Additive-Homomorphic Functional Commitments and Applications to Homomorphic Signatures*

Dario Catalano[1], Dario Fiore[2], and Ida Tucker[2]

[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 [LRY16], 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) [CFM08, LY10, CF13] and polynomial commitments (PC) [KZG10]—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 [LRY16, LM19] and semi-sparse polynomials [LP20].

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 [GW11].

In contrast, due to the falsifiability of the evaluation binding notion and as confirmed by the existing constructions [LRY16, LP20], 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 [CF13, KZG10, LRY16]). For this

* This article is the full version of the paper that appears in the proceedings of ASIACRYPT 2022,

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

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 [BF11]. 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 [LRY16, LM19].

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 multi-variate 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 [BGW05].

Compared to the FC for semi-sparse polynomials of [LP20] 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 whose security is based on established assumptions: the scheme of [LM19] relies on the generic group model, and that of [LP20] 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) [KW93] and show how to turn our FC for sQAPs into one for MSPs. Also,

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

via known transformations (see footnote 11) 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) [BF11] (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., [BF11, CFW14, GVW15] 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. [KNYY19] 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 [BF11, GVW15] or multilinear maps [CFW14]. Our result essentially shows that these powerful algebraic structures are not necessary to build such expressive HS.

In Appendix A.4 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 [LM19], 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. 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 [LM19] and prove its strong evaluation binding based on a falsifiable assumption, the parallel bilinear Diffie-Hellman exponent in [Wat11] (see Appendix D).

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 $\mathcal{L}$ (see Appendix A.3). 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 [GGPR13]. Therefore, due to the impossibility of Gentry and Wichs [GW11], 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

Functional commitments. Libert et al. [LRY16] 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 [LM19] extend the scheme of [LRY16] to support *linear maps* with *compact* openings, namely of size independent of both the input and the output lengths. Lipmaa and Pavlyk [LP20] 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 [LP20] and the one based on linearization are not additively homomorphic[5] and thus cannot be used in the applications discussed in this paper.

In a recent work, Peikert et al. [PPS21] 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.

Finally, as mentioned earlier, a construction of FCs can be obtained via succinct commitments and SNARKs for NP. This however yields an FC from *non-falsifiable assumptions* due to the latter being necessary for SNARKs [GW11]. The focus of this work is constructing FCs from falsifiable assumptions.

Homomorphic signatures. Homomorphic signatures (HS) allow a user Alice to sign a large collection of data $(x_1, \ldots, x_n)$ using her secret key. Then Alice can give this data and the associated signatures $(\sigma_1, \ldots, \sigma_n)$ to an untrusted server which can execute a computation f over it, producing the result $y = f(x_1, \ldots, x_n)$ along with a valid "signature" $\sigma_{f,y}$ on the result. The latter is a certificate that vouches for the correctness of y as the result of applying f on data signed by Alice, and it can be verified by anyone holding Alice's public key. A key feature of homomorphic signatures is *succinctness*: both fresh and "produced" signatures should be short, i.e. transmitting them should require much less bandwidth than sending out the original dataset. While there is a vast body of work on HS for linear functions, e.g., [BFKW09, GKKR10, AL11, CFW12, Fre12, LPJY13, CFN15], only few HS constructions support functions beyond linear ones. In particular, it appears that this primitive needs powerful algebraic structures, such as those needed for fully homomorphic encryption, i.e., lattices, or multilinear maps. This is indeed the case for [BF11, GVW15, CFW14]. An exception is a scheme by Katsumata et al. [KNYY19] that is based on pairings and supports NC^1 circuits (via monotone span programs). This scheme, however works in the weaker *single-input* model where the signer must sign the entire data set at once and is therefore less demanding in terms of homomorphic properties for combining different signatures. In this work, we obtain pairing-based HS for polynomials and NC^1 circuits that are multi-input. These HS schemes are much more versatile, e.g., they can be used in dynamic scenarios where a signer streams signed data to the server and the latter computes on the currently available snapshot.

[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.

As discussed in other works [KW20], the construction of homomorphic signatures by Gorbunov et al. [GVW15] can be interpreted as a generic construction from equivocable homomorphic commitments. This notion of homomorphic commitments can also be seen as a form of functional commitments except that *they are not succinct*. Our construction of HS from FCs is rather different, as it mainly relies on the succinctness of the FC and does not exploit any trapdoor property.

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 [LRY16, LM19] 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

$$X_0 = [p_{\boldsymbol{x}}^{(0)}(\alpha)]_1 = \sum_{j=1}^{n} x_j \cdot [\alpha^j]_1$$

[6] We use the bracket notation for bilinear groups of [EHK+13].

and consider for simplicity the case of $\delta = 1$ (i.e., opening to a polynomial of degree $d = 2$). 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

$$e(X_1, [1]_2) \stackrel{?}{=} e(X_0, \hat{X}_0) \quad \text{and} \quad e(X_2, [1]_2) \stackrel{?}{=} e(X_1, \hat{X}_1).$$

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 [LM19] 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 [LRY16, LM19], 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 [LM19]. 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 [LM19].

Compared to [LM19], 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 [LM19] 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. As we mentioned earlier, in Appendix D, we apply the same technique to show that the LMC of [LM19] is strong evaluation binding, without resorting to the GGM.

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 [LPJY13][7]. Finally, we notice that for the sake of this application the FC only needs to satisfy a

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

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 [EHK+13] 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) [LRY16]. A crucial feature that makes this primitive interesting and nontrivial is that both commitment and 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 [LM19], 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 [LRY16] is inspired by that of vector commitments [CF13]. 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* $\mathcal{A}$

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

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 [LM19]. 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 $f_i(\boldsymbol{x}) = \boldsymbol{y}_i$.

[8] This notion is similar in spirit to the basic security of accumulators [CL02].

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.*

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

3.2 Hiding and Zero-Knowledge

A functional commitment scheme can be hiding if the commitments generated by Com are hiding in the standard sense. In our work we define it via the notion of equivocation.

Definition 5 (Com-Hiding). *A FC has perfectly (resp. statistically) hiding commitments if there are simulator algorithms* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Setup}}, \mathsf{Sim}_{\mathsf{Com}}, \mathsf{Sim}_{\mathsf{Equiv}})$ *such that*

- *(i)* $\mathsf{Sim}_{\mathsf{Setup}}$ *generates indistinguishable keys, along with a trapdoor, i.e., the distributions* $\{\mathsf{ck} : \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n, m)\}$ *and* $\{\mathsf{ck} : (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda, n, m)\}$ *are identical (resp. statistically indistinguishable).*
- *(ii) for any vector* $\boldsymbol{x} \in \mathcal{X}^n$*, keys* $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda, n, m)$*, the following distributions are identical (resp. statistically indistinguishable):*

$$\{\mathsf{Com}(\mathsf{ck}, \boldsymbol{x})\} \approx \{(C, \mathsf{aux}) : (C, \widetilde{\mathsf{aux}}) \leftarrow \mathsf{Sim}_{\mathsf{Com}}(\mathsf{td}), \mathsf{aux} \leftarrow \mathsf{Sim}_{\mathsf{Equiv}}(\mathsf{td}, C, \widetilde{\mathsf{aux}}, \boldsymbol{x})\}$$

We define a notion of zero-knowledge for the openings produced by Open. Our notion is slightly more powerful than that of [LP20] as we require an independent simulator algorithm for creating fake commitments and for equivocating them, rather than letting the simulator simulate commitment and openings altogether in one shot. This variant is more versatile in applications where the commitment may have been created by another simulator.[9]

Definition 6 (Zero-knowledge openings). *An FC has perfect (resp. statistical) zero-knowledge openings if there is simulator* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Setup}}, \mathsf{Sim}_{\mathsf{Com}}, \mathsf{Sim}_{\mathsf{Equiv}}, \mathsf{Sim}_{\mathsf{Open}})$ *such that*

- *(i)* $\mathsf{Sim}_{\mathsf{Setup}}$ *generates indistinguishable keys, along with a trapdoor, i.e., the distributions* $\{\mathsf{ck} : \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n, m)\}$ *and* $\{\mathsf{ck} : (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda, n, m)\}$ *are identical (resp. statistically indistinguishable).*
- *(ii) for any vector* $\boldsymbol{x} \in \mathcal{X}^n$*, keys* $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda, n, m)$*, commitment* $(C, \mathsf{aux}) \leftarrow \mathsf{Sim}_{\mathsf{Com}}(\mathsf{ck})$*, equivocation* $\mathsf{aux}_x \leftarrow \mathsf{Sim}_{\mathsf{Equiv}}(\mathsf{td}, C, \mathsf{aux}, \boldsymbol{x})$*, and functions* $f_1, \ldots, f_Q \in \mathcal{F}$*, the following two distributions are identical (resp. statistically indistinguishable):*

$$(C, \{\mathsf{Sim}_{\mathsf{Open}}(\mathsf{td}, \mathsf{aux}, C, f_j, f_j(\boldsymbol{x}))\}_{j=1}^Q) \approx (C, \{\mathsf{Open}(\mathsf{ck}, \mathsf{aux}_x, f_j)\}_{j=1}^Q)$$

3.3 Additional properties of FCs

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.

[9] Our application to homomorphic signatures is an example (cf. Section 4.5).

Definition 7 (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* $(C, \mathsf{aux}) = \mathsf{Com}(\mathsf{ck}, \sum_{i=1}^{n} \boldsymbol{x}_i; r)$.

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 [CFW14] and preprocessing universal SNARKs [GKM+18], 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 8 (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 9. *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* $f^*(\boldsymbol{X}) = g(f_1(\boldsymbol{X}), \ldots, f_\ell(\boldsymbol{X}))$.

In Appendix A.2 we show a generic implication of the aggregation property: any functional commitment for linear forms $\boldsymbol{f}^\top \cdot \boldsymbol{x}$ that is evaluation binding and has linear aggregation, it also achieves strong evaluation binding. An interesting byproduct of the following theorem is a proof that the scheme for linear forms by Libert et al. [LRY16] satisfies strong evaluation binding. Indeed, it is easy to see that this scheme has linear aggregatable openings.

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. 10 [BGW05]).

More in detail, with our FC one can commit to a vector $\boldsymbol{x} \in \mathbb{F}^n$ and then open the commitment to a function $\boldsymbol{f} : \mathbb{F}^n \to \mathbb{F}^m$ such that the i-th output is $f_i(\boldsymbol{x})$, where $f_i(\boldsymbol{X})$ is a multivariate polynomial of total degree d, where d is a constant. The opening proofs of our FC scheme are compact, i.e., constant in both the input length n and the output length m.

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 4.6 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 discuss later in section for how to modify it in order to satisfy com-hiding and zero-knowledge openings.

$\underline{\mathsf{Setup}(1^\lambda, n, m, d)}$ 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

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

$\underline{\mathsf{Com}(\mathsf{ck}, \boldsymbol{x})}$ 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 $\mathsf{aux} = \boldsymbol{x}$.

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, \boldsymbol{f})}$ 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.[10] Next, set $\boldsymbol{x}^{(0)} := \boldsymbol{x}$ and proceed as follows.

[10] 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.

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

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

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 $e\left(X_k, [1]_2\right) \stackrel{?}{=} e\left(X_{k-1}, \hat{X}_{k-1}\right)$. 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

$$\mathbf{F} := \begin{bmatrix} \hat{\boldsymbol{f}}_1^\top \\ \vdots \\ \hat{\boldsymbol{f}}_m^\top \end{bmatrix} \in \mathbb{F}^{m \times N}$$

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

$\underline{\mathsf{Ver}(\mathsf{ck}, C, \pi, \boldsymbol{f}, \boldsymbol{y})}$ 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:

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

We refer the reader to Section 4.4 to see how this scheme has the properties of additive homomorphism, linear aggregation, and efficient amortized verification.

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 $2 \leq k \leq \delta - 1$ and $2^k \leq \mu \leq 2^{k+1} - 1$, the first step of the verification algorithm sets $\Psi_{k,\mu} = \Phi_\mu$, for $2 \leq k \leq \delta - 1$ and $2^k \leq \mu \leq 2^{k+1} - 1$, 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 $2 \leq \mu \leq 2^k - 1$, 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 $2 \leq \mu \leq 2^k - 1$, 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 at the end of this section), 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* $\hat{p}^{(k)}_{\boldsymbol{x}}(Z) = \sum_{j=1}^{n^{2^k}} x^{(k)}_j \cdot Z^{n^{2^k}(j-1)}$. *For* $2 \leq \ell \leq d-1$, *let* $\phi^{(\ell)}_{\boldsymbol{x}}(Z) = \sum_{j=1}^{n} x^{(0)}_j \cdot Z^{n^{\ell}(j-1)}$. *Then, it holds* $\hat{p}^{(k)}_{\boldsymbol{x}}(Z) = \prod_{\ell=2^k}^{2^{k+1}-1} \phi^{(\ell)}_{\boldsymbol{x}}(Z)$.

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

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

and by construction of Com, we have

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

Let us state the following claim (whose proof appears slightly below).

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* $\hat{p}^{(k)}_{\boldsymbol{x}}(Z) = \sum_{j=1}^{n^{2^k}} x^{(k)}_j \cdot Z^{n^{2^k}(j-1)}$. *Then for every* $k \in [\delta]$ *it holds* $p^{(k)}_{\boldsymbol{x}}(Z) = p^{(k-1)}_{\boldsymbol{x}}(Z) \cdot \hat{p}^{(k-1)}_{\boldsymbol{x}}(Z)$.

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

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

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

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

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

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

Proof (Proof of Claim 1). We prove the claim by induction on k.

For $k = 1$ we have

$$\begin{aligned}\hat{p}^{(1)}_{\boldsymbol{x}}(Z) &= \sum_{\ell=1}^{n^2} x^{(1)}_{\ell} \cdot Z^{n^2(\ell-1)} = \sum_{i,j=1}^{n} x^{(0)}_i \cdot x^{(0)}_j \cdot Z^{n^2(i+n(j-1)-1)} \\ &= \sum_{i,j=1}^{n} x^{(0)}_i \cdot x^{(0)}_j \cdot Z^{n^2(i-1)} \cdot Z^{n^2 n(j-1)} \\ &= \left(\sum_{i=1}^{n} x^{(0)}_i \cdot Z^{n^2(i-1)}\right) \cdot \left(\sum_{j=1}^{n} x^{(0)}_j \cdot Z^{n^3(j-1)}\right) = \phi^{(2)}_{\boldsymbol{x}}(Z) \cdot \phi^{(3)}_{\boldsymbol{x}}(Z)\end{aligned}$$

Next, for any $2 \le k \le \delta - 1$ assume that the claim holds for $k-1$, i.e.,

$$\hat{p}^{(k-1)}_{\boldsymbol{x}}(Z) = \prod_{\ell=2^{k-1}}^{2^k-1} \phi^{(\ell)}_{\boldsymbol{x}}(Z)$$

then we show that it holds for k. To this end, we first show below that

$$\hat{p}^{(k)}_{\boldsymbol{x}}(Z) = \hat{p}^{(k-1)}_{\boldsymbol{x}}(Z^{n^{2^{k-1}}}) \cdot \hat{p}^{(k-1)}_{\boldsymbol{x}}(Z^{n^{2^k}})$$

which main follows by the definition of $\boldsymbol{x}^{(k)}$:

$$\begin{aligned}\hat{p}^{(k)}_{\boldsymbol{x}}(Z) &= \sum_{\ell=1}^{n^{2^k}} x^{(k)}_{\ell} \cdot Z^{n^{2^k}(\ell-1)} = \sum_{i,j=1}^{n^{2^{k-1}}} x^{(k-1)}_i \cdot x^{(k-1)}_j \cdot Z^{n^{2^k}(i+n^{2^{k-1}}(j-1)-1)} \\ &= \sum_{i,j=1}^{n^{2^{k-1}}} x^{(k-1)}_i \cdot x^{(k-1)}_j \cdot Z^{n^{2^k}(i-1)} \cdot Z^{n^{2^k} n^{2^{k-1}}(j-1)} \\ &= \left(\sum_{i=1}^{n^{2^{k-1}}} x^{(k-1)}_i \cdot \left(Z^{n^{2^{k-1}}}\right)^{n^{2^{k-1}}(i-1)}\right) \cdot \left(\sum_{j=1}^{n^{2^{k-1}}} x^{(k-1)}_j \cdot Z^{n^{2^k} n^{2^{k-1}}(j-1)}\right) \\ &= \hat{p}^{(k-1)}_{\boldsymbol{x}}(Z^{n^{2^{k-1}}}) \cdot \hat{p}^{(k-1)}_{\boldsymbol{x}}(Z^{n^{2^k}})\end{aligned}$$

Next, let us apply the inductive assumption about $\hat{p}^{(k-1)}_{\boldsymbol{x}}(Z)$:

$$\begin{aligned}\hat{p}^{(k-1)}_{\boldsymbol{x}}(Z^{n^{2^{k-1}}}) &= \prod_{\ell=2^{k-1}}^{2^k-1} \phi^{(\ell)}_{\boldsymbol{x}}(Z^{n^{2^{k-1}}}) = \prod_{\ell=2^{k-1}}^{2^k-1} \phi^{(\ell+2^{k-1})}_{\boldsymbol{x}}(Z) \\ &= \prod_{\ell'=2^{k-1}+2^{k-1}}^{2^{k-1}+2^k-1} \phi^{(\ell')}_{\boldsymbol{x}}(Z) = \prod_{\ell'=2^k}^{3\cdot 2^{k-1}-1} \phi^{(\ell')}_{\boldsymbol{x}}(Z) \\ \hat{p}^{(k-1)}_{\boldsymbol{x}}(Z^{n^{2^k}}) &= \prod_{\ell=2^{k-1}}^{2^k-1} \phi^{(\ell)}_{\boldsymbol{x}}(Z^{n^{2^k}}) = \prod_{\ell=2^{k-1}}^{2^k-1} \phi^{(\ell+2^k)}_{\boldsymbol{x}}(Z) = \prod_{\ell'=2^k+2^{k-1}}^{2^k+2^k-1} \phi^{(\ell')}_{\boldsymbol{x}}(Z) \\ &= \prod_{\ell'=3\cdot 2^{k-1}}^{2^{k+1}-1} \phi^{(\ell')}_{\boldsymbol{x}}(Z)\end{aligned}$$

where above we applied the fact that for a positive integer ω it holds $\phi_{\boldsymbol{x}}^{(\ell)}(Z^{n^\omega}) = \phi_{\boldsymbol{x}}^{(\ell+\omega)}(Z)$. Finally, we can see that

$$\hat{p}_{\boldsymbol{x}}^{(k-1)}(Z^{n^{2^{k-1}}}) \cdot \hat{p}_{\boldsymbol{x}}^{(k-1)}(Z^{n^{2^k}}) = \prod_{\ell'=2^k}^{3\cdot 2^{k-1}-1} \phi_{\boldsymbol{x}}^{(\ell')}(Z) \prod_{\ell'=3\cdot 2^{k-1}}^{2^{k+1}-1} \phi_{\boldsymbol{x}}^{(\ell')}(Z) = \prod_{\ell'=2^k}^{2^{k+1}-1} \phi_{\boldsymbol{x}}^{(\ell')}(Z)$$

which concludes the proof of Claim 1. □

Proof (Proof of Claim 2). It follows via the following sequence of equations

$$\begin{aligned} p_{\boldsymbol{x}}^{(k-1)}(Z) \cdot \hat{p}_{\boldsymbol{x}}^{(k-1)}(Z) &= \left(\sum_{i=1}^{n^{2^{k-1}}} x_i^{(k-1)} \cdot Z^i\right)\left(\sum_{j=1}^{n^{2^{k-1}}} x_j^{(k-1)} \cdot Z^{n^{2^{k-1}}(j-1)}\right) \\ &= \sum_{i,j=1}^{n^{2^{k-1}}} x_i^{(k-1)} x_j^{(k-1)} \cdot Z^{i+(j-1)n^{2^{k-1}}} = \sum_{\ell=1}^{n^{2^k}} x_\ell^{(k)} \cdot \alpha^\ell \\ &= p_{\boldsymbol{x}}^{(k)}(Z) \end{aligned}$$

where the all but last equality holds by that $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} \otimes \boldsymbol{x}^{(k-1)}$.

4.3 Proof of Security

We prove the evaluation binding of our FC based on the N-Diffie-Hellman-Exponent (N-DHE) assumption [BGW05], which we recall below.

Definition 10 (N-DHE [BGW05]). *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 $(C, \boldsymbol{f}, \boldsymbol{y}, \pi, \boldsymbol{y}', \pi') \leftarrow \mathcal{A}(\mathsf{ck})$ 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

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

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

$$\begin{aligned} & e(\hat{\pi}, [1]_2) e([\alpha]_1, [\alpha^N]_2)^z = e(\hat{\pi}', [1]_2) e([\alpha]_1, [\alpha^N]_2)^{z'} \\ \Rightarrow \quad & \hat{\pi} - \hat{\pi}' = (z - z') \cdot [\alpha^{N+1}]_1 \end{aligned}$$ □

4.4 Additive homomorphism, aggregation, and efficient verification

It is easy to see that, by the construction of the elements of the commitment, our FC scheme for homogenous polynomials is additively homomorphic. Below is the formal specification of the homomorphic addition algorithms. Their correctness can be easily verified by inspection.

$\mathsf{Add}(\mathsf{ck}, C_1, C_2)$ Parse $C_i := (X_0^{(i)}, \hat{X}_0^{(i)}, \{\Phi_\ell^{(i)}\}_{\ell=2}^{d-1})$ for $i = 1, 2$ and return $C := (X_0, \hat{X}_0, \{\Phi_\ell\}_{\ell=2}^{d-1})$ computed by performing the group operation on each pair of elements of the commitment tuple, i.e., $X_0 \leftarrow X_0^{(1)} + X_0^{(2)}$, etc.

$\mathsf{Add}_{\mathsf{aux}}(\mathsf{ck}, \mathsf{aux}_1, \mathsf{aux}_2)$ return $\mathsf{aux}_1 + \mathsf{aux}_2 \in \mathbb{Z}_q^n$.

Linear Aggregation Next we show that the scheme has aggregatable openings with respect to linear functions over the vector of polynomials. Namely given ℓ openings for vectors of polynomials $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_\ell$ respectively, one can compute an opening for the (homogeneous) polynomial $\gamma_1 \cdot \boldsymbol{f}_1 + \cdots + \gamma_\ell \cdot \boldsymbol{f}_\ell$. This property follows from the linear aggregation of the linear map commitment that we are implicitly using in our construction for polynomials. For simplicity we show aggregation for two openings, the extension to ℓ openings is straightforward.

$\mathsf{AggLin}(\mathsf{ck}, C, (\pi_1, \boldsymbol{f}_1, \boldsymbol{y}_1), (\pi_2, \boldsymbol{f}_2, \boldsymbol{y}_2), (\gamma_1, \gamma_2))$ For $i = 1, 2$, parse

$$\pi_i := (\{X_k^{(i)}\}_{k=1}^{\delta}, \{\hat{X}_k^{(i)}, \{\Psi_{k,\mu}^{(i)}\}_{\mu=2}^{2^k-1}\}_{j=1}^{\delta-1}, \hat{\pi}_i)$$

and proceed as follows. If the first portion of the openings differs, i.e.,

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

then output $\perp$. Otherwise compute $\hat{\pi} = \gamma_1 \cdot \hat{\pi}_1 + \gamma_2 \cdot \hat{\pi}_2 \in \mathbb{G}_1$ and output

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

To show the correctness of the (linear) aggregation property, we mainly need to show that from two proofs $\hat{\pi}_1$, $\hat{\pi}_2$ that satisfy the verification equation (4) for matrices $\mathbf{F}_1$ and $\mathbf{F}_2$ (that are built from the vectors of polynomials $\boldsymbol{f}_1, \boldsymbol{f}_2$ as described in Open) and for results $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ respectively, we have that the proof $\hat{\pi} = \gamma_1 \cdot \hat{\pi}_1 + \gamma_2 \cdot \hat{\pi}_2$ computed by AggLin satisfies the verification equation (4) for the matrix $\mathbf{F} = \gamma_1 \cdot \mathbf{F}_1 + \gamma_2 \cdot \mathbf{F}_2$ (which in turn corresponds to the vector of polynomials $\boldsymbol{f} = \gamma_1 \cdot \boldsymbol{f}_1 + \gamma_2 \cdot \boldsymbol{f}_2$) and for the result $\boldsymbol{y} = \gamma_1 \cdot \boldsymbol{y}_1 + \gamma_2 \cdot \boldsymbol{y}_2$.

$$\begin{aligned} & e(X_\delta, \sum_{\substack{i \in [m] \\ j \in [N]}} (\gamma_1 F_{1,i,j} + \gamma_2 F_{2,i,j}) \cdot [\alpha^{N+1-j} \beta_i]_2) \\ & = \left(e\left(\hat{\pi}_1, [1]_2\right) e\left(\sum_{i=1}^{m} y_{1,i} \cdot [\alpha\beta_i]_1, [\alpha^N]_2 \right) \right)^{\gamma_1} \cdot \left(e\left(\hat{\pi}_2, [1]_2\right) e\left(\sum_{i=1}^{m} y_{2,i} \cdot [\alpha\beta_i]_1, [\alpha^N]_2 \right) \right)^{\gamma_2} \\ & = e\left(\hat{\pi}, [1]_2\right) \cdot e\left(\sum_{i=1}^{m} y_i \cdot [\alpha\beta_i]_1, [\alpha^N]_2 \right) \end{aligned}$$

Efficient amortized verification This property follows in a straightforward way from the observation that, given $\boldsymbol{f}$ and ck, one can precompute $\mathsf{vk}_{\boldsymbol{f}} := \sum_{i\in[m],j\in[N]} F_{i,j} \cdot [\alpha^{N+1-j}\beta_i]_2$. Given this $\mathsf{vk}_{\boldsymbol{f}}$ the efficient verification runs in time $O(m+d)$ in order to check equations (1)–(2)–(3), and then check equation $e\left(X_\delta, \mathsf{vk}_{\boldsymbol{f}}\right) \stackrel{?}{=} e\left(\hat{\pi}, [1]_2\right) \cdot e\left(\sum_{i=1}^m y_i \cdot [\alpha\beta_i]_1, [\alpha^N]_2\right)$ in place of equation (4).

4.5 Extension to hiding and zero-knowledge

We discuss how our construction can be extended so that the commitments become hiding and the openings are zero-knowledge. We start from the observation that a commitment consists of d group elements such that each of them is the evaluation of the same polynomial $p_{\boldsymbol{x}}(Z)$ on d distinct points $(\alpha, \alpha^n, \ldots, \alpha^{n^{d-1}})$. Hence, we can instantiate the same construction in order to commit to vectors of length $n+d$ so that the commitment of $\boldsymbol{x}$ is a commitment of the vector $(\boldsymbol{r}, \boldsymbol{x})$ where $\boldsymbol{r} \leftarrow\!\!\$\, \mathbb{F}^d$. This way a commitment is distributed like a tuple of d random group elements. Also, notice that the scheme remains additively homomorphic and Add_r simply computes the corresponding linear function over the $\boldsymbol{r}$ vectors of the commitments. Efficient verification and linear aggregation also remain preserved.

The other change consists into encoding an n-variate polynomial over $\boldsymbol{x}$ into a polynomial with $n+d$ variables in which the coefficients that touch the terms including one of the r_i's are set to 0. This ensures that the new scheme is also correct and evaluation binding remains unchanged.

Finally, the scheme has zero-knowledge openings based on the observation that for a fixed commitment C and function $\boldsymbol{f}$, the proof is unique. Let the trapdoor be the values $\alpha, \boldsymbol{\beta}$ used to generate the commitment key ck. The $\mathsf{Sim}_{\mathsf{Com}}$ algorithm is defined so that it generates a commitment to $\mathbf{0}$, which we recall is a commitment to the vector $\boldsymbol{\rho} := (\boldsymbol{r}, \mathbf{0})$.

The $\mathsf{Sim}_{\mathsf{Equiv}}$ algorithm, on input the trapdoor, the auxiliary information $\boldsymbol{r}$, the commitment C and a vector $\boldsymbol{x}$, uses α in order to find a vector $\boldsymbol{r}_x$ such that C can be also written as the result of computing $\mathsf{Com}(\mathsf{ck}, (\boldsymbol{r}_x, \boldsymbol{x}))$. It is not difficult to see that this can be done via interpolation as follows. Let $\chi_0 = \alpha(p_{\boldsymbol{\rho}}(\alpha) - p_{(\mathbf{0},\boldsymbol{x})}(\alpha))$ and, for $\ell = 1, \ldots, d-1$, let $\chi_\ell = p_\rho(\alpha^{n^\ell}) - p_{(\mathbf{0},\boldsymbol{x})}(\alpha^{n^\ell})$. Then, find the polynomial r' of degree $d-1$ such that for every $\ell = 0, \ldots, d-1$ it holds $r'(\alpha^{n^\ell}) = \chi_\ell$ (here we use that with overwhelming probability over the choice of α the d points α^{n^ℓ} are all distinct), and set $\boldsymbol{r}_x$ as the coefficients vector of r'.

Finally, given a commitment C and auxiliary information $\boldsymbol{r}$, the simulator $\mathsf{Sim}_{\mathsf{Open}}$ can create the proof elements $\{X_k\}_{k=1}^{\delta}, \{\hat{X}_k, \{\Psi_{k,\mu}\}_{\mu=2}^{2^k-1}\}_{k=1}^{\delta-1})$ by computing them as in the Open algorithm using $\boldsymbol{\rho}$ as the committed vector. Finally, the proof $\hat{\pi}$ can be simulated using the trapdoor by computing

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

4.6 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.

$\mathsf{Setup}(1^\lambda, n, m, d)$ run $\mathsf{Setup}'(1^\lambda, n+1, m, d')$, where d' is the smallest power of 2 greater than d, and return the same output.

$\mathsf{Com}(\mathsf{ck}, \boldsymbol{x})$ output $\mathsf{Com}'(\mathsf{ck}, (0, \boldsymbol{x}))$.

$\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, \boldsymbol{f})$ Assume aux is the auxiliary information of a commitment to a vector $(0, \boldsymbol{x})$, i.e., with 0 in the first position. The opening proceeds as follows.

- Compute a commitment to the vector $(1, \mathbf{0})$ without using random coins: $(C_1, \mathsf{aux}_1) \leftarrow \mathsf{Com}(\mathsf{ck}, (1, 0, \ldots, 0); \emptyset)$.
- Use the additive homomorphism to compute the auxiliary information corresponding to the commitment to vector $(1, \boldsymbol{x})$: $\hat{\mathsf{aux}} \leftarrow \mathsf{FC}'.\mathsf{Add}_{\mathsf{aux}}(\mathsf{ck}, \mathsf{aux}, \mathsf{aux}_1)$.
- Let $\boldsymbol{f} = (f_1, \ldots, f_m)$, where each f_i is an n-variate polynomial of degree d. Let d' be the smallest power of 2 greater than d. For every $i = 1$ to m, define the homogenized polynomial (whose degree is a power of 2)

$$\hat{f}_i(x_0, \ldots, x_n) := x_0^{d'} \cdot f_i\left(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}\right)$$

  and run $\hat{\pi} \leftarrow \mathsf{FC}'.\mathsf{Open}(\mathsf{ck}, \hat{\mathsf{aux}}, \hat{\boldsymbol{f}})$.

Return $\pi := \hat{\pi}$.

$\mathsf{Ver}(\mathsf{ck}, C, \pi, \boldsymbol{f}, \boldsymbol{y})$ Define the vector of homogeneous polynomials $\hat{\boldsymbol{f}}$ from $\boldsymbol{f}$ as in the Open algorithm. Compute $(C_1, \mathsf{aux}_1) \leftarrow \mathsf{Com}(\mathsf{ck}, (1, 0, \ldots, 0); \emptyset)$ and $\hat{C} \leftarrow \mathsf{FC}'.\mathsf{Add}(\mathsf{ck}, C, C_1)$. Output

$$\mathsf{FC}'.\mathsf{Ver}(\mathsf{ck}, \hat{C}, \hat{\pi}, \hat{f}, z)$$

We state the following theorem to formalize this transformation.

Theorem 2. *If FC' is a functional commitment for homogeneous polynomials that satisfies evaluation binding, additive homomorphism, linear aggregation, efficient verification, com-hiding and zero-knowledge openings, then FC is a functional commitment for multivariate polynomials that satisfy the same properties.*

The proof is rather straightforward and is omitted. For evaluation binding we note that an attack against FC immediately implies one for FC'. Indeed, if the attack is valid the commitment $\hat{C}$ computed by the verifier would be validly opened at two distinct values.

In Appendix B we also show a stronger transformation that starts from a strong evaluation binding FC and obtains one with the same property.

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 5.4, an FC for sQAPs is sufficiently powerful to build an FC for monotone span programs [KW93] and thus, using known transformations, an FC for NC^1 circuits.[11]

[11] 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. [LW11, Appendix G].

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:

Definition 11 (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

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

$\underline{\mathsf{Com}(\mathsf{ck}, \boldsymbol{x})}$ 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 the commitment $C := (C_z, C_y)$ and auxiliary information $\mathsf{aux} := (\boldsymbol{z}, \boldsymbol{y})$.

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, \mathbf{F})}$ 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.

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

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

Before moving to prove the correctness of the scheme 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 Correctness

Let us first focus on the equation (5), which follows easily by construction of C_z in Com and Φ in Ver:

$$\begin{aligned} e\left(C_z, \Phi\right) &= \left[p_{\boldsymbol{z}}(\gamma) \cdot \left(\sum_{i \in [m], j \in [n]} F_{i,j} \cdot (\alpha\gamma)^{n+1-j} \beta_i\right)\right]_T \\ &= \left[\sum_{i \in [m], j, \ell \in [n]} F_{i,j} \cdot z_\ell \cdot \alpha^{n+1-j} \beta_i \gamma^{n+1-j+\ell}\right]_T = e\left([1]_1, \Phi_z\right) \end{aligned}$$

Next, let us focus on the equation (6). By construction of W we have $e\left(W, \Phi_z\right) = [p_{\boldsymbol{w}}(\alpha) \cdot \phi_z]_T$, which is

$$\begin{aligned}
p_{\boldsymbol{w}}(\alpha) \cdot \phi_z &= \sum_{\substack{i\in[m]\\ j,k\in[n]}} F_{i,j} \cdot z_j \cdot w_k \cdot \alpha^{n+1-j+k}\beta_i\gamma^{n+1} + \sum_{\substack{i\in[m]\\ j,k,\ell\in[n]:\ell\neq j}} F_{i,j} \cdot z_\ell \cdot w_k\alpha^{n+1-j+k}\beta_i\gamma^{n+1-j+\ell} \\
&= \sum_{\substack{i\in[m]\\ j\in[n]}} F_{i,j} \cdot z_j \cdot w_j \cdot \alpha^{n+1}\beta_i\gamma^{n+1} + \sum_{\substack{i\in[m]\\ j,k\in[n]:j\neq k}} F_{i,j} \cdot z_j \cdot w_k \cdot \alpha^{n+1-j+k}\beta_i\gamma^{n+1} + \\
&\qquad \sum_{\substack{i\in[m]\\ j,k,\ell\in[n]:\ell\neq j}} F_{i,j} \cdot z_\ell \cdot w_k\alpha^{n+1-j+k}\beta_i\gamma^{n+1-j+\ell} \\
&= \sum_{i\in[m]} y_i \cdot (\alpha\gamma)^{n+1}\beta_i + \sum_{\substack{i\in[m]\\ j,k\in[n]:j\neq k}} F_{i,j} \cdot z_j \cdot w_k \cdot \alpha^{n+1-j+k}\beta_i\gamma^{n+1} + \\
&\qquad \sum_{\substack{i\in[m]\\ j,k,\ell\in[n]:\ell\neq j}} F_{i,j} \cdot z_\ell \cdot w_k\alpha^{n+1-j+k}\beta_i\gamma^{n+1-j+\ell}
\end{aligned}$$

where the last equality follows from the fact that for every $i \in [m]$, it holds $\sum_{j\in[n]} F_{i,j}z_jw_j = y_i$.

Notice that $C_y = [\sum_{i\in[m]} y_i \cdot \alpha\gamma\beta_i]_1$ and thus

$$e\left(C_y,\, [(\alpha\gamma)^n]_2\right) = \left[\sum_{i\in[m]} y_i \cdot (\alpha\gamma)^{n+1}\beta_i\right]_T .$$

Finally, by plugging the definition of $\hat{\pi}$ in Open we obtain that

$$e\left(\hat{\pi}, [1]_2\right) \cdot e\left(C_y,\, [(\alpha\gamma)^n]_2\right) = [p_{\boldsymbol{w}}(\alpha) \cdot \phi_z]_T$$

5.3 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 [Wat11]. In Appendix C we justify (n, m)-DP-BDHE in the generic group model.

Definition 12 ((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*

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

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

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 3. *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 $\mathcal{A}(\mathsf{ck})$.

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

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

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

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 $\sum_{k \in [m]} u_k \cdot y_k = 1$. □

5.4 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 [KW93].

Definition 13 (Monotone Span Programs [KW93]). *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., [LOS+10, CGW15, CGKW18], 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 11. 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 4. *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.

5.5 Extension to hiding and zero-knowledge

We discuss how the FC scheme for sQAPs can be extended so that the commitment to $\boldsymbol{z}$ becomes com-hiding and the openings zero-knowledge. Namely, we do not make C_y hiding, and note that this is sufficient for our application to MSPs in which the $\boldsymbol{y}$ vector is a known constant. The idea to make C_z hiding is similar to that for the FC for polynomials (Section 4.5).

First, we instantiate the scheme for longer vectors of length $n + 1$ and we commit to $\tilde{\boldsymbol{z}} = (r, \boldsymbol{z})$ where $r \leftarrow\!\!\$\, \mathbb{F}$. This makes C_z distributed like a uniformly chosen group element and so hiding trivially follows. For equivocation we can set the $\mathsf{Sim}_{\mathsf{Com}}$ algorithm to be the one that generates a commitment to $\mathbf{0}$, i.e., by what we just described, a commitment to $(r, \mathbf{0})$ using a random $r \leftarrow\!\!\$\, \mathbb{F}$,

and $\mathsf{Sim}_{\mathsf{Equiv}}$ the algorithm that solves a linear equation to find r' such that $r\gamma = p_{(r',\boldsymbol{z})}(\gamma)$, that is $r' = p_{(r,\boldsymbol{z})}(\gamma)/\gamma$.

Second, in order to maintain correctness, on input a matrix $\mathbf{F}$ the opening and verification algorithms internally define and run with a matrix $\tilde{\mathbf{F}} = (\mathbf{0} \mid \mathbf{F})$. This way the linear system remains functionally equivalent as r is ignored.

To make the opening proofs zero-knowledge we change the Open algorithm by committing to the vector $\tilde{\boldsymbol{w}} = (s, \boldsymbol{w})$ for a random $s \leftarrow\!\$\, \mathbb{F}$. This way the proof element W is uniformly distributed. Again, since we work with $\tilde{\mathbf{F}}$ the correctness remains preserved.

Finally, to see that this variant of our FC satisfies zero-knowledge openings we sketch how to simulate proofs. The simulator is supposed to know $\alpha, \gamma, \boldsymbol{\beta}$ as the trapdoor. The element $\boldsymbol{\Phi}_{\boldsymbol{z}}$ can be computed as in the Open algorithm but using an opening $\tilde{\boldsymbol{z}} = (r, \mathbf{0})$. The element W can be simulated as $[\omega]_1$ for a random $\omega \leftarrow\!\$\, \mathbb{F}$, and finally the proof $\hat{\pi}$ can be (perfectly) simulated as

$$\hat{\pi} := \left[\omega \cdot \sum_{i\in[m],j,\ell\in[n+1]} \tilde{F}_{i,j} \cdot \tilde{z}_\ell \cdot \alpha^{n+2-j} \beta_i \gamma^{n+2-j+\ell} - \sum_{i=1}^{m} y_i \cdot (\alpha\gamma)^{n+2} \beta_i \right]_1$$

6 Homomorphic signatures from additive-homomorphic functional commitments

Here we show an application of FCs to homomorphic signatures.

6.1 Homomorphic Signatures

We recall the definition of homomorphic signatures (HS) of [BF11], extended to work with labeled programs [GW13], as used in several prior works, e.g., [CFW14, CFN18].

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 [GW13]. 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 14 (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.* $|\sigma| \leq p(\lambda) \cdot \log n \cdot \log m$*).*

Remark 2 (One-hop vs. multi-hop HS). If evaluation correctness holds only for signatures $\{\sigma_i\}$ generated by Sign then HS is called *one-hop*. Otherwise, if evaluation correctness holds for signatures σ_i obtained after $k-1$ sequential executions of Eval on signature generated by Sign, then HS is called k-hop. An HS that is k-hop for any $k = \mathsf{poly}(\lambda)$ is called *multi-hop*.

Remark 3 (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

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

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

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

[CFN18] for a discussion on different notions of unforgeability. In this work we adopt the simplest and strongest notion from [CFN18], called strong-adaptive security.

Definition 15 (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* $\mathbf{Adv}^{\mathsf{strong\text{-}Ad\text{-}UF}}_{\mathcal{A},\mathsf{HS}}(\lambda) \leq \epsilon(\lambda)$*.*

Context Hiding HS can satisfy a privacy property called *context-hiding* which, informally speaking, guarantees that signatures on outputs do not reveal information about the signed inputs. In our work we use a statistical version of the notion proposed by Boneh and Freeman [BF11] called *weak context hiding.*[12]

Definition 16 (Weak Context-Hiding [BF11]). *An HS scheme is weakly context-hiding if there exists a simulator* Sim *such that for any keys* $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KeyGen}(1^\lambda, \mathcal{L})$*, data-set* Δ*, labels* $\tau_1, \ldots, \tau_n \in \mathcal{L}$*, messages* $x_{\tau_1}, \ldots, x_{\tau_n} \in \mathcal{X}$*, signatures* $\{\sigma_{\tau_i} \leftarrow \mathsf{Sign}(\mathsf{sk}, \Delta, \tau_i, x_{\tau_i})\}_{i=1}^n$*, and any tuple of labeled programs* $\{\mathcal{P}_j = (f_j, \tau_{j,1}, \ldots, \tau_{j,\ell_j})\}_{j=1}^Q$*, we have, for* $y_j = f_j(x_{\tau_{j,1}}, \ldots, x_{\tau_{j,\ell_j}})$*,*

$$\{\mathsf{Eval}(\mathsf{pk}, f_j, \sigma_{\tau_1}, \ldots, \sigma_{\tau_\ell})\}_{j=1}^Q \approx_s \mathsf{Sim}(\mathsf{sk}, \mathsf{pk}, \Delta, \{\mathcal{P}_j, y_j\}_{j=1}^Q)$$

where the probabilities are over the randomness of Eval *and* Sim*.*

Efficient Verification An HS scheme has efficient verification [BFR13, CFW14] if its verification can be split in two algorithms: an *offline* algorithm VerPrep that on input the public key pk and a labeled program $\mathcal{P}$ precomputes a concise verification key $\mathsf{vk}_\mathcal{P}$; an *online* algorithm EffVer that uses $\mathsf{vk}_\mathcal{P}$ to verify signatures w.r.t. $\mathcal{P}$ and *any* dataset Δ in time faster than that of Ver. More formally:

Definition 17 (Efficient Verification). *An homomorphic signature scheme* $\mathsf{HS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{Eval})$ *has* efficient verification *if there are two additional algorithms* $\mathsf{vk}_\mathcal{P} \leftarrow \mathsf{VerPrep}(\mathsf{pk}, \mathcal{P})$ *and* $b \leftarrow \mathsf{EffVer}(\mathsf{vk}_\mathcal{P}, \Delta, y, \sigma)$ *such that*

[12] The term "weak" refers to the fact that this is weaker than a notion by Ahn et al. [ABC+12] in which evaluated signatures should be indistinguishable from fresh signatures. This is not realizable in our case where signatures on outputs may have a different form than signatures on inputs.

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

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

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

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

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.

- *Correctness: For any keys* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathcal{L})$ *and any tuple* $(\mathcal{P}, \Delta, y, \sigma)$*, if* $\mathsf{vk}_{\mathcal{P}} \leftarrow \mathsf{VerPrep}(\mathsf{pk}, \mathcal{P})$*, then* $\mathsf{EffVer}(\mathsf{vk}_{\mathcal{P}}, \Delta, y, \sigma) = \mathsf{Ver}(\mathsf{vk}, \mathcal{P}, \Delta, y, \sigma)$ *with overwhelming probability.*
- *Amortized Efficiency: Let* $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ *be a labeled program and* Δ *be a dataset identifier, let* $(x_1, \ldots, x_n) \in \mathcal{X}^n$ *be any vector of inputs. If* $\mathsf{vk}_{\mathcal{P}} \leftarrow \mathsf{VerPrep}(\mathsf{pk}, \mathcal{P})$*, then* $\mathsf{EffVer}(\mathsf{vk}_{\mathcal{P}}, \Delta, x, \sigma)$ *runs in time* $\mathsf{poly}(\lambda, \log(n))$

6.2 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 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.

Theorem 5. *If* LHS *is strongly-adaptive secure and* FC *is weak evaluation binding, then* HS *is strongly-adaptive secure.*

Proof. The idea of the proof is that to break the security of HS an adversary must either return a commitment C which is a forgery under the FC.Add function for the LHS scheme, or, when the commitment is correct, generate an opening proof for a false result.

$\mathsf{Game}_0(\lambda)$**:** this is the same as the real experiment $\mathbf{Exp}^{\mathsf{strong\text{-}Ad\text{-}UF}}_{\mathcal{A},\mathsf{HS}}(\lambda)$.

$\mathsf{Game}_1(\lambda)$**:** this game proceeds as the previous one except that it outputs 0 if the following condition occurs. Let $(f^*, (i_1^*, \ldots, i_t^*), \Delta^*, \sigma^*, \boldsymbol{y}^*)$ be the forgery returned by $\mathcal{A}$, parse $\sigma^* = (\hat{\sigma}^*, C^*, \pi^*)$ and assume this is accepted by Ver. If this is a type-1 forgery (i.e., $\exists j \in [t]$ such that the adversary

never queried a tuple $(\Delta^*, i_j, \cdot)$) the game outputs 0. Otherwise, assuming that the adversary has queried all the labels $i_1^*, \ldots, i_t^*$ for the dataset Δ^*, let $C_1, \ldots, C_t$ be the commitments generated upon each of these queries. If $C^* \neq \mathsf{FC.Add}(\mathsf{ck}, C_1, \ldots, C_t)$, then the game outputs 0 as well.

To prove the theorem we first show that Game_0 and Game_1 are indistinguishable under the strongly adaptive unforgeability of LHS; next, we show that any adversary winning in Game_1 can be reduced to an adversary breaking the weak evaluation binding of FC.

Lemma 1. *For any PPT $\mathcal{A}$ there exists a PPT $\mathcal{B}$ such that*

$$|\Pr[\mathsf{Game}_0 = 1] - \Pr[\mathsf{Game}_1 = 1]| \leq \mathbf{Adv}^{\mathsf{strong\text{-}Ad\text{-}UF}}_{\mathcal{B},\mathsf{LHS}}(\lambda).$$

Proof. As one can see, the event by which Game_1 differs from Game_0 is the event that the forgery returned by $\mathcal{A}$ implies a forgery for the scheme LHS. The reduction is straightforward and is omitted.

Lemma 2. *For any PPT $\mathcal{A}$ there exists a PPT $\mathcal{B}$ such that*

$$\Pr[\mathsf{Game}_1 = 1] \leq \mathbf{Adv}^{\mathsf{wEvBind}}_{\mathcal{B},\mathsf{FC}}(\lambda).$$

Proof. The adversary $\mathcal{B}$ on input the FC commitment's key ck, generates the LHS keys $(\mathsf{sk}_{\mathsf{LHS}}, \mathsf{pk}_{\mathsf{LHS}}) \leftarrow \mathsf{LHS.KeyGen}(1^\lambda)$, sets $\mathsf{vk} := (\mathsf{pk}_{\mathsf{LHS}}, \mathsf{ck})$ and runs $\mathcal{A}(\mathsf{vk})$. The simulation of signing queries is straightforward since $\mathcal{B}$ has the secret key $\mathsf{sk}_{\mathsf{LHS}}$.

Consider $\mathcal{A}$'s output $(f^*, (i_1^*, \ldots, i_t^*), \Delta^*, \sigma^*, y^*)$. Given the changes introduced in Game_1, a forgery returned by $\mathcal{A}$ in this game must be a type-2 forgery. Also, the commitment C^* is the same as $\mathsf{FC.Add}(\mathsf{ck}, C_1, \ldots, C_t)$, where each C_j is the commitment generated by $\mathcal{B}$ at the signing query (Δ^*, i_j, x_j). If $\sigma^* = (\hat{\sigma}, C^*, \mathsf{aux}^*, i^*)$, i.e., it is a non-evaluated signature, then $\mathcal{B}$ computes $\pi^* \leftarrow \mathsf{FC.Open}(\mathsf{ck}, \mathsf{aux}^*)$. The rest of the reduction proceeds identically to the case of an evaluated signature as described below (considering that in such a case $t = 1$ and the program must be the identity function).

By definition of type-2 forgery it holds $\boldsymbol{y}^* \neq f^*(x_1, \ldots, x_t) = \hat{f}^*_{\boldsymbol{i}^*}(\hat{x}_1, \ldots, \hat{x}_n)$ as well as $\mathsf{FC.Ver}(\mathsf{ck}, C^*, \hat{f}^*_{\boldsymbol{i}^*}, y^*, \pi^*,) = 1$, where for every $k \in [n]$: $\hat{x}_k = x_j$ if $k = i_j^*$ and 0 everywhere else. Therefore, $\mathcal{B}$ computes $r \leftarrow \mathsf{FC.Add}_r(\mathsf{ck}, r_1, \ldots, r_t)$, where r_j is the randomness used to generate the commitment C_j at the signing query (Δ^*, i_j, x_j), and then it returns $(\hat{\boldsymbol{x}}, r, \hat{f}^*_{\boldsymbol{i}^*}, \boldsymbol{y}^*, \pi^*)$. It is easy to see that if $\mathcal{A}$ makes Game_1 return 1, then $\mathcal{B}$'s output is a successful break of weak evaluation binding.

6.3 Context-Hiding

Theorem 6. *If FC is com-hiding and zero-knowledge, then HS is weakly context hiding.*

Proof. To prove the theorem we consider a small variant of our scheme in which the commitment key ck is generated by using the Sim_1 algorithm along with a trapdoor td, which is included in sk (although not used for signing).[13]

The context hiding simulator, on input $\mathsf{sk} := (\mathsf{sk}_{\mathsf{LHS}}, \mathsf{td})$ works as follows.

[13] We could have changed the description of HS with this change but we prefer to keep it simpler. Another way to avoid this change is to assume a context hiding notion of HS in which (indistinguishable) keys are generated by the simulator.

$\mathsf{HS.Sim}(\mathsf{sk}, \Delta, \{(f_j, \boldsymbol{i}_j), \boldsymbol{y}_j\}_{j=1}^{Q})$

for $i = 1, \ldots, n$: $(C_i, \mathsf{aux}_i) \leftarrow \mathsf{Sim}_{\mathsf{Com}}(\mathsf{ck})$; $\hat{\sigma}_i \leftarrow \mathsf{LHS.Sign}(\mathsf{sk}_{\mathsf{LHS}}, \Delta, i, C_i)$

for $j = 1, \ldots, Q$:

$C_j \leftarrow \mathsf{FC.Add}(\mathsf{ck}, C_{i_{j,1}}, \ldots, C_{i_{j,\ell_j}})$

$\hat{\sigma}_j \leftarrow \mathsf{LHS.Eval}(\mathsf{pk}, \mathsf{FC.Add}, \hat{\sigma}_{i_{j,1}}, \ldots, \hat{\sigma}_{i_{j,\ell_j}})$

$\pi_j \leftarrow \mathsf{FC.Sim}_2(\mathsf{td}, C_j, \hat{f}_{\boldsymbol{i}_j}, \boldsymbol{y}_j)$

$\sigma_j := (\hat{\sigma}_j, C_j, \pi_j)$

return $(\sigma_1, \ldots, \sigma_Q)$

To prove context hiding we can define the following hybrid simulator $\mathsf{HS.Sim}$ that takes additionally as input the vector $\boldsymbol{x}$ and generates signatures correctly:

$\mathsf{HS.Sim}'(\mathsf{sk}, \Delta, \boldsymbol{x}, \{(f_j, \boldsymbol{i}_j), \boldsymbol{y}_j\}_{j=1}^{Q})$

for $i = 1, \ldots, n$: $(C_i, \widetilde{\mathsf{aux}}_i) \leftarrow \mathsf{Sim}_{\mathsf{Com}}(\mathsf{ck})$; $\hat{\sigma}_i \leftarrow \mathsf{LHS.Sign}(\mathsf{sk}_{\mathsf{LHS}}, \Delta, i, C_i)$

$\mathsf{aux}_i \leftarrow \mathsf{Sim}_{\mathsf{Equiv}}(\mathsf{td}, C_i, \widetilde{\mathsf{aux}}_i, x_i \boldsymbol{e}_i)$**endfor**

for $j = 1, \ldots, Q$:

$C_j \leftarrow \mathsf{FC.Add}(\mathsf{ck}, C_{i_{j,1}}, \ldots, C_{i_{j,\ell_j}})$

$\mathsf{aux}_j \leftarrow \mathsf{FC.Add}_{\mathsf{aux}}(\mathsf{ck}, \mathsf{aux}_{i_{j,1}}, \ldots, \mathsf{aux}_{i_{j,\ell_j}})$

$\hat{\sigma}_j \leftarrow \mathsf{LHS.Eval}(\mathsf{pk}, \mathsf{FC.Add}, \hat{\sigma}_{i_{j,1}}, \ldots, \hat{\sigma}_{i_{j,\ell_j}})$

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

$\sigma_j := (\hat{\sigma}_j, C_j, \pi_j)$

return $(\sigma_1, \ldots, \sigma_Q)$

It is easy to see that by the zero-knowledge of FC the output distributions of $\mathsf{HS.Sim}$ and $\mathsf{HS.Sim}'$ are indistinguishable.

Next, consider the real experiment in which each σ_j is generated via our $\mathsf{HS.Eval}$ algorithm. The only difference with the signatures generated by $\mathsf{HS.Sim}'$ is that in the former the commitments are directly generated as commitments to $x_i \boldsymbol{e}_i$ whereas in the latter they are generated by using $\mathsf{Sim}_{\mathsf{Com}}$ and later equivocated. This difference can be reduced to the com-hiding of the FC scheme.

6.4 Efficient Verification

It is easy to see that if both the homomorphic signature LHS and the functional commitment FC enjoy amortized efficient verification, then so does the resulting HS scheme. We show below how to construct $\mathsf{VerPrep}$ and EffVer algorithms for HS from the corresponding algorithms of LHS and FC. Their correctness follows from the efficient verification correctness properties of LHS and FC

$\mathsf{HS.VerPrep}(\mathsf{pk}, \mathcal{P} := (f, \boldsymbol{i}))$

$\mathsf{pk}_{\mathsf{LHS},\mathcal{P}} \leftarrow \mathsf{LHS.VerPrep}(\mathsf{pk}_{\mathsf{LHS}}, (\mathsf{FC.Add}, \boldsymbol{i}))$

$\mathsf{vk}_f \leftarrow \mathsf{FC.VerPrep}(\mathsf{ck}, \hat{f}_{\boldsymbol{i}})$

return $\mathsf{vk}_{\mathcal{P}} := (\mathsf{ck}, \mathsf{pk}_{\mathsf{LHS},\mathcal{P}}, \mathsf{vk}_f)$

$\mathsf{EffVer}(\mathsf{vk}_f, \Delta, \boldsymbol{y}, \sigma)$

$b_{\mathsf{LHS}} \leftarrow \mathsf{LHS.Ver}(\mathsf{pk}_{\mathsf{LHS},\mathcal{P}}, \Delta, C, \hat{\sigma})$

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

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

$b_{\mathsf{FC}} \leftarrow \mathsf{FC.EffVer}(\mathsf{vk}_f, C, \boldsymbol{y}, \pi)$

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

6.5 Instantiating LHS

Here we explain how to instantiate the linearly-homomorphic structure signature (LHS) scheme required by our construction of HS based on FCs, in the case when we instantiate the FC scheme using the pairing-based FCs proposed in our paper. Recall that the desiderata here is a scheme to sign the group elements comprised in the commitments of the FC schemes and that also provides efficient verification and strong adaptive security.

The simplest scheme satisfying all these requirements is the LHS from [CMP14], that we discuss here for completeness. Informally, Catalano *et al.* in [CMP14] present a very simple linearly homomorphic scheme to sign messages m that are element of some bilinear group $\mathbb{G}$. The scheme builds on the randomness-reuse technique originally introduced by Attrapadung and Libert [AL11] in the context of homomorphic network coding signatures: for each dataset Δ, the signer signs some random commitment g^r, via a standard signature scheme and then uses the randomness r as a key to sign all the messages from Δ. In particular, [CMP14] build their scheme in two steps: first they give a construction realizing random message security only (i.e. where the adversary has no control on the signed messages it receives). Next, they show a generic, yet efficient, transformation to achieve adaptive security. The original scheme however does not satisfy the strong adaptive security as defined in [CFN18] (in fact, this notion did not exist at the time). To address this, a possibility could be to resort to the efficient compiler relying on linearly homomorphic signatures from [CFN18]. In our setting there is a simpler and more efficient solution, though. Indeed, our transform from additive FC to homomorphic signatures requires the underlying LHS to guarantee strong adaptive security only when forgeries are restricted to very specific function families f^*. Specifically, letting $f^* = (\alpha_1^*, \ldots, \alpha_t^*)$, it has to be the case that $\alpha_i^* \in \{0, 1\}$ and, for all queried inputs τ_i it holds $\alpha_i \neq 0$. This is because, in our transformation the various commitments C_i are only combined via the FC.Add functionality and letting $\alpha_i^* = 0$ only means excluding the corresponding C_i from the computation. For the expert reader, this restriction on the classes of function implies that it can never occur that adversaries return forgeries that are of Type-3 according to the security notion proposed by Freeman [Fre12].

In Appendix E we described a slightly optimized variant of the construction from [CMP14] and prove that it achieves full adaptive security (under the restriction on the admissible functions discussed above).

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

ABC+12. Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 1–20. Springer, Heidelberg, March 2012.

AL11. Nuttapong Attrapadung and Benoît Libert. Homomorphic network coding signatures in the standard model. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 17–34. Springer, Heidelberg, March 2011.

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

BF11. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168. Springer, Heidelberg, May 2011.

BFKW09. Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 68–87. Springer, Heidelberg, March 2009.

BFR13. Michael Backes, Dario Fiore, and Raphael M. Reischuk. Verifiable delegation of computation on outsourced data. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 863–874. ACM Press, November 2013.

BGW05. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Heidelberg, August 2005.

Boy08. Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, September 2008.

CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Heidelberg, February / March 2013.

CFM08. Dario Catalano, Dario Fiore, and Mariagrazia Messina. Zero-knowledge sets with short proofs. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 433–450. Springer, Heidelberg, April 2008.

CFN15. Dario Catalano, Dario Fiore, and Luca Nizzardo. Programmable hash functions go private: Constructions and applications to (homomorphic) signatures with shorter public keys. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 254–274. Springer, Heidelberg, August 2015.

CFN18. Dario Catalano, Dario Fiore, and Luca Nizzardo. On the security notions for homomorphic signatures. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 183–201. Springer, Heidelberg, July 2018.

CFW12. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Efficient network coding signatures in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 680–696. Springer, Heidelberg, May 2012.

CFW14. Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 371–389. Springer, Heidelberg, August 2014.

CGKW18. Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded ABE via bilinear entropy expansion, revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 503–534. Springer, Heidelberg, April / May 2018.

CGW15. Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, April 2015.

CL02. Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Heidelberg, August 2002.

CMP14. Dario Catalano, Antonio Marcedone, and Orazio Puglisi. Authenticating computation on groups: New homomorphic primitives and applications. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 193–212. Springer, Heidelberg, December 2014.

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

Fre12. David Mandell Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 697–714. Springer, Heidelberg, May 2012.

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

GKKR10. Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 142–160. Springer, Heidelberg, May 2010.

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

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015.

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

GW13. Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 301–320. Springer, Heidelberg, December 2013.

KNYY19. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 622–651. Springer, Heidelberg, May 2019.

KW93. M. Karchmer and A. Wigderson. On span programs. In *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 102–111, 1993.

KW20. Sam Kim and David J. Wu. Multi-theorem preprocessing NIZKs from lattices. *Journal of Cryptology*, 33(3):619–702, July 2020.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.

LM19. Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Heidelberg, August 2019.

LOS+10. Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.

LP20. Helger Lipmaa and Kateryna Pavlyk. Succinct functional commitment for a large class of arithmetic circuits. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 686–716. Springer, Heidelberg, December 2020.

LPJY13. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Linearly homomorphic structure-preserving signatures and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 289–307. Springer, Heidelberg, August 2013.

LRY16. Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016.

LW11. Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.

LY10. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Heidelberg, February 2010.

PPS21. Chris Peikert, Zachary Pepin, and Chad Sharp. Vector and functional commitments from lattices. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 480–511. Springer, Heidelberg, November 2021.

Wat11. Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 53–70. Springer, Heidelberg, March 2011.

A More on FCs definitions

A.1 Evaluation binding implication

We prove the following simple proposition to show that evaluation binding implies weak evaluation binding.

Proposition 1. *If an* FC *satisfies evaluation binding then it also satisfies weak evaluation binding.*

Proof. Proceeding by contradiction, we show an adversary $\mathcal{B}$ against evaluation binding from an adversary $\mathcal{A}$ against weak evaluation binding. $\mathcal{B}(\mathsf{ck})$ runs $(\boldsymbol{x}, r, f, \boldsymbol{y}, \pi) \leftarrow \mathcal{A}(\mathsf{ck})$, computes a commitment $(C, \mathsf{aux}) \leftarrow \mathsf{Com}(\mathsf{ck}, \boldsymbol{x}; r)$ and an honest evaluation opening $\hat{\pi} \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{aux}, f)$ for $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$. $\mathcal{B}$ outputs $(C, f, \boldsymbol{y}, \pi, \hat{\boldsymbol{y}}, \hat{\pi})$. If $\mathcal{A}$ is successful then $\boldsymbol{y} \neq f(\boldsymbol{x}) = \hat{\boldsymbol{y}}$ and $\mathsf{Ver}(\mathsf{ck}, C, f, \boldsymbol{y}, \pi) = 1$. At the same time, by correctness of the FC scheme we have that $\mathsf{Ver}(\mathsf{ck}, C, f, \hat{\boldsymbol{y}}, \hat{\pi}) = 1$. Hence $\mathcal{B}$ succeeds in breaking evaluation binding with the same success probability of $\mathcal{A}$.

A.2 From Linear Aggregation to Strong Evaluation Binding

Theorem 7. *Let* FC *be an FC scheme for linear forms over a finite field that satisfies evaluation binding and linear aggregation. Then* FC *is strong evaluation binding.*

Proof. Consider an adversary $\mathcal{A}$ against strong evaluation binding that outputs $(C, \{\boldsymbol{f}_i, y_i, \pi_i\}_{i=1}^{Q})$, where $\boldsymbol{f}_i \in \mathcal{X}^n$ and $y_i \in \mathcal{X}$. Our goal is to build $\mathcal{B}$ against evaluation binding.

$\mathcal{B}$ runs $\mathcal{A}$ to obtain the output above. If $\mathcal{A}$ is successful each proof is valid, i.e., $\mathsf{Ver}(\mathsf{ck}, C, \boldsymbol{f}_i, y_i, \pi_i) = 1$, and $\nexists \boldsymbol{x} \in \mathcal{X}^n : \forall i \in [Q] : \boldsymbol{f}_i^\top \cdot \boldsymbol{x} = y_i$.

Define $\mathbf{F} \in \mathcal{X}^{Q \times n}$ and the vector $\boldsymbol{y}$ as the "concatenation" of these linear systems, i.e.,

$$\mathbf{F} := \begin{bmatrix} \boldsymbol{f}_1^\top \\ \vdots \\ \boldsymbol{f}_Q^\top \end{bmatrix}, \quad \boldsymbol{y} := \begin{pmatrix} y_1 \\ \vdots \\ y_Q \end{pmatrix}.$$

The winning condition of strong evaluation binding means that $(\mathbf{F} \mid \boldsymbol{y})$ is an inconsistent system of equations which means there exists an $\boldsymbol{\alpha} \in \mathcal{X}^Q$ such that $\boldsymbol{\alpha}^\top \cdot \mathbf{F} = \mathbf{0}^\top$ and $\boldsymbol{\alpha}^\top \cdot \boldsymbol{y} = \delta \neq 0$. Let $V := \{c \cdot \boldsymbol{\alpha} : c \in \mathcal{X}\}$. Then any vector $\boldsymbol{v} \in V$ is such that

$$\boldsymbol{v}^\top \cdot \mathbf{F} = \mathbf{0}^\top \wedge \boldsymbol{v}^\top \cdot \boldsymbol{y} \neq 0$$

In particular, one of them, $\boldsymbol{u} = \delta^{-1} \cdot \boldsymbol{\alpha}$ is such that $\boldsymbol{u}^\top \cdot \boldsymbol{y} = 1$. Then, $\mathcal{B}$ proceeds by finding $\boldsymbol{u}$ such that

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

and builds an attack against evaluation binding as follows.

First it defines the function $\boldsymbol{f}^* = u_1 \cdot \boldsymbol{f}_1$. Since $\boldsymbol{u}^\top \cdot \mathbf{F} = \sum_{j=1}^{Q} u_i \cdot \boldsymbol{f}_i^\top = \mathbf{0}^\top$, we have $\boldsymbol{f}^* = -\sum_{j=2}^{Q} u_j \cdot \boldsymbol{f}_j^\top$.

Second, it defines the outputs $y^* = u_1 \cdot y_1$ and $y = -\sum_{j=2}^{Q} u_j \cdot y_j$. Notice that $y - y' = \sum_{j=1}^{Q} u_j \cdot y_j = \boldsymbol{u}^\top \cdot \boldsymbol{y} = 1$ and thus $y \neq y'$.

Third, it builds the two openings by computing[14] $\pi \leftarrow \mathsf{Agg}(\mathsf{ck}, C, (\pi_1, \boldsymbol{f}_1, y_1), (u_1))$ and $\pi' \leftarrow \mathsf{Agg}(\mathsf{ck}, C, (\pi_2, \boldsymbol{f}_2, y_2), \ldots, (\pi_Q, \boldsymbol{f}_Q, y_Q), (u_2, \ldots, u_Q))$.

By the linear aggregation property we have that

$$\mathsf{Ver}(\mathsf{ck}, C, f^*, y, \pi) = \mathsf{Ver}(\mathsf{ck}, C, f^*, y', \pi') = 1$$

and thus $\mathcal{B}$'s output is a valid attack against evaluation binding. □

A.3 From Strong Evaluation Binding to SNARGs

Here we give more details about the observation that a strong evaluation binding FC with succinct proofs for a language $\mathcal{L}$ yields a SNARG for $\mathcal{L}$.

Let R be the NP relation associated to $\mathcal{L}$ such that a statement $x \in \mathcal{L}$ iff $\exists \boldsymbol{w} : R(x, \boldsymbol{w}) = \boldsymbol{y}$ for some known $\boldsymbol{y}$.[15] If FC is an FC scheme for a class of functions that include the computation of $R(x, \cdot)$ for any x, then we can build a SNARG Π for the language $\mathcal{L}$ as follows.

$\Pi.\mathsf{Setup}$ runs $\mathsf{ck} \leftarrow \mathsf{FC.Setup}$ and returns $crs = \mathsf{ck}$; $\Pi.\mathsf{P}(crs, x, w)$ computes $(C, \mathsf{aux}) \leftarrow \mathsf{FC.Com}(\mathsf{ck}, w)$, $\pi_f \leftarrow \mathsf{FC.Open}(\mathsf{ck}, \mathsf{aux}, f)$ where f is the function such that $f(\cdot) = R(x, \cdot)$, and returns $\pi = (C, \pi_f)$; $\Pi.\mathsf{Ver}(crs, x, \pi)$ returns $\mathsf{FC.Ver}(\mathsf{ck}, C, \pi_f, f, \boldsymbol{y})$.

The reduction from the soundness of Π to the strong evaluation binding of the FC works as follows. Let $\mathcal{A}$ be a SNARG adversary that returns $(x, \pi = (C, \pi_f))$ such that $x \notin \mathcal{L}$ and $\Pi.\mathsf{Ver}(crs, x, \pi) = \mathsf{FC.Ver}(ck, C, \pi_f, f, \boldsymbol{y}) = 1$. Since $x \notin \mathcal{L}$ there is no $\boldsymbol{w}$ such that $R(x, \boldsymbol{w}) = \boldsymbol{y}$, and thus the reduction can return $(C, f, \pi_f, \boldsymbol{y})$ as a break for the strong evaluation binding of FC.

Note that the resulting SNARG is adaptively sound and the language does not need to be honestly generated (unless this is required by the FC scheme, but for example this is not needed in the scheme for linear maps). Furthermore, the compactness of FC implies the succinctness of Π. Note that, when the NP relation has a constant-size output, the SNARG is succinct even if one starts from a succinct, not necessarily compact, FC.

A.4 Further applications of additive-homomorphic FCs

Here we discuss further applications that follow immediately from any functional commitment that is additively homomorphic.

More in general, we argue that having homomorphic properties in cryptographic primitives is notoriously useful to obtain primitives with more complex functionalities. Additive-homomorphic commitments have plenty of applications for example. For this reason, we believe that additive-homomorphic functional commitments may have even more applications than the ones we mention below.

[14] We denote the description of the linear function used in aggregation as a vector of coefficients.

[15] Usually, $\boldsymbol{y}$ is a bit but here we are considering a more general case in which $\boldsymbol{y}$ could be a constant value whose length could depend on $|\boldsymbol{w}|$, e.g., think of the vector of 0's if R is an R1CS checking that every gate of a circuit is correctly computed.

Updatable FCs. The first one is *updatability*, in a sense similar to the updatable VC notion of [CF13]). An FC is updatable if, given a tuple (C, i, x_i, x'_i) where C is a commitment to a vector $\boldsymbol{x}$, and x_i (resp. x'_i) is the old (resp. new) value at position i, there is an efficient algorithm that computes a commitment C' to the vector $\boldsymbol{x}'$ which is the same as $\boldsymbol{x}$ except for having x'_i at position i, i.e., $\forall j \neq i : x'_j = x_j$.

In an additive-homomorphic FC, the update algorithm can be obtained by generating a commitment C_i to the sparse vector $(0, \ldots 0, x'_i - x_i, 0, \ldots 0)$ and by homomorphically adding C_i to C.

Notice that this updatability approach does *not* work for FCs obtained through the linearization trick, i.e., if using an additive-homomorphic FC for linear functions to commit to all the monomials. In such a construction, when updating an entry x_i one would have to update all the vector entries with monomials involving x_i, and this would require the knowledge of the full vector.

Verifiable Databases with expressive queries (VDB) An application where updatable FCs are useful is an extension of the verifiable databases application shown in [CF13]. In VDB, a client outsources a large database consisting of key-value pairs (i, x_i) to an untrusted server, and later the client can retrieve records (i.e., obtain x_i) and update them (i.e., change x_i into x'_i) in a verifiable way. In brief, in the VDB construction from vector commitments the client stores a commitment to $(x_1, \ldots, x_n)$; the server proves the correctness of the i-th record by providing a VC opening for position i; to update the i-th record the client (who no longer stores the full database) uses the updatable property to update its local commitment in such a way that x_i is replaced with x'_i. We refer to [CF13] for the formal description of their VDB scheme.

By replacing updatable vector commitments with updatable functional commitments in this VDB construction, we obtain a VDB notion in which the client ask to the server queries more expressive than retrievals. In particular, the client can ask to the server to compute $f(\boldsymbol{x})$, for any f supported by the FC and be convinced of its correctness with a succinct proof. Also, the client can use the updatability of the FC in order to deal with updates, which is a desirable feature in outsourced storage applications.

B A stronger transformation from FC for homogeneous polynomials to FC for generic polynomials

It is interesting to note what happens if a malicious party creates a commitment which, instead of having 0 in the first position of the committed vector, it includes some other value $\delta \neq 0$. According to the notion of weak evaluation binding (Def. 3), this attack is not deemed valid as the adversary must open the commitment; hence it is forced to commit to the vector in the correct way. Curiously, although this attack is not explicitly caught by the evaluation binding experiment (Def. 2), the evaluation binding of the transformed scheme still holds if the commitment is malformed in this way. This is due to the fact that evaluation binding only asks the adversary to produce two inconsistent openings for the same function. However, if we consider strong evaluation binding, it is not hard to see that by setting the first entry of the vector to a value $\delta \neq 0$ then the FC of this transformation does not satisfy this property.

Nevertheless, we can strengthen our transformation of Section 4.6 by letting the prover add an opening to the first position to show that indeed $x_0 = 1$. Slightly more in detail, given FC' we define FC^* whose Setup and Com algorithms are the same as those of the scheme FC described earlier, while the opening and verification are as follows.

$\mathsf{FC}^*.\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, \boldsymbol{f})$ It works the same as $\mathsf{FC}.\mathsf{Open}$ except for the following step.

– Compute a proof that 1 is in the first position. Let $\hat{f}_0$ be the degree-1 (homogeneous) polynomial in $n+1$ variables that returns x_0, i.e., $\hat{f}_0(x_0, \ldots, x_n) := x_0$ and compute $\pi_1 \leftarrow \mathsf{FC}'.\mathsf{Open}(\mathsf{ck}, \hat{\mathsf{aux}}, \hat{f}_0)$.

Return $\pi := (\hat{\pi}, \pi_1)$.

$\mathsf{FC}^*.\mathsf{Ver}(\mathsf{ck}, C, \pi, \boldsymbol{f}, \boldsymbol{y})$ Define the homogeneous polynomial $\hat{f}$ from f as above. Compute $(C_1, \mathsf{aux}_1) \leftarrow \mathsf{Com}(\mathsf{ck}, (1, 0, \ldots, 0); \emptyset)$ and $\hat{C} \leftarrow \mathsf{FC}'.\mathsf{Add}(\mathsf{ck}, C, C_1)$. Output

$$\mathsf{FC}'.\mathsf{Ver}(\mathsf{ck}, \hat{C}, \hat{\pi}, \hat{\boldsymbol{f}}, \boldsymbol{y}) \wedge \mathsf{FC}'.\mathsf{Ver}(\mathsf{ck}, \hat{C}, \pi_1, \hat{f}_0, 1)$$

Theorem 8. *If* FC' *satisfies strong evaluation binding, so does* FC^*.

Proof. Consider an adversary $\mathcal{A}^*$ against the strong evaluation binding of FC^* who returns $(C, \{\boldsymbol{f}_\ell, \boldsymbol{y}_\ell, \pi_\ell\}_{\ell=1}^{Q})$ such that the proofs are all valid and $\nexists \boldsymbol{x} \in \mathcal{X}^n : \forall \ell \in [Q] : f_{\ell,i}(\boldsymbol{x}) = y_{\ell,i}$. We can build an adversary $\mathcal{B}$ who returns $(C, \{\hat{\boldsymbol{f}}_\ell, \boldsymbol{y}_\ell, \hat{\pi}_\ell\}_{i=0}^{Q})$ where $y_0 = 1$ and $\hat{\pi}_0 = \pi_{1,1}$ (i.e. any of the proofs that the first variable is 1 – recall that each proof $\pi_\ell = (\hat{\pi}_\ell, \pi_{\ell,1})$). Then, since the adversary's output is such that $\nexists \boldsymbol{x} \in \mathcal{X}^n : \forall \ell \in [Q] : \boldsymbol{f}_\ell(\boldsymbol{x}) = \boldsymbol{y}_\ell$ and by the construction of $\hat{\boldsymbol{f}}_\ell$ from $\boldsymbol{f}_\ell$, we obtain that for $\mathcal{B}$'s output it holds $\nexists (x_0, x_1, \ldots, x_n) \in \mathcal{X}^{n+1} : x_0 = 1 \wedge \forall \ell \in [Q] : \hat{\boldsymbol{f}}_\ell(x_0, x_1, \ldots, x_n) = \boldsymbol{y}_\ell$. □

C Analysis of the *DP-BDHE* assumption in the generic bilinear group model

Lemma 3. *For* $n, m = \mathsf{poly}(\lambda)$ *the* (n, m)*-DP-BDHE assumption holds in the generic bilinear group model.*

Proof. To justify that the (n, m)-DP-BDHE assumption holds in the generic bilinear group model we observe that it is an instance of the (computational) "Uber assumption" with Laurent polynomials of [BBG05, Boy08]. In particular, it is a special case in which all the polynomials given as input to the adversary consist of monomials. In this case, it is sufficient to show that the challenge monomial to be computed by the adversary in the target group, that is $[(\alpha\gamma)^{n+1}\delta]_T$, is not symbolically equivalent to any of the monomials that one can obtain by taking the product of all the terms of Ω given in $\mathbb{G}_1$ with those in $\mathbb{G}_2$.

For convenience, let us recall the adversary's input Ω.

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

Notice that since the target monomial contains the variable δ, we can only consider products that involve terms from the last row of Ω. Furthermore, since these elements in the third row are all in $\mathbb{G}_2$, we can only consider their product with elements of $\mathbb{G}_1$, namely terms in the first row of Ω. We also note that, among its inputs, the adversary also has $[1]_1, [2]_2$. However, it is clear that the challenge monomial cannot be obtained through a pairing with $[1]_1$ or $[1]_2$ since $[(\alpha\gamma)^{n+1}\delta]_T$ is not given in any of $\mathbb{G}_1$ or $\mathbb{G}_2$.

Let us begin by considering all the products of elements in $\left\{\left[\frac{\delta}{\beta_k}\right]_2\right\}_{k\in[m]}$ and the elements in the first row of Ω, which result in the following sets:

$$S_{1,1} := \left\{\left[\frac{\alpha^j\delta}{\beta_k}\right]_T, \left[\frac{\gamma^j\delta}{\beta_k}\right]_T\right\}_{j\in[n],k\in[m]},$$
$$S_{1,2} := \left\{\left[\frac{\alpha^j\beta_i\gamma^{n+1}\delta}{\beta_k}\right]_T\right\}_{\substack{i,k\in[m],j\in[2n]\\ j\neq n+1}}, S_{1,3} := \left\{\left[\frac{\alpha^j\beta_i\gamma^{\ell}\delta}{\beta_k}\right]_T\right\}_{\substack{i,k\in[m],j,\ell\in[2n]\\ \ell\neq n+1}}$$

Clearly, $[(\alpha\gamma)^{n+1}\delta]_T$ is not among the elements in $S_{1,1}$ since all of them contain a variable β_k^{-1}. For a similar reason, we can exclude those elements in $S_{1,2}\cup S_{1,3}$ where $i\neq k$. Let us then focus on the terms of $S_{1,2}\cup S_{1,3}$ where $i=k$, in which case the β_i variables cancel out. They are:

$$\left\{\left[\alpha^j\gamma^{n+1}\delta\right]_T\right\}_{j\in[2n]\setminus\{n+1\}}, \left\{\left[\alpha^j\gamma^{\ell}\delta\right]_T\right\}_{j,\ell\in[2n],\ell\neq n+1}$$

However, one can see that both sets do not include $[(\alpha\gamma)^{n+1}\delta]_T$.

Let us consider the products of elements in $\left\{\left[\frac{\alpha^j\beta_i\gamma^{n+1}\delta}{\beta_k}\right]_2\right\}_{\substack{j\in[n],i,k\in[m]\\ i\neq k}}$ and the elements in the first row of Ω, which result in the following sets:

$$S_{2,1} := \left\{\left[\frac{\alpha^{j+j'}\beta_i\gamma^{n+1}\delta}{\beta_k}\right]_T, \left[\frac{\alpha^{j}\beta_i\gamma^{j'+n+1}\delta}{\beta_k}\right]_T\right\}_{\substack{j,j'\in[n],i,k\in[m]\\ i\neq k}}$$
$$S_{2,2} := \left\{\left[\frac{\alpha^{j+j'}\beta_i\beta_{i'}\gamma^{2n+2}\delta}{\beta_k}\right]_T\right\}_{\substack{j\in[n],j'\in[2n],i,i',k\in[m]\\ i\neq k}},$$
$$S_{2,3} := \left\{\left[\frac{\alpha^{j+j'}\beta_i\beta_{i'}\gamma^{\ell+n+1}\delta}{\beta_k}\right]_T\right\}_{\substack{j\in[n],j',\ell\in[2n],i,i',k\in[m]\\ i\neq k,\ell\neq n+1}}$$

The monomial $[(\alpha\gamma)^{n+1}\delta]_T$ is not in $S_{2,1}$ since each of them contains the variables β_i/β_k for $i\neq k$; it is not in $S_{2,2}$ and $S_{2,3}$ as elements in these sets have the term γ^{2n+2} and $\gamma^{\ell+n+1}$, with $\ell\geq 1$, respectively.

Therefore, we consider the last set of products between the elements $\left\{\left[\frac{\alpha^j\beta_i\gamma^{\ell}\delta}{\beta_k}\right]_2\right\}_{\substack{i,k\in[m],j\in[n]\\ \ell\in[2n]\setminus\{n+1\}}}$ and the elements in the first row of Ω, which result in the following sets:

$$S_{3,1} := \left\{\left[\frac{\alpha^{j+j'}\beta_i\gamma^{\ell}\delta}{\beta_k}\right]_T,\right\}_{\substack{i,k\in[m],j,j'\in[n]\\ \ell\in[2n]\setminus\{n+1\}}}, S_{3,2} := \left\{\left[\frac{\alpha^{j}\beta_i\gamma^{\ell+j'}\delta}{\beta_k}\right]_T\right\}_{\substack{i,k\in[m],j,j'\in[n]\\ \ell\in[2n]\setminus\{n+1\}}},$$
$$S_{3,3} := \left\{\left[\frac{\alpha^{j+j'}\beta_i\beta_{i'}\gamma^{\ell+n+1}\delta}{\beta_k}\right]_T,\right\}_{\substack{i,i',k\in[m],j\in[n],j'\in[2n]\setminus\{n+1\}\\ \ell\in[2n]\setminus\{n+1\}}},$$
$$S_{3,4} := \left\{\left[\frac{\alpha^{j+j'}\beta_i\beta_{i'}\gamma^{\ell+\ell'}\delta}{\beta_k}\right]_T,\right\}_{\substack{i,i',k\in[m],j,j'\in[n]\\ \ell,\ell'\in[2n]\setminus\{n+1\}}}$$

Among the elements in $S_{3,1} \cup S_{3,2}$ let us consider the subsets with $i = k$ as the terms with $i \neq k$ contain the variables β_i/β_k and thus cannot include $[(\alpha\gamma)^{n+1}\delta]_T$. However, $[(\alpha\gamma)^{n+1}\delta]_T$ is not in the subset of $S_{3,1}$ with $i = k$ as none of the terms have γ^{n+1}; analogously, it is not in the subset of $S_{3,2}$ with $i = k$ as all the terms include α^j only for $j \leq n$. Finally, $[(\alpha\gamma)^{n+1}\delta]_T$ is not in $S_{3,3} \cup S_{3,4}$ as in these sets each term has at least a variable β_i.

D Strongly evaluation binding FC for linear maps from falsifiable assumptions

In this section we prove that the FC for linear maps by Lai and Malavolta [LM19] satisfies strong evaluation binding under a falsifiable assumption.

We recall the scheme in Fig. 3.

$\mathsf{Setup}(1^\lambda, n, m) \to \mathsf{ck}$

$\alpha \leftarrow\!\$\, \mathbb{F}, \boldsymbol{\beta} \leftarrow\!\$\, \mathbb{F}^m$

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

$\mathsf{Com}(\mathsf{ck}, \boldsymbol{x}) \to (C, \mathsf{aux})$

$C := \sum_{j=1}^{n} x_j \cdot [\alpha^j]_1$

$\mathsf{aux} := \boldsymbol{x}$

$\mathsf{Open}(\mathsf{ck}, \mathsf{aux}, \mathbf{F}) \to \pi_{\mathbf{F}}$

$\pi_{\mathbf{F}} := \sum_{i\in[m],j,k\in[n],j\neq k} F_{i,j} \cdot x_k \cdot [\alpha^{n+1-j+k}\beta_i]_1$

$\mathsf{Ver}(\mathsf{ck}, C, \pi_{\mathbf{F}}, \mathbf{F}, \boldsymbol{y})$

$e\left(C, \sum_{i\in[m],j\in[n]} F_{i,j} \cdot [\alpha^{n+1-j}\beta_i]_2\right) \stackrel{?}{=} e\left(\pi_{\mathbf{F}}, [1]_2\right) \cdot e\left(\sum_{i=1}^{m} y_i \cdot [\alpha\beta_i]_1, [\alpha^n]_2\right)$

Fig. 3: FC for linear maps $f : \mathbb{F}^n \to \mathbb{F}^m$ from [LM19].

Below we state the (m, n)-parallel bilinear Diffie-Hellman exponent assumption, which is essentially the computational version of the assumption defined by Waters in [Wat11], that was for $m = n$ and for symmetric bilinear groups.

Definition 18 ((n, m)-PBDHE 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)-PBDHE holds if for every $n, m = \mathsf{poly}(\lambda)$ and any PPT $\mathcal{A}$, the advantage $\mathbf{Adv}^{(n,m)\text{-}PBDHE}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A}(\mathsf{bgp}, \Omega) = [\alpha^{m+1}\delta]_T]$ is negligible, where*

$$\Omega := \begin{pmatrix} \{[\alpha^j]_1\}_{j\in[n]}, \{[\alpha^j\beta_i]_1\}_{i\in[m],j\in[2n]\setminus\{n+1\}}, \{[\beta_i \cdot \alpha^j]_2\}_{i\in[m],j\in[n]}, [\alpha^n]_2 \\ \{[\delta/\beta_k]_2\}_{k\in[m]}, \{[(\alpha^j\beta_i\delta)/\beta_k]_2\}_{j\in[n],i,k\in[m],i\neq k} \end{pmatrix}$$

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

We state and give the proof of the following theorem. This proof follows closely the security proof of our FC scheme for sQAPs (Theorem 3), with some simplifications due to the fact that here the language is only linear.

Theorem 9. *If the (n, m)-PBDHE assumption holds then the FC scheme of Figure 3 has strong evaluation binding.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the strong evaluation binding of the FC scheme. We use $\mathcal{A}$ to build a PPT adversary $\mathcal{B}$ against the (n,m)-PBDHE 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

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

and runs $\mathcal{A}(\mathsf{ck})$.

Let $\mathcal{A}$'s output be $(C, \{\mathbf{F}^{(\ell)}, \boldsymbol{y}^{(\ell)}, \pi_\ell\}_{i=1}^{Q})$, where for every ℓ, $\mathbf{F}^{(\ell)} \in \mathbb{F}^{n\times m}$ and $\boldsymbol{y}^{(\ell)} \in \mathbb{F}^n$. If $\mathcal{A}$ is successful we have that (i) each proof is valid, i.e.,

$$\forall \ell \in [Q] : e\left(C, \sum_{i\in[m],j\in[n]} F_{i,j}^{(\ell)} \cdot [\alpha^j\beta_i]_2\right) = e\left(\pi_\ell, [1]_2\right) \cdot e\left(\sum_{i=1}^{m} y_i^{(\ell)} \cdot [\alpha\beta_i]_1,\ [\alpha^n]_2\right)$$

$$\text{and (ii) } \nexists \boldsymbol{x} \in \mathbb{F}^n : \forall i \in [Q] : \mathbf{F}_i \cdot \boldsymbol{x} = \boldsymbol{y}_i.$$

Define $\mathbf{F} \in \mathbb{F}^{mQ\times n}$ and the vector $\boldsymbol{y} \in \mathbb{F}^{mQ}$ as the "concatenation" of these linear systems, i.e.,

$$\mathbf{F} := \begin{bmatrix} \mathbf{F}^{(1)} \\ \vdots \\ \mathbf{F}^{(Q)} \end{bmatrix}, \quad \boldsymbol{y} := \begin{pmatrix} \boldsymbol{y}^{(1)} \\ \vdots \\ \boldsymbol{y}^{(Q)} \end{pmatrix}.$$

The winning condition (ii) of strong evaluation binding means that $(\mathbf{F} \mid \boldsymbol{y})$ is an inconsistent system of equations which means there exists an $\boldsymbol{c} \in \mathbb{F}^{mQ}$ such that $\boldsymbol{c}^\top \cdot \mathbf{F} = \mathbf{0}^\top$ and $\boldsymbol{c}^\top \cdot \boldsymbol{y} = \delta \neq 0$. 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, $\boldsymbol{u} = \delta^{-1} \cdot \boldsymbol{c}$ is such that $\boldsymbol{u}^\top \cdot \boldsymbol{y} = 1$.

$\mathcal{B}$ finds $\boldsymbol{u}$ such that

$$\boldsymbol{u}^\top \cdot (\mathbf{F} \mid \boldsymbol{y}) = (0, \cdots, 0, 1) \tag{11}$$

(e.g., by Gaussian elimination) and then proceeds as follows.

Parse $\boldsymbol{u} = (\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(Q)})$ where each $\boldsymbol{u}^{(\ell)} = (u_1^{(\ell)}, \ldots, u_m^{(\ell)})^\top \in \mathbb{F}^m$.

For every $\ell \in [Q]$ and $k \in [m]$, $\mathcal{B}$ computes

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

and

$$\pi^*_{\ell,k} := \pi'_{\ell,k} \cdot e\left(C, \sum_{\substack{i\in[m]\setminus\{k\}\\ j\in[n]}} F^{(\ell)}_{i,j} \cdot \left[\frac{\alpha^{n+1-j}\beta_i\delta}{\beta_k}\right]_2\right)^{-1} \cdot \prod_{i\in[m]\setminus\{k\}} e\left([\alpha^n]_1, \left[\frac{\alpha\delta\beta_i}{\beta_k}\right]_2\right)^{y^{(\ell)}_i}$$

$$= e\left(C, \sum_{j\in[n]} F^{(\ell)}_{k,j} \cdot \left[\alpha^{n+1-j}\delta\right]_2\right) \cdot \left[-y^{(\ell)}_k \alpha^{n+1}\delta\right]_T$$

Finally, $\mathcal{B}$ computes and returns

$$\Delta^* = \prod_{\ell\in[Q], i\in[m]} (\pi^*_{\ell,k})^{-u^{(\ell)}_i}$$

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

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

$$\Delta^* = e\left(C, -\sum_{j\in[n]} \sum_{\ell\in Q, k\in[m]} u^{(\ell)}_k \cdot F^{(\ell)}_{k,j} \cdot \left[\alpha^{n+1-j}\delta\right]_2\right) \cdot \left[\alpha^{n+1}\delta \sum_{\ell\in[Q],k\in[m]} u^{(\ell)}_k \cdot y^{(\ell)}_k\right]_T$$

The equality $\Delta^* = [\alpha^{n+1}\delta]_T$ follows from the fact that, by equation (11), we have that for every $j \in [n]$, $\sum_{\ell\in Q,k\in[m]} u^{(\ell)}_k \cdot F^{(\ell)}_{k,j} = 0$ and that $\sum_{k\in[m]} u^{(\ell)}_k \cdot y^{(\ell)}_k = 1$. □

E An efficient instantiation of the LHS scheme

We stress that the scheme *does not* allow one to sign messages expressed as vectors of group elements (as most currently known linearly homomorphic structure preserving signature schemes do) but rather single group elements. This is not a problem in our context as the (few) group elements composing our commitments can be signed separately. For compatibility with our schemes we port the LHS of [CMP14] to the setting of asymmetric pairings (the original scheme is described for symmetric ones). Also, while the scheme described below works for messages in $\mathbb{G}_1$, it is straightforward to obtain the "dual" version of it for messages in $\mathbb{G}_2$.

$\mathsf{LHS.KeyGen}(1^\lambda, n)$ Run the $\mathsf{KeyGen}(1^\lambda)$ algorithm of a (standard) digital signature scheme, to obtain signing keys $(\mathsf{sk}, \mathsf{pk})$. Also run a (symmetric) bilinear group generator $\mathcal{BG}(1^\lambda)$ to get $\mathsf{bgp} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $\mathbb{G}_1$, $\mathbb{G}_1$, $\mathbb{G}_T$ are groups of prime order p and $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ are fixed generators. The algorithm proceeds by sampling $\omega_1, \omega_2 \in \mathbb{Z}^*_p$ and sets $W_1 = g_2^{\omega_1}$, $W_2 = g_2^{\omega_2}$. Also it picks random group elements (in $\mathbb{G}_1$)

$$h_1, \ldots, h_n; \quad k_1, \ldots, k_n$$

and sets $\mathsf{pk}_{\mathsf{LHS}} = (\mathsf{vk}, g_1, g_2, h_1, \ldots, h_n, k_1, \ldots, k_n, W_1, W_2)$ $\mathsf{sk}_{\mathsf{LHS}} = (\mathsf{sk}, \omega_1, \omega_2)$
The message space is $\mathbb{G}_1$.

$\mathsf{LHS.Sign}(\mathsf{sk}_{\mathsf{LHS}}, \Delta, \tau_i, m)$ First it (randomly) chooses m' in $\mathbb{G}_1$. The algorithm maintains a list T for previously returned dataset identifiers Δ, labels τ_i and related information.
If $\Delta \notin T$
– $r, s \leftarrow\$ \mathbb{Z}_p$, $\sigma_1 \leftarrow g_2^r, \sigma_2 \leftarrow g_2^s$
– $\gamma \leftarrow \mathsf{Sign}(\mathsf{sk}, \Delta, \sigma_1, \sigma_2)$
else retrieve associated $r, s, \sigma_1, \sigma_2, \gamma$ from memory.
Set

$$M' \leftarrow m'^{\omega_1}, M'' \leftarrow (m/m')^{\omega_2},\ V' \leftarrow (h_i M')^r, V'' \leftarrow (k_i M'')^s$$

Output $\pi \leftarrow (\sigma_1, \sigma_2, \gamma, V', V'', M', M'', m')$ as the signature for m with respect to the function $\boldsymbol{e}_i$ (where $\boldsymbol{e}_i$ is the i-th vector of the canonical basis of $\mathbb{Z}^n$).
$\mathsf{LHS.Eval}(\mathsf{pk}_{\mathsf{LHS}}, f, \pi_1, \ldots, \pi_t)$ if the γ_i are not all equal output $\perp$, else let $\gamma = \gamma_i$ and compute

$$V' \leftarrow \prod_{i=1}^{n} (V_i')^{\alpha_i}; V'' \leftarrow \prod_{i=1}^{n} (V_i'')^{\alpha_i}; M' \leftarrow \prod_{i=1}^{n} (M_i')^{\alpha_i}; M'' \leftarrow \prod_{i=1}^{n} (M_i'')^{\alpha_i}; m' \leftarrow \prod_{i=1}^{n} (m_i')^{\alpha_i};$$

Output $\pi \leftarrow (\sigma_1, \sigma_2, \gamma, V', V'', M', M'', m')$
$\mathsf{LHS.Ver}(\mathsf{pk}_{\mathsf{LHS}}, \mathcal{P}, \Delta, m, \pi)$ Let $\mathcal{P} = (f, \boldsymbol{\tau})$ and $f = (\alpha_1, \ldots, \alpha_n)$. Check that
– $\mathsf{Ver}(\mathsf{pk}, \gamma, (\Delta, \sigma_1, \sigma_2)) = 1$
– $e(M', g_2) = e(m', W_1)$
– $e(M'', g_2) = e(m/m', W_2)$
– $e(V', g_2) = e(\prod_{i=1}^{n} h_i^{\alpha_i} M', \sigma_1)$
– $e(V'', g_2) = e(\prod_{i=1}^{n} k_i^{\alpha_i} M'', \sigma_2)$
if any of the checks above fails output 0, else output 1.

We now prove the following theorem

Theorem 10 (informal). *If the 2-3CDH assumption [CMP14] holds and the underlying signature scheme guarantees unforgeability against adaptive chosen message attack then the scheme described above is a LHS scheme secure providing strong adaptive security (in the sense discussed above).*

Proof. We split the proof in 3 different cases. In each of them, we will show how an adversary that breaks the security of the scheme can be used to build a simulator that breaks the 2-3CDH assumption (or the security of the underlying signature scheme S). In particular, let $m^*, \mathcal{P}^* = (\Delta^*, f^*, \tau_1^*, \ldots, \tau_t^*)$, $\pi = (\sigma_1^*, \sigma_2^*, \gamma^*, V'^*, V''^*, M'^*, M''^*, m'^*)$ be the forgery produced by the adversary $\mathcal{A}$, Then (at least) one of the following conditions hold:

Type I The list T (defined as in the description of the scheme) is empty or $\Delta^* \notin T$
Type II For all $i \in [n]$, $\exists\, (\Delta^*, \tau_i, m_i) \in T$ and $m^* \neq f^*(m_1, \ldots, m_n)$, where m_i are the messages queried as part of the Δ^* dataset.
Type III Strong: there exists $j \in \{1, \ldots, n\}$ such that $(\tau_j^*, \cdot) \notin T$, even though Δ^* is in the list.

Type I forgeries. In this case it is easy to reduce security to the unforgeability of the underlying signature scheme S. The simulator is very simple: it uses its signing oracle for S to compute the component of each signature authenticating the dataset identifier, and can easily compute the remaining parts of each signature by creating the rest of the secret and public key as in the real case. When adversary $\mathcal{A}$ outputs a forgery, by definition of this case the simulator can output $(\Delta^*, \sigma_1^*, \sigma_2^*), \gamma^*$ as a forgery for S.

Type II forgeries. Notice that, by construction, the forgery condition implies that either

$$m'^* \neq f^*(m'_1, \ldots, m'_n) \tag{12}$$

or

$$m^*/m'^* \neq f^*(m_1/m'_1, \ldots, m_n/m'_n) \tag{13}$$

The simulator guesses which one of the two cases occurs and it will be correct with $1/2$ probability. Without loss of generality, in the remaining of the proof we will assume that condition 12 occurs. The reduction for the other case works analogously.

Notice also that because of the signature verification equations it must be the case that

$$V'^* \leftarrow \left(\prod_{i=1}^{n}(h_i)^{\alpha_i^*} M'^*\right)^r \; ; \quad M'^* = (m'^*)^{\omega_1} \tag{14}$$

Letting

$$V' \leftarrow \left(\prod_{i=1}^{n}(h_i)^{\alpha_i^*} M_i'^{\alpha_i^*}\right)^r \; ; \quad \prod_{i=1}^{n} M_i'^{\alpha_i^*} = (\prod_{i=1}^{n} m'^{\alpha_i^*})^{\omega_1} \tag{15}$$

the equations by computing f^* on the input originally queried for dataset Δ^*, we have

$$V'^*/V' = \left(M'^*/\prod_{i=1}^{n} M_i'^{\alpha_i^*}\right)^r \tag{16}$$

we show how to use this to solve the 2-3 CDH. In particular we construct the solver $\mathcal{B}$ as follows.

$\mathcal{B}$ takes as input the 2-3 CDH challenge $(g_1, g_2, g_1^{\omega}, g_1^r, g_2^{\omega}, g_2^r)$ and guesses both the dataset identifier Δ_i and the case (12 or 13 above) for which it will receive a forgery. Again, w.l.o.g, let us assume that case 12 occurs.

$\mathcal{B}$ sets $W_1 \leftarrow g_2^{\omega}$, it selects $b_i, \delta_i \leftarrow\$ \mathbb{Z}_p$, for $i = 1, \ldots, n$ and computes $\bar{m}_i \leftarrow g_1^{b_i}$ as part of its answers for signing queries regarding dataset Δ_i and $h_i \leftarrow g_1^{\delta_i}\bar{m}_i^{-\omega} = g^{\delta_i}(g_1^{\omega})^{-b_i}$. The rest of the public key/secret key material is produced as prescribed by key generation.

Signing queries are dealt with as follows. On input the query (Δ, τ_j, m), if $\Delta = \Delta_i$, i.e. the target dataset, it proceeds as follows

if $\Delta \notin T$ it sets

- $\sigma_1 \leftarrow g_2^r$
- Sample $s \leftarrow\$ \mathbb{Z}_p$ and set $\sigma_2 \leftarrow g_2^s$
- $\gamma \leftarrow \mathsf{Sign}(\mathsf{sk}, \Delta, \sigma_1, \sigma_2)$

else $\mathcal{B}$ retrieves $(\gamma, \sigma_1, \sigma_2)$ from memory. In any case it sets $m'_j \leftarrow \bar{m}_j$, $M'_j \leftarrow (g_1^{\omega})^{b_j}$, $V' \leftarrow (g_1^r)^{\delta_j} = (h_j M'_j)^r$ and $m''_j \leftarrow m/m'_j$, $M''_j \leftarrow m''^{\omega_2}_j$, $V'' \leftarrow (k_j M''_j)^s$

Queries for $\Delta \neq \Delta_i$ are answered as follows.

Sample $r_j, a_j \leftarrow\$ \mathbb{Z}_p$, $m'_j \leftarrow g_1^{a_j}$, $M'_j \leftarrow (g_1^{\omega})^{a_j}$, $V' \leftarrow (h_j M'_j)^{r_j}$. The rest is computed as the real signer would do.

When $\mathcal{A}$ produces a type II forgery $m^*, \mathcal{P}^* = (\Delta^*, f^*, \tau_1^*, \ldots, \tau_t^*)$, $\pi = (\sigma_1^*, \sigma_2^*, \gamma^*, V'^*, V''^*, M'^*, M''^*, m'^*)$, $\mathcal{B}$ outputs

$$(H, T) = \left(m'^* / \prod_{j=1}^{n} (m'_j)^{\alpha_i^*}, V'^* / V' \right)$$

where $V' \leftarrow \left(\prod_{j=1}^{n} (h_i)^{\alpha_i^*} M'_j \right)^r$. Thus, since by definition of type II forgeries $H = m'^* / \prod_{j=1}^{n} (m'_j)^{\alpha_i^*} \neq 1$, by combining the results of equations (14)–(15)–(16), we obtain that (H, T) is a valid solution of the 2-3 CDH problem, i.e., $H \neq 1$ and $T = H^{r\omega}$.

Type III forgeries. Here we assume that in the forgery produced by $\mathcal{A}$ there is some index j such that τ_j^* is not in T even though Δ^* is not a novel dataset identifier. We show that, in the restricted case highlighted above, given an adversary $\mathcal{A}$ that produces such a type 3 forgery we can use it to build a $\mathcal{B}$ adversary that produces a type two forgery. $\mathcal{B}$ runs $\mathcal{A}$ on the same pk material it receives. When $\mathcal{A}$ asks for a signing query $\mathcal{B}$ asks the same query to its oracle and sends it to $\mathcal{A}$. When $\mathcal{A}$ produces its type 3 forgery

$$m^*, \mathcal{P}^* = (\Delta^*, f^*, \tau_1^*, \ldots, \tau_t^*), \pi = (\sigma_1^*, \sigma_2^*, \gamma^*, V'^*, V''^*, M'^*, M''^*, m'^*)$$

we denote with J the (non empty) set of indexes for which τ_j^* is not in T and $\mathcal{B}$ proceeds sampling random messages m_j (for all $j \in J$) and by querying its signing oracle on $(\Delta^*, \tau_j^*, m_j)$ for each $j \in J$. $\mathcal{B}$ outputs what received from $\mathcal{A}$ as the required type II forgery. Notice that this is, indeed, a type II forgery as $\mathcal{B}$ made signing queries for all labels. Moreover, letting y the output obtained by computing f^* on all the msgs queried by $\mathcal{B}$, we have that $y \neq m^*$ with overwhelming probability. Indeed, since all α_i^* are assumed to be non zero, and the queries of A (information theoretically) independent from the randomly sampled messages later queried by $\mathcal{B}$, $y = m^*$ only with probability $1/p$.