Additive-Homomorphic Functional Commitments and Applications to Homomorphic Signatures

Dario Catalano[1][0000−0001−9677−944X], Dario Fiore[2][0000−0001−7274−6600], and Ida Tucker[2][0000−0003−4895−5896]

[1] University of Catania, Italy.
catalano@dmi.unict.it
[2] IMDEA Software Institute, Madrid, Spain.
dario.fiore@imdea.org

Abstract. Functional Commitments (FC) allow one to reveal functions of committed data in a succinct and verifiable way. In this paper we put forward the notion of additive-homomorphic FC and show two efficient, pairing-based, realizations of this primitive supporting multivariate polynomials of constant degree and monotone span programs, respectively. We also show applications of the new primitive in the contexts of *homomorphic signatures*: we show that additive-homomorphic FCs can be used to realize homomorphic signatures (supporting the same class of functionalities as the underlying FC) in a simple and elegant way. Using our new FCs as underlying building blocks, this leads to the (seemingly) first expressive realizations of multi-input homomorphic signatures not relying on lattices or multilinear maps.

1 Introduction

Functional commitments (FC), put forth by Libert, Ramanna and Yung [22], allow a sender to commit to a vector $\boldsymbol{x}$ of length n and later to open the commitment to functions of the committed vector, namely to prove that $f(\boldsymbol{x}) = y$. FCs are required to be *evaluation binding*, meaning that it is computationally hard to open a commitment at two distinct outputs $y \neq y'$ for the same function f. The distinguishing feature of FCs is that commitments and openings should be succinct, i.e., of size independent of n.

Functional commitments generalize the well known notions of vector commitments (VC) [5, 23, 4] and polynomial commitments (PC) [16]—two functionalities that, albeit specific, have nowadays a large number of applications. Besides VCs and PCs, state-of-the-art functional commitments capture linear forms [22, 18] and semi-sparse polynomials [24].

An FC for an arbitrary computation f can be built via succinct commitments and SNARKs for NP: simply, use the latter to generate a succinct argument that "$y = f(\boldsymbol{x})$ and $\boldsymbol{x}$ opens the commitment". However, such an FC holds under *non-falsifiable assumptions* that are required to build SNARKs [12].

In contrast, due to the falsifiability of the evaluation binding notion and as confirmed by the existing constructions [22, 24], functional commitments are

realizable from falsifiable assumptions. Thanks to these two properties – succinctness and security under falsifiable assumptions – FCs can be seen as a simple form of succinct non-interactive arguments.[3] Whenever evaluation binding is sufficient, FCs are an attractive building block: they provide communication-efficiency through succinctness, without having to sacrifice assumptions (e.g., see the applications of vector/polynomial commitments, and FCs for inner products in [4, 16, 22]). For this reason we believe that advancing the understanding of FCs could help us better understand the fundamental problem of constructing succinct argument systems from minimal assumptions.

1.1 Our results

In this work we make progress, along different fronts, in the study of functional commitments based on falsifiable assumptions.

We begin by exploring potential applications of FCs. While we know several applications of FCs for linear functionalities (and all the functionalities implied by them, such as vector and polynomial commitments), to the best of our knowledge, less is known about FCs for, say, multivariate polynomials or circuits.

We address this problem by showing a new application of FCs to building homomorphic signatures [1]. As it will be apparent later, this application becomes particularly interesting if the FC is *additively homomorphic*, namely if given two commitments to vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, one can compute the commitment to the vector $\boldsymbol{x}_1 + \boldsymbol{x}_2$. This is a basic and useful property of commitment schemes. Yet we know of no FC that is additive-homomorphic and supports a rich class of computations; the only known additive-homomorphic FCs are the ones for linear forms of [22, 18].

We bridge this gap by proposing the first additive-homomorphic FCs supporting the evaluation of functions beyond linear. Our techniques yield new homomorphic signatures that advance the state of the art, and a SNARG for a polynomial-time language from a falsifiable assumption.

Below we present our results in more detail, and in the next section we provide an overview of our techniques.

Additive-homomorphic FC for polynomials. We propose an additive-homomorphic FC scheme that allows one to commit to a vector $\boldsymbol{x}$ of length n and to open the commitment to $\boldsymbol{f}(\boldsymbol{x})$ where $\boldsymbol{f}$ is a collection of m multivariate polynomials of bounded constant degree. Our scheme enjoys *compact* openings, i.e., a single proof, of size constant in both n and m, for all the m evaluations. We build this FC using bilinear groups and prove its security based on the Diffie-Hellman exponent assumption [2].

Compared to the FC for semi-sparse polynomials of [24] and an FC for polynomials obtained via linearization (cf. section 1.2), the main novelty of ours is to be additively homomorphic. Also, ours is the first FC with compact openings

[3] Their functionality resembles commit-and-prove SNARKs except that FCs are evaluation binding rather than (knowledge) sound.

whose security is based on established assumptions: the scheme of [18] relies on the generic group model, and that of [24] uses a newly proposed assumption.

Additive-homomorphic FC for monotone span programs. Our second realization is an FC for a new polynomial time language, called *semi-quadratic arithmetic programs* (sQAPs, for short). In a nutshell, an sQAP is defined by a matrix $\mathbf{F}$ and accepts a pair of vectors $(\boldsymbol{z}, \boldsymbol{y})$ if there exists a solution $\boldsymbol{w}$ such that $\mathbf{F} \cdot (\boldsymbol{z} \circ \boldsymbol{w}) = \boldsymbol{y}$, where $\circ$ denotes entry-wise multiplication of vectors.[4] An FC for sQAPs allows one to commit to $(\boldsymbol{z}, \boldsymbol{y})$ and then open to $\mathbf{F}$, in the sense of proving that $\mathbf{F}$ accepts the pair of committed vectors. Our scheme is based on pairings, it is additively homomorphic and has constant size proofs consisting of three group elements. We prove its security based on a variant of the Diffie-Hellman exponent assumption that we justify in the generic group model.

We show that sQAPs are sufficiently expressive to capture the well known class of monotone span programs (MSPs) [15] and show how to turn our FC for sQAPs into one for MSPs. Also, via known transformations (see footnote 10) it is possible to build a monotone span program that models the satisfiability of an NC^1 circuit, which therefore allows us to obtain *the first FC for* NC^1 *circuits.*

Applications to homomorphic signatures, and more. To motivate additive-homomorphic FCs we present a novel application of this primitive to build homomorphic signatures (HS) [1] (see section 1.3 for an overview).

Notably, by plugging our new FCs in this transformation we obtain new HS that advance the state of the art as follows:

- Our FC for polynomials yields the first multi-input HS for polynomials based on pairings, and the first HS with "compact" signatures, where, again, by compact we mean that, for functions of the form $\boldsymbol{f} : \mathbb{F}^n \to \mathbb{F}^m$, the resulting signatures have size which is constant in both n and m. None of the previous schemes, e.g., [1, 7, 13] has compact signatures, as they need one signature for every output value.
- Through our FC for NC^1 we obtain the first *multi-input* HS based on pairings for NC^1 circuits. The most expressive HS based on pairings is that of Katsumata et al. [17] that also supports NC^1 circuits, but in the *single-input* model where the signer must sign the entire data vector at once. Prior multi-input HS for functions beyond linear instead need lattices [1, 13] or multilinear maps [7]. Our result essentially shows that these powerful algebraic structures are not necessary to build such expressive HS.

In the full version we discuss further applications of additive-homomorphic FCs, such as updatable FCs and verifiable databases with expressive queries.

A SNARG for linear systems from falsifiable assumptions. In [18], Lai and Malavolta put forth a stronger security property for FC, that we call strong evaluation binding, which considers as an attack not only two inconsistent openings for the same function but also inconsistent openings for multiple functions.

[4] sQAPs is in P as it can be decided via Gaussian elimination.

Namely it must be computationally hard to produce a commitment and a collection of valid openings for function-output pairs $\{f_i, y_i\}_{i=1}^{Q}$ for which there exists no vector $\boldsymbol{x}$ such that $f_i(\boldsymbol{x}) = y_i$ for every $i = 1$ to Q. Lai and Malavolta only show how to realize a strong evaluation binding FC for linear maps by resorting to the generic group model. This is unsatisfactory as a generic group model proof essentially uses non-black-box extractability techniques, which cannot be considered falsifiable, and would defeat the main goal of this work which is constructing FCs from falsifiable assumptions.

In our construction of FC for sQAPs we show a new proof technique that allows us to reduce an adversary that produces a valid proof for an inconsistent system of equations to an adversary against a falsifiable assumption. Interestingly, we can apply the same technique to the linear map FC of [18] and prove its strong evaluation binding based on a falsifiable assumption, the parallel bilinear Diffie-Hellman exponent in [26].

This is to the best of our knowledge *the first strong evaluation binding and compact FC from a falsifiable assumption.* This result is interesting since, as one could observe, a strong evaluation binding FC with compact proofs for a language $\mathcal{L}$ yields *de facto* a SNARG for $\mathcal{L}$. Also, a strong evaluation binding FC with compact proofs for quadratic polynomials would yield a SNARG for NP, since a system of quadratic equations can model circuit satisfiability, e.g., through R1CS [10]. Therefore, due to the impossibility of Gentry and Wichs [12], our SNARG for linear maps from falsifiable assumptions can be seen as optimal, in the sense that it is unlikely to have an analogous result for quadratic functions.

1.2 Related work

Libert et al. [22] introduce the notion of functional commitments and propose a construction for *linear forms* based on the Diffie-Hellman exponent assumption in bilinear groups. Lai and Malavolta [18] extend the scheme of [22] to support *linear maps* with *compact* openings, namely of size independent of both the input and the output lengths. Lipmaa and Pavlyk [24] propose an FC construction that supports, with compact proofs, a class of arithmetic circuits which roughly corresponds to semi-sparse polynomials. Their scheme is obtained by "scaling down" SNARK-based techniques and is proven secure from a newly proposed falsifiable assumption in bilinear groups. More generally, an FC for linear maps is sufficient to realize an FC for any *linearizable* function, that is a function f which can be implemented as $f(\boldsymbol{x}) = \langle \boldsymbol{p}(\boldsymbol{x}), \boldsymbol{\phi}_f \rangle$ where $\boldsymbol{p}(\cdot)$ is a vector of polynomial-time computable functions which do not depend on f and can be precomputed. Simply, the sender commits to the vector $\boldsymbol{p}(\boldsymbol{x})$ and then, for any f, opens the commitment to the linear form $\boldsymbol{\phi}_f$. Both the scheme of [24] and the one based on linearization are not additively homomorphic[5] and thus cannot be used in the applications discussed in this paper.

[5] Even if one starts from an additive-homomorphic FC for linear maps, one can notice that the transformation to FCs for linearizable functions does not preserve the additive-homomorphism.

In a recent work, Peikert et al. [25] propose the first construction of a vector commitment based on lattice assumptions and show an extension of it to a functional commitment for circuits. Their FC, however, works in a weaker model where a trusted authority uses secret information to generate an opening key for each function for which the prover wishes to generate an opening.

1.3 Technical overview

FC for polynomials. To illustrate the main ideas of our construction let us consider the simplified case where one opens the commitment to a single polynomial (i.e., no compactness) that is homogeneous. Note that a homogeneous polynomial of degree d, that we can write as $f(\boldsymbol{x}) = \sum_\ell f_\ell \cdot (x_1^{d_{\ell,1}} \cdots x_n^{d_{\ell,n}})$ with $\sum_j d_{\ell,j} = d$, can be linearized as an inner product between the vector of its coefficients and the vector of all degree-d terms. More precisely, assuming $d = 2^\delta$ a power of 2, given a homogeneous polynomial f we can build a vector $\hat{\boldsymbol{f}} \in \mathbb{F}^{n^d}$ such that for any $\boldsymbol{x} \in \mathbb{F}^n$ it holds $\langle \hat{\boldsymbol{f}}, \boldsymbol{x}^{(\delta)} \rangle = f(\boldsymbol{x})$, where $\boldsymbol{x}^{(\delta)}$ is the δ-fold Kronecker product of $\boldsymbol{x}$ with itself, i.e., $\boldsymbol{x}^{(1)} = \boldsymbol{x} \otimes \boldsymbol{x}$, $\boldsymbol{x}^{(2)} = \boldsymbol{x}^{(1)} \otimes \boldsymbol{x}^{(1)}$, etc.

Following this observation, one could use an FC for linear forms to commit to $\boldsymbol{x}^{(\delta)}$ and then open/verify the commitment using the appropriately computed linear form $\hat{\boldsymbol{f}}$. This idea however suffers the problem that the commitments would not be additively homomorphic.

Our approach to solve this problem is to generate a commitment C to $\boldsymbol{x}$ such that: (i) C is additively homomorphic, and (ii) the prover creates, at opening time, a linear-map commitment X_δ to $\boldsymbol{x}^{(\delta)}$ and convinces the verifier that the vector committed in X_δ is indeed the δ-fold Kronecker product of the vector committed in C. Once (ii) is achieved we could use the linear-map functionality to open X_δ to $\langle \hat{\boldsymbol{f}}, \boldsymbol{x}^{(\delta)} \rangle$. The challenge of achieving (ii) is to make this proof succinct without having to extract the committed vectors from the prover.

Our technique to solve this problem is algebraically involved. In what follows highlight the main ideas, without focusing too much on security.

For the X_δ produced in the opening we use the linear-map commitment of [22, 18] in which the vector $\boldsymbol{x}^{(\delta)}$ is encoded in a group element

$$X_\delta = [p_{\boldsymbol{x}}^{(\delta)}(\alpha)]_1 = \sum_{j=1}^{n^d} x_j^{(\delta)} \cdot [\alpha^j]_1$$

where the elements $[\alpha^j]_1$ are part of the public parameters.[6] For the commitment to $\boldsymbol{x}$, assume for now that it includes $X_0 = [p_{\boldsymbol{x}}^{(0)}(\alpha)]_1 = \sum_{j=1}^{n} x_j \cdot [\alpha^j]_1$, and consider for simplicity the case of $\delta = 1$ (i.e., opening to a polynomial of degree

[6] We use the bracket notation for bilinear groups of [9].

$d = 2$). Then our first key observation is that

$$p^{(0)}_{\boldsymbol{x}}(\alpha) \cdot (p^{(0)}_{\boldsymbol{x}}(\alpha^n)/\alpha^n) = \left(\sum_{i=1}^{n} x_i \cdot \alpha^i\right)\left(\sum_{j=1}^{n} x_j \cdot \alpha^{n(j-1)}\right)$$
$$= \sum_{i,j=1}^{n} x_i x_j \cdot \alpha^{i+n(j-1)} = \sum_{k=1}^{n^2} x^{(1)}_k \cdot \alpha^k = p^{(1)}_{\boldsymbol{x}}(\alpha)$$

Thus, if we include in the commitment the element $\hat{X}_0 = [\hat{p}^{(0)}_{\boldsymbol{x}}(\alpha)]_2 = [p^{(0)}_{\boldsymbol{x}}(\alpha^n)/\alpha^n]_2$, the verifier can test the correctness of X_1 via a pairing $e(X_1, [1]_2) = e(X_0, \hat{X}_0)$. Intuitively, this is secure because the pair $(X_0, \hat{X}_0)$ is part of the commitment and can be somehow considered "trusted"; so the pairing allows transferring this trust to X_1. To handle openings of polynomials of degree > 2, this is not sufficient though. Say that the prover includes in the opening the elements $X_2, X_1, \hat{X}_1$, and the verifier tests the correctness of X_2 via a "chain" of checks $e(X_1, [1]_2) \stackrel{?}{=} e(X_0, \hat{X}_0)$ and $e(X_2, [1]_2) \stackrel{?}{=} e(X_1, \hat{X}_1)$. The issue is that in the second check $(X_1, \hat{X}_1)$ is not "trusted"; in particular, while X_1 can be considered trusted due to the previous check, $\hat{X}_1$ is not, since it is generated by the prover and not tested.

Our second key idea is based on showing that the polynomial $\hat{p}^{(1)}_{\boldsymbol{x}}(\alpha)$ in $\hat{X}_1$ can be expressed as the product of two polynomials $\phi^{(2)}_{\boldsymbol{x}}(\alpha), \phi^{(3)}_{\boldsymbol{x}}(\alpha)$, each of them a linear function of $\boldsymbol{x}$. Precisely, it holds that (cf. Claim 2)

$$p^{(1)}_{\boldsymbol{x}}(\alpha^n)/\alpha^n = \phi^{(2)}_{\boldsymbol{x}}(\alpha) \cdot \phi^{(3)}_{\boldsymbol{x}}(\alpha) = (p^{(0)}_{\boldsymbol{x}}(\alpha^{n^2})/\alpha^{n^2}) \cdot (p^{(0)}_{\boldsymbol{x}}(\alpha^{n^3})/\alpha^{n^3})$$

So, if we include in the commitment group elements Φ_2, Φ_3 encoding $\phi^{(2)}_{\boldsymbol{x}}(\alpha)$ and $\phi^{(3)}_{\boldsymbol{x}}(\alpha)$ respectively, the verifier will be able to use a pairing to test the correctness of the element $\hat{X}_1$ included in the opening, and mark X_1 as "trusted", as it can establish a correct link with the group elements in the commitment.

To summarize, in this example of a degree-4 homogeneous polynomial f, the commitment C of $\boldsymbol{x}$ includes $(X_0, \hat{X}_0, \Phi_2, \Phi_3)$, and the opening includes $(X_1, \hat{X}_1, X_2)$ and a linear-map opening proof generated using [18] to show that X_2 (seen as a commitment to $\boldsymbol{x}^{(2)}$) opens to $\langle \hat{\boldsymbol{f}}, \boldsymbol{x}^{(2)} \rangle = f(\boldsymbol{x})$.

Importantly, all the group elements in the commitment can be expressed as a linear map of the vector $\boldsymbol{x}$, thus making C additively homomorphic.

Going beyond degree 4 requires further extensions of our technique since a polynomial $\hat{p}^{(k)}_{\boldsymbol{x}}(\alpha)$ factors into 2^k polynomials, which for $k > 1$ cannot be tested with a pairing. We bridge this gap by showing how to break each of these tests into a system of k quadratic equations using a tree-based encoding. This is our third key idea that allows us to generalize the techniques illustrated so far to handle degree-2^δ polynomials.

Eventually, we obtain an FC for arbitrary polynomials of constant degree d in which commitment and openings consist of exactly d group elements (notably, even if one opens m polynomials at the same time). Comparing to the techniques

of prior FCs for linear maps [22, 18], while our FC uses them in the final step of our opening algorithm, the remaining design ideas are novel and significantly different.

FC for semi-quadratic arithmetic programs. We recall that in an FC for sQAPs one commits to a pair of vectors $\boldsymbol{x} = (\boldsymbol{z}, \boldsymbol{y})$ and then opens to $\mathbf{F}$ in the sense of proving that $\exists \boldsymbol{w} : \mathbf{F} \cdot (\boldsymbol{z} \circ \boldsymbol{w}) = \boldsymbol{y}$. Similarly to the FC for polynomials, we start from the idea of linearizing the computation in such a way that we can eventually resort to a linear-map FC (LMC). Specifically, we use the LMC of [18]. However, to do this linearization we cannot use the same technique of the previous scheme to produce a commitment to, e.g., $\boldsymbol{z} \circ \boldsymbol{w}$ or $\boldsymbol{z} \otimes \boldsymbol{w}$. Roughly speaking, the issue is that in sQAPs $\boldsymbol{w}$ is not committed ahead of time together with $\boldsymbol{z}$; here $\boldsymbol{w}$ is a non-deterministic witness depending on each specific $\mathbf{F}$.

So we proceed differently. We let the prover compute a succinct encoding of the matrix $\mathbf{F}_{\boldsymbol{z}} = \mathbf{F} \circ \mathbf{Z}$, where $\mathbf{Z} \in \mathbb{F}^{m \times n}$ is the matrix with $\boldsymbol{z}^\top$ in every row, and we show how the verifier can check the validity of this encoding given $\mathbf{F}$ and a committed $\boldsymbol{z}$. This way, we are left with the problem of proving that $(\mathbf{F}_{\boldsymbol{z}} \mid \boldsymbol{y})$ is a satisfiable system of linear equations. To prove this, we let the prover generate a commitment W to the solution $\boldsymbol{w}$ and then generate an opening proof to argue that $\boldsymbol{y} = \mathbf{F}_{\boldsymbol{z}} \cdot \boldsymbol{w}$ for the committed $\boldsymbol{w}$. The generation of W and its opening to $\mathbf{F}_{\boldsymbol{z}}$ rely on the LMC of [18].

Compared to [18], we introduce two technical novelties. The first one deals with enabling the verifier to check the opening by having only an encoding of $\mathbf{F}_{\boldsymbol{z}}$, which can be linked to the public $\mathbf{F}$ and the commitment to $\boldsymbol{z}$. The second and most important novelty concerns the security proof. The challenge is the presence of this non-deterministic component $\boldsymbol{w}$ which requires the prover to show the satisfiability of a system – a task that goes beyond what is captured by the notion of evaluation binding since we need that an efficient adversary cannot generate a valid opening if $(\mathbf{F}_{\boldsymbol{z}} \mid \boldsymbol{y})$ is *not* satisfiable. This could be solved by resorting to the strong evaluation binding of the [18] LMC, but they only prove this property in the generic group model, essentially using a non-black-box extraction technique. In our paper we show a new proof technique for reducing an adversary producing a valid opening for an inconsistent system of equations into an adversary against a falsifiable assumption.

From FCs to homomorphic signatures. We present a novel approach to construct HS based on (additively homomorphic) FCs. The basic idea is that the signer generates a commitment C_x to the dataset $\boldsymbol{x}$ and a (standard) digital signature σ_{C_x} on the commitment. Given (C_x, σ_{C_x}), the server can compute a function f by giving to the verifier this pair (C_x, σ_{C_x}) (which is succinct) along with an opening of C_x to f (which is succinct as well). The resulting HS construction is clearly single-input since the signer must commit to the dataset all at once. We achieve a multi-input HS by exploiting FCs that are additively homomorphic. To sign the i-th element of the dataset, Alice commits to the sparse vector $x_i \cdot \boldsymbol{e}_i$ with x_i in position i and 0 everywhere else; let C_i be the resulting commitment. If the server is given these commitments one by one, eventually it can reconstruct a commitment C to the currently available dataset by computing

their sum homomorphically, and then proceed as in the single-input construction by opening C to the desired function f. This construction however is not secure as the verifier cannot be assured that C is validly obtained from commitments provided by Alice. Therefore we let Alice sign C_i using an homomorphic signature that only needs to support one functionality, the homomorphic sum in the commitment space. Interestingly, for pairing-based FCs, this HS can be implemented via well known linearly-homomorphic structure-preserving signatures [21][7]. Finally, we notice that for the sake of this application the FC only needs to satisfy a weaker notion of evaluation binding in which the adversary reveals the vector $\boldsymbol{x}$ committed in C, yet it manages to produce an opening to a function f and a result $y \neq f(\boldsymbol{x})$ that is accepted by the verification algorithm.

2 Preliminaries

Notation. We use $\lambda \in \mathbb{N}$ to denote the security parameter. If a function $\epsilon(\lambda) = O(\lambda^{-c})$ for every constant $c > 0$, then we say that ϵ is *negligible*, denoted $\epsilon(\lambda) = \mathsf{negl}(\lambda)$. A function $p(\lambda)$ is *polynomial* if $p(\lambda) = O(\lambda^c)$ for some constant $c > 0$. We say that an algorithm is *probabilistic polynomial time* (PPT) if its running time is bounded by some $p(\lambda) = \mathsf{poly}(\lambda)$. Given a finite set S, $x \leftarrow\!\!\$\, S$ denotes selecting x uniformly at random in S. For an algorithm A, we write $y \leftarrow A(x)$ for the output of A on input x. For a positive $n \in \mathbb{N}$, $[n]$ is the set $\{1, \ldots, n\}$. We denote vectors $\boldsymbol{x}$ and matrices $\mathbf{M}$ using bold fonts. For a ring $\mathcal{R}$, given two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{R}^n$, $\boldsymbol{x} \circ \boldsymbol{y}$ denotes their entry-wise product, i.e., the vector with entries $(x_i y_i)_i$, while $\boldsymbol{z} := (\boldsymbol{x} \otimes \boldsymbol{y}) \in \mathcal{R}^{n^2}$ denotes their Kronecker product (that is a vectorization of the outer product), i.e., $\forall i, j \in [n] : z_{i+(j-1)n} = x_i y_j$.

Bilinear Groups. Our FC constructions build on bilinear groups. A *bilinear group generator* $\mathcal{BG}(1^\lambda)$ outputs $\mathsf{bgp} := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are groups of prime order q, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are two fixed generators, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate, bilinear map. We present our results using Type-3 groups in which it is assumed that there is no efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$.

For group elements, we use the bracket notation of [9] in which, for $s \in \{1, 2, T\}$ and $x \in \mathbb{Z}_q$, $[x]_s$ denotes $g_s^x \in \mathbb{G}_s$. We use additive notation for $\mathbb{G}_1$ and $\mathbb{G}_2$ and multiplicative one for $\mathbb{G}_T$. For $s = 1, 2$, given an element $[x]_s \in \mathbb{G}_s$ and a scalar a, one can efficiently compute $a \cdot [x] = [ax] = g_s^{ax} \in \mathbb{G}_s$; given group elements $[a]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$, one can efficiently compute $[ab]_T = e([a]_1, [b]_2)$.

3 Functional Commitments

We recall the notion of functional commitments (FC) [22]. A crucial feature that makes this primitive interesting and nontrivial is that both commitment and

[7] Strictly speaking the signature does not need to be structure preserving as long as it allows to (homomorphically) sign group elements.

the openings are *succinct*, i.e., of size independent of the vector's length. In our work we also consider *compact* FCs, a notion introduced in [18], which requires openings size to be also independent of the function's output length.

Definition 1 (Functional Commitments). *A functional commitment scheme is a tuple of algorithms* $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{Ver})$ *with the following syntax and that satisfies correctness and succinctness (or compactness).*

$\mathsf{Setup}(1^\lambda, n, m) \to \mathsf{ck}$ *on input the security parameter* λ *and the vector length* n*, outputs a commitment key* ck*, which defines the message space* $\mathcal{X}$ *and the class of admissible functions* $\mathcal{F} \subseteq \{f : \mathcal{X}^n \to \mathcal{X}^m\}$ *for some* $n, m = \mathsf{poly}(\lambda)$*.*

$\mathsf{Com}(\mathsf{ck}, \boldsymbol{x}; r) \to (C, \mathsf{aux})$ *on input a vector* $\boldsymbol{x} \in \mathcal{X}^n$ *and (possibly) randomness* r*, outputs a commitment* C *and related auxiliary information* aux*. We often omit* r *from the inputs, in which case we assume it is randomly sampled in the appropriate space.*

$\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f) \to \pi$ *on input an auxiliary information* aux *and a function* $f \in \mathcal{F}$*, outputs an opening proof* π*.*

$\mathsf{Ver}(\mathsf{ck}, C, f, \boldsymbol{y}, \pi) \to b \in \{0, 1\}$ *on input a commitment* C*, an opening proof* π*, a function* $f \in \mathcal{F}$ *and a value* $y \in \mathcal{X}^m$*, accepts* $(b = 1)$ *or rejects* $(b = 0)$*.*

Correctness. FC *is correct if for any* $n, m \in \mathbb{N}$*, all* $\mathsf{ck} \leftarrow\!\!\$\, \mathsf{Setup}(1^\lambda, n)$*, any* $f : \mathcal{X}^n \to \mathcal{X}^m$ *in the class* $\mathcal{F}$*, and any vector* $\boldsymbol{x} \in \mathcal{X}^n$*, if* $(C, \mathsf{aux}) \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{x})$*, then it holds* $\mathsf{Ver}(\mathsf{ck}, C, f, f(\boldsymbol{x}), \mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f)) = 1$ *with probability 1.*

Succinctness/Compactness. *A functional commitment* FC *is* succinct *if there exists a fixed polynomial* $p(\lambda) = \mathsf{poly}(\lambda)$ *such that that for any* $n, m = \mathsf{poly}(\lambda)$*, any admissible function* $f \in \mathcal{F}$ *such that* $f : \mathcal{X}^n \to \mathcal{X}^m$*, honestly generated commitment key* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n, m)$*, vector* $\boldsymbol{x} \in \mathcal{X}^n$*, commitment* $(C, \mathsf{aux}) \in \mathsf{Com}(\mathsf{ck})$ *and opening* $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f)$*, it holds that* $|C| \leq p(\lambda)$ *and* $|\pi| \leq p(\lambda) \cdot m$*. Furthermore, we say that* FC *is* compact *if* $|\pi| \leq p(\lambda)$*.*

3.1 Binding notions of FCs

Intuitively, the security of FCs should model the hardness of computing openings for false statements that are accepted by the verification algorithm. The first definition in [22] is inspired by that of vector commitments [4]. It states that it should be computationally hard to open a commitment to two distinct outputs for the same function. Formally, it is defined as follows.

Definition 2 (Evaluation Binding). *For any PPT adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathsf{EvBind}}_{\mathcal{A},\mathsf{FC}}(\lambda) = \Pr\left[\begin{array}{l}\mathsf{Ver}(\mathsf{ck}, C, f, \boldsymbol{y}, \pi) = 1 \\ \quad \wedge\ \boldsymbol{y} \neq \boldsymbol{y}' \wedge \\ \mathsf{Ver}(\mathsf{ck}, C, f, \boldsymbol{y}', \pi') = 1\end{array} : \begin{array}{r}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n) \\ (C, f, \boldsymbol{y}, \pi, \boldsymbol{y}', \pi') \leftarrow \mathcal{A}(\mathsf{ck})\end{array}\right] = \mathsf{negl}(\lambda)$$

We define a weaker notion of evaluation binding in which the adversary is required to fully open the commitment (i.e., to show the vector $\boldsymbol{x}$ it contains) and

to generate a valid opening for a false output, i.e., for some $\boldsymbol{y} \neq f(\boldsymbol{x})$.[8] Intuitively, this is sufficient in applications where the verifier has either computed once the commitment or has received the commitment from a trusted party (e.g., the commitment comes with a valid signature of this party). We show in Section 6 that this notion is sufficient to construct homomorphic signatures from FCs.

Definition 3 (Weak Evaluation Binding). *For any PPT adversary* $\mathcal{A}$

$$\mathbf{Adv}^{\mathsf{wEvBind}}_{\mathcal{A},\mathsf{FC}}(\lambda) = \Pr\left[\begin{matrix}(C,\cdot) = \mathsf{Com}(\mathsf{ck},\boldsymbol{x};r) \\ \wedge\ y \neq f(\boldsymbol{x}) \wedge \\ \mathsf{Ver}(\mathsf{ck},C,f,\boldsymbol{y},\pi) = 1\end{matrix} : \begin{matrix}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda,n) \\ (\boldsymbol{x},r,f,\boldsymbol{y},\pi) \leftarrow \mathcal{A}(\mathsf{ck})\end{matrix}\right] = \mathsf{negl}(\lambda)$$

One may observe that if an FC satisfies evaluation binding, then it also satisfies weak evaluation binding. For a formal proof, we refer to the full version.

Finally, we also mention a stronger version of evaluation binding, put forward by Lai and Malavolta [18]. Here, the adversary outputs a commitment, and a collection of openings to one or more functions. It is successful if all the claimed outputs define an inconsistent system of equations. Namely, it outputs $\{f_i, \boldsymbol{y}_i\}$ for which there exists no $\boldsymbol{x}$ such that for all i $f_i(\boldsymbol{x}) = \boldsymbol{y}_i$.

Definition 4 (Strong Evaluation Binding). *For any PPT adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathsf{sEvBind}}_{\mathcal{A},\mathsf{FC}}(\lambda)$ *defined below is negligible.*

$$\Pr\left[\begin{matrix}\forall i \in [Q] : \mathsf{Ver}(\mathsf{ck},C,f_i,\boldsymbol{y}_i,\pi_1) = 1 \\ \wedge\ \nexists \boldsymbol{x} \in \mathcal{X}^n : \forall i \in [Q] : f_i(\boldsymbol{x}) = \boldsymbol{y}_i\end{matrix} : \begin{matrix}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda,n) \\ (C,\{f_i,\boldsymbol{y}_i,\pi_i\}_{i=1}^{Q}) \leftarrow \mathcal{A}(\mathsf{ck})\end{matrix}\right]$$

3.2 Additional properties of FCs

In the full version we give the notions of hiding commitments and zero-knowledge openings. Here we define some extra properties of functional commitments that can be useful in applications and that are enjoyed by our constructions.

Additive-homomorphic FCs We consider additively homomorphic FCs in which, given two commitments C_1 and C_2 to vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ respectively, one can compute a commitment to $\boldsymbol{x}_1 + \boldsymbol{x}_2$. Below, we formalize this property, considering also how to obtain the corresponding random coins and auxiliary information of the commitment.

Definition 5 (Additive-homomorphic FCs). *Let* FC *be a functional commitment scheme where* $\mathcal{X}$ *is a ring. Then* FC *is additive homomorphic if there exist deterministic algorithms* $\mathsf{FC.Add}(\mathsf{ck},C_1,\ldots,C_n) \to C$, $\mathsf{FC.Add}_{\mathsf{aux}}(\mathsf{ck},\mathsf{aux}_1,\ldots,\mathsf{aux}_n) \to \mathsf{aux}$ *and* $\mathsf{FC.Add}_r(\mathsf{ck},r_1,\ldots,r_n) \to r$ *such that for any* $\boldsymbol{x}_i \in \mathcal{X}$ *and* $(C_i,\mathsf{aux}_i) \leftarrow \mathsf{Com}(\mathsf{ck},\boldsymbol{x}_i;r_i)$*, if* $C \leftarrow \mathsf{FC.Add}(\mathsf{ck},C_1,\ldots,C_n)$*,* $\mathsf{aux} \leftarrow \mathsf{FC.Add}_{\mathsf{aux}}(\mathsf{ck},\mathsf{aux}_1,\ldots,\mathsf{aux}_n)$*, and* $r \leftarrow \mathsf{FC.Add}_r(\mathsf{ck},r_1,\ldots,r_n)$*, then* $(C,\mathsf{aux}) = \mathsf{Com}(\mathsf{ck},\sum_{i=1}^n \boldsymbol{x}_i;r)$.

[8] This notion is similar in spirit to the basic security of accumulators [3].

Efficient Verification In FCs the verification algorithm must read the function's description, which can be as large as its running time for certain computational models (e.g., linear forms, polynomials, circuits) and thus can make verifying and output of f as expensive as running f. To address this problem, we define a notion of amortized efficient verification for FCs. Similarly to homomorphic signatures [7] and preprocessing universal SNARKs [14], an FC has this property if the verifier can precompute a short verification key vk_f associated to f, and later can verify any opening for f by using only vk_f.

Definition 6 (Amortized efficient verification). *A functional commitment scheme* FC *has* amortized efficient verification *if there are two additional algorithms* $\mathsf{vk}_f \leftarrow \mathsf{VerPrep}(\mathsf{ck}, f)$ *and* $b \leftarrow \mathsf{EffVer}(\mathsf{vk}_f, C, \boldsymbol{y}, \pi)$ *such that for any honestly generated commitment key* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n, m)$*, vector* $\boldsymbol{x} \in \mathcal{X}^n$*, commitment* $(C, \mathsf{aux}) \in \mathsf{Com}(\mathsf{ck})$ *and opening* $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f)$ *with* $f \in \mathcal{F}$*, it holds: (a)* $\mathsf{EffVer}(\mathsf{VerPrep}(\mathsf{ck}, f), C, \boldsymbol{y}, \pi) = \mathsf{Ver}(\mathsf{ck}, C, f, \boldsymbol{y}, \pi)$*, and (b)* EffVer *running time is a fixed polynomial* $p(\lambda, |\boldsymbol{y}|)$*.*

Aggregation. Intuitively, we say that FC has aggregatable openings if given several openings $\pi_1, \ldots, \pi_\ell$ such that each π_i verifies for the same commitment C and function-output pair $(f_i, \boldsymbol{y}_i)$, and given a function $g : \mathcal{X}^{m_1} \times \cdots \times \mathcal{X}^{m_\ell} \to \mathcal{X}^m$ one can compute an opening π that verifies for the composed function $g(f_1, \ldots, f_\ell)$ and the output $g(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\ell)$.

Definition 7. *A functional commitment scheme* FC *satisfies* aggregation *if there is an algorithm* $\pi \leftarrow \mathsf{Agg}(\mathsf{ck}, C, ((\pi_1, f_1, \boldsymbol{y}_1), \ldots, (\pi_\ell, f_\ell, \boldsymbol{y}_\ell)), g)$ *such that, for honestly generated commitment key* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n, m)$*, commitment* C *and triples* $\{(\pi_i, f_i, \boldsymbol{y}_i)\}_{i=1}^{\ell}$ *such that for all* $i \in [\ell]$ *it holds* $\boldsymbol{y}_i \in \mathcal{X}^{m_i}$ *and* $\mathsf{Ver}(\mathsf{ck}, C, \pi_i, f_i, \boldsymbol{y}_i) = 1$*, then for any admissible function* $g : \mathcal{X}^{m_1} \times \cdots \times \mathcal{X}^{m_\ell} \to \mathcal{X}^m$*,*

$$\mathsf{Ver}(\mathsf{ck}, C, \mathsf{Agg}(\mathsf{ck}, C, ((\pi_1, f_1, \boldsymbol{y}_1), \ldots, (\pi_\ell, f_\ell, \boldsymbol{y}_\ell)), g), f^*, g(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\ell)) = 1$$

where f^* *is the composed function* $f^*(\boldsymbol{X}) = g(f_1(\boldsymbol{X}), \ldots, f_\ell(\boldsymbol{X}))$*.*

4 Additive-Homomorphic FC for Polynomials

In this section we propose our FC for polynomials, which supports the following features: additive-homomorphic, opening to multiple (multivariate) polynomials of the committed vector with a *compact* proof, efficient verification and linear aggregation. We build our scheme in bilinear groups and prove that it satisfies evaluation binding under the DHE assumption (Def. 8 [2]).

We build this FC in two steps. We begin by constructing an FC that only supports homogeneous multivariate polynomials whose degree is a power of two (see next section). Next, in Section 4.4 we show how an additive-homomorphic FC for homogeneous polynomials can be turned into one for all multivariate polynomials by letting one commit to vectors $(1, \boldsymbol{x})$.

4.1 Additive-homomorphic FC for Homogeneous Polynomials

Below we describe our FC for homogeneous polynomials. See section 1.3 for an intuition. To keep the exposition simpler we present a deterministic version of our FC which is not hiding, and refer to the full version for how to modify it in order to satisfy com-hiding and zero-knowledge openings.

$\mathsf{Setup}(1^\lambda, n, m, d)$ Let $n, m, d \geq 1$ be three integers representing the length of the vectors to be committed, the number of the polynomials to be computed at opening time, and the degree of these polynomials, respectively. Define $N := n^d$, generate a bilinear group description $\mathsf{bgp} := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{BG}(1^\lambda)$, and let $\mathbb{F} := \mathbb{Z}_q$. Next, sample random $\alpha \leftarrow\!\$\, \mathbb{Z}_q$, $\boldsymbol{\beta} \leftarrow\!\$\, \mathbb{F}^m$ and output

$$\mathsf{ck} := \begin{pmatrix} \{[\alpha^j]_1, [\alpha^j]_2\}_{j\in[N]}, \{[\beta_i \cdot \alpha^j]_2\}_{i\in[m], j\in[N]} \\ \{[\alpha\beta_i]_1\}_{i\in[m]}, \{[\alpha^j\beta_i]_1\}_{i\in[m], j\in[2N]\setminus\{N+1\}} \end{pmatrix}$$

$\mathsf{Com}(\mathsf{ck}, \boldsymbol{x})$ We encode the vector $\boldsymbol{x}$ with the polynomial $p_{\boldsymbol{x}}(Z) := \sum_{j=1}^{n} x_j \cdot Z^j$. Also, for $\ell = 1, \ldots, d-1$, we define the polynomials

$$\phi_{\boldsymbol{x}}^{(\ell)}(Z) := p_{\boldsymbol{x}}(Z^{n^\ell})/Z^{n^\ell} = \sum_{j=1}^{n} x_j \cdot Z^{n^\ell(j-1)} \quad \text{of degree } \leq n^{\ell+1} - n^\ell$$

Next, we compute

$$X_0 := \sum_{j=1}^{n} x_j \cdot [\alpha^j]_1 = [p_{\boldsymbol{x}}(\alpha)]_1, \quad \hat{X}_0 := \sum_{j=1}^{n} x_j \cdot [\alpha^{n(j-1)}]_2 = [p_{\boldsymbol{x}}(\alpha^n)/\alpha^n]_2$$

$$\forall \ell = 2, \ldots, d-1 : \Phi_\ell := \begin{cases} \sum_{j=1}^{n} x_j \cdot [\alpha^{n^\ell(j-1)}]_1 = \left[\phi_{\boldsymbol{x}}^{(\ell)}(\alpha)\right]_1 & \textbf{if } \ell \text{ even} \\ \sum_{j=1}^{n} x_j \cdot [\alpha^{n^\ell(j-1)}]_2 = \left[\phi_{\boldsymbol{x}}^{(\ell)}(\alpha)\right]_2 & \textbf{if } \ell \text{ odd} \end{cases}$$

Output $C := (X_0, \hat{X}_0, \{\Phi_\ell\}_{\ell=2}^{d-1})$ and $\mathsf{aux} = \boldsymbol{x}$.

$\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, \boldsymbol{f})$ Let $\boldsymbol{f} = (f_1, \ldots, f_m)$ be a vector of m n-variate homogeneous polynomials of degree d, where $d = 2^\delta$ is a power of 2. We use a representation of each polynomial f_i via a linear form $\hat{\boldsymbol{f}}_i : \mathbb{F}^{n^d} \to \mathbb{F}$ such that $f_i(\boldsymbol{x}) = \hat{\boldsymbol{f}}_i^\top \cdot \boldsymbol{x}^{(\delta)}$, where $\boldsymbol{x}^{(\delta)} = (\boldsymbol{x} \otimes \cdots \otimes \boldsymbol{x})$ is the result of taking the Kronecker product of $\boldsymbol{x}$ with itself δ times.[9] Next, set $\boldsymbol{x}^{(0)} := \boldsymbol{x}$ and proceed as follows.

– For $k = 1, \ldots, \delta - 1$, compute

$$\boldsymbol{x}^{(k)} := \boldsymbol{x}^{(k-1)} \otimes \boldsymbol{x}^{(k-1)}, \quad X_k := \sum_{j=1}^{n^{2^k}} x_j^{(k)} \cdot [\alpha^j]_1, \quad \hat{X}_k := \sum_{j=1}^{n^{2^k}} x_j^{(k)} \cdot [\alpha^{n^{2^k}(j-1)}]_2$$

[9] Since $\boldsymbol{x}^{(\delta)}$ has several terms repeated multiple times (e.g., after one product, the resulting vector contains both $x_i x_j$ and $x_j x_i$), we assume $\hat{\boldsymbol{f}}_i$ to always use the first of them, according to lexicographic order, and have 0 coefficients for the others.

Let us define the polynomials

$$p_{\boldsymbol{x}}^{(k)}(Z) := \sum_{j=1}^{n^{2^k}} x_j^{(k)} \cdot Z^j, \quad \hat{p}_{\boldsymbol{x}}^{(k)}(Z) := p_{\boldsymbol{x}}^{(k)}(Z^{n^{2^k}})/Z^{n^{2^k}} = \sum_{j=1}^{n^{2^k}} x_j^{(k)} \cdot Z^{n^{2^k}(j-1)}$$

and note that for every k the pair $(X_k, \hat{X}_k)$ is $([p_{\boldsymbol{x}}^{(k)}(\alpha)]_1, [\hat{p}_{\boldsymbol{x}}^{(k)}(\alpha)]_2)$.
$(X_k, \hat{X}_k)$ can be seen as a commitment to the vector $\boldsymbol{x}^{(k)} \in \mathbb{F}^{n^{2^k}}$.

– Compute the last vector $\boldsymbol{x}^{(\delta)} := \boldsymbol{x}^{(\delta-1)} \otimes \boldsymbol{x}^{(\delta-1)}$, and its commitment $X_\delta := \sum_{j=1}^{n^d} x_j^{(\delta)} \cdot [\alpha^j]_1 = [p_{\boldsymbol{x}}^{(\delta)}(\alpha)]_1$.
For $k = 1$ to δ, one can verify the correctness of the element X_k based on the correctness of the previous pair $(X_{k-1}, \hat{X}_{k-1})$ (which eventually reduces to the correctness of the commitment pair $(X_0, \hat{X}_0)$) by testing $e\,(X_k, [1]_2) \stackrel{?}{=} e\left(X_{k-1}, \hat{X}_{k-1}\right)$. This equality holds based on the fact that, for every k, $p_{\boldsymbol{x}}^{(k)}(Z) = p_{\boldsymbol{x}}^{(k-1)}(Z) \cdot \hat{p}_{\boldsymbol{x}}^{(k-1)}(Z)$ (see Claim 2).
The checks above can be seen as a way to progressively build trust in the elements $X_1, \ldots, X_\delta$. However for it to work we need that for a given k both elements of the previous pair $(X_{k-1}, \hat{X}_{k-1})$ are deemed correct.
– In this step we show how to enable the verification of the correctness of $\hat{X}_k$. This cannot be done via a quadratic equation, as we observed for X_k, but it is possible by letting the prover provide additional hints to the verifier.
The main idea of this step is that, for $k = 1$ to $\delta - 1$, we can factor $\hat{p}_{\boldsymbol{x}}^{(k)}(Z)$ as the product of 2^k polynomials (implicitly) known to the verifier, namely

$$\hat{p}_{\boldsymbol{x}}^{(k)}(Z) = \prod_{\ell=2^k}^{2^{k+1}-1} \phi_{\boldsymbol{x}}^{(\ell)}(Z) \quad \text{(cf. Section 4.2, Claim 1)}$$

To let the verifier check this factorization with a pairing computation, we break the verification of this product into a set of $\approx 2^k$ quadratic equations. The idea is that, for every k, the prover builds a binary tree of height k in which the 2^k polynomials are the leaves and then are multiplied pair-wise in a bottom-up tree fashion, i.e., each node of the tree is the multiplication of its child nodes. More precisely, if we index the nodes of the k-th tree with an integer $1 \leq \mu \leq 2^{k+1} - 1$, then an internal node $\mu \in \{1, \ldots, 2^k - 1\}$ of the k-th tree is a group element $\Psi_{k,\mu}$, which encodes the product of the polynomials encoded in the two child nodes $\Psi_{k,2\mu}$ and $\Psi_{k,2\mu+1}$. Instead, the leaves are the elements $\{\Phi_\ell = [\phi_{\boldsymbol{x}}^{(\ell)}(\alpha)]_b\}_{\ell=2^k}^{2^{k+1}-1}$ (where $b = (\ell \bmod 2) + 1$) that are included in the commitment. In detail, the computation of all the internal nodes $\Psi_{k,\mu}$ proceed as follows.
For every $k = 2, \ldots, \delta - 1$ and $\mu = 2^k, \ldots, 2^{k+1} - 1$, initialize the polynomials $\psi_{k,\mu}(Z) := \phi_{\boldsymbol{x}}^{(\mu)}(Z)$. These are the leaves of the k-th tree. Next, for $\mu = 2^k - 1, \ldots, 2$, compute

$$\Psi_{k,\mu} := \begin{cases} [\psi_{k,2\mu}(\alpha) \cdot \psi_{k,2\mu+1}(\alpha)]_1 & \textbf{if } \mu \text{ even} \\ [\psi_{k,2\mu}(\alpha) \cdot \psi_{k,2\mu+1}(\alpha)]_2 & \textbf{if } \mu \text{ odd} \end{cases}$$

Note that we do not compute the root node $\Psi_{k,1}$ but only stop at its children $\Psi_{k,2}$, $\Psi_{k,3}$. The root is indeed the element $\hat{X}_k$ already computed in the first step of this Open algorithm.

– Finally, we compute a linear-map evaluation proof for the commitment X_δ as follows. For every $i = 1$ to m, take the linear form $\hat{\boldsymbol{f}}_i : \mathbb{F}^{n^d} \to \mathbb{F}$ such that $f_i(\boldsymbol{x}) = \hat{\boldsymbol{f}}_i^\top \cdot \boldsymbol{x}^{(\delta)}$, and define the matrix $\mathbf{F} \in \mathbb{F}^{m \times N}$ that has $\hat{\boldsymbol{f}}_i^\top$ in the i-th row. We generate a proof $\hat{\pi} \in \mathbb{G}_1$ for $\boldsymbol{y} = \mathbf{F} \cdot \boldsymbol{x}^{(\delta)}$ as

$$\hat{\pi} := \sum_{\substack{i \in [m] \\ j,k \in [N] : j \neq k}} F_{i,j} \cdot x_k^{(\delta)} \cdot [\alpha^{N+1-j+k} \beta_i]_1$$

– Return $\pi := (\{X_k\}_{k=1}^{\delta}, \{\hat{X}_k, \{\Psi_{k,\mu}\}_{\mu=2}^{2^k-1}\}_{k=1}^{\delta-1}, \hat{\pi})$.

Ver$(\mathsf{ck}, C, \pi, \boldsymbol{f}, \boldsymbol{y})$ Parse the commitment as $C := (X_0, \hat{X}_0, \{\Phi_\ell\}_{\ell=2}^{d-1})$, and the proof $\pi := (\{X_k\}_{k=1}^{\delta}, \{\hat{X}_k, \{\Psi_{k,\mu}\}_{\mu=2}^{2^k-1}\}_{k=1}^{\delta-1}, \hat{\pi})$ as returned by Open.
Output 1 if all the following checks pass and 0 otherwise:

– For $k = 1$ to $\delta - 1$ and $\mu \in [2^k, 2^{k+1} - 1]$ set $\Psi_{k,\mu} := \Phi_\mu$.

– For $k = 2$ to $\delta - 1$, check the validity of the k-th tree of elements $\{\Psi_{k,\mu}\}_{\mu=2}^{2^k-1}$. First, check all the internal nodes, bottom-up:

$$\textbf{for } k = 2 \ldots \delta - 1, \textbf{for } \mu = 2^k - 1 \ldots 2 : \\ e(\Psi_{k,2\mu}, \Psi_{k,2\mu+1}) \stackrel{?}{=} \begin{cases} e(\Psi_{k,\mu}, [1]_2) & \textbf{if } \mu \text{ even} \\ e([1]_1, \Psi_{k,\mu}) & \textbf{if } \mu \text{ odd} \end{cases} \tag{1}$$

Second, check the roots of the trees:

$$\textbf{for } k = 1 \ldots \delta - 1 : e([1]_1, \hat{X}_k) \stackrel{?}{=} e(\Psi_{k,2}, \Psi_{k,3}) \tag{2}$$

– Check the validity of the chain of commitments:

$$\textbf{for } k = 1 \ldots \delta : e(X_k, [1]_2) \stackrel{?}{=} e(X_{k-1}, \hat{X}_{k-1}) \tag{3}$$

– Define the matrix $\mathbf{F}$ from $\boldsymbol{f}$ as in the Open algorithm, and check the proof for the linear map:

$$e\left(X_\delta, \sum_{\substack{i \in [m] \\ j \in [N]}} F_{i,j} \cdot [\alpha^{N+1-j} \beta_i]_2\right) \stackrel{?}{=} e\left(\hat{\pi}, [1]_2\right) \cdot e\left(\sum_{i=1}^{m} y_i \cdot [\alpha \beta_i]_1, [\alpha^N]_2\right) \tag{4}$$

It is easy to see that our commitments are additively homomorphic and that the scheme has efficient amortized verification as the verifier can precompute $\sum_{i \in [m], j \in [N]} F_{i,j} \cdot [\alpha^{N+1-j} \beta_i]_2$. We refer to the full version for further details about these properties as well as for a proof that the openings are linearly aggregatable in the sense of Def. 7.

Compactness. In our scheme, an opening consists of $2\delta + \sum_{k=2}^{\delta-1}(2^k - 2) = d$ group elements, and a commitment also comprises d elements. Since the degree is assumed to be a constant, $d = O(1)$, compactness follows.

Efficiency. It is easy to see that the complexity of Com is $O(nd)$, while Ver takes time $O(d+|\boldsymbol{y}|+|\boldsymbol{f}|)$ (and the $O(|\boldsymbol{f}|)$ part can be precomputed when using efficient verification). The most complex and computationally heavy procedure of our scheme is the Open algorithm, whose time complexity is $O(mdn^d \log n)$, which we justify as follows. Computing the commitments $(X_1, \ldots, X_\delta, \hat{X}_1, \ldots, \hat{X}_{\delta-1})$ in the first and second step takes time at most $\sum_{k=0}^{\delta} O(n^{2^k})$ which is $O(\delta n^d)$. Computing all the group elements $\Psi_{k,\mu}$ in the third step can take time at most $O(d^2 n^d \log n)$. This estimation is obtained by observing that: every $\psi_{k,\mu}(Z)$ has degree $< n^d$ (this is a non-tight worst case analysis, as many of them actually have much lower degree); for each node of the tree the polynomial $\psi_{k,\mu}(Z)$ can be computed via a multiplication of its children polynomials which, using FFT, takes time $O(dn^d \log n)$. So by summing over all the d elements $\{\psi_{k,\mu}\}_{k,\mu}$, we obtain the above estimation. Finally, the generation of $\hat{\pi}$ in the last step takes $O(mN \log N) = O(mdn^d \log n)$. This follows from an observation that, for every row $i = 1$ to m, the coefficients of the polynomial in α of degree $< 2N$ can be computed using an FFT-based multiplication instead of going over all the N^2 indices j, k.

4.2 Proof of Correctness

To prove correctness we proceed one by one on the equations of the verification algorithm. We begin recalling the definition of the polynomials

$$p_{\boldsymbol{x}}^{(k)}(Z) := \sum_{j=1}^{n^{2^k}} x_j^{(k)} \cdot Z^j,\ \hat{p}_{\boldsymbol{x}}^{(k)}(Z) := p_{\boldsymbol{x}}^{(k)}(Z^{n^{2^k}})/Z^{n^{2^k}},\ \phi_{\boldsymbol{x}}^{(\ell)}(Z) := p_{\boldsymbol{x}}^{(0)}(Z^{n^\ell})/Z^{n^\ell}$$

Verification equation (1). For $2 \le k \le \delta - 1$ and $2^k \le \mu \le 2^{k+1} - 1$, the first step of the verification algorithm sets $\Psi_{k,\mu} = \Phi_\mu$, for $2 \le k \le \delta - 1$ and $2^k \le \mu \le 2^{k+1} - 1$, where each Φ_μ is defined in Com as

$$\Phi_\mu = \left[\phi_{\boldsymbol{x}}^{(\mu)}(\alpha)\right]_b = \left[p_{\boldsymbol{x}}(\alpha^{n^\mu})/\alpha^{n^\mu}\right]_b \quad : b = 1 \text{ if } \mu \text{ even}, b = 0 \text{ if } \mu \text{ odd}$$

On the other hand, Open initializes the polynomials $\psi_{k,\mu}(Z) := \phi_{\boldsymbol{x}}^{(\mu)}(Z)$ and then, for $2 \le \mu \le 2^k - 1$, it constructs $\Psi_{k,\mu} = [\psi_{k,2\mu}(\alpha) \cdot \psi_{k,2\mu+1}(\alpha)]_b$, with $b = 1$ if μ is even and $b = 2$ if μ is odd. By the construction of $\Psi_{k,\mu}$ for $2 \le \mu \le 2^k - 1$, and having observed that both algorithms start from the same leaves, it is therefore clear that each check of equation (1) is satisfied.

Verification equation (2). The intuition is that the check $e([1]_1, \hat{X}_k) \stackrel{?}{=} e(\Psi_{k,2}, \Psi_{k,3})$ is verifying whether the element $\hat{X}_k = \left[\hat{p}_{\boldsymbol{x}}^{(k)}(\alpha)\right]_2$ is the root of the

k-th binary tree computed starting from the leaf nodes $\{\phi_{\boldsymbol{x}}^{(\mu)}(\alpha)\}_{\mu=2^k,\ldots,2^{k+1}-1}$, and where each node is the multiplication of its two children.

To show this, we observe that by the construction of the polynomials $\psi_{k,\mu}(Z)$ in Open as a multiplication tree, we have that

$$\psi_{k,2}(Z) \cdot \psi_{k,3}(Z) = \prod_{\ell=2^k}^{2^{k+1}-1} \phi_{\boldsymbol{x}}^{(\ell)}(Z)$$

The correctness of equation (2) then follows from the following Claim (whose proof appears in the full version), which shows that the polynomial $\hat{p}_{\boldsymbol{x}}^{(k)}(Z)$ encoded in $\hat{X}_k$ can be factored into the product $\prod_{\ell=2^k}^{2^{k+1}-1} \phi_{\boldsymbol{x}}^{(\ell)}(Z)$.

Claim 1 *Fix any vector $\boldsymbol{x}^{(0)} \in \mathbb{F}^n$ and for any $k \in [\delta-1]$, let $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} \otimes \boldsymbol{x}^{(k-1)}$ and $\hat{p}_{\boldsymbol{x}}^{(k)}(Z) = \sum_{j=1}^{n^{2^k}} x_j^{(k)} \cdot Z^{n^{2^k}(j-1)}$. For $2 \le \ell \le d-1$, let $\phi_{\boldsymbol{x}}^{(\ell)}(Z) = \sum_{j=1}^{n} x_j^{(0)} \cdot Z^{n^{\ell}(j-1)}$. Then, it holds $\hat{p}_{\boldsymbol{x}}^{(k)}(Z) = \prod_{\ell=2^k}^{2^{k+1}-1} \phi_{\boldsymbol{x}}^{(\ell)}(Z)$.*

Verification equation (3). By construction of Open, we have

$$\forall k \in [\delta] : X_k = [p_{\boldsymbol{x}}^{(k)}(\alpha)]_1, \quad \forall k \in [\delta-1] : \hat{X}_k = [p_{\boldsymbol{x}}^{(k)}(\alpha^{n^{2^k}})/\alpha^{n^{2^k}}]_2$$

and by construction of Com, we have

$$X_0 = [p_{\boldsymbol{x}}^{(0)}(\alpha)]_1, \quad \hat{X}_0 = [p_{\boldsymbol{x}}^{(0)}(\alpha^n)/\alpha^n]_2$$

Let us state the following claim (whose proof appears in the full version).

Claim 2 *Fix any vector $\boldsymbol{x}^{(0)} \in \mathbb{F}^n$ and for any $k \in [\delta]$, let $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} \otimes \boldsymbol{x}^{(k-1)}$ and $\hat{p}_{\boldsymbol{x}}^{(k)}(Z) = \sum_{j=1}^{n^{2^k}} x_j^{(k)} \cdot Z^{n^{2^k}(j-1)}$. Then for every $k \in [\delta]$ it holds $p_{\boldsymbol{x}}^{(k)}(Z) = p_{\boldsymbol{x}}^{(k-1)}(Z) \cdot \hat{p}_{\boldsymbol{x}}^{(k-1)}(Z)$.*

Then for every $1 \le k \le \delta$ it holds

$$e\left(X_{k-1}, \hat{X}_{k-1}\right) = \left[p_{\boldsymbol{x}}^{(k-1)}(\alpha) \cdot p_{\boldsymbol{x}}^{(k-1)}(\alpha^{n^{2^{k-1}}})/\alpha^{n^{2^{k-1}}}\right]_T = \left[p_{\boldsymbol{x}}^{(k)}(\alpha)\right]_T = e\left(X_k, [1]_2\right)$$

Verification equation (4). By construction of Open we have

$$\hat{\pi} = \sum_{\substack{i\in[m] \\ j,k\in[N]:j\neq k}} F_{i,j} \cdot x_k^{(\delta)} \cdot [\alpha^{N+1-j+k}\beta_i]_1$$

Thus, consider a correct output $y_i = f_i(\boldsymbol{x})$ which, by the definition of $\mathbf{F}$ in Open and Ver, is $y_i = \sum_{j\in[N]} F_{i,j} \cdot x_j^{(\delta)}$. Then it holds

$$
\begin{aligned}
& e\left(\sum_{\substack{i\in[m]\\ j,k\in[N]:j\neq k}} F_{i,j}\cdot x_k^{(\delta)}\cdot[\alpha^{N+1-j+k}\beta_i]_1,\,[1]_2\right)\cdot e\left(\sum_{i=1}^{m} y_i\cdot[\alpha\beta_i]_1,\,[\alpha^N]_2\right)\\
&= \left[\sum_{\substack{i\in[m]\\ j,k\in[N]:j\neq k}} F_{i,j}\cdot x_k^{(\delta)}\cdot\alpha^{N+1-j+k}\beta_i + \sum_{i\in[m],j\in[N]} F_{i,j}\cdot x_j^{(\delta)}\cdot[\alpha^{N+1}\beta_i]_1\right]_T\\
&= \left[\left(\sum_{k\in[N]} x_k^{(\delta)}\cdot\alpha^k\right)\left(\sum_{\substack{i\in[m]\\ j\in[N]}} F_{i,j}\cdot\alpha^{N+1-j}\beta_i\right)\right]_T\\
&= e\left(X_\delta,\sum_{i\in[m],j\in[N]} F_{i,j}\cdot[\alpha^{N+1-j}\beta_i]_2\right)
\end{aligned}
$$

4.3 Proof of Security

We prove the evaluation binding of our FC based on the N-Diffie-Hellman-Exponent (N-DHE) assumption [2], which we recall below.

Definition 8 (N-DHE [2]). *Let* $\mathsf{bgp} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ *be a bilinear group setting. The N-DHE holds if for every PPT $\mathcal{A}$ the following advantage is negligible*

$$\mathbf{Adv}_{\mathcal{A}}^{N\text{-}DHE}(\lambda) = \Pr[\mathcal{A}(\mathsf{bgp}, \{[\alpha^i]_1, [\alpha^i]_2\}_{i\in[2N]\setminus\{N+1\}}) = [\alpha^{N+1}]_1]$$

where the probability is over the random choice of $\alpha \leftarrow\!\!\$\, \mathbb{Z}_q$ and $\mathcal{A}$'s random coins.

Theorem 1. *If the n^d-DHE assumption holds, then the scheme* FC *of Section 4.1 satisfies evaluation binding.*

Proof. Consider an adversary $\mathcal{A}$ who returns a tuple $(C, \boldsymbol{f}, \boldsymbol{y}, \pi, \boldsymbol{y}', \pi')$ that breaks evaluation binding. Parse

$$\pi = (\{X_k\}_{k=1}^{\delta}, \{\hat{X}_k, \{\Psi_{k,\mu}\}_{\mu=2}^{2^k-1}\}_{k=1}^{\delta-1}, \hat{\pi}),\ \pi' = (\{X'_k\}_{k=1}^{\delta}, \{\hat{X}'_k, \{\Psi'_{k,\mu}\}_{\mu=2}^{2^k-1}\}_{j=1}^{\delta-1}, \hat{\pi}')$$

and recall that by definition both proofs verify for the same commitment $C = (X_0, \hat{X}_0, \{\Phi_\ell\}_{\ell=2}^{d-1})$ and that $\boldsymbol{y} \neq \boldsymbol{y}'$. Let us call this event Win.

Let us define Coll as the event that $\mathcal{A}$'s output is such that $\boldsymbol{\beta}^\top \cdot (\boldsymbol{y} - \boldsymbol{y}') = 0$, where $\boldsymbol{\beta}$ is the vector sampled in ck.

We can partition adversaries in two classes: those that make Coll occur and those that do not. Clearly it holds.

$$\Pr[\mathsf{Win}] \leq \Pr[\mathsf{Win} \wedge \mathsf{Coll}] + \Pr[\mathsf{Win} \mid \overline{\mathsf{Coll}}]$$

To prove the theorem we show that under the n^d-DHE assumption both probabilities are negligible.

For the first probability, $\Pr[\mathsf{Win} \wedge \mathsf{Coll}]$, it is easy to see that we can reduce it to the discrete logarithm assumption (which is implied by n^d-DHE). The idea of the reduction is that, if $\boldsymbol{\beta}^\top \cdot (\boldsymbol{y} - \boldsymbol{y}') = 0$ occurs then one can recover the value of β_i such that $y_i - y'_i \neq 0$. Hence a discrete logarithm adversary that receives as input $[\eta]_1, [\eta]_2$ can choose a random index $i^* \leftarrow\!\!\$\, [m]$, implicitly set $\beta_{i^*} = \eta$ and perfectly simulate all the group elements of ck. If $y_{i^*} \neq y'_{i^*}$ (which happens with probability $\geq 1/m$), then one can recover $\beta_{i^*} = \eta$. We don't formalize this reduction further as it is rather standard.

In the rest of the proof we focus on proving the remaining case, namely that $\Pr[\mathsf{Win} \mid \overline{\mathsf{Coll}}]$ is negligible. In particular, we show that for any PPT $\mathcal{A}$ there is a PPT $\mathcal{B}$ such that

$$\Pr[\mathsf{Win} \mid \overline{\mathsf{Coll}}] \leq \mathbf{Adv}_{\mathcal{B}}^{n^d\text{-}DHE}(\lambda)$$

$\mathcal{B}$ takes as input $\{[\alpha^i]_1, [\alpha^i]_2\}_{i \in [2N] \setminus \{N+1\}}$, samples $\boldsymbol{\beta} \leftarrow\!\!\$\, \mathbb{F}^m$ and generates ck, which is distributed identically to that generated by Setup.

Next, $\mathcal{B}$ runs $(C, \boldsymbol{f}, \underline{\boldsymbol{y}}, \pi, \boldsymbol{y}', \pi') \leftarrow \mathcal{A}(\mathsf{ck})$ and proceeds as follows.

It computes $z := \boldsymbol{\beta}^\top \cdot \boldsymbol{y}$ and $z' := \boldsymbol{\beta}^\top \cdot \boldsymbol{y}'$ (recall that conditioned on $\overline{\mathsf{Coll}}$, $z \neq z'$) and then outputs

$$(z' - z) \cdot (\hat{\pi} - \hat{\pi}')\,.$$

Next, we claim that for a successful adversary $\mathcal{A}$, $\mathcal{B}$'s output is $[\alpha^{N+1}]_1$.

Consider the executions of the Ver algorithm for π and π'.

First, for $k = 1$ to $\delta - 1$ and $\mu \in [2^k, 2^{k+1} - 1]$, let $\Psi_{k,\mu}$ and $\Psi'_{k,\mu}$ be the internal variables set in the first step of the verification algorithm. We observe that $\Psi_{k,\mu} = \Psi'_{k,\mu}$ since in both cases (cf. the first step of Ver) they are built from the same set of values $\{\Phi_\ell\}_{\ell=2}^{d-1}$ included in C, which is common to both executions of Ver.

Second, we argue that by the validity of the verification equation (1) for both proofs (and by the non-degeneracy of the pairing function) we obtain that $\Psi_{k,\mu} = \Psi'_{k,\mu}$ for every $k = 2, \ldots, \delta - 1$ and $\mu = 2^j - 1, \ldots, 2$. We show this by induction. Let us consider the case of μ even (μ odd is analogous). For $\mu = 2^k - 1, \ldots, 2^{k-1}$, we are checking the parents of the leaves, and it holds $\Psi_{k,2\mu} = \Psi'_{k,2\mu} = \Phi_{2\mu}$, $\Psi_{k,2\mu+1} = \Psi'_{k,2\mu+1} = \Phi_{2\mu+1}$ since $2\mu \in [2^k, 2^{k+1}-2]$ and $2\mu+1 \in [2^k+1, 2^{k+1}-1]$. Therefore, by the non-degeneracy of the pairing function we have

$$\left.\begin{aligned} e(\Phi_{2\mu}, \Phi_{2\mu+1}) &= e(\Psi_{k,\mu}, [1]_2) \\ e(\Phi_{2\mu}, \Phi_{2\mu+1}) &= e(\Psi'_{k,\mu}, [1]_2) \end{aligned}\right\} \;\Rightarrow\; \Psi_{k,\mu} = \Psi'_{k,\mu}$$

Next, using the fact $\Psi_{k,\mu} = \Psi'_{k,\mu}$ for $\mu = 2^k - 1, \ldots, 2^{k-1}$, we can apply the same argument inductively to obtain that $\Psi_{k,\mu'} = \Psi'_{k,\mu'}$ for $\mu' = 2^{k-1} - 1, \ldots, 2^{k-2}$. Eventually, we obtain that for all k, $\Psi_{k,\mu} = \Psi'_{k,\mu}$ for $\mu = 2, 3$.

Third, notice that by the validity of verification equations (3) and (2) for $k = 1$ (and by the non-degeneracy of the pairing function) we obtain that $X_1 = X'_1$ and $\hat{X}_1 = \hat{X}'_1$. Moving to $k > 1$, we can see that from the equalities $X_{k-1} = X'_{k-1}$ and $\hat{X}_{k-1} = \hat{X}'_{k-1}$, we can derive in a similar way $X_k = X'_k$ and $\hat{X}_k = \hat{X}'_k$. In particular for the latter we use the conclusion of the second claim. Notice that this argument leads to conclude that it must be the case that $X_\delta = X'_\delta$.

Finally, by the validity of the verification equation (4) for both proofs with the same X_δ, we have

$$e(\hat{\pi}, [1]_2)e([\alpha]_1, [\alpha^N]_2)^z = e(\hat{\pi}', [1]_2)e([\alpha]_1, [\alpha^N]_2)^{z'}$$
$$\Rightarrow \quad \hat{\pi} - \hat{\pi}' = (z - z') \cdot [\alpha^{N+1}]_1 \qquad \square$$

4.4 From Homogeneous to Generic Polynomials

We show how to go from an additive homomorphic FC scheme for homogenous polynomials to an FC that supports generic multivariate polynomials of the same degree. The basic idea is to extend vectors by prepending a 1 in the first position and then, instead of evaluating $f(\boldsymbol{x})$ one evaluates $\hat{f}(1, \boldsymbol{x})$ where $\hat{f}$ is the homogeneous polynomial in $n + 1$ variables defined as $\hat{f}(x_0, \ldots, x_n) := x_0^d \cdot f\left(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}\right)$, which is such that $\forall x : \hat{f}(1, \boldsymbol{x}) = f(\boldsymbol{x})$.

In order to preserve the additive homomorphic property, we actually let one commit to vectors $(0, \boldsymbol{x})$. Then a commitment to $(1, \boldsymbol{x})$ is obtained by adding homomorphically $(1, \mathbf{0})$ at verification time.

In terms of security, we show that the scheme from this transformation satisfies evaluation binding (and thus weak evaluation binding) provided that so does the FC we start from. See the full version for the transformation's details.

5 Additive-Homomorphic FC for Semi-Quadratic Arithmetic Programs

In this section we propose our second FC scheme that supports a new language called *semi-quadratic arithmetic programs* (sQAP). As we show in Section 5.3, an FC for sQAPs is sufficiently powerful to build an FC for monotone span programs [15] and thus, using known transformations, an FC for NC^1 circuits.[10]

In a nutshell, an sQAP checks the satisfiability of a class of quadratic equations (from which the name). More in detail, an sQAP defined by a matrix $\mathbf{M}$ accepts a pair of vectors $(\boldsymbol{z}, \boldsymbol{y})$ if the linear system of equations $(\mathbf{M} \mid \boldsymbol{y})$ has a solution $\boldsymbol{w}'$ which is in multiplicative relation with the input $\boldsymbol{z}$, i.e., $\boldsymbol{w}' = \boldsymbol{w} \circ \boldsymbol{z}$ for some $\boldsymbol{w}$. More formally:

[10] It is known that a circuit in the class NC^1 can be converted into a polynomial-size boolean formula, and the latter can be turned into a monotone span program of equivalent size, e.g. [20, Appendix G].

Definition 9 (Semi-Quadratic Arithmetic Programs). *A semi-quadratic arithmetic program (sQAP) $f : \mathbb{F}^n \times \mathbb{F}^m \to \{\mathsf{true}, \mathsf{false}\}$ over a finite field $\mathbb{F}$ is defined by a matrix $\mathbf{F} \in \mathbb{F}^{m \times n}$. On an input $\boldsymbol{x} = (\boldsymbol{z}, \boldsymbol{y})$, f accepts (i.e., outputs* true*) iff*

$$\exists \boldsymbol{w} \in \mathbb{F}^n : \mathbf{F} \cdot (\boldsymbol{w} \circ \boldsymbol{z}) = \boldsymbol{y}$$

We observe that sQAPs are a polynomial time language. Given $(\boldsymbol{z}, \boldsymbol{y})$, one can decide as follows. Define $\mathbf{F}'$ as the matrix of entries $F'_{i,j} = F_{i,j} \cdot z_j$ and output true if and only if $\exists \boldsymbol{w} \in \mathbb{F}^n : \mathbf{F}' \cdot \boldsymbol{w} = \boldsymbol{y}$ (e.g., using Gaussian elimination).

5.1 Our FC for sQAPs

We present our additive-homomorphic FC for sQAPs (see sec. 1.3 for an overview).

$\underline{\mathsf{Setup}(1^\lambda, n, m)}$ Let $m, n \geq 1$ be two integers representing the size of the sQAPs supported by the scheme (i.e., matrices in $\mathbb{F}^{m \times n}$) and thus the length of the input vectors (pairs in $\mathbb{F}^n \times \mathbb{F}^m$). Generate a bilinear group description $\mathsf{bgp} := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{BG}(1^\lambda)$, and let $\mathbb{F} := \mathbb{Z}_q$. Next, sample random $\alpha, \gamma \leftarrow\!\!\$\, \mathbb{F}, \boldsymbol{\beta} \leftarrow\!\!\$\, \mathbb{F}^m$ and output

$$\mathsf{ck} := \begin{pmatrix} \{[\alpha^j]_1, [\gamma^j]_1\}_{j \in [n]}, [(\alpha\gamma)^n]_2, \{\left[\alpha^j \beta_i \gamma^\ell\right]_2\}_{i \in [m], j \in [n], \ell \in [2n]}, \\ \{[\alpha^j \beta_i \gamma^{n+1}]_1\}_{i \in [m], j \in [2n] \setminus \{n+1\}}, \{\left[\alpha^j \beta_i \gamma^\ell\right]_1\}_{i \in [m], j, \ell \in [2n] : \ell \neq n+1} \end{pmatrix}$$

$\underline{\mathsf{Com}(\mathsf{ck}, \boldsymbol{x})}$ Given an input $\boldsymbol{x} = (\boldsymbol{z}, \boldsymbol{y})$, we compute

$$C_z := \sum_{j \in [n]} z_j \cdot [\gamma^j]_1, \quad C_y := \sum_{i \in [m]} y_i \cdot [\alpha\gamma\beta_i]_1$$

Note, we encode $\boldsymbol{z}$ with the polynomial $p_{\boldsymbol{z}}(X) = \sum_{j=1}^{m} z_j \cdot X^j$, and thus $C_z = [p_{\boldsymbol{z}}(\gamma)]_1$. We output $C := (C_z, C_y)$ and $\mathsf{aux} := (\boldsymbol{z}, \boldsymbol{y})$.

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, \mathbf{F})}$ Let $\mathbf{F} \in \mathbb{F}^{m \times n}$ be a sQAP which accepts the input $(\boldsymbol{z}, \boldsymbol{y})$ in aux. The opening algorithm performs the following steps:

- Compute a witness $\boldsymbol{w} \in \mathbb{F}^n$ such that $\mathbf{F} \cdot (\boldsymbol{w} \circ \boldsymbol{z}) = \boldsymbol{y}$ and compute a commitment to it as $W := [p_{\boldsymbol{w}}(\alpha)]_1 = \sum_{j \in [n]} w_j \cdot [\alpha^j]_1$.
- Next, we compute an encoding Φ_z of the matrix $\mathbf{F} \circ \mathbf{Z}$ where $\mathbf{Z} \in \mathbb{F}^{m \times n}$ is the matrix with $\boldsymbol{z}^\top$ in every row:

$$\Phi_z := \sum_{\substack{i \in [m] \\ j, \ell \in [n]}} F_{i,j} \cdot z_\ell \cdot \left[\alpha^{n+1-j} \beta_i \gamma^{n+1+\ell-j}\right]_2$$

Precisely, note that $\mathbf{F} \circ \mathbf{Z}$ is encoded in the terms including γ^{n+1} of the above polynomial, i.e., the (i,j)-th entry is in the term $F_{i,j} \cdot z_j \cdot [\alpha^{n+1-j} \gamma^{n+1} \beta_i]_2$.

– Finally, we compute an evaluation proof to show that the vector $\boldsymbol{w}$ committed in W is a solution to the linear system $((\mathbf{F} \circ \mathbf{Z}) \mid \boldsymbol{y})$, i.e., $(\mathbf{F} \circ \mathbf{Z}) \cdot \boldsymbol{w} = \mathbf{F} \cdot (\boldsymbol{w} \circ \boldsymbol{z}) = \boldsymbol{y}$:

$$\hat{\pi} := \sum_{\substack{i\in[m]\\ j,k\in[n]:j\neq k}} F_{i,j} \cdot z_j \cdot w_k \cdot [\alpha^{n+1-j+k}\beta_i\gamma^{n+1}]_1 + \sum_{\substack{i\in[m]\\ j,k,\ell\in[n]:\ell\neq j}} F_{i,j} \cdot z_\ell \cdot w_k \cdot \left[\alpha^{n+1-j+k}\beta_i\gamma^{n+1-j+\ell}\right]_1$$

Output $\pi := (W, \Phi_z, \hat{\pi})$.

$\underline{\mathsf{Ver}(\mathsf{ck}, C, \pi, \mathbf{F}, \mathsf{true})}$ First, compute $\Phi \leftarrow \sum_{i\in[m],j\in[m]} F_{i,j} \cdot [(\alpha\gamma)^{n+1-j}\beta_i]_2$ and then output 1 if all the following checks are satisfied.

$$e\left(C_z, \Phi\right) \stackrel{?}{=} e\left([1]_1, \Phi_z\right) \tag{5}$$

$$e\left(W, \Phi_z\right) \stackrel{?}{=} e\left(\hat{\pi}, [1]_2\right) \cdot e\left(C_y, [(\alpha\gamma)^n]_2\right) \tag{6}$$

We refer to the full version for the correctness proof. Here we observe that: proofs are succinct (three group elements), and commitments are additively homomorphic. Also, it is easy to see that the scheme enjoys efficient amortized verification: $\mathsf{VerPrep}$ is the algorithm that on input $\mathbf{F}$ computes the element Φ, and EffVer performs the two checks described in Ver.

5.2 Proof of Security

We prove the weak evaluation binding of our FC for sQAPs based on the following assumption that we call *double parallel bilinear Diffie-Hellman exponent* (DP-BDHE) assumption, as it can be seen as a "double version" of the PBDHE assumption introduced by Waters in [26]. In the full version we justify (n, m)-DP-BDHE in the generic group model.

Definition 10 ((n, m)-DP-BDHE assumption). *Let* $\mathsf{bgp} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ *be a bilinear group setting. The* (n, m)*-DP-BDHE holds if for every* $n, m = \mathsf{poly}(\lambda)$ *and any PPT* $\mathcal{A}$*, the following advantage is negligible*

$$\mathbf{Adv}^{(n,m)\text{-}DP\text{-}BDHE}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A}(\mathsf{bgp}, \Omega) = [\alpha^{n+1}\gamma^{n+1}\delta]_T] \quad \textit{where}$$

$$\Omega := \left(\begin{array}{l} \left\{[\alpha^j]_1, [\gamma^j]_1\right\}_{j\in[n]}, \left\{[\alpha^j\beta_i\gamma^{n+1}]_1\right\}_{\substack{i\in[m],j\in[2n]\\ j\neq n+1}}, \left\{\left[\alpha^j\beta_i\gamma^\ell\right]_1\right\}_{\substack{i\in[m],j,\ell\in[2n]\\ \ell\neq n+1}} \\ [(\alpha\gamma)^n]_2, \left\{\left[\alpha^j\beta_i\gamma^\ell\right]_2\right\}_{i\in[m],j\in[n],\ell\in[2n]}, \\ \left\{\left[\frac{\delta}{\beta_k}\right]_2\right\}_{k\in[m]}, \left\{\left[\frac{\alpha^j\beta_i\gamma^{n+1}\delta}{\beta_k}\right]_2\right\}_{\substack{j\in[n],i,k\in[m]\\ i\neq k}}, \left\{\left[\frac{\alpha^j\beta_i\gamma^\ell\delta}{\beta_k}\right]_2\right\}_{\substack{i,k\in[m],j\in[n]\\ \ell\in[2n]\setminus\{n+1\}}} \end{array} \right)$$

and the probability is over the random choices of $\alpha, \gamma, \delta \leftarrow\!\!\$\, \mathbb{Z}_q$*,* $\boldsymbol{\beta} \leftarrow\!\!\$\, \mathbb{Z}_q^m$ *and* $\mathcal{A}$*'s random coins.*

Theorem 2. *If the (n, m)-DP-BDHE assumption holds then the FC scheme of Section 5.1 satisfies weak evaluation binding.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the weak evaluation binding of the FC scheme. We use $\mathcal{A}$ to build a PPT adversary $\mathcal{B}$ against the (n, m)-DP-BDHE assumption. $\mathcal{B}$ runs on input the bilinear group description and the list of group elements Ω.

$\mathcal{B}$ takes a subset of the elements in Ω, sets ck as below, and runs $\mathcal{A}(\mathsf{ck})$.

$$\mathsf{ck} := \left(\begin{array}{l} \{[\alpha^j]_1, [\gamma^j]_1\}_{j\in[n]}, [(\alpha\gamma)^n]_2, \{\left[\alpha^j\beta_i\gamma^\ell\right]_2\}_{i\in[m],j\in[n],\ell\in[2n]}, \\ \{[\alpha^j\beta_i\gamma^{n+1}]_1\}_{i\in[m],j\in[2n]\setminus\{n+1\}}, \left\{\left[\alpha^j\beta_i\gamma^\ell\right]_1\right\}_{i\in[m],j,\ell\in[2n]:\ell\neq n+1} \end{array} \right)$$

Let $\mathcal{A}$'s output be $((\boldsymbol{z}, \boldsymbol{y}), \mathbf{F}, \mathsf{true}, \pi)$. If $\mathcal{A}$ is successful we have that: (i) the proof is valid for the commitment $C = \mathsf{Com}(\mathsf{ck}, (\boldsymbol{z}, \boldsymbol{y}))$, and (ii) the sQAP does not accept $(\boldsymbol{z}, \boldsymbol{y})$. If we parse $\pi := (W, \Phi_z, \hat{\pi})$, condition (i) means

$$e\left([p_{\boldsymbol{z}}(\gamma)]_1, \Phi\right) = e\left([1]_1, \Phi_z\right) \tag{7}$$

$$e\left(W, \Phi_z\right) = e\left(\hat{\pi}, [1]_2\right) \cdot e\left([\alpha\gamma\boldsymbol{\beta}^\top\boldsymbol{y}]_1, [(\alpha\gamma)^n]_2\right) \tag{8}$$

while condition (ii) means that for $\mathbf{F}' = (F_{i,j} \cdot z_j)_{i,j}$

$$\nexists \boldsymbol{w} \in \mathbb{F}^n : \mathbf{F}' \cdot \boldsymbol{w} = \boldsymbol{y} \tag{9}$$

As first step, for every $k \in [m]$, $\mathcal{B}$ computes $\pi'_k := e\left(\hat{\pi}, \left[\frac{\delta}{\beta_k}\right]_2\right)$. By the construction of Φ in the Ver algorithm and by equation (7) we have:

$$\begin{aligned} \Phi_z &= \left[\left(\sum_{\ell\in[n]} z_\ell \cdot \gamma^\ell \right) \cdot \left(\sum_{i\in[m],j\in[n]} F_{i,j} \cdot (\alpha\gamma)^{n+1-j}\beta_i \right) \right]_2 \\ &= \sum_{\substack{i\in[m]\\ j\in[n]}} F_{i,j} \cdot z_j \cdot [\alpha^{n+1-j}\beta_i\gamma^{n+1}]_2 + \sum_{\substack{i\in[m]\\ j,\ell\in[n]:\ell\neq j}} F_{i,j} \cdot z_\ell \cdot \left[\alpha^{n+1-j}\beta_i\gamma^{n+1-j+\ell}\right]_2 \end{aligned}$$

Hence, by applying equation (8), for every $k \in [m]$, it holds

$$\begin{aligned} \pi'_k = e\left(W, \sum_{j\in[n]} F_{k,j} \cdot z_j \cdot \left[\alpha^{n+1-j}\gamma^{n+1}\delta\right]_2\right) \cdot \left[-y_k \cdot (\alpha\gamma)^{n+1}\delta\right]_T \cdot \\ e\left(W, \sum_{\substack{i\in[m]\setminus\{k\}\\ j\in[n]}} F_{i,j} \cdot z_j \cdot \left[\frac{\alpha^{n+1-j}\beta_i\gamma^{n+1}\delta}{\beta_k}\right]_2\right) \cdot \\ e\left(W, \sum_{\substack{i\in[m]\\ j,\ell\in[n]:\ell\neq j}} F_{i,j} \cdot z_\ell \cdot \left[\frac{\alpha^{n+1-j}\beta_i\gamma^{n+1-j+\ell}\delta}{\beta_k}\right]_2\right) \cdot \left[-\sum_{i\in[m]\setminus\{k\}} y_i \cdot \frac{(\alpha\gamma)^{n+1}\beta_i\delta}{\beta_k}\right]_T \end{aligned}$$

As the second step, for every $k \in [m]$, $\mathcal{B}$ computes

$$\pi_k^* := \pi'_k \cdot e\left(W, -\sum_{\substack{i\in[m]\setminus\{k\}\\ j\in[n]}} F_{i,j}\cdot z_j \cdot \left[\frac{\alpha^{n+1-j}\beta_i\gamma^{n+1}\delta}{\beta_k}\right]_2\right)\cdot$$
$$e\left(W, -\sum_{\substack{i\in[m]\\ j,\ell\in[n]:\ell\neq j}} F_{i,j}\cdot z_\ell \cdot \left[\frac{\alpha^{n+1-j}\beta_i\gamma^{n+1-j+\ell}\delta}{\beta_k}\right]_2\right)\cdot$$
$$e\left(\left[(\alpha\gamma)^n\right]_1, \sum_{i\in[m]\setminus\{k\}} y_i \cdot \left[\frac{\alpha\gamma\delta\beta_i}{\beta_k}\right]_2\right)$$
$$= e\left(W, \sum_{j\in[n]} F_{k,j}\cdot z_j \cdot \left[(\alpha\gamma)^{n+1-j}\delta\right]_2\right)\cdot\left[-y_k\cdot(\alpha\gamma)^{n+1}\delta\right]_T$$

Notice that the elements above can be efficiently computed by $\mathcal{B}$, given the group elements included in its input Ω. In particular, for every $j, \ell \in [n]$ such that $\ell \neq j$ (and any $i, k \in [m]$), notice that $\left[\frac{\alpha^{n+1-j}\beta_i\gamma^{n+1-j+\ell}\delta}{\beta_k}\right]_2$ is part of $\left\{\left[\frac{\alpha^{j'}\beta_i\gamma^{\ell'}\delta}{\beta_k}\right]_2\right\}_{j'\in[n],\ell\in[2n]\setminus\{n+1\}}$.

If the sQAP is not satisfied, i.e., condition (9) holds, it means that $(\mathbf{F}' \mid \boldsymbol{y})$ is an inconsistent system of equations, thus there exists a vector $\boldsymbol{c} \in \mathbb{F}^m$ such that $\boldsymbol{c}^\top \cdot \mathbf{F}' = \mathbf{0}^\top$ and $\boldsymbol{c}^\top \cdot \boldsymbol{y} = \tau \neq 0$. Let $V := \{v \cdot \boldsymbol{c} : v \in \mathbb{F}\}$. Then any vector $\boldsymbol{v} \in V$ is such that

$$\boldsymbol{v}^\top \cdot \mathbf{F}' = (0, \ldots, 0) \ \wedge \ \boldsymbol{v}^\top \cdot \boldsymbol{y} \neq 0$$

In particular, one of them, $\boldsymbol{u} = \tau^{-1} \cdot \boldsymbol{c}$, is such that $\boldsymbol{u}^\top \cdot \boldsymbol{y} = 1$. So, $\mathcal{B}$ finds $\boldsymbol{u}$ such that

$$\boldsymbol{u}^\top \cdot (\mathbf{F}' \mid \boldsymbol{y}) = (0, \cdots, 0, 1) \tag{10}$$

(e.g., by Gaussian elimination), and then $\mathcal{B}$ computes and returns

$$\Delta^* = \prod_{k\in[m]} (\pi_k^*)^{-u_k}$$

We show below that, conditioned on $\mathcal{A}$ being successful, $\Delta^* = [(\alpha\gamma)^{n+1}\delta]_T$, and thus $\mathcal{B}$ succeeds in breaking the (n, m)-DP-BDHE assumption.

Expanding each term $\pi^*_{\ell,k}$ we have

$$\Delta^* = e\left(W, -\sum_{j\in[n]}\left[(\alpha\gamma)^{n+1-j}\delta\right]_2 \sum_{k\in[m]} F_{k,j}\cdot z_j\cdot u_k\right)\cdot\left[(\alpha\gamma)^{n+1}\delta\sum_{k\in[m]} y_k u_k\right]_T$$

The equality $\Delta^* = [(\alpha\gamma)^{n+1}\delta]_T$ follows from the fact that, by equation (10), we have that for every $j \in [n]$, $\sum_{k\in[m]} u_k \cdot F_{k,j} \cdot z_j = \sum_{k\in[m]} u_k \cdot F'_{k,j} = 0$ and that $\sum_{k\in[m]} u_k \cdot y_k = 1$. □

5.3 From FC for sQAPs to an FC for monotone span programs

Here we show how to construct an FC for monotone span programs from an additive-homomorphic FC for sQAPs, which can be instantiated using the scheme presented in section 5.1. We instantiate the same construction with vectors of length $n+1$ so that the commitment to

We recall the notion of (monotone) span programs (MSP) of Karchmer and Wigderson [15].

Definition 11 (Monotone Span Programs [15]). *A (monotone) span program for attribute universe $[n]$ is a pair $(\mathbf{M}, \rho)$ where $\mathbf{M} \in \mathbb{F}^{\ell \times m}$ and $\rho : [\ell] \to [n]$. Given an input $\boldsymbol{x} \in \{0,1\}^n$, we say that*

$$(\mathbf{M}, \rho) \text{ accepts } \boldsymbol{x} \text{ iff } (1, 0 \ldots, 0) \in \text{span}\langle \mathbf{M}_{\boldsymbol{x}} \rangle$$

where $\mathbf{M}_{\boldsymbol{x}}$ denotes the matrix obtained from $\mathbf{M}$ by taking only the i-th rows $\mathbf{M}_i$ for which $x_{\rho(j)} = 1$, and span *is the linear span of row vectors over $\mathbb{F}$.*

So, $(\mathbf{M}, \rho)$ accepts $\boldsymbol{x}$ iff there exist $\boldsymbol{w} \in \mathbb{F}^{\ell}$ such that

$$\sum_{i : x_{\rho(i)} = 1} w_i \cdot \mathbf{M}_i = (1, 0, \ldots, 0)$$

Notice that the MSP can be evaluated in polynomial time by using Gaussian elimination to find $\boldsymbol{w}$.

As in other cryptographic works, e.g., [19], we work with a restricted version of MSPs in which each input x_i is read only once. Hence, $\ell = n$ and ρ is a permutation, which (up to a reordering of the rows of $\mathbf{M}$) can be assumed to be the identity function. Notice that the one-use restriction can be removed by working with larger input vectors of length $k \cdot n$ in which each entry x_i is repeated k times, if k is an upper bound on the input's fan out.

Therefore, in what follows we assume a monotone span program defined by a matrix $\mathbf{M} \in \mathbb{F}^{n \times m}$ and we say that

$$\mathbf{M} \text{ accepts } \boldsymbol{x} \quad \text{iff} \quad \exists \boldsymbol{w} \in \mathbb{F}^n : (\boldsymbol{w} \circ \boldsymbol{x})^{\top} \cdot \mathbf{M} = (1, 0 \ldots, 0)$$

It is easy to see that MSPs are an instance of the sQAPs of Definition 9. Given $\mathbf{M}$, set $\mathbf{F} := \mathbf{M}^{\top}$ and consider sQAP inputs $(\boldsymbol{z}, \boldsymbol{y}) := (\boldsymbol{x}, (1, 0 \ldots, 0)^{\top})$. Then it is clear that the MSP $\mathbf{M}$ accepts $\boldsymbol{x}$ iff the sQAP $\mathbf{M}^{\top}$ accepts $(\boldsymbol{x}, (1, 0 \ldots, 0)^{\top})$.

We can use this relation to build an FC for monotone span programs from an FC for sQAP. In particular, we can do it in such a way to preserve the additive-homomorphic property, which allows us to use this scheme in the application to homomorphic signatures of section 6.

FC for MSPs from FC for sQAPs. Let FC' be a functional commitment scheme for sQAPs. We build a scheme FC for monotone span programs as follows.

$\underline{\mathsf{Setup}(1^{\lambda}, n, m)}$ output $\mathsf{ck} \leftarrow \mathsf{Setup}'(1^{\lambda}, n, m)$

$\underline{\mathsf{Com}(\mathsf{ck}, \boldsymbol{x})}$ Output $(C, \mathsf{aux}) \leftarrow \mathsf{Com}'(\mathsf{ck}, (\boldsymbol{x}, \mathbf{0}))$

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, \mathbf{M})}$ Assume aux is the auxiliary information of a commitment to a pair of vectors $(\boldsymbol{x}, \mathbf{0})$. The opening proceeds as follows.

- Compute a commitment to the vector $(\mathbf{0}, (1, \mathbf{0}))$ without using random coins: $(C_1, \mathsf{aux}_1) \leftarrow \mathsf{Com}(\mathsf{ck}, (\mathbf{0}, (1, \mathbf{0})); \emptyset)$.
- Use the additive homomorphism to compute the auxiliary information corresponding to the commitment to $(\boldsymbol{x}, (1, \mathbf{0}))$: $\hat{\mathsf{aux}} \leftarrow \mathsf{FC}'.\mathsf{Add}_{\mathsf{aux}}(\mathsf{ck}, \mathsf{aux}, \mathsf{aux}_1)$.
- Let $\mathbf{F} := \mathbf{M}^\top$ and run $\pi \leftarrow \mathsf{FC}'.\mathsf{Open}(\mathsf{ck}, \hat{\mathsf{aux}}, \mathbf{F})$.

Return π.

$\underline{\mathsf{Ver}(\mathsf{ck}, C, \pi, \mathbf{M}, \mathsf{true})}$ Compute $(C_1, \mathsf{aux}_1) \leftarrow \mathsf{Com}(\mathsf{ck}, (\mathbf{0}, (1, \mathbf{0})); \emptyset)$ and $\hat{C} \leftarrow \mathsf{FC}'.\mathsf{Add}(\mathsf{ck}, C, C_1)$. Output $\mathsf{FC}'.\mathsf{Ver}(\mathsf{ck}, \hat{C}, \pi, \mathbf{M}^\top, \mathsf{true})$.

We state the following theorem. The proof easily follows from the characterization of MSPs from sQAP mentioned earlier.

Theorem 3. *If* FC' *is a weak evaluation binding FC for sQAP, then* FC *is a weak evaluation binding FC for MSPs.*

Remark 1. We note that our FCs for sQAPs and MSPs allow the prover to show that the program accepts, but not that it rejects. We believe that the schemes could be changed to achieve this property and we leave it for future work. However, we observe that proving only acceptance is sufficient when the MSP is used to express that a circuit C outputs 1, due to the following observation. If the claim is that C outputs 0, prover and verifier could switch to use $\bar{C}$ (that is C with a negated output), build an MSP for it, and show it accepts.

6 Homomorphic signatures from additive-homomorphic functional commitments

Homomorphic Signatures. We recall the definition of homomorphic signatures (HS) of [1], extended to work with labeled programs [11], as used in several prior works, e.g., [7, 6].

In an HS scheme, the signer can sign a set of messages $\{x_i\}$ so that anyone can later compute a function f on the signed messages and obtain a signature that certifies the correctness of the result. Each set of messages is grouped into a "dataset" which has an identifier Δ (e.g., the filename); inside such dataset each message x_i is assigned a "label" τ_i (e.g., its position). So, more precisely, in HS the signer signs a collection of messages (x_i) with respect to a dataset identifier Δ and a label τ. Evaluation instead consists in executing a function f on the messages associated to some labels $\tau_1, \ldots, \tau_n$ of a dataset Δ. A property that makes HS an interesting primitive is that the signatures resulted from the evaluation are succinct, i.e., of size at most logatithmic in the input size. In this paper we generalize HS to the case of functions with multiple outputs and define the notion of *compactness*, which says that signatures are succinct with respect to both input and output size. We provide below formal definitions.

Labeled Programs [11]. Let $\mathcal{L}$ be the label space (e.g., $\mathcal{L} = \{0,1\}^*$ or $\mathcal{L} = [n]$). A *labeled program* $\mathcal{P}$ is a tuple $(f, \tau_1, ..., \tau_n)$ where $f : \mathcal{X}^n \to \mathcal{X}^m$ and every $\tau_i \in \mathcal{L}$ is the label of the i-th input of f. Given a function $g : \mathcal{X}^\ell \to \mathcal{X}^m$, we can compose t labeled programs $\mathcal{P}_1, \ldots, \mathcal{P}_t$ with $m_1, \ldots, m_t$ outputs respectively, into $\mathcal{P}^*$. The latter, denoted as $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_t)$, is the program obtained by evaluating g on the $\ell = \sum_{i=1}^{t} m_i$ outputs of $\mathcal{P}_1, \ldots, \mathcal{P}_t$. The labeled inputs of $\mathcal{P}^*$ are the distinct labeled inputs of $\mathcal{P}_1, \ldots, \mathcal{P}_t$ (all inputs with the same label are merged into a single input of $\mathcal{P}^*$). If $f_{id} : \mathcal{X} \to \mathcal{X}$ denotes the identity function and $\tau \in \mathcal{L}$, $\mathcal{I}_\tau = (f_{id}, \tau)$ is the identity program with label τ.

Definition 12 (Homomorphic Signature). *A homomorphic signature scheme* HS *is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{Eval})$ *that work as follows and satisfy* authentication correctness, evaluation correctness *and* succinctness.

$\mathsf{KeyGen}(1^\lambda, \mathcal{L}) \to (\mathsf{sk}, \mathsf{pk})$ *Given the security parameter λ and the label space $\mathcal{L}$, outputs a public key pk and a secret key sk. The public key pk defines the message space $\mathcal{X}$ and the set $\mathcal{F}$ of admissible functions.*

$\mathsf{Sign}(\mathsf{sk}, \Delta, \tau, x) \to \sigma$ *On input the secret key sk, a dataset identifier $\Delta \in \{0,1\}^*$, a label $\tau \in \mathcal{L}$, and a message $x \in \mathcal{X}$, outputs a signature σ.*

$\mathsf{Eval}(\mathsf{pk}, f, \sigma_1, \ldots, \sigma_n) \to \sigma$ *On input the public key pk, a function $f : \mathcal{X}^n \to \mathcal{X}^m$ in the class $\mathcal{F}$ and a tuple of signatures $(\sigma_i)_{i=1}^n$, outputs a new signature σ.*

$\mathsf{Ver}(\mathsf{pk}, \mathcal{P}, \Delta, \boldsymbol{y}, \sigma) \to \{0,1\}$ *On input the public key pk, a labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ with $f : \mathcal{X}^n \to \mathcal{X}^m$, a dataset identifier Δ, a value $\boldsymbol{y} \in \mathcal{X}^m$, and a signature σ, outputs either 0 (reject) or 1 (accept).*

Authentication Correctness. *Informally, authentication correctness means that a "fresh" signature generated by* Sign *on message x and label τ verifies correctly for x as output of the identity program $\mathcal{I}_\tau$. More formally, a scheme* HS *satisfies authentication correctness if for a given label space $\mathcal{L}$, all key pairs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathcal{L})$, any label $\tau \in \mathcal{L}$, dataset identifier $\Delta \in \{0,1\}^*$, and any signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \Delta, \tau, x)$, $\mathsf{Ver}(\mathsf{pk}, \mathcal{I}_\tau, \Delta, x, \sigma) = 1$ holds with all but negligible probability.*

Evaluation Correctness. *Informally, this property means that executing* Eval *with a function g on signatures $(\sigma_1, \ldots, \sigma_t)$, where σ_i verifies for x_i as output of $\mathcal{P}_i$, produces a signature σ that verifies for $g(x_1, \ldots, x_t)$ as output of the composed program $g(\mathcal{P}_1, \ldots, \mathcal{P}_t)$. More formally, fix a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathcal{L})$, a function $g : \mathcal{X}^\ell \to \mathcal{X}^m$, and a set of program/message/signature triples $\{(\mathcal{P}_i, \boldsymbol{x}_i, \sigma_i)\}_{i=1}^t$ such that $\mathsf{Ver}(\mathsf{pk}, \mathcal{P}_i, \Delta, \boldsymbol{x}_i, \sigma_i) = 1$. If $\boldsymbol{x}^* = g(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$, $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_t)$, and $\sigma^* = \mathsf{Eval}(\mathsf{pk}, g, \sigma_1, \ldots, \sigma_t)$, then $\mathsf{Ver}(\mathsf{pk}, \mathcal{P}^*, \Delta, \boldsymbol{x}^*, \sigma^*) = 1$ holds with all but negligible probability.*

Succinctness/Compactness. *An HS scheme* HS *is* succinct *(resp.* compact*) if there exists a universal polynomial $p(\lambda)$ such that for any keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathcal{L})$, integer $n = \mathsf{poly}(\lambda)$ and function $f : \mathcal{X}^n \to \mathcal{X}$ in $\mathcal{F}$ (resp. integers $n, m = \mathsf{poly}(\lambda)$ and function $f : \mathcal{X}^n \to \mathcal{X}^m$ in $\mathcal{F}$), messages $(x_1, \ldots, x_n) \in \mathcal{X}^n$, labels $(\tau_1, \ldots, \tau_n) \in \mathcal{L}^n$, and dataset $\Delta \in \{0,1\}^*$, if $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, \Delta, \tau_i, x_i)$ and $\sigma \leftarrow \mathsf{Eval}(\mathsf{pk}, f, \sigma_1, \ldots, \sigma_n)$, then $|\sigma| \leq p(\lambda) \cdot \log n$ (resp. $|\sigma| \leq p(\lambda) \cdot \log n \cdot \log m$).*

$\mathbf{Exp}^{\mathsf{strong\text{-}Ad\text{-}UF}}_{\mathcal{A},\mathsf{HS}}(\lambda)$

$T \leftarrow \emptyset$; $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, \mathcal{L})$

$(\mathcal{P}^*, \Delta^*, x^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}(\cdot)}(\mathsf{pk})$ // signing query phase

$b_{\mathsf{Ver}} \leftarrow \mathsf{Ver}(\mathsf{pk}, \mathcal{P}^*, \Delta^*, m^*, \sigma^*)$ // the signature verifies

$b_1 \leftarrow \exists j : (\Delta^*, \tau_j^*, \cdot, \cdot) \notin T$ // type-1: new dataset/label

$b_2 \leftarrow x^* \neq f^*(x_1, \ldots, x_n)$ // type-2: all inputs queried

where $\forall i : (\Delta^*, \tau_i^*, x_i, \cdot) \in T$ // but wrong result

return $b_{\mathsf{Ver}} \wedge (b_1 \vee b_2)$

Oracle $\mathcal{O}_{\mathsf{Sign}}(\Delta, \tau, m)$

if $(\Delta, \tau, \cdot, \cdot) \notin T$ **then**

$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \Delta, \tau, x)$

$T \leftarrow T \cup \{(\Delta, \tau, x, \sigma)\}$

return σ

else return $\perp$

Fig. 1: Strong adaptive security experiment for homomorphic signatures.

Remark 2 (Single-input vs. multi-input HS). The HS notion presented here allows one to sign the messages of a dataset one by one. We call such a scheme a *multi-input* HS. In contrast, *single-input* HS are HS schemes where Sign only works on input *all* the messages of the dataset.

Security Informally, an HS is secure if an adversary, without knowledge of the secret key, can only produce signatures that are either the ones obtained from the signer, or they are signatures obtained through the Eval algorithm on signatures obtained from the signer. The formalization of this intuition can have different strengths according to how a forgery is defined. We refer to [6] for a discussion on different notions of unforgeability. In this work we adopt the simplest and strongest notion from [6], called strong-adaptive security.

Definition 13 (Strong Adaptive Security). *Let* $\mathbf{Exp}^{\mathsf{strong\text{-}Ad\text{-}UF}}_{\mathcal{A},\mathsf{HS}}(\lambda)$ *be the security experiment of Fig. 1, and let* $\mathbf{Adv}^{\mathsf{strong\text{-}Ad\text{-}UF}}_{\mathcal{A},\mathsf{HS}}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{strong\text{-}Ad\text{-}UF}}_{\mathcal{A},\mathsf{HS}}(\lambda) = 1]$ *be the advantage of* $\mathcal{A}$ *against the strong adaptive security of scheme* HS*. We say that* HS *is* strong adaptive secure *if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $\epsilon(\lambda)$ *such that* $\mathbf{Adv}^{\mathsf{strong\text{-}Ad\text{-}UF}}_{\mathcal{A},\mathsf{HS}}(\lambda) \leq \epsilon(\lambda)$.

HS can also satisfy a privacy property, called context hiding [1], which informally says that signatures on outputs do not leak information about the inputs. An HS can have efficient amortized verification [7]; in brief this means that given f one can precompute a function-specific verification key which can be used later to verify any signature for f's outputs in at most polylogarithmic time. We give formal definitions of these properties in the full version.

6.1 From FCs to HS

Let FC be an additively homomorphic functional commitment scheme for a class of functions $\mathcal{F}$, such that the commitments are in $\mathcal{C}$ and $\mathsf{FC.Add} : \mathcal{C}^n \to \mathcal{C}$ is its homomorphic addition algorithm. Let LHS be an HS with message space $\mathcal{C}$ and that supports the evaluation of FC.Add. We use these two schemes to build an

$\mathsf{KeyGen}(1^\lambda, [n])$

$\mathsf{ck} \leftarrow \mathsf{FC.Setup}(1^\lambda, n, m)$

$(\mathsf{sk}_{\mathsf{LHS}}, \mathsf{pk}_{\mathsf{LHS}}) \leftarrow \mathsf{LHS.KeyGen}(1^\lambda, [n])$

$\mathsf{pk} := (\mathsf{pk}_{\mathsf{LHS}}, \mathsf{ck}), \mathsf{sk} := \mathsf{sk}_{\mathsf{LHS}}$

return $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \Delta, i, x_i)$

Let $\boldsymbol{e}_i$ s.t. $e_{i,i} = 1, e_{i,j} = 0\ \forall i \neq j$

$(C_i, \mathsf{aux}_i) \leftarrow \mathsf{FC.Com}(\mathsf{ck}, x_i \cdot \boldsymbol{e}_i)$

$\hat{\sigma}_i \leftarrow \mathsf{LHS.Sign}(\mathsf{sk}_{\mathsf{LHS}}, \Delta, i, C_i)$

return $\sigma_i := (\hat{\sigma}_i, C_i, \mathsf{aux}_i, i)$

$\mathsf{Eval}(\mathsf{pk}, f, \sigma_1, \ldots, \sigma_t)$

$C \leftarrow \mathsf{FC.Add}(\mathsf{ck}, C_1, \ldots, C_t)$

$\mathsf{aux} \leftarrow \mathsf{FC.Add}_{\mathsf{aux}}(\mathsf{ck}, \mathsf{aux}_1, \ldots, \mathsf{aux}_t)$

$\hat{\sigma} \leftarrow \mathsf{LHS.Eval}(\mathsf{pk}_{\mathsf{LHS}}, \mathsf{FC.Add}, \hat{\sigma}_1, \ldots, \hat{\sigma}_t)$

$\pi \leftarrow \mathsf{FC.Open}(\mathsf{ck}, \mathsf{aux}, \hat{f}_{\boldsymbol{i}})$

return $\sigma_{f,y} := (\hat{\sigma}, C, \pi_f)$

$\mathsf{Ver}(\mathsf{pk}, (f, \boldsymbol{i}), \Delta, \boldsymbol{y}, \sigma)$

$b_{\mathsf{LHS}} \leftarrow \mathsf{LHS.Ver}(\mathsf{pk}_{\mathsf{LHS}}, (\mathsf{FC.Add}, \boldsymbol{i}), \Delta, C, \hat{\sigma})$

if $\sigma = (\hat{\sigma}, C, \mathsf{aux}, i)$

$\pi \leftarrow \mathsf{FC.Open}(\mathsf{ck}, \mathsf{aux}, \hat{f}_{id,i})$

$b_{\mathsf{FC}} \leftarrow \mathsf{FC.Ver}(\mathsf{ck}, C, \hat{f}_{\boldsymbol{i}}, \boldsymbol{y}, \pi)$

return $b_{\mathsf{FC}} \wedge b_{\mathsf{LHS}}$

Fig. 2: HS from additive FC and LHS for FC.Add.

HS scheme HS for functions in $\mathcal{F}$. The scheme is described in Fig. 2. We refer to the introduction for an intuitive explanation of the construction.

Below, given a labeled program $(f, \boldsymbol{i})$ with $f : \mathcal{X}^t \to \mathcal{X}^m$ and $\boldsymbol{i} = (i_1, \ldots, i_t) \in [n]^t$, we define $\hat{f}_{\boldsymbol{i}} : \mathcal{X}^n \to \mathcal{X}^m$ as the n-input function that, ignoring inputs at positions $j \notin \boldsymbol{i}$, works identically as f.

The correctness of the scheme can be checked by inspection. In the following theorem we prove its security. For lack of space, we defer to the full version for the proof, further propertied of this construction, and a discussion on how to instantiate the LHS scheme based on [8].

Theorem 4. *If* LHS *is strongly-adaptive secure and* FC *is weak evaluation binding, then* HS *is strongly-adaptive secure.*

Acknowledgements. We would like to thank Pierre Bourse for initial discussions that inspired this work, and Ignacio Cascudo for a useful discussion.

This work has received funding in part from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under project PICOCRYPT (grant agreement No. 101001283), by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), and RED2018-102321-T, by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), by a research grant from Nomadic Labs and the Tezos Foundation, and by the Programma ricerca di ateneo UNICT 35 2020-22 linea 2 and by research gifts from Protocol Labs.

References

1. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer,

Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_10
2. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_16
3. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_5
4. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_5
5. Catalano, D., Fiore, D., Messina, M.: Zero-knowledge sets with short proofs. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 433–450. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_25
6. Catalano, D., Fiore, D., Nizzardo, L.: On the security notions for homomorphic signatures. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18. LNCS, vol. 10892, pp. 183–201. Springer, Heidelberg (Jul 2018). https://doi.org/10.1007/978-3-319-93387-0_10
7. Catalano, D., Fiore, D., Warinschi, B.: Homomorphic signatures with efficient verification for polynomial functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 371–389. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_21
8. Catalano, D., Marcedone, A., Puglisi, O.: Authenticating computation on groups: New homomorphic primitives and applications. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 193–212. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45608-8_11
9. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_8
10. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_37
11. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 301–320. Springer, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42045-0_16
12. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 99–108. ACM Press (Jun 2011). https://doi.org/10.1145/1993636.1993651
13. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC. pp. 469–477. ACM Press (Jun 2015). https://doi.org/10.1145/2746539.2746576
14. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_24

15. Karchmer, M., Wigderson, A.: On span programs. In: [1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference. pp. 102–111 (1993)
16. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_11
17. Katsumata, S., Nishimaki, R., Yamada, S., Yamakawa, T.: Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 622–651. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17656-3_22
18. Lai, R.W.F., Malavolta, G.: Subvector commitments with application to succinct arguments. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 530–560. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_19
19. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_4
20. Lewko, A.B., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_30
21. Libert, B., Peters, T., Joye, M., Yung, M.: Linearly homomorphic structure-preserving signatures and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 289–307. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_17
22. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) ICALP 2016. LIPIcs, vol. 55, pp. 30:1–30:14. Schloss Dagstuhl (Jul 2016). https://doi.org/10.4230/LIPIcs.ICALP.2016.30
23. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 499–517. Springer, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_30
24. Lipmaa, H., Pavlyk, K.: Succinct functional commitment for a large class of arithmetic circuits. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 686–716. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_23
25. Peikert, C., Pepin, Z., Sharp, C.: Vector and functional commitments from lattices. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 480–511. Springer, Heidelberg (Nov 2021). https://doi.org/10.1007/978-3-030-90456-2_16
26. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_4