Sim}(\mathsf{ck}, \mathsf{td}, \{\boldsymbol{\beta}_i\}, \{\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta}_i)\}))\} \ .$$

Next, we will define evaluation-binding. Evaluation-binding can be weaker than binding, but sometimes the two notions are equivalent. (Consider the case of the inner product when the adversary asks the committer to open a commitment for $\boldsymbol{\beta} = \boldsymbol{e}_i$ for each i.) In the context of FC schemes, evaluation-binding is *the* distinguishing security notion.

Definition 5 (Computational evaluation-binding). *A functional commitment scheme* $\mathsf{FC} = (\mathsf{KC}, \mathsf{com}, \mathsf{open}, \mathsf{V})$ *for circuit class* $\mathbf{CC}$ *is* computationally evaluation-binding *if for any* λ*,* $\mathcal{C} \in \mathbf{CC}$*, and a non-uniform PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathrm{bind}}_{\mathsf{FC},\lambda,\mathcal{C},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$*, where* $\mathsf{Adv}^{\mathrm{bind}}_{\mathsf{FC},\lambda,\mathcal{C},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\begin{array}{l}(\mathsf{ck}, \mathsf{tk}) \leftarrow \mathsf{KC}(1^\lambda, \mathcal{C}); (C, \boldsymbol{\beta}, \boldsymbol{\xi}, \mathrm{op}_{\boldsymbol{\xi}}, \widetilde{\boldsymbol{\xi}}, \widetilde{\mathrm{op}}_{\boldsymbol{\xi}}) \leftarrow \mathcal{A}(\mathsf{ck}) : \boldsymbol{\beta} \in \mathcal{D}^{\mu_\beta} \wedge \\ \boldsymbol{\xi} \neq \widetilde{\boldsymbol{\xi}} \in \mathcal{D}^\kappa \wedge \mathsf{V}(\mathsf{ck}, C, \mathrm{op}_{\boldsymbol{\xi}}, \boldsymbol{\beta}, \boldsymbol{\xi}) = \mathsf{V}(\mathsf{ck}, C, \widetilde{\mathrm{op}}_{\boldsymbol{\xi}}, \boldsymbol{\beta}, \widetilde{\boldsymbol{\xi}}) = 1\end{array}\right] \ .$$

An FC scheme is *succinct* (SFC), if both the commitments and openings have length that is polylogarithmic in $|\boldsymbol{\alpha}|$ and $|\boldsymbol{\beta}|$.

3 The New SFC Scheme

In this section, we will construct a succinct functional commitment (SFC) scheme for (almost) all polynomial-size arithmetic circuits by mixing techniques from SNARKs with original ideas, needed to construct a SFC scheme. Let $\boldsymbol{\mathcal{F}}$ be a fixed vector function that takes inputs from two parties, the committer and the verifier. Let α_j be private inputs of the committer, used when committing. Let β_j be public inputs of the verifier, used when opening the commitment.

Let $\mathcal{C}$ be an arithmetic circuit that inputs α_j and β_j and computes $\boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}))_{i=1}^\kappa$, where $\boldsymbol{\alpha}$ is the private input of the committer and $\boldsymbol{\beta}$ is chosen by the verifier, possibly only later. We compile $\mathcal{C}$ to a circuit $\mathcal{C}^*$ that consists of four subcircuits $\mathcal{C}_\phi$, $\mathcal{C}_\psi$, $\mathcal{C}_\chi$ and $\mathcal{C}_\xi$. We need the division to four subcircuits to prove evaluation-binding; we will give more details later.

After that, we use the QAP-representation [20] (more precisely, the approach of [33]) of arithmetic circuits, obtaining polynomials $\mathsf{A}(X, Y)$, $\mathsf{B}(X, Y)$ (the "commitment polynomials" to all left/right inputs of all gates of $\mathcal{C}^*$, correspondingly), and $\mathsf{C}(X, Y)$ (the "opening polynomial"), such that $\mathsf{C}(X, Y)$ is in the linear span

of the "polynomial commitment key" $\mathsf{ck}_1 = (\varrho(X,Y) : \varrho \in \mathcal{R})$ if and only if the committer was honest. The circuit compilation allows us to divide the polynomials to "private" parts (transmitted during the commitment) and "public" parts (trasmitted during the opening), such that one can, given two different openings for the same commitment, break a computational assumption. We then use SNARK-based techniques to construct the SFC for $\mathcal{C}^*$ with succinct commitment and opening. We postpone security proofs to Section 5; we currently emphasize that the evaluation-binding proof is novel (in particular, not related to the knowledge-soundness proofs of SNARKs at all).

Circuit Compilation. Let $\mathcal{C}$ be a polynomial-size arithmetic circuit that, on input $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, outputs $\boldsymbol{\xi} = \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}))_{i=1}^{\kappa}$. We compile $\mathcal{C}$ to a *compiled circuit* $\mathcal{C}^*$, see Fig. 1, that consists of the public subcircuits $\mathcal{C}_\phi$, $\mathcal{C}_\psi$, $\mathcal{C}_\chi$, and $\mathcal{C}_\xi$ that are combined as follows. In the commitment phase, the committer uses the circuit $\mathcal{C}_\phi$ to compute a number of polynomials $\phi_i(\boldsymbol{\alpha})$ depending on only 1 and $\boldsymbol{\alpha}$. More precisely, $\boldsymbol{\phi}(\boldsymbol{\alpha}) = (\phi_1(\boldsymbol{\alpha}), \ldots, \phi_{\mu_\phi}(\boldsymbol{\alpha}))$ denotes the set of the outputs of all (including intermediate) gates in $\mathcal{C}_\phi$ (the same is the case of other circuits and corresponding polynomials). The commitment depends only on 1, $\boldsymbol{\alpha}$, and $\boldsymbol{\phi}(\boldsymbol{\alpha})$. In the opening phase, the verifier sends $\boldsymbol{\beta}$ to the committer, who uses the circuit $\mathcal{C}_\psi$ to compute some polynomials $\psi_i(\boldsymbol{\beta})$ depending on 1 and $\boldsymbol{\beta}$. This part of the computation is public and can be redone by the verifier.

After that, the committer uses the circuit $\mathcal{C}_\chi$ to compute a number of polynomials $\chi_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$ from the inputs and outputs of $\mathcal{C}_\phi$ and $\mathcal{C}_\psi$, i.e., from $(1, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\phi}(\boldsymbol{\alpha}), \boldsymbol{\psi}(\boldsymbol{\beta}))$. $\mathcal{C}_\chi$ has multiplicative depth 1, and thus, w.l.o.g., each $\chi_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is a product of some $\phi_j(\boldsymbol{\alpha})$ with some $\psi_k(\boldsymbol{\beta})$. Finally, the committer uses $\mathcal{C}_\xi$ to compute the outputs $\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of $\mathcal{C}^*$. We will explain the need for such compilation after Eqs. (7) and (8). We will summarize all actual restrictions on the circuits in Theorem 1. In the introduction, we gave an intuitive explanation of how this compilation reduces the circuit class that we can handle. See Section 4 for an additional discussion on the power of this circuit class.

Next, let $\mathbf{a} \in \mathbb{Z}_p^\mu$ be the value of all wires of $\mathcal{C}^*$. We write

$$\mathbf{a} = 1//\boldsymbol{\alpha}//\boldsymbol{\phi}(\boldsymbol{\alpha})//\boldsymbol{\beta}//\boldsymbol{\psi}(\boldsymbol{\beta})//\boldsymbol{\chi}(\boldsymbol{\alpha}, \boldsymbol{\beta})//\boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \ . \tag{3}$$

Here, $\boldsymbol{\alpha} \in \mathbb{Z}_p^{\mu_\alpha}$, $\boldsymbol{\phi}(\boldsymbol{\alpha}) \in \mathbb{Z}_p^{\mu_\phi}$, $\boldsymbol{\beta} \in \mathbb{Z}_p^{\mu_\beta}$, $\boldsymbol{\psi}(\boldsymbol{\beta}) \in \mathbb{Z}_p^{\mu_\psi}$, $\boldsymbol{\chi}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathbb{Z}_p^{\mu_\chi}$, and $\boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathbb{Z}_p^{\kappa}$. Thus, $\mu = 1+\mu_\alpha+\mu_\beta+\mu_\phi+\mu_\psi+\mu_\chi+\kappa$. To use the RC1S approach, we construct matrices U, V, and W, such that $U\mathbf{a} \circ V\mathbf{a} = W\mathbf{a}$ iff $\mathcal{C}^*$ is correctly computed. Let $\boldsymbol{\alpha}^* = (1//\boldsymbol{\alpha}//\boldsymbol{\phi}(\boldsymbol{\alpha})) \in \mathbb{Z}_p^{1+\mu_\alpha+\mu_\phi}$ and $\boldsymbol{\beta}^* = (1//\boldsymbol{\beta}//\boldsymbol{\psi}(\boldsymbol{\beta})) \in \mathbb{Z}_p^{1+\mu_\beta+\mu_\psi}$. First, we define R1CS-matrices $U_\phi, U_\psi, U_\chi, U_\xi, V_\phi, V_\psi, V_\chi$ such that (various subcircuits of) $\mathcal{C}^*$ are correctly computed iff

$$\begin{gathered} U_\phi\boldsymbol{\alpha}^* \circ V_\phi\boldsymbol{\alpha}^* = \boldsymbol{\phi}(\boldsymbol{\alpha}) \ , \quad U_\psi\boldsymbol{\beta}^* \circ V_\psi\boldsymbol{\beta}^* = \boldsymbol{\psi}(\boldsymbol{\beta}) \ , \\ U_\chi \begin{pmatrix} \boldsymbol{\alpha}^* \\ \boldsymbol{\beta} \\ \boldsymbol{\psi}(\boldsymbol{\beta}) \end{pmatrix} \circ V_\chi \begin{pmatrix} \boldsymbol{\alpha}^* \\ \boldsymbol{\beta} \\ \boldsymbol{\psi}(\boldsymbol{\beta}) \end{pmatrix} = \boldsymbol{\chi}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \ , \quad U_\xi\boldsymbol{\chi}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \circ \mathbf{1} = \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \ . \end{gathered} \tag{4}$$

Here, $U_\phi, V_\phi \in \mathbb{Z}_p^{\mu_\phi \times (1+\mu_\alpha+\mu_\phi)}$, $U_\psi, V_\psi \in \mathbb{Z}_p^{\mu_\psi \times (1+\mu_\beta+\mu_\psi)}$, $U_\chi, V_\chi \in \mathbb{Z}_p^{\mu_\chi \times (1+\mu_\alpha+\mu_\beta+\mu_\phi+\mu_\psi)}$, and $U_\xi \in \mathbb{Z}_p^{\kappa \times \mu_\chi}$. In particular,

$$\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \textstyle\sum_{j=1}^{\mu_\chi} U_{\xi ij} \chi_j(\boldsymbol{\alpha}, \boldsymbol{\beta}) \ , \quad i \in [1 \,..\, \kappa] \ . \tag{5}$$

Next, we define $U, V, W \in \mathbb{Z}_p^{\nu \times \mu}$, as

$$\left(\begin{array}{c|c|c|c|c|c|c|c} 1 & \boldsymbol{\alpha} & \boldsymbol{\phi}(\boldsymbol{\alpha}) & \boldsymbol{\beta} & \boldsymbol{\psi}(\boldsymbol{\beta}) & \boldsymbol{\chi}(\boldsymbol{\alpha},\boldsymbol{\beta}) & \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \hline \multicolumn{3}{c|}{U_\phi} & & & & \\ \hline U_\psi & & & U_\psi & & & \\ \hline \multicolumn{5}{c|}{U_\chi} & & \\ \hline & & & & & U_\xi & \end{array}\right) \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & \boldsymbol{\alpha} & \boldsymbol{\phi}(\boldsymbol{\alpha}) & \boldsymbol{\beta} & \boldsymbol{\psi}(\boldsymbol{\beta}) & \boldsymbol{\chi}(\boldsymbol{\alpha},\boldsymbol{\beta}) & \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \hline \multicolumn{3}{c|}{V_\phi} & & & & \\ \hline V_\psi & & & V_\psi & & & \\ \hline \multicolumn{5}{c|}{V_\chi} & & \\ \hline \mathbf{1}_\kappa & & & & & & \end{array}\right) \left(\begin{array}{c|c|c|c|c|c|c|c} 1 & \boldsymbol{\alpha} & \boldsymbol{\phi}(\boldsymbol{\alpha}) & \boldsymbol{\beta} & \boldsymbol{\psi}(\boldsymbol{\beta}) & \boldsymbol{\chi}(\boldsymbol{\alpha},\boldsymbol{\beta}) & \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \hline & & I_{\mu_\phi} & & & & \\ \hline & & & & I_{\mu_\psi} & & \\ \hline & & & & & I_{\mu_\chi} & \\ \hline & & & & & & I_\kappa \end{array}\right) \tag{6}$$

correspondingly. Clearly, $\nu := \mu_\phi + \mu_\psi + \mu_\chi + \kappa$. Here, we labeled vertically each column of each matrix by the supposed value of the corresponding coefficients of $\mathbf{a} = 1//\boldsymbol{\alpha}//\ldots//\boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Some submatrices ($U_\psi$ and V_ψ) are divided between non-continuous areas. The empty submatrices are all-zero in the compiled instance. Clearly, $U\mathbf{a} \circ V\mathbf{a} = W\mathbf{a}$ iff Eq. (4) holds.

QAP Representation. Recall that $\ell_i(X) \in \mathbb{Z}_p^{(\leq \nu-1)}[X]$, $i \in [1 \,..\, \nu]$, interpolates the ν-dimensional unit vector $\boldsymbol{e}_i$. To obtain a QAP representation of the equation $U\mathbf{a} \circ V\mathbf{a} = W\mathbf{a}$, we use interpolating polynomials; e.g., $u_j(X)$ interpolates the jth column of U. (See Section 2.) To simplify notation, we introduce polynomials like $u_{\phi j}(X)$ and $u_{\chi j}(X)$, where say $u_{\chi j}(X)$ interpolates (*all ν rows of the*) the jth column of the $\nu \times (1 + \mu_\alpha + \mu_\beta + \mu_\phi + \mu_\psi)$ submatrix of U that contains U_χ. E.g., $u_{\chi j}(X)$ interpolates the jth column of U_χ (preceded and followed by 0 rows), $u_{\chi j}(X) = \sum_{i=1}^{\mu_\chi} U_{\chi ij} \ell_{\mu_\phi+\mu_\psi+i}(X)$.

We divide the polynomials $u(X)$ and $v(X)$ into two addends: one polynomial (u_s, v_s, resp.) that depends on $\boldsymbol{\alpha}$ but not on $\boldsymbol{\beta}$, and another polynomial (u_p, v_p, resp.) that depends on public values ($\boldsymbol{\beta}$ and $\{\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})\}$) but not on $\boldsymbol{\alpha}$ otherwise. Such a division is possible due to the way $\mathcal{C}^*$ is composed from the subcircuits. Thus, $u(X) = \sum_{j=1}^{\mu} \mathsf{a}_j u_j(X) = u_s(X) + u_p(X)$ and $v(X) = \sum_{j=1}^{\mu} \mathsf{a}_j v_j(X) = v_s(X) + v_p(X)$, where

$$\begin{aligned} u_s(X) =& \textstyle\sum_{j=2}^{\mu_\alpha+\mu_\phi+1} \mathsf{a}_j u_j(X) \\ =& \textstyle\sum_{j=1}^{\mu_\alpha} \alpha_j (u_{\phi,1+j}(X) + u_{\chi,1+j}(X)) + \\ & \textstyle\sum_{j=1}^{\mu_\phi} \phi_j(\boldsymbol{\alpha})(u_{\phi,1+\mu_\alpha+j}(X) + u_{\chi,1+\mu_\alpha+j}(X)) \ , \\ u_p(X) =& u_1(X) + \textstyle\sum_{j=\mu_\alpha+\mu_\phi+2}^{\mu} \mathsf{a}_j u_j(X) \\ =& u_1(X) + \textstyle\sum_{j=1}^{\mu_\beta} \beta_j (u_{\psi,1+j}(X) + u_{\chi,1+\mu_\alpha+\mu_\phi+j}(X)) + \\ & \textstyle\sum_{j=1}^{\mu_\psi} \psi_j(\boldsymbol{\beta})(u_{\psi,1+\mu_\beta+j}(X) + u_{\chi,1+\mu_\alpha+\mu_\phi+\mu_\beta+j}(X)) + \\ & \underbrace{\textstyle\sum_{j=1}^{\mu_\chi} \chi_j(\boldsymbol{\alpha}, \boldsymbol{\beta}) u_{\xi,1+j}(X)}_{= \sum_{i=1}^{\kappa} \mathcal{F}_i(\boldsymbol{\alpha},\boldsymbol{\beta}) \ell_{\nu-\kappa+i}(X)} \ , \end{aligned} \tag{7}$$

and

$$\begin{aligned} v_s(X) =& \textstyle\sum_{j=2}^{\mu_\alpha+\mu_\phi+1} \mathsf{a}_j v_j(X) \\ =& \textstyle\sum_{j=1}^{\mu_\alpha} \alpha_j (v_{\phi,1+j}(X) + v_{\chi,1+j}(X)) + \\ & \textstyle\sum_{j=1}^{\mu_\phi} \phi_j(\boldsymbol{\alpha})(v_{\phi,1+\mu_\alpha+j}(X) + v_{\chi,1+\mu_\alpha+j}(X)) \ , \\ v_p(X) =& \mathsf{a}_1 v_1(X) + \textstyle\sum_{j=\mu_\alpha+\mu_\phi+2}^{\mu} \mathsf{a}_j v_j(X) \\ =& v_1(X) + \textstyle\sum_{j=1}^{\mu_\beta} \beta_j \left(v_{\psi,1+j}(X) + v_{\chi,1+\mu_\alpha+\mu_\phi+j}(X)\right) + \\ & \textstyle\sum_{j=1}^{\mu_\psi} \psi_j(\boldsymbol{\beta})(v_{\psi,1+\mu_\beta+j}(X) + v_{\chi,1+\mu_\alpha+\mu_\phi+\mu_\beta+j}(X)) \ . \end{aligned} \tag{8}$$

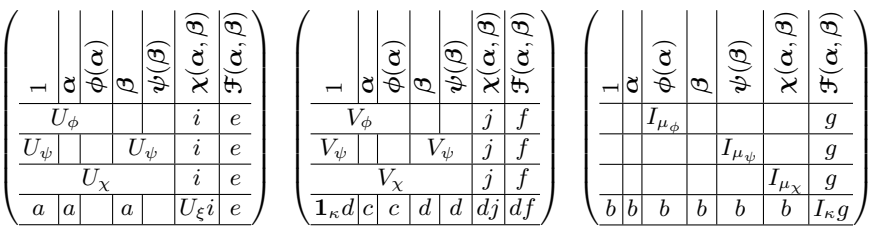
(In particular, recall that $\mathsf{a}_1 = 1$.) Here, $u_1(X) = u_{\phi 1}(X) + u_{\psi 1}(X) + u_{\chi 1}(X)$ and $v_1(X) = v_{\phi 1}(X) + v_{\psi 1}(X) + v_{\chi 1}(X) + \sum_{i=1}^{\kappa} \ell_{\nu-\kappa+i}(X)$. The concrete shape of all these polynomials follows from Eqs. (3) and (6).

In Theorems 2 and 3(see their claims and proofs), we will need several conditions to hold. Next, we will state and prove that these conditions hold for $\mathcal{C}^*$. One can observe directly that most of the guarantees, given by $\mathcal{C}^*$ about the shape of U, V, W, are actually required by the following conditions. Since the addition of the circuit $\mathcal{C}_\xi$ is essentially for free (it only means the addition of κ gates), many of the following conditions are very easy to satisfy; we will denote such conditions by a superscript + as in $(\mathrm{a})^+$. *We emphasize that the only restrictive conditions are Items i and j that basically state that $\mathcal{C}_\chi$ can only have multiplicative depth* 1. (See Remark 1 for discussion.) That is, the new SFC scheme will work for all circuits $\mathcal{C}$ that have a polynomial-size compiled circuit $\mathcal{C}^*$, such that $\mathcal{C}_\chi$ has multiplicative depth 1.

Theorem 1. *Let $\mathcal{C}$ be an arithmetic circuit and let $\mathcal{C}^*$ be its compiled version, so that U, V, W are defined as in Eq.* (6)*. Then the following holds.*

(a)$^+$ For $j \in [1 \,..\, \mu - \kappa]$: if $\boldsymbol{W}^{(j)} = \mathbf{0}$ then $U_{\nu-\kappa+i,j} = 0$ for $i \in [1 \,..\, \kappa]$.
(b)$^+$ For $I \in [1 \,..\, \kappa]$ and $j \in [1 \,..\, \mu - \kappa]$, $W_{\nu-\kappa+I,j} = 0$.
(c)$^+$ For $j \in [2 \,..\, 1 + \mu_\alpha + \mu_\phi]$, $v_{\phi j}(X), v_{\chi j}(X)$ are in the span of $(\ell_i(X))_{i=1}^{\nu-\kappa}$.
(d)$^+$ For $j \in [2 + \mu_\alpha + \mu_\phi \,..\, \mu]$, $v_1(X) - \sum_{i=1}^{\kappa} \ell_{\nu-\kappa+i}(X)$ and $v_j(X)$ are in the span of $(\ell_i(X))_{i=1}^{\nu-\kappa}$.
(e)$^+$ For $j \in [\mu - \kappa \,..\, \mu]$, $\boldsymbol{U}^{(j)} = \mathbf{0}$.
(f)$^+$ For $j \in [\mu - \kappa \,..\, \mu]$, $\boldsymbol{V}^{(j)} = \mathbf{0}$.
(g)$^+$ For $i \in [1 \,..\, \kappa]$, $w_{\mu-\kappa+i}(X) = \ell_{\nu-\kappa+i}(X)$.
(h) The set of non-zero $\boldsymbol{W}^{(j)}$, $j \in [1 \,..\, \mu - \kappa]$, is linearly independent.
(i) For $j \in [\mu - \mu_\chi - \kappa + 1 \,..\, \mu - \kappa]$, $U_{ij} = 0$ if $i \le \nu - \kappa$, while the last κ rows of this column range define a matrix U_ξ that satisfies Eq. (4)*.*
(j) For $j \in [\mu - \mu_\chi - \kappa + 1 \,..\, \mu - \kappa]$, $\boldsymbol{V}^{(j)} = \mathbf{0}$.

Proof. First, we summarize the requirements, denoting each submatrix of U, V, and W by the number of condition that ascertains that this submatrix is 0 (or has a well-defined non-zero form); moreover, Item h states that the columns of W, that contain identity matrices, are linearly independent. That is, $U, V, W =$



Item a: follows since $\boldsymbol{W}^{(j)} = \mathbf{0}$ in the columns labeled by 1, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and the last rows of U in all these columns are equal to 0, according to Eq. (6).

Item b: obvious from W in Eq. (6).

Item c: follows since the last rows of V, corresponding to columns labeled by $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, are equal to 0.

Item d: follows since the last rows of V, corresponding to columns labeled by $\boldsymbol{\beta}$, $\boldsymbol{\psi}(\boldsymbol{\beta})$, $\boldsymbol{\chi}(\boldsymbol{\alpha}, \boldsymbol{\beta})$, and $\boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$, are equal to 0, and the last rows of $\boldsymbol{V}^{(1)}$ are equal to $\mathbf{1}_\kappa$.

Items e to g, i and j: follows from direct observation.

Item h: follows from the fact that $\boldsymbol{W}^{(j)} = \mathbf{0}$ for some columns j, and the submatrix of W that consists of the rest of the columns is an identity matrix. □

Remark 1. The compiled circuit $\mathcal{C}^*$ satisfies some conditions, not required by Theorem 1. First, by Item h, the set of non-zero $\boldsymbol{W}^{(j)}$ has to be linearly independent (not necessarily an identity matrix), while in Eq. (6), the corresponding columns constitute an identity matrix. Second, by Item a, last rows of $\boldsymbol{U}^{(j)}$ need to be zero only if $\boldsymbol{W}^{(j)}$ is 0; one can insert dummy gates to $\mathcal{C}^*$ such that W has no zero columns. This essentially just corresponds to the fact that we start with an arithmetic circuit and each constraint is about a concrete gate being correctly evaluated. Third, several submatrices of U, V, W are all-zero in our template while there is no actual need for that. For example, U_ξ can be generalized, and U_ϕ and U_ψ can also both depend on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. For the sake of simplicity, we stick to the presented compilation process, and leave the possible generalizations to future work.

SNARK-Related Techniques. Next, we follow [33] to derive polynomials related to the SNARK, underlying the new SFC. We simplify the derivation a bit, and refer to [33] for full generality. Let $\mathsf{A}(X, Y) = r_a + u(X)Y$ and $\mathsf{B}(X, Y) = r_b + v(X)Y$ for $r_a, r_b \leftarrow\!\!\$\, \mathbb{Z}_p$. ([33] considered the general case where $\mathsf{A}(X, Y) = r_a Y^\alpha + u(X)Y^\beta$ and $\mathsf{B}(X, Y) = r_b Y^\alpha + v(X)Y^\beta$ for some small integers α, β to be fixed later.) The addends r_a and r_b are needed to protect the secret information hidden by $\mathsf{A}(X, Y)$ and $\mathsf{B}(X, Y)$, and we use the indeterminate Y to simplify the security proofs. As with u and w, we divide the polynomials $\mathsf{A}, \mathsf{B}, \mathsf{C}$ into two addends: (i) a polynomial ($\mathsf{A}_s, \mathsf{B}_s$, C_{sp}), where A_s and B_s depend on $\boldsymbol{\alpha}$ but not on $\boldsymbol{\beta}$ while C_{sp} depends on both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and (ii) a polynomial ($\mathsf{A}_p, \mathsf{B}_p, \mathsf{C}_p$, resp.) that depends on public values ($\boldsymbol{\beta}$ and $\{\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})\}$) but not on $\boldsymbol{\alpha}$ otherwise.

(Such a division was not possible in [33] since there one did not work with a compiled circuit $\mathcal{C}^*$.) Then,

$$\begin{aligned} \mathsf{A}_s(X,Y) =& r_a + u_s(X)Y \ , \quad \mathsf{A}_p(X,Y) = u_p(X)Y \ , \\ \mathsf{B}_s(X,Y) =& r_b + v_s(X)Y \ , \quad \mathsf{B}_p(X,Y) = v_p(X)Y \ . \end{aligned} \tag{9}$$

For integer constants δ and η that we will fix later, define

$$\begin{aligned} \mathsf{C}(X,Y) =& (\mathsf{A}(X,Y) + Y^\delta)(\mathsf{B}(X,Y) + Y^\eta) - Y^{\delta+\eta} \\ =& (r_a + u(X)Y + Y^\delta)(r_b + v(X)Y + Y^\eta) - Y^{\delta+\eta} \\ =& r_a(v(X)Y + Y^\eta) + r_b(\mathsf{A}(X,Y) + Y^\delta) + (u(X)v(X) - w(X))Y^2 + \\ & u(X)Y^{\eta+1} + v(X)Y^{\delta+1} + w(X)Y^2 \\ =& r_a(v(X)Y + Y^\eta) + r_b(\mathsf{A}(X,Y) + Y^\delta) + \mathcal{H}(X)\ell(X)Y^2 + \\ & u(X)Y^{\eta+1} + v(X)Y^{\delta+1} + w(X)Y^2 \ , \end{aligned}$$

where the last equation holds iff the committer is honest (see Eq. (2)). Intuitively, we want that a committer must be able to compute $\mathsf{C}(X,Y)$ iff he was honest.

Following [33], the inclusion of Y^δ and Y^η in the definition of $\mathsf{C}(X,Y)$ serves two goals. First, it introduces the addend $u(X)Y^{\eta+1} + v(X)Y^{\delta+1} + w(X)Y^2 = \sum_{j=1}^{\mu} \mathsf{a}_j(u_j(X)Y^{\eta+1} + v_j(X)Y^{\delta+1} + w_j(X)Y^2)$ that makes it easier to verify that P uses the same coefficients α_j when computing $[\mathsf{A}]_1$, $[\mathsf{B}]_2$, and $[\mathsf{C}]_1$. Second, the coefficient of Y^2 is $u(X)v(X) - w(X)$ that divides by $\ell(X)$ iff the committer is honest. That is, the coefficient of Y^2 is $\mathcal{H}(X)\ell(X)$ for some polynomial $\mathcal{H}(X)$ iff the prover is honest and thus $\boldsymbol{\xi} = \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

Let γ be another small integer, fixed later. Let $\mathsf{C}(X,Y) = \mathsf{C}_{sp}(X,Y) + \mathsf{C}_p(X,Y)Y^\gamma$, where $\mathsf{C}_p(X,Y)$ depends only on $\boldsymbol{\xi}$. (In [33], $\mathsf{C}_{sp}(X,Y)$ was multiplied with Y^α but here $\alpha = 0$.) The factor Y^γ is used to "separate" the public and the secret parts. In the honest case,

$$\begin{aligned} \mathsf{C}_{sp}(X,Y) =& r_a(v(X)Y + Y^\eta) + r_b(\mathsf{A}(X,Y)Y + Y^\delta) + \\ & \mathcal{H}(X)\ell(X)Y^2 + \textstyle\sum_{j=1}^{\mu-\kappa} \mathsf{a}_j(u_j(X)Y^{\eta+1} + v_j(X)Y^{\delta+1} + w_j(X)Y^2) \ , \\ \mathsf{C}_p(X,Y) =& \textstyle\sum_{j=\mu-\kappa+1}^{\mu} \mathsf{a}_j(u_j(X)Y^{\eta+1-\gamma} + v_j(X)Y^{\delta+1-\gamma} + w_j(X)Y^{2-\gamma}) \\ =& \textstyle\sum_{i=1}^{\kappa} \mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})(u_{\mu-\kappa+i}(X)Y^{\eta+1-\gamma} + v_{\mu-\kappa+i}(X)Y^{\delta+1-\gamma} + w_{\mu-\kappa+i}(X)Y^{2-\gamma}) \ . \end{aligned}$$

Intuitively, the verifier checks that $\mathsf{C}_{sp}(X,Y)$ is correctly computed by checking that $\mathcal{V}(X,Y) = 0$, where

$$\begin{aligned} \mathcal{V}(X,Y) :=& (\mathsf{A}_s(X,Y) + \mathsf{A}_p(X,Y) + Y^\delta)(\mathsf{B}_s(X,Y) + \mathsf{B}_p(X,Y) + Y^\eta) - \\ & (\mathsf{C}_{sp}(X,Y) + \mathsf{C}_p(X,Y)Y^\gamma) - Y^{\delta+\eta} \ . \end{aligned}$$

Here, $(\mathsf{A}_s, \mathsf{B}_s)$ (the part of (A, B) that only depends on private information) is the functional commitment, C_{sp} is the opening, and A_p, B_p, and C_p can be recomputed by the verifier given public information.

$\mathsf{KC}(1^\lambda, \mathcal{C})$**:** $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda, \nu)$; Let $\mathcal{C}^*$ be the compiled arithmetic circuit; $\mathcal{C}^*$ defines ν, μ, and other parameters. For $\mathsf{tk} = (\chi, y) \leftarrow\!\!\$\ (\mathbb{Z}_p^*)^2$ s.t. $\chi^\nu \neq 1$, $\mathsf{ck} =$

$$\begin{pmatrix} [1, (\chi^i y)_{i=0}^{\nu-1}, y^\eta, (\chi^i \ell(\chi) y^2)_{i=0}^{\nu-2}, (u_j(\chi) y^{\eta+1} + v_j(\chi) y^{\delta+1} + w_j(\chi) y^2)_{j=1}^{\mu-\kappa}]_1, \\ [(u_{\mu-\kappa+i}(\chi) y^{\eta+1-\gamma} + v_{\mu-\kappa+i}(\chi) y^{\delta+1-\gamma} + w_{\mu-\kappa+i}(\chi) y^{2-\gamma})_{i=1}^{\kappa}, y^\delta]_1, \\ [1, (\chi^i y)_{i=0}^{\nu-1}, y^\gamma, y^\eta]_2, [y^{\delta+\eta}]_T \end{pmatrix}.$$

Return $(\mathsf{ck}, \mathsf{tk})$;

$\mathsf{com}(\mathsf{ck}; \boldsymbol{\alpha}; r_a, r_b)$**:** // $r_a, r_b \leftarrow\!\!\$\ \mathbb{Z}_p$;
Compute $(\mathsf{a}_j)_{j=2}^{\mu_\alpha+\mu_\phi+1}$ from $\boldsymbol{\alpha}$;
Let $\mathsf{A}_s(X,Y) \leftarrow r_a + \sum_{i=0}^{\nu-1} A_i X^i Y$ be as in Eq. (9);
Let $\mathsf{B}_s(X,Y) \leftarrow r_b + \sum_{i=0}^{\nu-1} B_i X^i Y$ be as in Eq. (9);
For $i \in [1\,..\,\kappa]$: $\mathsf{B}_i^{\mathsf{aux}}(X,Y) \leftarrow \ell_{\nu-\kappa+i}(X)\mathsf{B}_s(X,Y)Y$;
$[\mathsf{A}_s]_1 \leftarrow r_a[1]_1 + \sum_{i=0}^{\nu-1} A_i[\chi^i y]_1$; $[\mathsf{B}_s]_2 \leftarrow r_b[1]_2 + \sum_{i=0}^{\nu-1} B_i[\chi^i y]_2$;
For $i \in [1\,..\,\kappa]$: $[\mathsf{B}_i^{\mathsf{aux}}]_1 \leftarrow [\mathsf{B}_i^{\mathsf{aux}}(\chi, y)]_1$;
$C \leftarrow ([\mathsf{A}_s, \{\mathsf{B}_i^{\mathsf{aux}}\}_{i=1}^{\kappa}]_1, [\mathsf{B}_s]_2)$; $D \leftarrow (\boldsymbol{\alpha}, r_a, r_b)$; return (C, D);

$\mathsf{open}(\mathsf{ck}; C = ([\mathsf{A}_s, \{\mathsf{B}_i^{\mathsf{aux}}\}_{i=1}^{\kappa}]_1, [\mathsf{B}_s]_2), D = (\boldsymbol{\alpha}, r_a, r_b), \boldsymbol{\beta})$**:**
Compute $\mathbf{a}$ from $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$;
Compute $[(\ell_j(\chi) y)_{j=1}^{\nu}]_1$ from $[(\chi^i y)_{i=0}^{\nu-1}]_1$; // Needs to be done once
Compute $[(u_j(\chi) y, v_j(\chi) y)_{j=1}^{\mu}]_1$ from $[(\chi^i y)_{i=0}^{\nu-1}]_1$; // Needs to be done once
Compute $[(w_j(\chi) y)_{j=1}^{\mu}]_1$ from $[(\chi^i y)_{i=0}^{\nu-1}]_1$; // Needs to be done once
$u(X) \leftarrow \sum_{j=1}^{\mu} \mathsf{a}_j u_j(X)$; $v(X) \leftarrow \sum_{j=1}^{\mu} \mathsf{a}_j v_j(X)$; $w(X) \leftarrow \sum_{j=1}^{\mu} \mathsf{a}_j w_j(X)$;
$\mathcal{H}(X) \leftarrow (u(X)v(X) - w(X))/\ell(X)$;
$[\mathsf{A}_p]_1 \leftarrow [\mathsf{A}_p(\chi, y)]_1$ where $\mathsf{A}_p(X,Y)$ is as in Eq. (9);
$[\mathsf{C}_{sp}]_1 \leftarrow r_a([v(\chi)y]_1 + [y^\eta]_1) + r_b([\mathsf{A}_s]_1 + [\mathsf{A}_p]_1 + [y^\delta]_1) +$
$\sum_{i=0}^{\nu-2} \mathcal{H}_i[\chi^i \ell(\chi) y^2]_1 + \sum_{j=1}^{\mu-\kappa} \mathsf{a}_j[u_j(\chi) y^{\eta+1} + v_j(\chi) y^{\delta+1} + w_j(\chi) y^2]_1$;
return $\mathsf{op}_{\boldsymbol{\xi}} \leftarrow [\mathsf{C}_{sp}]_1$;

$\mathsf{V}(\mathsf{ck}, C = ([\mathsf{A}_s, \{\mathsf{B}_i^{\mathsf{aux}}\}_{i=1}^{\kappa}]_1, [\mathsf{B}_s]_2), [\mathsf{C}_{sp}]_1, \boldsymbol{\beta}, \{\xi_i\}_{i=1}^{\kappa})$**:** // $\xi_i =^? \mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})$
Compute $[(\ell_{\nu-\kappa+i}(\chi) y)_{i=1}^{\kappa}]_1$ from $[(\chi^i y)_{i=0}^{\nu-1}]_1$; // Needs to be done once
Compute $[(\ell_{\nu-\kappa+i}(\chi) y^{2-\gamma})_{i=1}^{\kappa}]_1$ from $[(\chi^i y^{2-\gamma})_{i=0}^{\nu-1}]_1$; // Needs to be done once
Compute $[(\ell_{\nu-\kappa+i}(\chi) y)_{i=1}^{\kappa}]_2$ from $[(\chi^i y)_{i=0}^{\nu-1}]_2$; // Needs to be done once
Compute needed $[u_j(\chi) y]_1$ from $[(\chi^i y)_{i=0}^{\nu-1}]_2$; // done once
Compute needed $[v_j(\chi) y]_2$ from $[(\chi^i y)_{i=0}^{\nu-1}]_2$; // done once
$[\mathsf{A}_p]_1 \leftarrow [\mathsf{A}_p(\chi, y)]_1$ where $\mathsf{A}_p(X,Y)$ is as in Eq. (9);
$[\mathsf{B}_p]_2 \leftarrow [\mathsf{B}_p(\chi, y)]_2$ where $\mathsf{B}_p(X,Y)$ is as in Eq. (9);
$[\mathsf{C}_p]_1 \leftarrow \sum_{i=1}^{\kappa} \xi_i [\ell_{\nu-\kappa+i}(\chi) y^{2-\gamma}]_1$;
Check $([\mathsf{A}_s]_1 + [\mathsf{A}_p]_1 + [y^\delta]_1) \bullet ([\mathsf{B}_s]_2 + [\mathsf{B}_p]_2 + [y^\eta]_2) = [\mathsf{C}_{sp}]_1 \bullet [1]_2 + [\mathsf{C}_p]_1 \bullet [y^\gamma]_2 + [y^{\delta+\eta}]_T$;
For $i \in [1\,..\,\kappa]$: check $[\ell_{\nu-\kappa+i}(\chi) y]_1 \bullet [\mathsf{B}_s]_2 = [\mathsf{B}_i^{\mathsf{aux}}]_1 \bullet [1]_2$;

Fig. 2. SNARK-based SFC scheme $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ for arithmetic circuit $\mathcal{C}$

The New SFC Scheme $\mathsf{FC}_{\mathsf{sn}}$: Details. We are now ready to describe the new succinct functional commitment scheme $\mathsf{FC}_{\mathsf{sn}}$, see Fig. 2. Here, instead of operating with bivariate polynomials like $\mathsf{A}(X,Y)$, one operates with their encodings like $[\mathsf{A}_s(\chi, y)]_\iota$ in the source groups, where χ and y are secret trapdoors. The

commitment key of the SFC scheme contains the minimal amount of information needed to perform commitment, opening, and verification by honest parties. The expression of ck in KC has a generic form; one can replace the polynomials $u_j(X)$, $v_j(X)$, $w_j(X)$ with their values evident from Eq. (6). Finally, $\ell_j(X)$ (and thus also $u_j(X)$, $v_j(X)$, and $w_j(X)$) has degree $\nu-1$ and can thus be computed from $(X^i)_{i=0}^{\nu-1}$, while $\ell(X)$ has degree ν. We explain in the correctness proof of Theorem 3 how to compute $[\mathsf{B}_i^{\mathsf{aux}}(\chi,y)]_1$.

Note that $\mathsf{FC}_{\mathsf{sn}}$ can also be seen as a SNARK proving that $\mathcal{F}(\boldsymbol{\alpha},\boldsymbol{\beta})=\boldsymbol{\xi}$, if we let the prover to compute $[\mathsf{A}_p]_1$, $[\mathsf{B}_p]_2$, and $[\mathsf{C}_p]_1$.

Instantiation. Let $\mathcal{C}$ be a fixed circuit. Let $\mathcal{R}$ and $\mathcal{S}$ be two sets of bivariate polynomials, such that the commitment key of $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ is equal to $\mathsf{ck}=([\mathcal{R}(\chi,y)]_1,[\mathcal{S}(\chi,y)]_2)$. Similarly to [33], let

$$\mathsf{Mon}_1=\{0,1,2,2-\gamma,\delta,\delta+1,\delta+1-\gamma,\eta,\eta+1,\eta+1-\gamma\} \tag{10}$$

be the set of exponents of Y in all polynomials from $\mathcal{R}$. Let $\mathsf{Crit}=\{2,\eta+1\}$ and $\overline{\mathsf{Crit}}=\mathsf{Mon}_1\setminus\mathsf{Crit}$. For the evaluation-binding proof to hold, we need to fix values of $\gamma,\delta,\eta\in\mathbb{Z}_p$, such that the coefficients from Crit are unique, i.e.,

$$2,\eta+1\notin\{0,1,2-\gamma,\delta,\delta+1,\delta+1-\gamma,\eta,\eta+1-\gamma\}\quad\text{and}\quad\eta+1\neq 2\ . \tag{11}$$

That is, $\mathsf{Crit}\cap\overline{\mathsf{Crit}}=\emptyset$ and $|\mathsf{Crit}|=2$. It follows from Theorem 1 that the polynomial $f_i(X,Y):=\ell_{\nu-\kappa+i}(X)Y^{\eta+1}$, $i\in[1\,..\,\kappa]$, does not belong to $\mathrm{span}(\mathcal{R})$.

We will later consider two different evaluations for γ, δ, and η. Replacing γ, δ, and η with 1, 0, and 3 guarantees that Eq. (11) holds (see Theorem 2, Item 1, for more). Then,

$$\mathsf{ck}=\begin{pmatrix}[1,(\chi^i y)_{i=0}^{\nu-1},y^3,(\chi^i\ell(\chi)y^2)_{i=0}^{\nu-2},(u_j(\chi)y^4+v_j(\chi)y^1+w_j(\chi)y^2)_{j=1}^{\mu-\kappa}]_1,\\ [(u_{\mu-\kappa+i}(\chi)y^3+v_{\mu-\kappa+i}(\chi)y^0+w_{\mu-\kappa+i}(\chi)y^1)_{i=1}^{\kappa},y^0]_1,\\ [1,(\chi^i y)_{i=0}^{\nu-1},y^1,y^3]_2,[y^3]_T\end{pmatrix}\ .$$

In this case, the ck has one element (namely, $[1]_1$) twice, and thus ck can be shortened by one element.

Alternatively, replacing γ, δ, and η with 4, 0, and 7 (this choice is sufficient for the evaluation-binding reduction to uber-assumption in $\mathbb{G}_T$ to work and will be explained in Theorem 2, Item 2), we get

$$\mathsf{ck}=\begin{pmatrix}[1,(\chi^i y)_{i=0}^{\nu-1},y^7,(\chi^i\ell(\chi)y^2)_{i=0}^{\nu-2},(u_j(\chi)y^8+v_j(\chi)y^1+w_j(\chi)y^2)_{j=1}^{\mu-\kappa}]_1,\\ [(u_{\mu-\kappa+i}(\chi)y^4+v_{\mu-\kappa+i}(\chi)y^{-3}+w_{\mu-\kappa+i}(\chi)y^{-2})_{i=1}^{\kappa},y^0]_1,\\ [1,(\chi^i y)_{i=0}^{\nu-1},y^4,y^7]_2,[y^7]_T\end{pmatrix}\ .$$

Then, ck has one element ($[1]_1$) twice, and thus it can be shortened.

Efficiency. The CRS length is $1+\nu+1+(\nu-1)+(\mu-\kappa)+\kappa+1=2\nu+\mu+2$ elements from $\mathbb{G}_1$, $\nu+3$ elements from $\mathbb{G}_2$, and 1 element from $\mathbb{G}_T$. In the case

of fixed γ, δ and η in the previous two paragraphs, the CRS length will shorten by 1 element of $\mathbb{G}_1$.

The functional commitment takes $(\nu+1)+\kappa(\nu+1)=(\kappa+1)(\nu+1)$ exponentiations in $\mathbb{G}_1$ and $\nu+1$ exponentiations in $\mathbb{G}_2$. The length of the functional commitment is $\kappa+1$ elements of $\mathbb{G}_1$ and 1 element of $\mathbb{G}_2$.

The opening takes $\mu_\beta+\mu_\psi+\kappa$ (to compute $[\mathsf{A}_p]_1$; note that $u_1(X)$ and other simular polynomials are precomputed), $\mu_\alpha+\mu_\beta+\mu_\phi+\mu_\psi$ (to compute $[v(\chi)y]_1$) and $2+(\nu-1)+(\mu-\kappa)=\nu+\mu-\kappa+1$ (to compute $[\mathsf{C}_{sp}]_1$) exponentiations in $\mathbb{G}_1$, in total, $\nu+\mu+\mu_\alpha+2\mu_\beta+\mu_\phi+2\mu_\psi+1$ exponentiations. The length of the opening is 1 element of $\mathbb{G}_1$.

The verification takes $(\mu_\beta+\mu_\psi+\kappa)+\kappa=\mu_\beta+\mu_\psi+2\kappa$ (to compute $[\mathsf{A}_p,\mathsf{C}_p]_1$) exponentiations in $\mathbb{G}_1$, $\mu_\beta+\mu_\psi$ (to compute $[\mathsf{B}_p]_2$) exponentiations in $\mathbb{G}_2$, and $2\kappa+3$ pairings. Here, we do not count computations (e.g., computation of $[\ell_{\nu-\kappa+i}(\chi)y]_1$ from $[(\chi^i y)_{i=0}^{\nu-1}]_1$) that are only done once per the CRS.

The real efficiency depends of course significantly on the concrete application. We will give some detailed examples in the full version [34].

4 On the Circuit Class and Example Applications

Next, we study the power of the implementable circuit class $\mathbf{CC}_{\Sigma\Pi\forall}$, and we show that many known functional commitment scheme are for functionalities that belong to this class, and thus can be implemented by $\mathsf{FC}_{\mathsf{sn}}$.

In this section, we assume basic knowledge of the algebraic complexity theory. See [37] for necessary background. $\mathbf{VP}$ is the class of polynomial families $\{f_n\}$, where f_n is an univariate polynomial of $\mathsf{poly}(n)$ variables of $\mathsf{poly}(n)$ degree that has an arithmetic circuit of $\mathsf{poly}(n)$ size [39]. $\Sigma\Pi\Sigma$ (resp., $\Sigma\Pi\Sigma\Pi$) is the class of depth-3 (resp., depth-4) circuits composed of alternating levels of sum and product gates with a sum gate at the top [37, Section 3.5]. *Sparse polynomials* are n-variate polynomials that have $\mathsf{poly}(n)$ monomials.

Recall that a compiled circuit $\mathcal{C}^*$ can evaluate a vector polynomial $\boldsymbol{f}(\boldsymbol{\alpha},\boldsymbol{\beta})=(f_i(\boldsymbol{\alpha},\boldsymbol{\beta}))_{i=1}^{\kappa}$ iff $\kappa\in\mathsf{poly}(\lambda)$ and each f_i can be written as

$$f_i(\boldsymbol{\alpha},\boldsymbol{\beta})=\sum\phi_j(\boldsymbol{\alpha})\psi_k(\boldsymbol{\beta})\ , \tag{12}$$

where all polynomials ϕ_j and ψ_k are in the complexity class $\mathbf{VP}$, and there are a polynomial number of additions in the representation Eq. (12) (thus, also a polynomial number of polynomials ϕ_j and ψ_k). We call such representation an *efficient* $\Sigma\Pi\forall$*-representation* (here, $\forall$ denotes "any") of $\boldsymbol{f}$, and we denote by $\mathbf{CC}_{\Sigma\Pi\forall}$ the class of circuits (or vector polynomials) that have an efficient $\Sigma\Pi\forall$-presentation. Clearly, $\mathsf{FC}_{\mathsf{sn}}$ can implement $\boldsymbol{f}$ iff $\boldsymbol{f}\in\mathbf{CC}_{\Sigma\Pi\forall}$.

It is clear that all sparse polynomials in $\mathbf{VP}$ have an efficient $\Sigma\Pi\forall$-representation, and thus $\mathsf{FC}_{\mathsf{sn}}$ can implement all sparse polynomials. However, we can do more. For example, consider the polynomial $f'(\alpha,\boldsymbol{\beta})=\prod_{i=1}^{n}(\alpha+\beta_i)$ for $n=\mathsf{poly}(\lambda)$. Since f' has 2^n monomials, it is not sparse. However, we can rewrite f' as $f'(\alpha,\boldsymbol{\beta})=\sum_{d=0}^{n}\alpha^d\sigma_{n-d}(\boldsymbol{\beta})$, where $\sigma_{n-d}(\boldsymbol{\beta})=\sum_{T\subseteq[1\,..\,n],|T|=d}\prod_{i\in T}\beta_i$ is

Table 1. Rewriting the functionalites of various SFC as sparse polynomials

Type	μ_α	μ_β	f_i
Inner-product commitment [25,29]	n	n	$\sum_{j=1}^{n} \alpha_j \beta_j$
Polynomial commitment [27]	n	n	$\sum_{j=0}^{n-1} \alpha_j \beta^j$
Vector commitment [12]	n	1	$\alpha_I = \sum_{j=1}^{n} \alpha_j e_{Ij}$
Accumulator [5,3]	1	n	$\sum_{j=0}^{\mu_\alpha - 1} \chi_{\boldsymbol{\alpha} j} \beta^j$
Evaluation-point commitment	1	n	$\sum_{j=0}^{n-1} \alpha^j \beta_j$
c-variate polynomial commitment [36,10]	$\binom{n+c}{c}$	c	$\sum \alpha_j \prod_{k=1}^{c} \beta_k^{j_k}$

the $(n-d)$th symmetric polynomial. There exists a $\Sigma\Pi\Sigma$ circuit of size $O(n^2)$, due to Ben-Or (see [37, Section 3.5]), that computes all n symmetric polynomials in parallel. Thus, f has an efficient $\Sigma\Pi\forall$ -representation, and thus $\mathsf{FC}_{\mathsf{sn}}$ can implement at least one non-sparse polynomial.

On the other hand, $\mathbf{CC}_{\Sigma\Pi\forall} \subseteq \mathbf{VP}$. To see that $\mathbf{CC}_{\Sigma\Pi\forall} \subsetneq \mathbf{VP}$, consider the polynomial $f''(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{i=1}^{n} (\alpha_i + \beta_i)$ for $n = \mathsf{poly}(\lambda)$. Since f'' has 2^n monomials, it is not sparse. Considering β_i as coefficients, it also has 2^n monomials in $\boldsymbol{\alpha}$ (the case of considering α_i as coefficients is dual), and thus any $\Sigma\Pi\forall$-representation of f'' requires at least 2^n addition gates. Since f'' can be implemented by a $\Pi\Sigma$ circuit [37], it means $\Pi\Sigma \not\subset \mathbf{CC}_{\Sigma\Pi\forall}$; however, clearly, $\Pi\Sigma \not\subset \mathbf{CC}_{\Sigma\Pi\forall}$ so $\mathbf{CC}_{\Sigma\Pi\forall}$ is incomparable to $\Pi\Sigma$. Thus

$$\text{the class of sparse polynomials} \subsetneq \mathbf{CC}_{\Sigma\Pi\forall} \subsetneq \mathbf{VP} \ .$$

It is an interesting open problem to characterize $\mathbf{CC}_{\Sigma\Pi\forall}$. Motivated by our analysis of α'', it seems we can implement all polynomials $f(\boldsymbol{\alpha}, \boldsymbol{\beta})$, where either the dimension μ_α of $\boldsymbol{\alpha}$ or the dimension μ_β of $\boldsymbol{\beta}$ is logarithmic in λ. Really, if $\mu_\alpha = O(\log \lambda)$ then there are at most $2^\mu_\alpha = \mathsf{poly}(\lambda)$ possible monomials $\phi_i(\boldsymbol{\alpha})$ in $\boldsymbol{\alpha}$, and thus there exists an efficient $\Sigma\Pi\forall$-representation of f.

Known Types of SFCs as (Semi-)Sparse Polynomials. In Table 1, we write down the functionalities of several previous known types of SFCs. This shows that in all such cases, one has a sparse polynomial and thus can use $\mathsf{FC}_{\mathsf{sn}}$ to implement them. In none of these cases, one needs the power of non-sparse semi-sparse polynomials, and we leave it as another open question to find an application where such power is needed. In the case of the vector commitment scheme (resp., accumulator), one implements the inner-product scheme with $\boldsymbol{\beta} = \boldsymbol{e}_I$ (resp., $\chi_{\boldsymbol{\alpha}}(X) = \prod(X - \alpha_i)$). In the case of say the polynomial commitment scheme, $\boldsymbol{\beta} = (1, \beta, \ldots, \beta^{n-1})$ and thus $\mu_\beta = n$.

Aggregation. The next lemma is straightforward.

Lemma 1. *Assume that $\mathcal{C}_i \in \mathbf{CC}_{\Sigma\Pi\forall}$, where $i \in \{i\}Q$, and $Q = \mathsf{poly}(\lambda)$. Then their parallel composition $\mathcal{C}^{\|} = (\mathcal{C}_1 \| \ldots \| \mathcal{C}_Q) \in \mathbf{CC}_{\Sigma\Pi\forall}$.*

Proof. Obvious since we can just "parallelize" the representation in Eq. (12). □

In practice, Lemma 1 is very important since it means that $\mathsf{FC}_{\mathsf{sn}}$ allows to aggregate a polynomial number of SFCs for which $\mathsf{FC}_{\mathsf{sn}}$ is efficient. It just results in a larger circuit $\mathcal{C}^{\|}$(and thus larger parameters like μ and κ. However, as the length of the commitment in $\mathsf{FC}_{\mathsf{sn}}$ depends on κ, it means that the commitment stays succinct when $Q < |\mathsf{wit}|$. On the other hand, the length of the opening will be one group element, independently of Q.

As a corollary of Lemma 1, we can construct succinct aggregated inner-product SFCs, accumulators, (multi-point / multi-polynomial) polynomial commitment schemes, vector commitment schemes (including subvector commitment schmes), but also aggregate all these SFC variants with each other. Due to the lack of space, we will give more details and examples in the full version [34].

Example: Succinct Aggregated Inner-Product Functional Commitment. In an aggregated SIPFC, the committer commits to $\boldsymbol{\alpha}$ and then opens it simultaneously to $\langle \boldsymbol{\alpha}, \boldsymbol{\beta}_i \rangle = \sum_{j=1}^{n} \alpha_j \beta_{ij}$ for κ different verifier-provided vectors $\boldsymbol{\beta}_i$, where $i \in [1\,..\,\kappa]$. Assume $\boldsymbol{\alpha}$ and each $\boldsymbol{\beta}_i$ are n-dimensional vectors. There is no circuit $\mathcal{C}_\phi$ or $\mathcal{C}_\psi$. Given $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}_i$, $\mathcal{C}_\chi$ computes κn products $\chi_{ij}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \alpha_j \beta_{ij}$, $i \in [1\,..\,\kappa]$ and $j \in [1\,..\,n]$, and $\mathcal{C}_\xi$ sums them together to obtain κ outputs $\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{j=1}^{n} \alpha_j \beta_{ij}$. Thus, $U_\chi = \mathbf{1}_\kappa \otimes I_n \in \mathbb{Z}_p^{\kappa n \times n}$, $V_\chi = I_{\kappa n}$, $U_\xi = I_n \otimes \mathbf{1}_\kappa^\top \in \mathbb{Z}_p^{n \times \kappa n}$ (note that $\mathcal{C}_\chi$ does not take 1 as an input), and

$$U = \left(\begin{array}{c|c|c|c|c} 1 & \boldsymbol{\alpha} & \boldsymbol{\beta} & \boldsymbol{\chi}(\boldsymbol{\alpha},\boldsymbol{\beta}) & \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \hline & U_\chi & & & \\ \hline & & & U_\xi & \end{array}\right), \quad V = \left(\begin{array}{c|c|c|c|c} 1 & \boldsymbol{\alpha} & \boldsymbol{\beta} & \boldsymbol{\chi}(\boldsymbol{\alpha},\boldsymbol{\beta}) & \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \hline & & V_\chi & & \\ \hline 1 & & & & \end{array}\right), \quad W = \left(\begin{array}{c|c|c|c|c} 1 & \boldsymbol{\alpha} & \boldsymbol{\beta} & \boldsymbol{\chi}(\boldsymbol{\alpha},\boldsymbol{\beta}) & \boldsymbol{\mathcal{F}}(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \hline & & & I_{\kappa n} & \\ \hline & & & & I_\kappa \end{array}\right).$$

Here, $\nu = \kappa(n+1)$, $\mu = 1 + n + \kappa n + \kappa n + \kappa = (\kappa+1)n + \kappa + 1$, $\mathsf{A}_s(X,Y) = r_a + \sum_{j=1}^{n} \alpha_j u_{\chi j}(X) Y$, $\mathsf{A}_p(X,Y) = \sum_{i=1}^{\kappa} \langle \boldsymbol{\alpha}, \boldsymbol{\beta}_i \rangle \ell_{\nu-\kappa+i}(X) Y$. Importantly, $\mathsf{B}_s(X,Y) = 0$ (since there is nothing to hide, one can set $r_b \leftarrow 0$; hence, also $\mathsf{B}_i^{\mathsf{aux}}(X,Y) = 0$; thus the commitment is only one group element, $[\mathsf{A}_s]_1$), and $\mathsf{B}_p(X,Y) = \sum_{i=1}^{\kappa} \ell_{\nu-\kappa+i}(X) Y + \sum_{i=1}^{\kappa} \sum_{j=1}^{n} \beta_{ij} v_{\chi,n(i-1)+j}(X) Y$. The verifier has to execute 2κ exponentiations in $\mathbb{G}_1$ to compute $[\mathsf{A}_p]_1$ and $[\mathsf{C}_p]_1$, κn exponentiations in $\mathbb{G}_2$ to compute $[\mathsf{B}_p]_2$, and 3 pairings. We emphasize that here, both the functional commitment and the opening will consist of a single group element. One obtains IPFC by setting $\kappa \leftarrow 1$; in this case, the verification executes 2 exponentiations in $\mathbb{G}_1$, n exponentiations in $\mathbb{G}_2$, and 3 pairings.

Let us briefly compare the resulting *non-aggregated* IPFC with the IPFC of [25]. Interestingly, while the presented IPFC is a simple specialization of the general SFC scheme, it is only slightly less efficient than [25]. Let g_ι denote the bitlength of an element of the group $\mathbb{G}_\iota$. The CRS length is $2ng_1 + (n+1)g_2$ in [25], and $(3(\kappa+1) + (4\kappa+1)n)g_1 + (\kappa + \kappa n + 3)g_2 + 1g_T$ (this shortens to $(5n+6)g_1 + (n+4)g_2 + 1g_T$ when $\kappa = 1$) in our case. The commitment takes $n+1$ exponentiations in [25], and $n+2$ in our case. Interestingly, a straightforward [25]

opening takes $\Theta(n^2)$ multiplications (this can be probably optimized), while in our case it takes $\Theta(n \log n)$ multiplications. The verifier takes n exponentiations in [25], and $n+3$ here. The commitment and opening are both 1 group elements in both schemes. Thus, our *generic, unoptimized* scheme is essentially as efficient as the most efficient known prior IPFC, losing ground only in the CRS length. On the other hand, we are not aware of *any* previous aggregated IPFC schemes.

5 Security of $\mathsf{FC}_{\mathsf{sn}}$

Next, we prove the security of $\mathsf{FC}_{\mathsf{sn}}$. While its correctness and hiding proofs are straightforward, evaluation-binding is far from it. As before, for a fixed $\mathcal{C}$, let $\mathcal{R}$ and $\mathcal{S}$ be two sets of bivariate polynomials, s.t. $\mathsf{ck} = ([\mathcal{R}(\chi,y)]_1, [\mathcal{S}(\chi,y)]_2)$. For a fixed $\mathcal{C}$, in Theorem 3, we will reduce evaluation-binding of $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ to a $(\mathcal{R},\mathcal{S},\{f_i\})$-*span-uber-assumption* in $\mathbb{G}_1$, a new assumption that states that it is difficult to output an element $\sum \Delta_i [f_i(\chi,y)]_1$ together with the coefficient vector $\boldsymbol{\Delta} \neq \mathbf{0}$, where $f_i \notin \operatorname{span}(\mathcal{R})$. Thus, it is a generalization of the $(\mathcal{R},\mathcal{S},\cdot)$-computational uber-assumption in $\mathbb{G}_1$. Importantly, if $\kappa = 1$ then it is equivalent to the latter. To motivate the span-uber-assumption, we will show that it follows from the more conventional $(\mathcal{R},\mathcal{S},f'_I)$-computational uber-assumption (for a related set of polynomials f'_i) in $\mathbb{G}_T$ [7]; see Lemma 2. Thus, for the concrete parameters $\mathcal{R},\mathcal{S},\{f_i\}$, and $\{f'_i\}$,

uber-assumption in $\mathbb{G}_T$ $\Rightarrow$ span-uber-assumption in $\mathbb{G}_1$ $\Rightarrow$ uber-assumption in $\mathbb{G}_1$

For the reduction to the PDL and HAK assumptions in the full version [34] to work, we also prove that $f_i \notin \operatorname{span}(\mathcal{R})$ and $f'_i \notin \operatorname{span}(\mathcal{RS})$; see Theorem 2. (Intuitively, this is needed for the span-uber-assumptions to be secure in the generic model.) Each concrete proof (e.g., the proof of correctness, the proof of evaluation-binding, and the proofs that $f_i \notin \operatorname{span}(\mathcal{R})$ and $f'_i \notin \operatorname{span}(\mathcal{RS})$) puts some simple restrictions on the matrices U, V, W. They can usually be satisfied by slightly modifying the underlying arithmetic circuit.

Definition 6. *Let $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ be three tuples of bivariate polynomials over $\mathbb{Z}_p[X,Y]$. Let f_i be bivariate polynomial over $\mathbb{Z}_p[X,Y]$. The $(\mathcal{R},\mathcal{S},\mathcal{T},\{f_i\}_{i=1}^{\kappa})$* computational span-uber-assumption *for Pgen in group $\mathbb{G}_\iota$, where $\iota \in \{1,2,T\}$, states that for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{setuber}}_{\mathsf{Pgen},\mathcal{R},\mathcal{S},\mathcal{T},\{f_i\},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$, where $\mathsf{Adv}^{\mathrm{setuber}}_{\mathsf{Pgen},\mathcal{R},\mathcal{S},\mathcal{T},\{f_i\},\mathcal{A}}(\lambda) :=$*

$$\Pr\left[\begin{array}{l}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); \chi, y \leftarrow_{\$} \mathbb{Z}_p^*; \\ \mathsf{ck} \leftarrow ([\mathcal{R}(\chi,y)]_1, [\mathcal{S}(\chi,y)]_2, [\mathcal{T}(\chi,y)]_T); \\ (\boldsymbol{\Delta} \in \mathbb{Z}_p^\kappa, [z]_\iota) \leftarrow \mathcal{A}(\mathsf{ck}) : \boldsymbol{\Delta} \neq \mathbf{0} \wedge [z]_\iota = \sum_{i=1}^{\kappa} \Delta_i [f_i(\chi,y)]_\iota\end{array}\right].$$

If $\kappa = 1$ then the $(\mathcal{R},\mathcal{S},\mathcal{T},\{f\})$ span-uber-assumption is the same as the $(\mathcal{R},\mathcal{S},\mathcal{T},f_1)$ uber-assumption: in this case the adversary is tasked to output $\mathbb{Z}_p \ni \Delta \neq 0$ and $\Delta[f_1(\chi,y)]_\iota$ which is equivalent to outputting $[f_1(\chi,y)]_\iota$.

We will now show that the used polynomials are linearly independent.

Theorem 2. *Write* $\mathsf{ck} = ([\varrho(X,Y) : \varrho \in \mathcal{R}]_1, [\sigma(X,Y) : \sigma \in \mathcal{S}]_2)$ *as in Fig. 2. For* $i \in [1\,..\,\kappa]$, *let* $f_i(X,Y) := \ell_{\nu-\kappa+i}(X)Y^{\eta+1}$ *and* $f'_i(X,Y) := (\ell_{\nu-\kappa+i}(X))^2 Y^{\eta+2}$.

1. *Assume* $\gamma = 1$, $\delta = 0$, *and* $\eta = 3$. *Assume Items a and h of Theorem 1 hold. Then* $f_i(X,Y) \notin \mathrm{span}(\mathcal{R})$ *for* $i \in [1\,..\,\kappa]$.
2. *Assume* $\gamma = 4$, $\delta = 0$, $\eta = 7$, *and that Items a, b and h of Theorem 1 hold. Then* $f'_i(X,Y) \notin \mathrm{span}(\mathcal{RS})$ *for* $i \in [1\,..\,\kappa]$.

Proof. **(1: $f_I \notin \mathrm{span}(\mathcal{R})$).** Let Mon_1 be as in Eq. (10) and $\mathsf{Crit} = \{2, \eta+1\}$. For the rest of the proof to make sense, as we will see in a few paragraphs, we need to fix γ, δ, and η so that the coefficients in Mon_1 and in $\mathsf{Mon}_1 \setminus \mathsf{Crit}$ are different (in particular, the coefficients in Crit are different from each other). A small exhaustive search shows that one can define $\gamma = 1$, $\delta = 0$, $\eta = 3$, as in the claim. This setting can be easily manually verified, by noticing that $\mathsf{Mon}_1 = \{0,1,2,3,4\}$, $\mathsf{Crit} = \{2,4\}$, and thus $\mathsf{Mon}_1 \setminus \mathsf{Crit} = \{0,1,3\}$.

Assume that, for some I, $f_I(X,Y) = \ell_{\nu-\kappa+I}(X)Y^{\eta+1}$ belongs to the span of $\mathcal{R}$. We consider the coefficients of Y^i, for $i \in \mathsf{Crit}$, in the resulting equality (for some unknown coefficients in front of the polynomials from $\mathcal{R}$), and derive a contradiction from this. Thus, we write down an arbitrary linear combination of polynomials in $\mathcal{R}$ as a linear combination of $u_j(X)Y^{\eta+1} + v_j(X)Y^{\delta+1} + w_j(X)Y^2$, $X^i\ell(X)Y^2$, and $T(X,Y)$, where $T(X,Y)$ is some polynomial with monomials that do not have Y^i for $i \in \mathsf{Crit}$. That is,

$$\begin{aligned}\ell_{\nu-\kappa+I}(X)Y^{\eta+1} = \textstyle\sum_{j=1}^{\mu-\kappa} t'_j(u_j(X)Y^{\eta+1} + v_j(X)Y^{\delta+1} + w_j(X)Y^2) + \\ t(X)\ell(X)Y^2 + T(X,Y)\end{aligned} \tag{13}$$

for some $t(X) \in \mathbb{Z}_p[X]$ (thus $t(X)\ell(X)Y^2$ encompasses all $X^i\ell(X)Y^2$) and integers t'_j.

First, considering only the coefficient of Y^2 in both the left-hand side and the right hand side of Eq. (13),

$$\textstyle\sum_{j=1}^{\mu-\kappa} t'_j w_j(X) + t(X)\ell(X) = 0 \ .$$

Due to Item h of Theorem 1, either $w_j(X) = 0$ or $t'_j = 0$ for $j \in [1\,..\,\mu-\kappa]$. Let $\mathcal{J} \subset [1\,..\,\mu-\kappa]$ be the set of indices $j \in [1\,..\,\mu-\kappa]$ so that $w_j(X) = 0$.

Second, considering only the coefficient of $Y^{\eta+1}$ in Eq. (13),

$$\ell_{\nu-\kappa+I}(X) = \textstyle\sum_{j=1}^{\mu-\kappa} t'_j u_j(X) = \sum_{j\in\mathcal{J}} t'_j u_j(X) \ .$$

Due to Item a of Theorem 1, $\ell_{\nu-\kappa+I}(X)$ is linearly independent of (the non-zero elements of) $\{u_j(X)\}_{j\in\mathcal{J}}$, a contradiction. Hence, $f_I(X,Y) \notin \mathrm{span}(\mathcal{R})$.

(Item 2: $f'_I \notin \mathrm{span}(\mathcal{RS})$). For the proof to make sense, as we will see in a few paragraphs, we need that the set of critical coefficients $\mathsf{Crit}' := \{3, \eta+2\}$ (that is different from Crit above) is different from the set $\mathsf{Mon}' \setminus \mathsf{Crit}'$ all other coefficients in $\mathcal{RS}$, where $\mathsf{Mon}' :=$

$$\left\{\begin{array}{l} 0, 1, 2, 3, 2-\gamma, 3-\gamma, \gamma, \gamma+1, \gamma+2, \delta, 1+\delta, 2+\delta, 1-\gamma+\delta, 2-\gamma+\delta, \\ \gamma+\delta, 1+\gamma+\delta, \eta, 2\eta, \eta+1, \eta+2, -\gamma+\eta+1, -\gamma+\eta+2, \gamma+\eta, \\ \gamma+\eta+1, \delta+\eta, 1+\delta+\eta, 1-\gamma+\delta+\eta, 1+2\eta, 1-\gamma+2\eta \end{array}\right\} \ .$$

is defined by $\mathsf{Mon}' = \mathsf{Mon}_1 + \mathsf{Mon}_2$, where Mon_1 is as in Eq. (10) and $\mathsf{Mon}_2 = \{0, 1, \gamma, \eta\}$ is the set of exponents of Y in all polynomials from $\mathcal{S}$. A small exhaustive search, performed by using computer algebra, shows that one can define $\gamma = 4$, $\delta = 0$, $\eta = 7$, as in the claim. This setting can be easilymanually verified, by noticing that $\mathsf{Mon}' \setminus \mathsf{Crit}' = \{-3, -2, -1, 0, 1, 2, 4, 5, 6, 7, 8, 11, 12, 14, 15\}$ and $\mathsf{Crit}' = \{3, 9\}$.

Assume now in contrary that $f'_I \in \operatorname{span}(\mathcal{RS})$. Then, as in Item 1, $(\ell_{\nu-\kappa+I}(X))^2 Y^{\eta+2}$ is in the span of some polynomials containing Y^i for $i \in \mathsf{Crit}'$ (and we need to quantify the coefficients of these polynomials) and of all other polynomials. Clearly, the first type of polynomials are in the span of $X^i \ell(X) Y^2$ times Y^η, $X^i \ell(X) Y^2$ times $X^k Y$, $u_j(X) Y^{\eta+1} + v_j(X) Y^{\delta+1} + w_j(X) Y^2$ times Y^η, and $u_j(X) Y^{\eta+1} + v_j(X) Y^{\delta+1} + w_j(X) Y^2$ times $X^k Y$, for properly chosen i, j, and k. Thus,

$$\begin{aligned}(\ell_{\nu-\kappa+I}(X))^2 Y^{\eta+2} =& t(X)\ell(X)Y^{\eta+2} + t''(X)\ell(X)Y^3 + \\ & \textstyle\sum_{j=1}^{\mu-\kappa} t'_j(X)(u_j(X)Y^{2\eta+1} + v_j(X)Y^{\delta+\eta+1} + w_j(X)Y^{\eta+2}) + \\ & \textstyle\sum_{j=1}^{\mu-\kappa} t^*_j(X)(u_j(X)Y^{\eta+2} + v_j(X)Y^{\delta+2} + w_j(X)Y^3) + T(X,Y)\end{aligned}$$

where $t'_j(X)$, $t^*_j(X)$, $t(X)$ and $t''(X)$ are univariate polynomials, and $T(X, Y)$ is a polynomial that does not contain monomials with Y^i, $i \in \mathsf{Crit}'$. We now consider separately the coefficients of Y^i in this equation for each $i \in \mathsf{Crit}'$ and derive a contradiction.

First, considering the coefficients of Y^3, we get $\sum_{j=1}^{\mu-\kappa} t^*_j(X) w_j(X) + t''(X)\ell(X) = 0$. Due to Item h of Theorem 1, either $t^*_j(X) = 0$ or $w_j(X) = 0$ for $1 \le j \le \mu - \kappa$. Let $\mathcal{J} \subset [1 \,..\, \mu - \kappa]$ be the set of indices j so that $w_j(X) = 0$.

Second, the coefficients of $Y^{\eta+2}$ give us

$$\begin{aligned}(\ell_{\nu-\kappa+I}(X))^2 &= \textstyle\sum_{j=1}^{\mu-\kappa} t^*_j(X)u_j(X) + \sum_{j=\mu_\alpha+2}^{\mu-\kappa} t'_j(X)w_j(X) + t(X)\ell(X) \\ &= \textstyle\sum_{j\in\mathcal{J}} t^*_j(X)u_j(X) + \sum_{j\notin\mathcal{J}} t'_j(X)w_j(X) + t(X)\ell(X)\ .\end{aligned}$$

Due to Items a, b and h of Theorem 1 (and of the fact that $(\ell_{\nu-\kappa+i}(X))^2$ has degree 2ν), $\{(\ell_{\nu-\kappa+I}(X))^2\} \cup \{u_j(X)\}_{j\in\mathcal{J}} \cup \{w_j(X)\}_{j\notin\mathcal{J}} \cup \{X^i\ell(X)\}_{i=0}^{\nu-2}$ is linearly independent. Contradiction, and thus $f'_I(X, Y) \notin \operatorname{span}(\mathcal{RS})$. □

Next, we show that for the concrete choice of the parameters $\mathcal{R}$, $\mathcal{S}$, f_i, and f'_i, the span-uber-assumption in $\mathbb{G}_1$ is at least as strong as the uber-assumption in $\mathbb{G}_T$. The new assumption may be weaker since the latter assumption argues about elements in $\mathbb{G}_T$, which may not always be possible [26]. However, the proof of Lemma 2 depends crucially on the concrete parameters.

Lemma 2 (Uber-assumption in $\mathbb{G}_T \Rightarrow$ span-uber-assumption). *Assume $\gamma = 4$, $\delta = 0$, and $\eta = 7$. Let $\mathsf{FC}^{\mathcal{C}}_{\mathsf{sn}}$ be the SFC scheme for arithmetic circuits in Fig. 2. Write $\mathsf{ck} = ([\varrho(X, Y) : \varrho \in \mathcal{R}]_1, [\sigma(X, Y) : \sigma \in \mathcal{S}]_2)$ as in Fig. 2. For $i \in [1 \,..\, \kappa]$, let $f_i(X, Y) := \ell_{\nu-\kappa+i}(X) Y^{\eta+1}$ and $f'_i(X, Y) := (\ell_{\nu-\kappa+i}(X))^2 Y^{\eta+2}$. If the $(\mathcal{R}, \mathcal{S}, f'_I)$ computational uber-assumption holds in $\mathbb{G}_T$ for each $I \in [1 \,..\, \kappa]$ then the $(\mathcal{R}, \mathcal{S}, \{f_i\}_{i=1}^{\kappa})$ computational span-uber-assumption holds in $\mathbb{G}_1$.*

Proof (Sketch). Assume $\mathcal{A}$ is an adversary against the $(\mathcal{R}, \mathcal{S}, \{f_i\}_{i=1}^{\kappa})$ computational span-uber-assumption that has successfully output $\boldsymbol{\Delta} \neq \mathbf{0}$ and $[z]_1 = \sum_{i=1}^{\kappa} \Delta_i [f_i(\chi, y)]_1 = \sum_{i=1}^{\kappa} \Delta_i [\ell_{\nu-\kappa+i}(X) Y^{\eta+1}]_1$.

Since $\boldsymbol{\Delta} \neq \mathbf{0}$, then there exists at least one coordinate I such that $\Delta_I \neq 0$. Let $\mathcal{B}$ be the following adversary against the $(\mathcal{R}, \mathcal{S}, f_I)$ computational uber-assumption in $\mathbb{G}_T$. Given ck and $[z]_1$, $\mathcal{B}$ computes

$$1/\Delta_I \cdot [z]_1 \bullet [\ell_{\nu-\kappa+I}(\chi) y]_2 = \sum_{i=1}^{\kappa} \Delta_i/\Delta_I \cdot [\ell_{\nu-\kappa+i}(\chi) y^{\eta+1}]_1 \bullet [\ell_{\nu-\kappa+I}(\chi) y]_2 \ .$$

Let $d_i(X)$ be the rational function satisfying $d_i(X)\ell(X) = \ell_{\nu-\kappa+i}(X)\ell_{\nu-\kappa+I}(X)$. Clearly, $d_i(X)$ is a polynomial for $i \neq I$. Thus, $d(X) := \sum_{i \neq I} \Delta_i/\Delta_I \cdot d_i(X)$ is a polynomial of degree $\leq \nu - 2$. Since $[y^\eta]_2$ is a part of the commitment key, $\mathcal{B}$ can efficiently compute $\sum_{i \neq I} \Delta_i/\Delta_I \cdot [\ell_{\nu-\kappa+i}(\chi) y^{\eta+1}]_1 \bullet [\ell_{\nu-\kappa+I}(\chi) y]_2 = \sum_{i \neq I} \Delta_i/\Delta_I \cdot [d_i(\chi)\ell(\chi) y^2]_1 \bullet [y^\eta]_2 = [d(\chi)\ell(\chi) y^2]_1 \bullet [y^\eta]_2$. Thus, $\mathcal{B}$ can compute

$$\begin{aligned}[z^*]_T =& [f_I(\chi, y)]_T \leftarrow [\ell_{\nu-\kappa+I}(\chi) y^{\eta+1}]_1 \bullet [\ell_{\nu-\kappa+I}(\chi) y]_2 \\ =& 1/\Delta_I \cdot [z]_1 \bullet [\ell_{\nu-\kappa+I}(\chi) y]_2 - [d(\chi)\ell(\chi) y^2]_1 \bullet [y^\eta]_2\end{aligned}$$

and break the $(\mathcal{R}, \mathcal{S}, f'_I)$-computational uber-assumption in $\mathbb{G}_T$. □

Theorem 3 (Security of $\mathsf{FC}_{\mathsf{sn}}$). *Let $\mathcal{C}$ be a fixed circuit and let $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ be the SFC scheme in Fig. 2. Let $\mathsf{ck} = ([\varrho(X, Y) : \varrho \in \mathcal{R}]_1, [\sigma(X, Y) : \sigma \in \mathcal{S}]_2)$ as in Fig. 2. For $i \in [1 \,..\, \kappa]$, let $f_i(X, Y) := \ell_{\nu-\kappa+i}(X) Y^{\eta+1}$.*

1. *Assume Item c of Theorem 1 holds. Then $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ is correct.*
2. *$\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ is perfectly com-hiding.*
3. *$\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ is perfectly open-hiding.*
4. *$\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ is perfectly zero-knowledge.*
5. *Assume that either $\gamma = 1$, $\delta = 0$, and $\eta = 3$ or $\gamma = 4$, $\delta = 0$, and $\eta = 7$. Assume that Items d to g, i and j of Theorem 1 hold. If the $(\mathcal{R}, \mathcal{S}, \{f_i\})$-computational span-uber-assumption holds in $\mathbb{G}_1$ then the SFC scheme $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ is computationally evaluation-binding.*

Proof. **(1: correctness).** We first show that the prover can compute $\mathsf{B}_i^{\mathsf{aux}}(X, Y)$, and then that the verification equation holds. Recall that for $i \in [1 \,..\, \kappa]$, $\mathsf{B}_i^{\mathsf{aux}}(X, Y) = \ell_{\nu-\kappa+i}(X)\mathsf{B}_s(X, Y)Y = \ell_{\nu-\kappa+i}(X)(r_b + v_s(X, Y)Y)Y$, where $v_s(X)$ is as in Eq. (8). First, the addend $r_b \ell_{\nu-\kappa+i}(X) Y$ belongs to the span of $(X^i Y)_{i=0}^{\nu-1} \subset \mathcal{R}$. Second, due to Item c of Theorem 1, for all $j \in [2 \,..\, 1 + \mu_\alpha + \mu_\phi]$,

$$\ell(X) \mid \ell_{\nu-\kappa+i}(X) v_{\phi j}(X) \quad \text{and} \quad \ell(X) \mid \ell_{\nu-\kappa+i}(X) v_{\chi j}(X) \ ,$$

and thus $\mathsf{B}_i^{\mathsf{aux}}(X, Y) - r_b \ell_{\nu-\kappa+i}(X) Y$ is equal to $b'_i(X)\ell(X)Y^2$ for some polynomial $b'_i(X) \in \mathbb{Z}_p^{\leq(\nu-2)}[X]$. Thus, $\mathsf{B}_i^{\mathsf{aux}}(X) \in \mathrm{span}(\mathcal{R})$ and the committer can compute $[\ell_{\nu-\kappa+i}(\chi)\mathsf{B}_s y]_2 = [\mathsf{B}_i^{\mathsf{aux}}(\chi, y)]_2$.

Assume that $\mathsf{ck} \leftarrow \mathsf{KC}(1^\lambda, \mathcal{C})$, $([\mathsf{A}_s, \{\mathsf{B}_i^{\mathsf{aux}}\}_{i=1}^{\kappa}]_1, [\mathsf{B}_s]_2) \leftarrow \mathsf{com}(\mathsf{ck}; \boldsymbol{\alpha}; r_a, r_b)$ and $[\mathsf{C}_{sp}]_1 \leftarrow \mathsf{open}(\mathsf{ck}, ([\mathsf{A}_s, \{\mathsf{B}_i^{\mathsf{aux}}\}_{i=1}^{\kappa}]_1, [\mathsf{B}_s]_2), (\boldsymbol{\alpha}, r_a, r_b), \boldsymbol{\beta})$. It is clear that then the verifier accepts.

(2: perfect com-hiding). Follows from the fact that $([\mathsf{A}_s]_1, [\mathsf{B}_s]_2)$ is perfectly masked by uniformly random $r_a, r_b \leftarrow_\$ \mathbb{Z}_p$. Moreover, $[\mathsf{B}_i^{\mathsf{aux}}]_1$ are publicly verifiable functions of $[\mathsf{B}_s]_2$.

(3: perfect open-hiding). Due to com-hiding and the fact that $[\mathsf{A}_p]_1$, $[\mathsf{B}_p]_2$, and $[\mathsf{C}_p]_1$ only depend on $(\boldsymbol{\beta}, \{\mathcal{F}_i(\boldsymbol{\alpha}, \boldsymbol{\beta})\})$ (and not on $\boldsymbol{\alpha}$ otherwise), it means that the distribution of all elements in the opening (except possibly $[\mathsf{C}_{sp}]_1$) is the same for any two vectors $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ that satisfy $\mathcal{F}_i(\boldsymbol{\alpha}_1, \boldsymbol{\beta}) = \mathcal{F}_i(\boldsymbol{\alpha}_2, \boldsymbol{\beta})$ for all i, Since $[\mathsf{C}_{sp}]_1$ is the unique element that makes the verifier to accept, this means that the same claim holds for the whole opening, and $\mathsf{FC}^{\mathcal{C}}_{\mathsf{sn}}$ is open-hiding.

(4: perfect zero-knowledge). We construct Sim as follows. It has (χ, y) as the trapdoor. It samples random $\mathsf{A}_s, \mathsf{B}_s \leftarrow_\$ \mathbb{Z}_p$, and then sets $[\mathsf{B}_i^{\mathsf{aux}}]_1 \leftarrow [\ell_{\nu-\kappa+i}(\chi) y \mathsf{B}_s]_1$ for all i. It computes B_p (by using the trapdoors), $[\mathsf{A}_p]_1$, and $[\mathsf{C}_p]_1$. It then computes the unique $[\mathsf{C}_{sp}]_1$ that makes the verifier to accept,

$$[\mathsf{C}_{sp}]_1 \leftarrow ((\mathsf{A}_s + y^\delta)(\mathsf{B}_s + \mathsf{B}_p) + \mathsf{A}_s y^\eta)[1]_1 + (\mathsf{B}_s + \mathsf{B}_p + y^\eta)[\mathsf{A}_p]_1 - y^\gamma [\mathsf{C}_p]_1 \ .$$

(5: evaluation-binding). Assume that $\mathcal{A}$ is an evaluation-binding adversary that, with probability $\varepsilon_{\mathcal{A}}$ and in time $\tau_{\mathcal{A}}$, returns a collision $((([\mathsf{A}_s, \{\mathsf{B}_i^{\mathsf{aux}}\}_{i=1}^{\kappa}]_1, [\mathsf{B}_s]_2); \boldsymbol{\beta}; \{\xi_i\}, [\mathsf{C}_{sp}]_1, \{\tilde{\xi}_i\}, [\tilde{\mathsf{C}}_{sp}]_1)$ with $\boldsymbol{\xi} \neq \tilde{\boldsymbol{\xi}}$, such that (here, $[\mathsf{A}_p, \mathsf{C}_p]_1$ / $[\tilde{\mathsf{A}}_p, \tilde{\mathsf{C}}_p]_1$ is the opening in the collision),

$$[\mathsf{A}_s + \mathsf{A}_p + y^\delta]_1 \bullet [\mathsf{B}_s + \mathsf{B}_p + y^\eta]_2 = [\mathsf{C}_{sp}]_1 \bullet [1]_2 + [\mathsf{C}_p]_1 \bullet [y^\gamma]_2 + [y^{\delta+\eta}]_T \ ,$$
$$[\mathsf{A}_s + \tilde{\mathsf{A}}_p + y^\delta]_1 \bullet [\mathsf{B}_s + \mathsf{B}_p + y^\eta]_2 = [\tilde{\mathsf{C}}_{sp}]_1 \bullet [1]_2 + [\tilde{\mathsf{C}}_p]_1 \bullet [y^\gamma]_2 + [y^{\delta+\eta}]_T \ ,$$

and $[\ell_{\nu-\kappa+i}(\chi) y]_1 \bullet [\mathsf{B}_s]_2 = [\mathsf{B}_i^{\mathsf{aux}}]_1 \bullet [1]_2$ for $i \in [1 \,..\, \kappa]$. Here we used the fact that by Items f and j of Theorem 1 (see also the definition of $u_p(X)$ and $v_p(X)$ in Eqs. (7) and (8)), the value of $[\mathsf{B}_p]_2$ stays the same in both openings.

We now construct an adversary $\mathcal{B}$ against the computational uber-assumption in $\mathbb{G}_1$. From the collision, by subtracting the second equation from the first equation and then moving everything from $\mathbb{G}_T$ (the result of pairings) to $\mathbb{G}_1$,

$$[(\mathsf{A}_p - \tilde{\mathsf{A}}_p)(\mathsf{B}_s + \mathsf{B}_p + y^\eta)]_1 = [\mathsf{C}_{sp}]_1 - [\tilde{\mathsf{C}}_{sp}]_1 + [(\mathsf{C}_p - \tilde{\mathsf{C}}_p) y^\gamma]_1 \ . \tag{14}$$

Denote $\Delta_i := \xi_i - \tilde{\xi}_i$. Let $\mathbf{a}$ and $\tilde{\mathbf{a}}$ be witnesses, used by $\mathcal{A}$ when creating the collision. Without any further assumptions (see Eqs. (7) and (9)), $A_p(X) - \tilde{A}_p(X) = \sum_{j=\mu_\alpha+\mu_\phi+2}^{\mu} (\mathsf{a}_j - \tilde{\mathsf{a}}_j) u_j(X) Y = \sum_{j=\mu-\mu_\chi-\kappa}^{\mu-\kappa} (\mathsf{a}_j - \tilde{\mathsf{a}}_j) u_j(X) Y + \sum_{j=\mu-\kappa+1}^{\mu} \Delta_{j-(\mu-\kappa)} u_j(X) Y$. (This is since for $j \le \mu_\alpha + \mu_\phi + 1$, $\mathsf{a}_j = \tilde{\mathsf{a}}_j$ is either fixed by the commitment or can be recomputed by the verifier from $\boldsymbol{\beta}$ alone.) Thus, Eq. (14) is equivalent to

$$\begin{aligned} &\left(\textstyle\sum_{j=1}^{\mu_\chi} (\mathsf{a}_{\mu-\mu_\chi-\kappa+j} - \tilde{\mathsf{a}}_{\mu-\mu_\chi-\kappa+j}) u_{\xi j}(\chi) y + \sum_{i=1}^{\kappa} \Delta_i u_{\mu-\kappa+i}(\chi) y\right) (\mathsf{B}_s + \mathsf{B}_p + y^\eta) \\ &\quad = (\mathsf{C}_{sp} - \tilde{\mathsf{C}}_{sp}) + \textstyle\sum_{i=1}^{\kappa} \Delta_i \left(u_{\mu-\kappa+i}(\chi) y^{\eta+1} + v_{\mu-\kappa+i}(\chi) y^{\delta+1} + w_{\mu-\kappa+i}(\chi) y^2\right) \ . \end{aligned}$$

Assuming Items e to g of Theorem 1,

$$\textstyle\sum_{i=1}^{\mu_\chi} (\mathsf{a}_{\mu-\mu_\chi-\kappa+i} - \tilde{\mathsf{a}}_{\mu-\mu_\chi-\kappa+i}) u_{\xi j}(\chi) y (\mathsf{B}_s + \mathsf{B}_p + y^\eta) = (\mathsf{C}_{sp} - \tilde{\mathsf{C}}_{sp}) + \sum_{i=1}^{\kappa} \Delta_i \ell_{\nu-\kappa+i}(\chi) y^2.$$

Assuming additionally Item i of Theorem 1,

$$\sum_{i=1}^{\kappa} \Delta_i \left[\ell_{\nu-\kappa+i}(\chi) y (\mathsf{B}_s + \mathsf{B}_p + y^{\eta})\right]_1 = [\mathsf{C}_{sp}]_1 - [\tilde{\mathsf{C}}_{sp}]_1 + \sum_{i=1}^{\kappa} \Delta_i [\ell_{\nu-\kappa+i}(\chi) y^2]_1 \ .$$

Let $[z]_1 := \sum_{i=1}^{\kappa} \Delta_i [\ell_{\nu-\kappa+i}(\chi) y^{\eta+1}]_1 (= \sum \Delta_i [f_i(\chi, y)]_1)$. In what follows, we show that $\mathcal{B}$ can compute $[z]_1$ and thus break the span-uber-assumption. From the last displayed equation, we get

$$\begin{aligned}[z]_1 + \sum_{i=1}^{\kappa} \Delta_i [\ell_{\nu-\kappa+i}(\chi)(\mathsf{B}_p - y) y]_1 =& [\mathsf{C}_{sp}]_1 - [\tilde{\mathsf{C}}_{sp}]_1 - \sum_{i=1}^{\kappa} \Delta_i [\ell_{\nu-\kappa+i}(\chi) \mathsf{B}_s y]_1 \\ =& [\mathsf{C}_{sp}]_1 - [\tilde{\mathsf{C}}_{sp}]_1 - \sum_{i=1}^{\kappa} \Delta_i [\mathsf{B}_i^{\mathsf{aux}}]_1 \ .\end{aligned}$$

(The last equation is guaranteed by $[\ell_{\nu-\kappa+i}(\chi) y]_1 \bullet [\mathsf{B}_s]_2 = [\mathsf{B}_i^{\mathsf{aux}}]_1 \bullet [1]_2$.)

We now show how to efficiently compute $[\ell_{\nu-\kappa+i}(\chi)(\mathsf{B}_p - y) y]_1$. Let $t(X) = v_p(X) - \sum_{i=1}^{\kappa} \ell_{\nu-\kappa+i}(X)$. Let $h_i'(X)$ be the rational function that satisfies

$$\begin{aligned}h_i'(X)\ell(X) =& \ell_{\nu-\kappa+i}(X) \left(\mathsf{B}_p(X, Y)/Y - 1\right) \\ =& \ell_{\nu-\kappa+i}(X) \left(t(X) + \sum_{i=1}^{\kappa} \ell_{\nu-\kappa+i}(X) - 1\right) \\ =& \ell_{\nu-\kappa+i}(X)(t(X) + \sum_{j \neq i} \ell_{\nu-\kappa+j}(X)) + \ell_{\nu-\kappa+i}(X)(\ell_{\nu-\kappa+i}(X) - 1) \ .\end{aligned} \tag{15}$$

Due to Item d of Theorem 1 and the definition of $t(X)$ (see also Eqs. (7) and (8)),

$$\ell(X) \mid \ell_{\nu-\kappa+i}(X) t(X) \ .$$

Moreover, $\ell(X) \mid \ell_{\nu-\kappa+i}(X)\ell_{\nu-\kappa+j}(X)$, for $i \neq j$, and $\ell(X) \mid \ell_{\nu-\kappa+i}(X)(\ell_{\nu-\kappa+i}(X) - 1)$. Thus, the polynomial on the right-hand side of Eq. (15) divides by $\ell(X)$. Thus, $h_i'(X)$ is a polynomial of degree $\leq \nu - 2$ and thus $\mathcal{B}$ can compute efficiently

$$[\ell_{\nu-\kappa+i}(\chi)(\mathsf{B}_p - y) y]_1 = [\ell_{\nu-\kappa+i}(\chi)(\mathsf{B}_p/y - 1) y^2]_1 = [h_i'(\chi)\ell(\chi) y^2]_1 \ ,$$

and then

$$\begin{aligned}[z]_1 =& \sum_{i=1}^{\kappa} \Delta_i [\ell_{\nu-\kappa+i}(\chi) y^{\eta+1}]_1 \\ \leftarrow& ([\mathsf{C}_{sp}]_1 - [\tilde{\mathsf{C}}_{sp}]_1) - \sum_{i=1}^{\kappa} \Delta_i \left([\mathsf{B}_i^{\mathsf{aux}}]_1 + [h_i'(\chi)\ell(\chi) y^2]_1\right) \ .\end{aligned}$$

Thus, given the collision, $\mathcal{B}$ outputs $(\boldsymbol{\Delta}, [z]_1 = \sum \Delta_i [f_i(\chi, y)]_1)$ for $f_i(X, Y) \notin \mathrm{span}(\mathcal{R})$. Thus, $\mathcal{B}$ breaks (w.p. $\varepsilon_{\mathcal{A}}$ and time close to $t_{\mathcal{A}}$) the $(\mathcal{R}, \mathcal{S}, \{f_i\})$-computational span-uber-assumption in $\mathbb{G}_1$ in the case $f_i \notin \mathrm{span}(\mathcal{R})$. □

The following Corollary follows from Item 5 in Theorem 3 and Lemma 2.

Corollary 1. *Let $\mathcal{C}$ be a fixed circuit. Let $\gamma = 4$, $\delta = 0$, and $\eta = 7$. Let $f_i'(X, Y) := (\ell_{\nu-\kappa+i}(X))^2 Y^{\eta+2}$. If the $(\mathcal{R}, \mathcal{S}, f_I')$-computational uber-assumption holds in $\mathbb{G}_T$ for all $I \in [1 \,..\, \kappa]$ then $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ is computationally evaluation-binding.*

Remark 2. Importantly, the indeterminate Y is crucial in establishing the independence of f_i from $\mathcal{R}$. Let $\mathcal{R}^* := \{(X^i)_{i=0}^{\nu-1}, (X^i \ell(X))_{i=0}^{\nu-2}\}$, $\mathcal{S}^* := \{(X^i)_{i=0}^{\nu-1}\}$, and $f_i^* := \ell_{\nu-\kappa+i}(X)$. One can establish that $\mathsf{FC}_{\mathsf{sn}}^{\mathcal{C}}$ is evaluation-binding under

the $(\mathcal{R}^*, \mathcal{S}^*, \{f_i^*\})$-computational span-uber-assumption in $\mathbb{G}_1$. Really, consider the following $(\mathcal{R}^*, \mathcal{S}^*, \{f_i^*\})$-span-uber-assumption adversary $\mathcal{B}^*$ that will create y herself, generate a new ck based on her input and y, and then use $\mathcal{B}$ in Theorem 3 to break the $(\mathcal{R}^*, \mathcal{S}^*, \{f_i^*\})$-computational span-uber-assumption. $\mathcal{B}^*$ will have similar success as $\mathcal{B}$. However, $f_i^* \in \mathrm{span}(\mathcal{R}^*)$ and thus the $(\mathcal{R}^*, \mathcal{S}^*, \{f_i^*\})$-computational span-uber-assumption itself is not secure.

On the Security of the Span-Uber-Assumption. It is known that in composite-order bilinear groups, the computational uber-assumption in $\mathbb{G}_T$ holds under appropriate subgroup hiding assumptions [13]. Hence, a composite-order group version of the span-uber-assumption (and also of the new SFC) is secure under a subgroup hiding assumption. In the full version [34], we will use the Déjà Q approach of [14] directly to prove that the span-uber-assumption in $\mathbb{G}_\iota$, $\iota \in \{1, 2\}$, is secure under a subgroup hiding assumption. More precisely, we establish the following corollary. (See the full version [34] for the definition of subgroup hiding and extended adaptive parameter hiding.)

Theorem 4. *The $(\mathcal{R}, \mathcal{S}, \{f_i\}_{i=1}^{\kappa})$-computational span-uber-assumption holds in the source group $\mathbb{G}_1$ with all but negligible probability if*

1. *subgroup hiding holds in $\mathbb{G}_1$ with respect to $\mu = \{\mathsf{P}_1^2, \mathsf{P}_2^1\}$,*
2. *subgroup hiding holds in $\mathbb{G}_2$ with respect to $\mu = \{\mathsf{P}_1^1\}$,*
3. *extended adaptive parameter hiding holds with respect to $\mathcal{R} \cup \{f_i\}_{i=1}^{\kappa}$ and $\mathsf{aux} = \{\mathsf{P}_2^{1\sigma(\cdot)}\}_{\sigma \in \mathcal{S}}$ for any $\mathsf{P}_2^1 \in \mathbb{G}_2$.*
4. *the polynomials in $\mathcal{R}$ have maximum degree $\mathsf{poly}(\lambda)$.*

Here, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are additive groups of composite order $N = p_1 p_2$ ($p_1 \neq p_2$) and $\mathsf{P}_\iota^1 \in \mathbb{G}_{\iota,p_1}$, $\mathsf{P}_\iota^2 \in \mathbb{G}_{\iota,p_2}$ are randomly sampled subgroup generators, where $\mathbb{G}_{\iota,p_j}$ is the subgroup of $\mathbb{G}_\iota$ of order p_j and $\mathsf{P}_\iota \in \mathbb{G}_\iota = \mathbb{G}_{\iota,p_1} \oplus \mathbb{G}_{\iota,p_2}$.

The direct proof in the full version [34] is simpler than the mentioned two-step proof since it does not rely on the intermediate step of reducing the span-uber-assumption to a uber-assumption in $\mathbb{G}_T$. Moreover, the Déjà Q approach is more straightforward in case one works in the source group. We will leave it up to future work to reduce *prime-order* span-uber-assumption to a simpler assumption; there has been almost no prior work on reducing prime-order *assumptions*.

Finally, in the full version [34], by following [33], we will prove that the span-uber-assumption is secure under a hash algebraic knowledge (HAK) assumption and the well-known PDL assumption [31], from which it follows that is secure in the algebraic group model (with hashing) [19] under the PDL assumption.[1] Following the semi-generic-group model of [26], the HAK assumptions of [33] are defined only in the case when the adversary outputs elements in the source groups (but not in $\mathbb{G}_T$), and thus one cannot prove the security of the computational uber-assumption in $\mathbb{G}_T$ using the approach of [33]. Thus, in a well-defined sense, the span-uber-assumption is weaker than the uber-assumption in $\mathbb{G}_T$.

[1] As a corollary of independent interest, we also show in the full version [34] that if $f \notin \mathrm{span}(\mathcal{R})$ then the $(\mathcal{R}, \mathcal{S}, \mathcal{T})$-uber-assumption follows from HAK and PDL.

References

1. Abdolmaleki, B., Baghery, K., Lipmaa, H., Zajac, M.: A subversion-resistant SNARK. In: ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 3–33
2. Abdolmaleki, B., Lipmaa, H., Siim, J., Zajac, M.: On QA-NIZK in the BPK model. In: PKC 2020, Part I. LNCS, vol. 12110, pp. 590–620
3. Bari, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: EUROCRYPT'97. LNCS, vol. 1233, pp. 480–494
4. Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: Security in the face of parameter subversion. In: ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 777–804
5. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: EUROCRYPT'93. LNCS, vol. 765, pp. 274–285
6. Bitansky, N.: Verifiable random functions from non-interactive witness-indistinguishable proofs. In: TCC 2017, Part II. LNCS, vol. 10678, pp. 567–594
7. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456
8. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 561–586
9. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Efficient polynomial commitment schemes for multiple points and polynomials. Technical Report 2020/081, IACR (2020)
10. Bowe, S., Grigg, J., Hopwood, D.: Halo: Recursive Proof Composition without a Trusted Setup. Technical report (2019) Available from `https://electriccoin.co/wp-content/uploads/2019/09/Halo.pdf`.
11. Boyen, X.: The uber-assumption family (invited talk). In: PAIRING 2008. LNCS, vol. 5209, pp. 39–56
12. Catalano, D., Fiore, D.: Vector commitments and their applications. In: PKC 2013. LNCS, vol. 7778, pp. 55–72
13. Chase, M., Maller, M., Meiklejohn, S.: Déjà Q all over again: Tighter and broader reductions of q-type assumptions. In: ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 655–681
14. Chase, M., Meiklejohn, S.: Déjà Q: Using dual systems to revisit q-type assumptions. In: EUROCRYPT 2014. LNCS, vol. 8441, pp. 622–639
15. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768
16. Dent, A.W.: Adapting the weaknesses of the random oracle model to the generic group model. In: ASIACRYPT 2002. LNCS, vol. 2501, pp. 100–109
17. Fischlin, M.: A note on security proofs in the generic model. In: ASIACRYPT 2000. LNCS, vol. 1976, pp. 458–469
18. Fuchsbauer, G.: Subversion-zero-knowledge SNARKs. In: PKC 2018, Part I. LNCS, vol. 10769, pp. 315–347
19. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62
20. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645

21. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: 43rd ACM STOC, pp. 99–108
22. Gorbunov, S., Reyzin, L., Wee, H., Zhang, Z.: Pointproofs: Aggregating Proofs for Multiple Vector Commitments. Technical Report 2020/419, IACR (2020)
23. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340
24. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326
25. Izabachène, M., Libert, B., Vergnaud, D.: Block-wise P-signatures and non-interactive anonymous credentials with efficient attributes. In: 13th IMA International Conference on Cryptography and Coding. LNCS, vol. 7089, pp. 431–450
26. Jager, T., Rupp, A.: The semi-generic group model and applications to pairing-based cryptography. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 539–556
27. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194
28. Lai, R.W.F., Malavolta, G.: Subvector commitments with application to succinct arguments. In: CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 530–560
29. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: ICALP 2016. LIPIcs, vol. 55, pp. 30:1–30:14
30. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: TCC 2010. LNCS, vol. 5978, pp. 499–517
31. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189
32. Lipmaa, H.: Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In: ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 41–60
33. Lipmaa, H.: Simulation-Extractable ZK-SNARKs Revisited. Technical Report 2019/612, IACR (2019) `https://eprint.iacr.org/2019/612`, updated on 8 Feb 2020.
34. Lipmaa, H., Pavlyk, K.: Succinct Functional Commitment for a Large Class of Arithmetic Circuits. Technical Report 2020/?, IACR (2020)
35. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: ACM CCS 2019, pp. 2111–2128
36. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: TCC 2013. LNCS, vol. 7785, pp. 222–242
37. Shpilka, A., Yehudayoff, A.: Arithmetic Circuits: A Survey of Recent Results and Open Questions. Foundations and Trends in Theoretical Computer Science, vol. 5. Now Publishers Inc (2010)
38. Tomescu, A., Abraham, I., Buterin, V., Drake, J., Feist, D., Khovratovich, D.: Aggregatable Subvector Commitments for Stateless Cryptocurrencies. Technical Report 2020/527, IACR (2020)
39. Valiant, L.G.: Completeness Classes in Algebra. In: STOC 1979, pp. 249–261
40. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-Efficient zkSNARKs Without Trusted Setup. In: IEEE SP 2018, pp. 926–943
41. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases. In: IEEE SP 2017, pp. 863–880