Indistinguishability Obfuscation from DDH on 5-linear Maps and Locality-5 PRGs

Huijia Lin[*]
University of California, Santa Barbara

Abstract

We present a new construction of Indistinguishability Obfuscation (IO) from the following:

- asymmetric L-linear maps [Boneh and Silverberg, Eprint 2002] with subexponential Decisional Diffie-Hellman (DDH) assumption,
- locality-L polynomial-stretch pseudorandom generators (PRG) with subexponential security, and
- the subexponential hardness of Learning With Errors (LWE).

When plugging in a candidate PRG with locality-5 (*e.g.*, [Goldreich, ECCC 2010, O'Donnell and Witmer, CCC 2014]), we obtain a construction of IO from subexponential DDH on *5-linear maps* and LWE. Previous IO constructions rely on multilinear maps or graded encodings with higher degrees (at least larger than 30), more complex functionalities (*e.g.*, graded encodings with complex label structures), and stronger assumptions (*e.g.*, the joint-SXDH assumption).

[*]`rachel.lin@cs.ucsb.edu`. Huijia Lin was partially supported by NSF grants CNS-1528178 and CNS-1514526.

Contents

1 Introduction

Indistinguishability obfuscation, defined first in the seminal work of Barak *et al.* [BGI$^+$01a], aims to obfuscate programs into "unintelligible" ones while preserving functionality. IO is an extraordinarily powerful object. Starting from the elegant work of Sahai and Waters [SW14], IO now gives us a treasure-chest of cryptographic constructions, providing answers to a long list of open problems (*e.g.*, [SW14, GGH$^+$13b, GGHR14, GHRW14, CHN$^+$15, GP15, BPW16, BGJ$^+$16]), solutions to new cryptographic goals (*e.g.*, [GGSW13, CHJV15, BGL$^+$15, KLW15]), and even implications outside cryptography [BPR15].

Unfortunately, so far, the existence of IO remain uncertain. Most known candidate IO schemes [GGH$^+$13b, BR14, BGK$^+$14, PST14, AGIS14, GLSW15, Zim15, AB15, GMS16, MSZ16] are built from the so-called *graded encoding schemes* [GGH13a], a framework of complex algebraic structures that, in essence, enables evaluating *polynomial-degree* polynomials on secret encoded values and revealing whether the output is zero or not. The security of most IO candidates are either analyzed in the ideal model or based on strong uber assumptions [PST14], with the only exception that Gentry, Lewko, Sahai and Waters [GLSW15] came up with an IO construction under the *multilinear subgroup elimination* assumption. However, instantiating graded encodings from concrete mathematical objects has proved elusive: Vulnerabilities were demonstrated in all known instantiations [GGH13a, CLT13, LSS14, GGH15, CLT15].

The state-of-affairs motivates one of the most important questions in theory of cryptography today: What mathematical objects and assumptions imply IO? In a recent works [Lin16], the author took the first step in simplifying the algebraic structure needed for constructing IO, and showed that graded encodings supporting only evaluation of constant-degree polynomials, called constant-degree graded encodings, already suffice, assuming the existence of polynomial-stretch PRG in NC^0 and hardness of Learning With Errors (LWE). Following that, the author and Vaikuntanathan [LV16] further weakened the assumption on constant-degree graded encodings from a uber assumption in [Lin16] to the joint-SXDH assumption, which resembles the classical Decisional Diffie-Hellman (DDH) assumption. Following the trajectory of recent developments, we ask,

How much can we narrow the gap between mathematical objects and assumptions that imply IO, and well-studied mathematical objects, like bilinear pairing groups with DDH assumption?

In this work, we show that assuming LWE and the existence of a PRG with small output locality L (*i.e.*, every output bit depends on at most L input bits), IO can be constructed from degree-L (asymmetric) multilinear paring groups, with exactly the DDH assumption. In the literature, candidate PRGs with locality-5 [Gol00, OW14] exist.[1] Assuming their security and LWE, we immediately obtain a construction of IO from *the DDH assumption on* 5*-linear maps*.

Our result improves the previous state-of-the-art at multiple fronts. First, our construction is based on the simpler algebraic structure of multilinear pairing groups than graded encodings. Multilinear pairing groups, introduced by Boneh and Silverberg [BS02], are direct generalization of bilinear pairing groups to higher degree, whereas graded encodings provides more complex functionalities that in particular allows one to pose constraints on what type of polynomials can and cannot be evaluated on a set of secret encoded values. Second, we reduce the concrete degree of graded encodings / multilinear maps needed for IO construction from $6L$ in [LV16] to now L, matching exactly the locality L of the PRG. This is, in some sense, optimal under current techniques that require using the graded encoding / multilinear maps to evaluate the PRG on secret

[1]There is no PRG with locality-4, achieving polynomial stretch [CM01, MST03].

encoded seeds. Last but not least, we simplify the assumption from joint-SXDH that resembles the DDH assumption to the DDH assumption itself on multilinear maps.

1.1 Our Results

SXDH on Multilinear Maps Asymmetric multilinear pairing groups as introduced by Boneh and Silverberg [BS02] generalize asymmetric bilinear pairing maps to a collection of source groups $G_1, \cdots, G_D$, whose elements can be paired to produce elements in a target group G_T via a multilinear map $e(g_1^{a_1}, \cdots, g_D^{a_D}) = g_T^{a_1 \cdots a_D}$. The degree (a.k.a. multilinearity) of the (asymmetric) multilinear map is the number of elements that can be paired together, which equals to the number of source groups D. We say that the multilinear pairing groups have *prime order* if all source groups and the target group have the same prime order, and *composite order* if all groups have the same composite order. In this work, we consider constant-degree multilinear paring groups, and in particular 5-linear pairing groups, with either prime or composite order.

The SXDH assumption on (asymmetric) multilinear pairing groups is a natural generalization of the standard symmetric external Diffie-Hellman (SXDH) assumption on (asymmetric) bilinear pairing groups. In short, SXDH states that the decisional Diffie-Hellman assumption holds in every source group: It postulates that the distribution of g_d^a, g_d^b, g_d^{ab} in any source group d should be indistinguishable to that of g_d^a, g_d^b, g_d^r. Formally

SXDH over D-linear maps: $\forall d \in [D]$,

$$\left\{ g_d^a,\ g_d^b \xleftarrow{\$} G_d \ :\ \{g_i\}_{i\in[D]},\ g_d^a,\ g_d^b,\ g_d^{ab} \right\} \approx \left\{ g_d^a,\ g_d^b,\ g_d^r \xleftarrow{\$} G_d \ :\ \{g_i\}_{i\in[D]},\ g_d^a,\ g_d^b,\ g_d^r \right\},$$

where $\{g_i\}$ is the set of generators in all groups. When $D = 2$, this gives exactly the SXDH assumption on bilinear pairing groups.

Multilinear maps are much simpler than graded encodings. The interface of multilinear pairing groups is much simpler than that of graded encoding schemes introduced by [GGH13a]. First, graded encoding schemes support *graded multiplication* over a collection of groups $\{G_l\}$: Graded multiplication can pair elements of two groups G_{l_1}, G_{l_2}, indexed by two labels l_1, l_2, to produce an element in the group $G_{l_1+l_2}$, indexed by label $l_1 + l_2$ (according to some well-defined addition operation over the labels; for example, if labels are integers, $+$ is integer addition, and if labels are sets, $+$ is set union). In particular, the output element in $G_{l_1+l_2}$ can be further paired with elements in other groups to produce elements in group $G_{l_1+l_2+l_3+\cdots}$ and so on. In contrast, multilinear map allows only "one-shot" multiplication, where the output element belongs to the target group G_T that cannot be paired anymore. Second, graded encoding schemes support the notion of "pairable groups" in the sense that only elements from groups G_{l_1}, G_{l_2} that satisfy a "pairable" relation can be paired (*e.g.*, if labels are sets, then two groups are pairable, if their label-sets l_1, l_2 are disjoint).

The support for graded multiplication between pairable groups provides powerful capabilities. In essence, GES allows one to "engineer" the labels of a set of group elements $\{g_{l_i}^{a_i}\}$, so that, only polynomials of certain specific forms can be evaluated on values in the exponent. In contrast, the simple interface of multilinear maps does not provide such capabilities.

SXDH is simpler than Joint-SXDH Lin and Vaikuntanathan introduced the joint-SXDH assumption on graded encoding schemes, and showed that IO for P/poly can be based on subexponential joint-SXDH and PRG in NC^0. Their joint-SXDH assumption further generalizes the SXDH assumption above: It considers the joint distribution of elements $(g_l^a, g_l^b, g_l^{ab})_{l\in S}$ in a set S of groups. The intuition is that as long as *no* pairs of groups G_{l_1}, G_{l_2} in the set S are pairable, in the same spirit as SXDH, the distribution is possibly indistinguishable to the joint distribution of elements $(g_l^a, g_l^b, g_l^r)_{l \in S}$ in the same set of groups, with random exponents (a, b, r) (note that the same random exponents are used in all groups in S). Though joint-SXDH is a natural generalization of the SXDH assumption, its relation with the SXDH assumption is unknown and is potentially much stronger than the SXDH assumption.

Our Main Result: IO from SXDH on Low-degree Multilinear Maps and Local PRG We are now ready to state our main result:

Theorem 1 (Main Theorem)**.** *Let L be any positive integer. Assume the subexponential hardness of LWE. Then, IO for* $\mathsf{P/poly}$ *is implied by the subexponential SXDH assumption on L-linear pairing groups (with prime or composite order), and the existence of a subexponentially secure locality-L PRG with $n^{1+\varepsilon}$-stretch for any $\varepsilon > 0$.*

We remark that the subexponential hardness of SXDH, PRG, and LWE required by our theorem is weaker than standard notions of subexponential hardness of decisional problems, in the sense that we only require the distinguishing gap to be subexponentially small against polynomial time adversaries, as opposed to subexponential time adversaries (See Section 3 for definition).

Our result establishes a direct and tight connection between the degree D of multilinear maps needed for constructing IO and the locality L of PRGs — *they are the same $D = L$* — assuming subexponential LWE. In the literature, there are many works studying local PRGs. On the negative side, it was shown that there is *no* PRG with locality 4 that achieves super-linear stretch [CM01, MST03]. On the positive side, candidate PRGs with locality 5 and polynomial stretch exist [Gol00, OW14][2]; they are the so-called "random" NC^0 functions [CEMT09, BQ12, OW14, AL16] — a variant of Goldreich's OWFs [Gol00] (See section 1.2 for a brief survey of local PRG.) Plugging a locality-5 PRG in our main theorem immediately gives the following corollary that IO can be based on SXDH on 5-linear maps, assuming subexponential LWE.

Corollary 1 (IO from 5-linear maps and locality-5 PRG)**.** *Assume the subexponential hardness of LWE. IO for* $\mathsf{P/poly}$ *is implied by the subexponential SXDH assumption on* 5-linear pairing groups *(with prime or composite order), and the existence of a subexponentially-secure* locality-5 *PRG with $n^{1+\varepsilon}$-stretch for any $\varepsilon > 0$.*

Our result improves previous works at several fronts. Most candidate IO schemes are built from polynomial-degree graded encodings. Recently, Lin [Lin16] presented the first IO construction from constant-degree graded encodings, assuming the existence of subexponentially secure PRGs in NC^0 and LWE. A drawback of her result is that the security of IO relies on strong subexponential uber assumptions (similar to the semantic security of [PST14]) on the graded encodings. This is improved by Lin and Vaikuntanathan [LV16], who showed that it suffices to assume joint-SXDH on constant-degree graded encodings (without subexponential LWE). Our result further simplifies the algebraic structures and assumptions needed for constructing IO towards the "minimal": First, we give the first construction of IO from constant-degree multilinear maps, as opposed to graded encodings, which provide much more complex functionalities (in particular, graded multiplication between pairable groups). Second, we simplify the assumption from joint-SXDH to SXDH. Third, the multilinear maps that our IO construction are based on have a much smaller degree than that of the graded encodings used in previous works.

[2]The best known attacks on these candidates takes certain specific subexponential time.

So far, the IO construction that is based on graded encodings with the smallest degree is given by [LV16], which additionally relies on a PRG in NC^0. Suppose that the NC^0-PRG can be evaluated, in the exponent of graded encodings, using a degree-D polynomial P (that is, $g^{\mathbf{PRG}(\mathbf{s})} = g^{P(\mathbf{s})}$). Then the graded encodings just need to support evaluation of degree-$(6D+4)$ polynomials (in the exponent). Since the degree D is at least 2 (for there are no linearly-computable PRGs), the degree of their graded encodings is at least 16. In fact, to the best of our knowledge, it is not even clear whether there exist candidate PRGs that admit a degree D lower than 5. On the other hand, the locality L of a PRG upper bounds the degree D (no matter what exponent space the graded encodings have). This is because the PRG has *binary* input strings and any polynomial that computes the PRG is multilinear (for $x^2 = x$ when $x \in \{0,1\}$). Therefore, the degree of the graded encodings needed in [LV16] is only upper bounded by $(6L+4)$, much higher than degree L needed in our IO construction.

Our Approach via Bootstrapping: IO from Locality-L PRG and Degree-L FE We follow the same two-step approaches in all previous IO constructions: First, construct IO for $\mathsf{P/poly}$ from some simpler primitives — call this the *bootstrapping step* — and then instantiate the primitives needed, using graded encodings or multilinear maps. In recent works, simplifying the intermediate primitives acts as the "catalyst" for simplifying the underlying algebraic structures and assumptions. Often, additional cryptographic assumptions, like LWE and the existence of NC^0-PRGs [LV16], are used in the bootstrapping step, in order to "trade" for the simplicity of the intermediate primitives, and eventually the simplicity of the underlying algebra and assumptions. This provides a framework for using more well-studied or just completely different types of assumptions, to weaken the requirements on graded encodings and multilinear maps.

In this work, one of our goals is minimizing the degree of multilinear maps needed. To achieve this, we first build IO from FE for computing low degree, L, polynomials in some ring $\mathcal{R}$ (which eventually corresponds to the exponent space of multilinear maps used for instantiating the FE), assuming the existence of a polynomial-stretch PRG with locality-L.

Theorem 2 (Bootstrapping Theorem). *Let L be any positive integer. Assume the subexponential hardness of LWE. IO for $\mathsf{P/poly}$ is implied by the existence of sub-exponentially secure (collusion resistant) secret-key FE schemes for computing degree-L polynomials in any ring $\mathcal{R}$, with linear efficiency and a sub-exponentially secure locality-L PRG with $n^{1+\varepsilon}$-stretch for any $\varepsilon > 0$.*

(In the case that the FE schemes are public-key, the assumption on the hardness of LWE is not needed.)

Above, the linear efficiency of FE schemes means that encryption time is linear in the input length $N(\lambda)$, that is, $\mathsf{Time}_{\mathsf{FE.Enc}} = N(\lambda)\,\mathrm{poly}(\lambda)$. In fact, we only need the FE scheme to achieve the weaker functionality of revealing whether the output of a degree-L polynomial is zero in $\mathcal{R}$ (see Section 3.6 for the formal definition). We refer to such FE schemes as degree-L FE in $\mathcal{R}$ with linear efficiency.

Previous bootstrapping theorems build IO for $\mathsf{P/poly}$ from either of the following: *i)* from IO for NC^1 [GGH$^+$13b], *or ii)* from subexponentially secure FE for NC^1 [AJ15, BV15, AJS15, BNPW16], *or iii)* from subexponentially secure IO for constant degree computations and PRG in NC^0 [Lin16], *or iv)* from subexponentially secure FE for NC^0 and PRG in NC^0 [LV16]. (Some bootstrapping theorems additionally assume LWE [GGH$^+$13b, Lin16] or the existence of public key encryption [BNPW16]). The the bootstrapping theorem most related to this work is that of [LV16]: The parameterized version of their theorem states that IO can be built from degree-$(3L+2)$ FE and locality-L PRG. We here reduce the degree of FE to exactly L.

An overview of our bootstrapping step is given in Section 2.1 and details provided in Section 5.1.

Degree Preserving Construction of FE: Degree-L FE from SXDH on Degree -L Multilinear Maps Using multilinear paring groups whose exponent space corresponds to a ring $\mathcal{R}$, we can instantiate degree-L FE in ring $\mathcal{R}$, if the multilinear map has degree L.

Theorem 3 (Degree-Preserving Construction of FE). *Let D be any positive integer and $\mathcal{R}$ any ring. Assuming SXDH on D-linear maps over ring $\mathcal{R}$, there exist secret key FE schemes for degree-D polynomials in $\mathcal{R}$, with linear efficiency.*

Previous constructions of FE for NC^1 either relies on IO for NC^1 or high degree multilinear maps [GGH+13b, GGHZ16], whose degree is polynomial in the circuit-size of the computations. In [LV16], Lin and Vaikuntanathan constructed FE for NC^0 from constant-degree graded encodings. Their construction, however, is *not* degree-preserving: To compute NC^0 functions that can be evaluated in degree D, they require degree $2D$ graded encodings. Our FE construction is the first one that supports degree-D computations using only degree-D multilinear maps. An intriguing question to ask is whether we can rely on multilinear maps with degree even less than D. This can be done when the degree D of the computations is high enough, for instance, polynomial, since one can apply randomized encodings [IK02, AIK04] to represent such computations with ones that have a much smaller degree. However, when the degree D of the computations is already small, and cannot be reduced via randomized encodings, it is unclear whether one can construct FE for such degree-D computations, using multilinear maps with degree less than D. In this setting, it is not unclear how to achieve functionality, and would require completely new ideas.

See Section 2.2 and 2.3 for an overview of our degree-preserving FE construction, and details in Section 8.

Additional Contributions. Along the way of designing our degree-preserving FE construction, we also construct the following primitives that are of independent interests.

Simple Function Hiding IPE Schemes from SXDH on Bilinear Maps Without using the heavy hammers of multilinear maps or IO, the state-of-the-art collusion resistant FE schemes can only compute inner products, they are called Inner Product Encryption (IPE). In the literature, Abdalla, Bourse, De Caro and Pointcheval (ABCP) [ABCP15] came up with a public key IPE scheme based on one of a variety of assumptions, such as the decisional Diffie-Hellman assumption, the Paillier assumption and the learning with errors assumption. Bishop, Jain and Kowalczyk [BJK15] (BJK) constructed the first secret-key IPE scheme based on the SXDH assumption over asymmetric bilinear pairing groups; however, their scheme achieves a stronger notion of security than the ABCP scheme, called *weak function-hiding*. A followup construction [DDM16] further achieved *fully function hiding* IPE. Moreover, Lin and Vaikuntanathan [LV16] showed how to generically transform any weak function hiding IPE to full function hiding IPE.

While the ABCP public-key IPE scheme is simple, the secret-key (weak) function hiding IPE schemes [BJK15, DDM16] are much more complex. This is attributed by the fact that they achieve the stronger function hiding property, which guarantees hiding of *both* inputs and functions (revealing only the function outputs), whereas standard security only hides inputs. In this work, we give a *simple* construction of weak function hiding IPE scheme from SXDH on bilinear maps (which can then be lifted to function hiding IPE using [LV16]). Our IPE scheme is built from the ABCP public-key IPE scheme in a modular way, and inherits its efficiency and simplicity:

Ciphertexts and secret keys of length-N vectors consists of $(N+2)$ group elements, and the construction and security proof of our scheme fits within 2 pages (reducing to the security of the ABCP IPE scheme). In addition, the new scheme satisfies certain special properties that are important for our construction of degree-L FE schemes above, which are not satisfied by previous IPE schemes [BJK15, DDM16]. (See Section 2.4 for an overview of our simple function hiding IPE and Section 6 for details.)

High-Degree IPE We also generalize IPE to the notion of *high-degree IPE*, or *HIPE* for short. They are *multi-input* FE scheme, introduced by [GGG+14], for computing, what we call, *degree-D inner product* defined as

$$\langle \mathbf{x}^1, \cdots, \mathbf{x}^D \rangle = \Sigma_{i \in [N]} x_i^1 x_i^2 \cdots x_i^D \ .$$

We construct HIPE for degree-D inner products from degree-D multilinear maps, which is then used as a key tool in our construction of degree-D FE schemes. We believe that this notion is interesting on its own and may have other applications. (See Section 2.3 for an overview of HIPE and Section 7 for details.)

1.2 Local Pseudo-Random Generators

We briefly survey constructions of local PRGs. On the negative side, it was shown that there is no PRG in NC^0_4 (with output locality 4) achieving super-linear stretch [CM01, MST03]. On the positive side, Applebaum, Ishai, and Kushilevitz [AIK04] showed that any PRG in NC^1 can be efficiently "compiled" into a PRG in NC^0 using randomized encodings, but with only *sub-linear* stretch. They further constructed a *linear-stretch* PRG in NC^0 under a specific intractability assumption related to the hardness of decoding "sparsely generated" linear codes [AIK08], previously conjectured by Alekhnovich [Ale03]. Applebaum [App12] showed that based on the one-wayness of "random" NC^0 functions (with appropriate output length) – a variant of Goldreich's one-way functions [Gol00], there exists a linear stretch PRG in NC^0, as well as a polynomial-stretch weak PRG (where the distinguishing advantage is $1/\operatorname{poly}(n)$). In fact, the random NC^0 functions themselves are polynomial-stretch weak PRGs.

Random NC^0 functions are also candidate polynomial-stretch PRGs. They are defined w.r.t. a D-ary predicate P and a stretch parameter $m(n)$: Let $\mathcal{F}_{P,m}$ be a distribution over D-local functions $f : \{0,1\}^n \to \{0,1\}^m$ defined by setting every output bit as P evaluated on D randomly chosen input bits. Several works investigated the (in)security of random NC^0 functions as one-way functions or pseudo-random generators. So far, best known attacks take (certain specific) subexponential time when the choice of the predicate P avoids degenerate cases [CEMT09, BQ12, OW14, AL16]. In particular, O'Donnell and Witmer [OW14] gave evidence for the security of random NC^0 functions defined by the 5-local predicate $P(x_1, x_2, x_3, x_4, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 x_5 \pmod 2$. They showed that when the stretch is $n^{1.499}$, this family is secure against both subexponential-time $\mathbb{F}_2$-linear attacks, as well as subexponential-time attacks using SDP hierarchies such as Sherali-Adams$^+$ and Lasserre/Parrilo. We remark that to build IO, local PRG with any non-trivial stretch factor $1 + \varepsilon > 1$ suffices; we can in particular use the O'Donnell-Witmer PRG with $n^{1.0001}$-stretch.

1.3 Concurrent and Independent Work

In a concurrent work[3], Ananth and Sahai introduce a new primitive called projective arithmetic FE (PAFE) – a version of FE tailored to arithmetic circuits. They obtain sub-linear secret key FE, which suffices to build IO, from PAFE for degree d polynomials and randomizing polynomials for degree d polynomials, assuming sub-exponential LWE. The randomizing polynomials they need are required to satisfy some special properties. They give a degree preserving reduction from degree d multilinear maps to PAFE for degree d polynomials. Finally, they show some instantiations of the randomizing polynomials that they need, including an instantiation from degree 5 multilinear maps. This yields a construction of IO from degree-5 multilinear maps, assuming subexponential LWE, subexponentially secure PRGs of locality 5, and a specific family of subexponential assumptions over degree-5 multilinear maps.

1.4 Organization

In Section 2, we give an overview of our constructions and techniques. We present in Section 3 basic notations and definitions, and in Section 4 the definition of multilinear pairing groups and the SXDH assumption on them. In Section 5, we show how to bootstrap FE for degree-L polynomials and locality-L PRGs to IO for P/poly, assuming LWE. In Section 6, we construct various function hiding IPE schemes that are building blocks of our constructions of FE. In Section 7, we define and construct high-degree IPE, HIPE, schemes. Finally, in Section 8, we use HIPE schemes to construct our FE schemes for degree-L polynomials from degree-L multilinear pairing groups.

2 Technical Overview

2.1 Bootstrapping

Our bootstrapping theorem follows the same two step approach as [Lin16, LV16]. To construct IO for P/poly,

Step 1**.** First, construct sub-exponentially secure single-key FE schemes **CFE** for NC^1 that are *weakly compact*, meaning that encryption time scales polynomially in the security parameter λ and the input length N, but also scales *sublinearly* in the maximal size S of the circuits for which secret keys are generated. More precisely, a FE scheme is said to be $(1-\varepsilon)$-weakly-compact if its encryption time is $\mathrm{poly}(\lambda, N)S^{1-\varepsilon}$.

Step 2**.** If the FE schemes obtained from Step 1 are public-key schemes, invoke the result of [AJ15, BV15] that any public-key (single-key) weakly-compact FE schemes (for any $\varepsilon > 0$) imply IO for P/poly.

[3]We have engaged in several amicable exchanges with Ananth and Sahai about our respective results over the few weeks preceding the initial posting of our papers. During this time, with regard to the minimum level of multilinearity needed for constructing iO, certain milestones were reached at different times. In particular, Ananth and Sahai had the first paper claiming iO from degree-15 maps. After that, we had the first paper claiming iO from degree-5 maps, and thereafter Ananth and Sahai were also able to modify their paper to claim iO from degree-5 maps. Both groups independently relied on PRGs of locality 5 to achieve these results, and the common bottleneck at degree 5 reflects this. Prior to posting, however, the groups did not exchange any manuscripts, and worked independently. There are several other differences in our results, most notably with regard to the assumptions. In addition, the techniques are substantially different.

Otherwise, if the FE schemes obtained are secret-key schemes, then invoke the recent result of [BNPW16] that any secret-key weakly-compact FE schemes also imply IO for P/poly, assuming additionally the sub-exponential hardness of LWE.

The challenging task is constructing (public-key or secret-key) weakly-compact FE schemes for NC^1 from simpler primitives. In [LV16] (LV), they constructed such schemes from (public key or secret key respectively) *collusion resistant* FE schemes for NC^0 with *linear efficiency*, assuming the existence of a polynomial-stretch PRG in NC^0. There, linear efficiency means that encryption time is linear in input length, $N \operatorname{poly}(\lambda)$. Our first observation is that in their construction, the NC^0-FE schemes can be replaced with FE schemes for computing low degree D polynomials in any $\mathcal{R}$ (also collusion resistant and has linear efficiency), where the degree D is bounded by $(3L+2)$ if PRG has locality L. In this work, we show that the degree of the FE schemes (*i.e.*, the degree of the polynomials supported) can be reduced to L, which gives our bootstrapping theorem. Below, we start with reviewing the LV construction of weakly-compact FE for NC^1, and then modify their construction to reduce the degree of the underlying FE schemes. (In the exposition below, we do not differentiate public-key vs secret-key schemes, since they are handled in the same way.)

The LV Weakly-Compact FE for NC^1. To construct weakly-compact FE schemes for NC^1 from FE schemes for NC^0, LV uses randomized encodings to represent every NC^1 function $f(\mathbf{x})$, as a simpler NC^0 randomized function $\hat{f}(\mathbf{x};\ \mathbf{r})$. Then, to enable computing $f(\mathbf{x})$, it suffices to publish a secret key for $\hat{f} \in \mathsf{NC}^0$, which can be done using the NC^0-FE scheme, together with a ciphertext encrypting $(\mathbf{x}, \mathbf{r})$. However, the resulting ciphertext is not compact, since the randomness $\mathbf{r}$ for computing the randomized encoding is at least of length $S(\lambda) \operatorname{poly}(\lambda)$, where $S(\lambda)$ is the size of the circuit computing f. The key idea of LV is using a polynomial-stretch PRG $\mathbf{PRG} : \{0,1\}^n \to \{0,1\}^{n^{1+\alpha}}$ in NC^0 to generate pseudo-randomness for RE, that is, computing instead $g(\mathbf{x}, \mathbf{s}) = \hat{f}(\mathbf{x}; \mathbf{PRG}(\mathbf{s}))$. Now the input of the function becomes $(\mathbf{x}, \mathbf{s})$, whose length is sublinear in $S(\lambda)$ thanks to the fact that the PRG has polynomial stretch. Since the NC^0-FE scheme has linear efficiency, the ciphertext size is also sublinear in $S(\lambda)$. In addition, the function g to be computed is still in NC^0.

Observe that if g can be computed by a degree-D polynomial in some ring $\mathcal{R}$, then one can instantiate the LV construction with degree-D FE schemes in $\mathcal{R}$. The question is how large is the degree D? Plug in the randomized encoding scheme by Applebaum, Ishai, and Kushilevitz [AIK04], whose encodings $\hat{f}(\mathbf{x}; \mathbf{r})$ are computable in NC^0_4 and has degree 1 in $\mathbf{x}$ and degree 3 in $\mathbf{r}$. Then, the degree of g is determined by the degree D_{PRG} of the PRG (*i.e.*, the minimal degree of polynomials that computes PRG in $\mathcal{R}$), namely, $D = 3D_{\text{PRG}} + 1$. Since the degree of the PRG is upper bounded by its locality $D_{\text{PRG}} \le L$, the degree of g is bounded by $3L+1$. For the security proof to work out, the actual functions used in the LV construction are more complicated and has degree $3L+2$; we omit details here. In summary, in the LV construction, it suffices to use with a degree-$(3L+2)$ FE scheme.

Relying on Degree-L FE To reduce the degree of polynomials computed using the low-degree FE, our key idea is *pre-processing* the input $(\mathbf{x}, \mathbf{s})$, so that, part of the computation of the function g is already done at *encryption time*. To illustrate the idea, recall that g is linear in $\mathbf{x}$. Thus, if one pre-computes $\mathbf{x} \otimes \mathbf{s}$ (where $\mathbf{x} \otimes \mathbf{s}$ is the tensor product of $\mathbf{x}$ and $\mathbf{s}$), then g can be computed with one degree less. More specifically, there exists another function g' that takes input $(\mathbf{x}, \mathbf{s}, \mathbf{x} \otimes \mathbf{s})$ and computes $g(\mathbf{x}, \mathbf{s})$ in degree $3L$, by replacing every monomial of form $x_i s_{i_1} s_{i_2} \cdots$ with $(x_i s_{i_1})\, s_{i_2} \cdots$, where $x_i s_{i_1}$ is taken directly from $\mathbf{x} \otimes \mathbf{s}$. Therefore, we can modify the LV construction to encrypt

$(\mathbf{x}, \mathbf{s}, \mathbf{x} \otimes \mathbf{s})$, whose length is still sublinear in $S(\lambda)$, and generate keys for functions g' that have degree $3L$.

The more tricky part is how to further reduce the degree of g in s. The naive method of pre-computing s $\otimes$ s at encryption time would not work, since it would make encryption time exceed $S(\lambda)$, losing compactness. To avoid this, consider a simple case where the NC^1 function f to be computed is *decomposable*, in the sense that it has $I = S(\lambda)/\operatorname{poly}(\lambda)$ output bits, and every output bit $i \in [I]$ can be computed by a function f_i of fixed polynomial size $\operatorname{poly}(\lambda)$. (In fact, it is w.l.o.g. to assume this, since every function f can be turned into one that is decomposable using Yao's garbled circuits.) Then, the AIK randomized encoding of f consists of $\{\hat{f}_i(\mathbf{x}, \mathbf{r}[i])\}_{i\in[I]}$, where the random tape $\mathbf{r}[i]$ for every encoding has a fixed polynomial length $Q = \operatorname{poly}(\lambda)$, since $|f_i| = \operatorname{poly}(\lambda)$.

In LV, all the random tapes $\{\mathbf{r}[i]\}$ are generated by evaluating a PRG on a single seed $\mathbf{r} = \mathbf{PRG}(\mathbf{s})$. We first modify how these random tapes are generated. Parse s as Q equally-long seeds, $\mathbf{s}_1, \cdots \mathbf{s}_Q$, and use $\mathbf{s}_q$ to generate the q^{th} bit in all the random tapes, that is,

$$\forall\, q \in [Q],\ i \in [I], \quad \mathbf{r}[i]_q = \mathbf{PRG}(\mathbf{s}_q)|_i = \mathbf{PRG}_i(\{s_{q,\gamma}\}_{\gamma\in\Gamma(i)})\ ,$$

where $\mathbf{PRG}_i$ is the function that computes the i^{th} output bit of the PRG, which depends on at most L seed bits with indexes $\gamma \in \Gamma(i)$. $\mathbf{PRG}(\mathbf{s}_q)$ is a length-I string, and hence the length $|\mathbf{s}_q|$ of each seed $\mathbf{s}_q$ is sublinear in $S(\lambda)$. Since each encoding $\hat{f}_i$ has degree 3 in its random tape $\mathbf{r}[i]$, consider an arbitrary degree 3 monomial $\mathbf{r}[i]_{q_1}\mathbf{r}[i]_{q_2}\mathbf{r}[i]_{q_2}$.

$$\begin{aligned}\mathbf{r}[i]_{q_1}\mathbf{r}[i]_{q_2}\mathbf{r}[i]_{q_2} &= \mathbf{PRG}_i(\{s_{q_1,\gamma}\}_{\gamma\in\Gamma(i)})\,\mathbf{PRG}_i(\{s_{q_2,\gamma}\}_{\gamma\in\Gamma(i)})\,\mathbf{PRG}_i(\{s_{q_3,\gamma}\}_{\gamma\in\Gamma(i)})\\ &= \sum_{\substack{\text{Monomials}\\ X,Y,Z \text{ in } \mathbf{PRG}_i}} \begin{pmatrix} & X(s_{q_1,\gamma_1}, & \cdots, & s_{q_1,\gamma_L})\\ \times & Y(s_{q_2,\gamma_1}, & \cdots, & s_{q_2,\gamma_L})\\ \times & Z(s_{q_3,\gamma_1}, & \cdots, & s_{q_3,\gamma_L})\end{pmatrix}, \qquad \text{where } \Gamma(i) = \{\gamma_1, \cdots, \gamma_L\}\end{aligned}$$

Now, suppose that for every index $\gamma \in [|vs_q|]$ in all seeds, the encryptor pre-compute all the degree ≤ 3 monomials over the γ^{th} bits in all Q seeds; denote this set as

$$M^3(\mathbf{s}, \gamma) = \{\text{ degree} \leq 3 \text{ monomials over } \{\mathbf{s}_{q,\gamma}\}_{q\in[Q]}\ \}\ .$$

Note that given $M^3(\mathbf{s}, \gamma)$ for every $\gamma \in \Gamma(i)$, the above monomial $\mathbf{r}[i]_{q_1}\mathbf{r}[i]_{q_2}\mathbf{r}[i]_{q_2}$ can be computed in just degree L. Therefore, given $M^3(\mathbf{s}, \gamma)$ for every $\gamma \in [|\mathbf{s}_q|]$, the function g can be computed in degree L (with additionally the above-mentioned trick for reducing the degree in $\mathbf{x}$). More precisely, there exists a degree-L polynomial g'' that, on input $\mathbf{x}$, $\{M^3(\mathbf{s}, \gamma)\}_\gamma$, and their tensor product, computes $g(\mathbf{x}, \mathbf{s})$.

Finally, we need to make sure that the total number of such degree ≤ 3 monomials is sublinear in $S(\lambda)$, so that, encryption remains weakly-compact. Note that, for each $\gamma \in [|\mathbf{s}_q|]$, the number of degree ≤ 3 monomials over the γ^{th} bits in these Q seeds is bounded by $(Q+1)^3 = \operatorname{poly}(\lambda)$. Moreover, the length of each seed $|s_q|$ is still sublinear in $S(\lambda)$. Thus, the total number of monomials to be pre-computed is sublinear in $S(\lambda)$.

2.2 Quadratic Secret-Key FE

Before proceeding to constructing degree-D FE schemes from SXDH on degree-D MMaps, we describe a self-contained construction of FE for quadratic polynomials from SXDH on bilinear maps. The degree-D scheme is a generalization of the quadratic scheme.

In the literature, the only constructions of collusion-resistant FE from standard assumptions are for computing inner products, referred to as Inner Product Encryption (IPE). Roughly speaking, a (public key or secret key) IPE scheme allows to encode vectors $\mathbf{y}$ and $\mathbf{x}$ in a ring $\mathcal{R}$, in a function key $\mathsf{iSK}(\mathbf{y})$ and ciphertext $\mathsf{iCT}(\mathbf{x})$ respectively, and decryption evaluates the inner product $\langle \mathbf{y}, \mathbf{x} \rangle$. In this work (like in [LV16]), we use specific IPEs that compute the inner product *in the exponent*, which, in particular, allows the decryptor to test whether the inner product is zero, or whether it falls into any polynomial-sized range. [4]

Given IPE schemes, it is trivial to implement FE for quadratic polynomials, or quadratic FE schemes for short: Simply write a quadratic function f as a linear function over quadratic monomials $f(x) = \Sigma_{i,j} c_{i,j} x_i x_j = \langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle$. Then, generate an IPE secret key $\mathsf{iSK}(\mathbf{c})$, and an IPE ciphertext $\mathsf{iSK}(\mathbf{x} \otimes \mathbf{x})$, from which the function output can be computed. However, such a scheme has encryption time quadratic in the input length $N = |\mathbf{x}|$. and improving the encryption time to become linear in the input length under standard assumptions (*e.g.* bilinear maps) has proved elusive.

In this work, we construct the first quadratic FE schemes in $\mathcal{R}$ with linear encryption time, based on SXDH on bilinear maps over $\mathcal{R}$. At a high-level, our key idea is starting with the above trivial quadratic FE schemes, and "compress" the encryption time from quadratic to linear, by publishing only certain "compressed information" of linear size at encryption time, which can later be expanded to an IPE ciphertext of $\mathbf{x} \otimes \mathbf{x}$ at decryption time. To make this idea work, we will use, as our basis, the public key IPE scheme by Abdalla, Bourse, De Caro, Pointcheval (ABCP) [ABCP15] based on the DDH assumption. Let us start with reviewing their scheme.

The ABCP public key IPE scheme The ABCP scheme $\mathbf{IPE}$ resembles the El Gamal encryption and is extremely simple. Let G be a cyclic group of order p with generator g, in which DDH holds. A master secret key of the ABCP scheme is a random vector $\mathbf{s} = s_1, \cdots, s_N \xleftarrow{\$} \mathbb{Z}_p^N$, and its corresponding public key is $\mathsf{iMPK} = g^{s_1}, \cdots g^{s_N}$. A ciphertext encrypting a vector $\mathbf{x} = x_1, \cdots, x_N$ looks like $\mathsf{iCT} = g^{-r},\ g^{rs_1 + x_1},\ \cdots,\ g^{rs_N + x_N}$, where r is the random scalar "shared" for encrypting every coordinate. It is easy to see that it follows from DDH that this encryption is semantically secure.

To turn El Gamal into an IPE scheme, observe that given a vector $\mathbf{y} \in \mathbb{Z}_p^N$, and in addition the inner product $\langle \mathbf{y}, \mathbf{s} \rangle$ in the clear, one can homomorphically compute inner product in the exponent to obtain $g^{-r\langle \mathbf{y}, \mathbf{s} \rangle} g^{r\langle \mathbf{s}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{y} \rangle} = g^{\langle \mathbf{x}, \mathbf{y} \rangle}$, which reveals whether the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ is zero or not. Therefore, the ABCP scheme simply sets the secret key to be $\mathsf{iSK} = \langle \mathbf{s}, \mathbf{y} \rangle \,||\mathbf{y}$.

In this work, we will use the bracket notation $[x]_l = g_l^x$ to represent group elements. Under this notation, the ABCP scheme can be written as,

$$\mathsf{iMSK} = \mathbf{s} \xleftarrow{\$} \mathbb{Z}_p, \qquad \mathsf{iMPK} = [\mathbf{s}], \qquad \mathsf{iCT} = [-r \,||\, (r\,\mathbf{s} + \mathbf{x})] \qquad \mathsf{iSK} = \langle \mathbf{s}, \mathbf{y} \rangle \,||\mathbf{y}$$

where $a\mathbf{u}$ denotes coordinate-wise multiplication with a scalar a and $\mathbf{u} + \mathbf{v}$ denotes coordinate-wise addition between two vectors. We will also refer to $[x]_l$ as an encoding of x in group G_l.

Compress an ABCP ciphertext $\mathsf{iCT}(\mathbf{x} \otimes \mathbf{x})$ Our intuitive idea of "compressing" a ciphertext $\mathsf{iCT}(\mathbf{x} \otimes \mathbf{x})$ encrypting all quadratic monomials of $\mathbf{x}$ immediately hits a barrier: The ciphertext has form $\mathsf{iCT} = [-r \,||\, (r\,\mathbf{s} + \mathbf{x} \otimes \mathbf{x})]$ and contains information of the master secret key $\mathbf{s}$ of quadratic length, which is truely random and cannot be "compressed".

Our first idea is replacing the truly random secret key $\mathbf{s}$ with the tensor product of two length-N vectors $\mathbf{s}^1 \otimes \mathbf{s}^2$, so that, the new ciphertext $\mathsf{iCT} = \left[-r \,||\, (r\,\mathbf{s}^1 \otimes \mathbf{s}^2 \,+\, \mathbf{x} \otimes \mathbf{x})\right]$ depends only

[4]Such IPEs should be contrasted with functional encryption for testing the orthogonality of two vectors (see, e.g., [KSW08, LOS+10] and many others), which reveals *only* whether the inner product is zero and nothing else; in particular, they do not compute the inner product in the exponent in a way that allows for further computation.

on information, namely $(r, \mathbf{s}^1, \mathbf{s}^2, \mathbf{x})$, of linear size in total. Roughly speaking, the reason that we resolve to using the tensor product $\mathbf{s}^1 \otimes \mathbf{s}^2$ is that, under the DDH assumption, encodings $[\mathbf{s}^1 \otimes \mathbf{s}^2]$ is indistinguishable to encodings of N^2 truely random elements. Thus, there is hope that generating the master secret key as a tensor product is just "as good as" using truly random master secret keys. As we will see later, this hope is true, however through complicated security proof.

Now, it is information theoretically possible to compress $\mathsf{iCT}(\mathbf{x} \otimes \mathbf{x})$. However, simply publishing $(r, \mathbf{s}^1, \mathbf{s}^2, \mathbf{x})$ would blatantly violate security. We need a way to securely and succinctly encode them so that only the ciphertext iCT is revealed. Classical cryptographic tools for hiding computation like garbled circuits or randomized encodings do not help here, since the output length is quadratic, garbled circuits or randomized encodings would have at least quadratic size as well. Instead, we leverage the special structure of iCT: The last N^2 encodings of iCT each encodes an element (or exponent) that is the inner product of two length-2 vectors, that is,

$$\mathsf{iCT}[0] = [-r], \quad \Big(\mathsf{iCT}[i,j] = [\ \langle\ x_i || s_i^1\ ,\ x_j || r s_j^2\ \rangle\ ] \Big)_{i \in [N], j \in [N]}$$

Here, for convenience, we use 0 and $\{(i,j)\}$ to index different encodings in iCT.

Suppose that we have a (secret key) IPE scheme **cIPE** that is function hiding (defined shortly) from bilinear maps, and has certain *canonical form*: In particular, its ciphertexts and secret keys encodes the input and function vectors in different source groups G_1, G_2 of the bilinear map, and decryption simply uses pairing to produce an *encoding of the output inner product* in the target group G_3. (Unfortunately, off-the-shelf function hiding IPEs [BJK15, LV16, DDM16] do not have the canonical form and we discuss how to construct such a scheme later.)

Then, we can use a canonical function hiding IPE, to generate the last N^2 encodings $\{\mathsf{iCT}[i,j]\}$: Publish N ciphertext $\{\mathsf{cCT}_i\}$ where each cCT_i encrypts vector $(x_i || s_i^1)$, and N secret keys $\{\mathsf{cSK}_j\}$ where each cSK_j encrypts vector $(x_j || r s_j^2)$. To obtain the $(i,j)^{\text{th}}$ encoding, one can simply decrypt the i^{th} ciphertext using the j^{th} secret key, which produces

$$\mathsf{iCT}[i,j] = [\ \langle\ x_i || s_i^1\ ,\ x_j || r s_j^2\ \rangle\ ] = \mathsf{cIPE.Dec}(\mathsf{cSK}_j, \mathsf{cCT}_i)$$

In order to hide r, x_j's, and s_j^2's, the IPE scheme needs to have the stronger function hiding property, which guarantees that secret keys and ciphertexts for two sets of vectors $\{\mathbf{u}_i, \mathbf{v}_i\}$ and $\{\mathbf{u}'_i, \mathbf{v}'_i\}$ are indistinguishable if they produce identical inner products $\langle \mathbf{u}_i, \mathbf{v}_j \rangle = \langle \mathbf{u}'_i, \mathbf{v}'_j \rangle$. Intuitively, the hope is that function hiding ensures that only the set of possible outputs $\{\mathsf{iCT}[i,j]\}$ is revealed, and all other information of $(r, \mathbf{x}, \mathbf{s}^1, \mathbf{s}^2)$ is hidden.

In summary, we now have the first version of our quadratic FE schemes.

VERSION 1 OF OUR SECRET KEY QUADRATIC FE SCHEME **qFE**

- SETUP: A master secret key msk consists of two random vectors $\mathbf{s}^1, \mathbf{s}^2$ of length N.
- KEY GENERATION: A secret key $\mathsf{SK}(\mathbf{c})$ of a function $f_{\mathbf{c}}(\mathbf{x}) = \langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle$ consists of

  $$\mathsf{SK}(\mathbf{c}) = \left(\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c} \rangle\ ,\ \mathbf{c}\ \right)\ .$$

  Note that the secret key is identical that of the ABCP scheme for vector $\mathbf{c}$.
- ENCRYPTION: Sample a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$. A ciphertext $\mathsf{CT}(\mathbf{x})$ of input vector $\mathbf{x}$ contains

$$\mathsf{CT}(\mathbf{x}) = \Big(\ [-r],\ \{\mathsf{cCT}_i(\boldsymbol{\chi}_i^1),\ \mathsf{cSK}_i(\boldsymbol{\chi}_i^2)\}_{i \in [N]}\ ,\ \Big) \quad \text{where } \boldsymbol{\chi}_i^d = \begin{cases} x_i || s_i^1 & \text{if } d = 1 \\ x_i || r s_i^2 & \text{if } d = 2 \end{cases} \qquad (1)$$

and $\{\mathsf{cSK}_j,\ \mathsf{cCT}_i\}$ are generated using a *freshly sampled* master secret key cMSK of a canonical function hiding IPE **cIPE**.

- DECRYPTION: For every $(i, j) \in [N]^2$, decrypt cCT_i using cSK_j to obtain

$$\mathsf{cIPE.Dec}(\mathsf{cSK}_j, \mathsf{cCT}_i) = \left[\langle \boldsymbol{\chi}^1_i, \boldsymbol{\chi}^2_j \rangle\right] = \left[r s^1_i s^2_j + x_i x_j\right] = \mathsf{iCT}[i, j] \ . \qquad (2)$$

  Homomorphically compute encoding $\Lambda_1 = \langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c}\rangle [-r] = \left[-r \langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c}\rangle\right]$, and $\Lambda_2 = \langle \{\mathsf{iCT}[i, j]\}\ ,\ \mathbf{c}\rangle$. Homomorphically add $\Lambda_1 + \Lambda_2$ to produce an encoding of the output $[f_{\mathbf{c}}(\mathbf{x})]$.

Next, we move to describing ideas for the security proof. As we develop the proof ideas, we will need to make a few modifications to the above scheme.

Selective IND-Security of Our Quadratic FE Scheme. We want to show that ciphertexts of **qFE** of one set of inputs $\{\mathbf{u}_i\}$ is indistinguishable from that of another $\{\mathbf{v}_i\}$, as long as all the secret keys published are associated with functions $\{f_{\mathbf{c}_j}\}$ that do not separate these inputs, that is, $f_{\mathbf{c}_j}(\mathbf{u}_i) = f_{\mathbf{c}_j}(\mathbf{v}_i)$ for all i, j. For simplicity of this overview, we restrict our attention to the simpler case where only a single ciphertext and many secret keys are published. Proving security in this case already requires overcoming all the main challenges. The security proof for the general case with many ciphertexts follows from a hybrid argument where the encrypted vectors are switched one by one from $\mathbf{u}_i$ to $\mathbf{v}_i$, and the indistinguishability of each step is proven using the same ideas to the single-ciphertext case.

Naturally, we want to reduce the security of **qFE** the security of the ABCP IPE scheme **IPE** and the function hiding of **cIPE**. Our intuition is that given a ciphertext $\mathsf{CT}(\mathbf{x})$ for $\mathbf{x} = \mathbf{u}$ or $\mathbf{v}$, the security of **cIPE** ensures that the N ciphertexts and secret keys $\{\mathsf{cCT}_i\}, \{\mathsf{cSK}_j\}$ contained in ciphertext $\mathsf{CT}(\mathbf{x})$ reveals only the output encodings $\{\mathsf{iCT}[i, j]\}$ and nothing else. Then, the security of the ABCP scheme ensures that the derived ciphertext iCT encrypting either $\mathbf{u} \otimes \mathbf{u}$ or $\mathbf{v} \otimes \mathbf{v}$ is indistinguishable, at the presence of secret keys for vectors $\{\mathbf{c}_j\}$ that do not separate them. This intuition would go through if the two building blocks **cIPE** and **IPE** provide very strong security guarantees: Naturally, **cIPE** has simulation security, so that, its ciphertexts and secret keys $\{\mathsf{cCT}_i\}, \{\mathsf{cSK}_j\}$ can be simulated from the set of output encodings $\{\mathsf{iCT}[i, j]\}$, and second, the ABCP scheme is secure even when the master secret keys are generated as a tensor product $\mathbf{s}^1 \otimes \mathbf{s}^2$ as opposed to be truely random.

Unfortunately, our building blocks do not provide such strong security guarantees, and proving security of **qFE** without relying on such strong security requirements are the main technical challenges.

- **Challenge 1 — Relying only on indistinguishability-based function hiding of cIPE.** The simulation security of **cIPE** essentially allows one to easily reduce the security of **qFE** to that of **IPE**. With only indistinguishability-based security of **cIPE**, the reduction to security of **IPE** becomes significantly harder. Typically, one build a black-box security reduction that receives from its challenger **IPE** secret keys and a ciphertext, in this case $\{\mathsf{SK}_j\}, \mathsf{iCT}$, and embeds them in the view of the adversary attacking the **qFE** scheme. However, the ciphertext CT of **qFE** has only linear size, but iCT has quadratic size — there is not enough space for embedding. [5]

[5]Non-black-box security reduction may get around this difficulty, but is unclear how one can design a non-black-box reduction here.

To resolve this problem, our idea is to *embed* iCT *in "piecemeal"*. Observe that the ABCP scheme encrypts its input vector *bit by bit* using different master secret key bits, and a shared random scalar. Thus, we can flexibly view its ciphertext iCT either as a single ciphertext, or as a list of many ciphertexts encrypting a list of vectors of shorter length. In particular, we will "cut" the ciphertext into N pieces, each of length N and indexed by $i \in [N]$.

$$\mathsf{iCT} = [r], \quad \left\{ \mathsf{iCT}[i, \star] = \{ \left[r s_i^1 s_j^2 + x_i x_j \right] \}_{j \in [N]} \right\}_{i \in [N]} .$$

Since the i^{th} ciphertext-piece can be viewed as an **IPE** ciphertext of vector $x_i\mathbf{x}$, generated with master secret key $s_i^1\mathbf{s}^2$ and shared random scalar r. Our idea is gradually switching the values of $x_i\mathbf{x}$ from $u_i\mathbf{u}$ to $v_i\mathbf{v}$ piece by piece in N steps. In each step, we first apply the function hiding of **cIPE** to move to a hybrid distribution where the challenge-piece $\mathsf{iCT}[i, \star]$ is directly hardwired the **qFE** ciphertext — since $|\mathsf{iCT}[i, \star]| = N$, there is enough space for it. Then, we rely on the indistinguishability-security of **IPE** to argue that switching the plaintext-piece underlying $\mathsf{iCT}[i, \star]$ from $u_i\mathbf{u}$ to $v_i\mathbf{v}$ is indistinguishable.

- **Challenge 2 — Relying on the security of the ABCP scheme under correlated randomness.** Arguing the indistinguishability of switching the vectors underlying each ciphertext-piece $\mathsf{iCT}[i, \star]$ from $u_i\mathbf{u}$ to $v_i\mathbf{v}$ turns out to be tricky. First, An acute reader might have already noticed the problem that changing pieces in the tensor product would affect the function output, which is noticeable. For example, after switching the first plaintext piece to $v_i\mathbf{v}$, the function output changes to $\langle \mathbf{c}_j, \mathbf{u} \otimes \mathbf{u} \rangle \neq \langle \mathbf{c}_j, v_1\mathbf{v} || \mathbf{u}_{\geq 1} \otimes \mathbf{u} \rangle$. To resolve this problem, we modify the scheme to build in an *offset* value Δ_j in every secret key SK_j to ensure that the function output remains the same throughout all steps.

  Second, the challenge ciphertext-piece is generated with master secret key $s_i^1\mathbf{s}^2$, which is not truly random, since the vector $\mathbf{s}^2$ is used for generating the master secret keys $s_k^1\mathbf{s}^2$ of other ciphertext-pieces for $k \neq i$. We overcome this by relying on the SXDH assumption to argue that encodings of $s_i^1\mathbf{s}^2$, given encodings of s_i^1 and $\mathbf{s}^2$, are indistinguishable to encodings of random elements, and hence as good as a truly random master secret key. Similar idea was used in [LV16].

Overcoming Challenge 1 — Embed ABCP IPE ciphertext in piecemeal. Our goal is switching *piece by piece* the tensor product underlying the derived **IPE** ciphertext from $\mathbf{u} \otimes \mathbf{u}$ to $\mathbf{v} \otimes \mathbf{v}$, which corresponds to changing the encrypted input from $\mathbf{u}$ to $\mathbf{v}$. To do so, we build a sequence of $2N$ hybrids $\{H_\rho^b\}_{\rho \in [N], b \in \{0,1\}}$ satisfying the following desiderata:

1. In H_ρ^b, the ρ^{th} ciphertext-piece $\mathsf{iCT}[\rho, \star]$ is embedded in the **qFE** ciphertext CT,
2. The derived **IPE** ciphertext iCT encrypts the following "hybrid" vectors.

$$\begin{aligned} &\text{In } H_\rho^0, \qquad v_1\mathbf{v}|| \cdots ||v_{\rho-1}\mathbf{v}|| \underline{u_\rho \mathbf{u}} ||u_{\rho+1}\mathbf{u}|| \cdots ||u_N\mathbf{u} \\ &\text{In } H_\rho^1, \qquad v_1\mathbf{v}|| \cdots ||v_{\rho-1}\mathbf{v}|| \underline{v_\rho \mathbf{v}} ||u_{\rho+1}\mathbf{u}|| \cdots ||u_N\mathbf{u} \end{aligned}$$

To build such hybrids, we need to modify our **qFE** scheme to build in more "redundant space" in its ciphertext.

VERSION 2 OF OUR SECRET KEY QUADRATIC FE SCHEME **qFE**

- ENCRYPTION: A ciphertext $\mathsf{CT}(\mathbf{x})$ consists of

$$\mathsf{CT}(\mathbf{x}) = \left([-r],\ \{\mathsf{cCT}_i(\underline{\mathbf{X}^1_i})\}_{i\in[N]},\ \left\{\mathsf{cSK}_j(\underline{\mathbf{X}^2_j})\right\}_{j\in[N]} \right),\ \text{where } \underline{\mathbf{X}^d_i = (\boldsymbol{\chi}^d_i || \mathbf{0}, 0)} \tag{3}$$

where $\{\mathsf{cCT}_i\}$ and $\{\mathsf{cSK}_j\}$ encode vectors $\boldsymbol{\chi}^d_i$ like before, but now padded with 3 zeros.

We refer to the first 4 elements in $\mathbf{X}$'s as the first slot, which holds two vectors of length 2, and the last element as the second slot. In the honest executions, these vectors $\{\mathbf{X}^d_i\}$ are set to either $(\boldsymbol{\mu}^d||\mathbf{0}, 0)$ if $\mathbf{u}$ is encrypted, or $(\boldsymbol{\nu}^d||\mathbf{0}, 0)$ if $\mathbf{v}$ is encrypted, with $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ defined as $\boldsymbol{\chi}$ in Equation 1 but replacing x_i with u_i or v_i respectively.

Set the vector $\mathbf{X}$'s in hybrid H^b_ρ. Hybrid H^b_ρ uses the following set of vectors $\mathbf{X}$'s, which leverages the "space" of the additional zeros to satisfy the above desiderata.

$$\mathbf{X}^1_i = \begin{cases} \mathbf{0} \,||\, \boldsymbol{\nu}^1_i & \text{if } i < \rho \\ \boldsymbol{\mu}^1_i \,||\, \mathbf{0} & \text{if } i > \rho \\ \mathbf{0} \,||\, \mathbf{0} & \text{if } i = \rho \end{cases},\ \begin{cases} 0 & \text{if } i < \rho \\ 0 & \text{if } i > \rho \\ 1 & \text{if } i = \rho \end{cases} \qquad \mathbf{X}^2_j = \boldsymbol{\mu}^2_j || \boldsymbol{\nu}^2_j,\ \begin{cases} \left\langle \boldsymbol{\mu}^1_\rho, \boldsymbol{\mu}^2_j \right\rangle & \text{in } H^0_\rho \\ \left\langle \boldsymbol{\nu}^1_\rho, \boldsymbol{\nu}^2_j \right\rangle & \text{in } H^1_\rho \end{cases} \tag{4}$$

Let us first see how the challenge ciphertext-piece $\mathsf{iCT}[\rho,\star]$ is hardwired. Observe that the last slots of $\mathbf{X}^2_j$'s contain exactly the values encoded in $\mathsf{iCT}[\rho,\star]$: In H^0_ρ, they are set to $\{\left\langle \boldsymbol{\mu}^1_\rho, \boldsymbol{\mu}^2_j \right\rangle = rs^1_\rho s^2_j + u_\rho u_j\}_{j\in[N]}$ (see Equation 2), corresponding to encrypting $u_\rho \mathbf{u}$, while in H^1_ρ, they are set to $\{\left\langle \boldsymbol{\nu}^1_\rho, \boldsymbol{\nu}^2_j \right\rangle = rs^1_\rho s^2_j + v_\rho v_j\}_j$, encrypting $v_\rho \mathbf{v}$. By the fact that **cIPE** encodes its function vectors, $\mathbf{X}^2_j$'s here, in a bilinear source group, $\left[\mathbf{X}^2_j\right]$ is effectively embedded in cSK_j's and hence so is $\mathsf{iCT}[\rho,\star]$. Next, we check that the **IPE** ciphertext derived by decrypting every pair $(\mathsf{cCT}_i, \mathsf{cSK}_j)$ indeed encrypts the right hybrid vector.

$$\mathsf{cIPE.Dec}(\mathsf{cSK}_j, \mathsf{cCT}_i) = \left[\left\langle \mathbf{X}^1_i, \mathbf{X}^2_j \right\rangle\right] = \left[\begin{cases} \left\langle \mathbf{0} \,||\, \boldsymbol{\nu}^1_i \,||\, 0,\ \boldsymbol{\mu}^2_j \,||\, \boldsymbol{\nu}^2_j \,||\, \star \right\rangle = \left\langle \boldsymbol{\nu}^1_i,\ \boldsymbol{\nu}^2_j \right\rangle & \text{if } i < \rho \\ \left\langle \boldsymbol{\mu}^1_i \,||\, \mathbf{0} \,||\, 0,\ \boldsymbol{\mu}^2_j \,||\, \boldsymbol{\nu}^2_j \,||\, \star \right\rangle = \left\langle \boldsymbol{\mu}^1_i,\ \boldsymbol{\mu}^2_j \right\rangle & \text{if } i > \rho \\ \left\langle \mathbf{0} \,||\, \mathbf{0} \,||\, 1,\ \boldsymbol{\mu}^2_j \,||\, \boldsymbol{\nu}^2_j \,||\, \star \right\rangle = \quad \star & \text{if } i = \rho \end{cases} \right]$$

In the case $i = \rho$, $\mathsf{iCT}[\rho,\star]$ encodes exactly the values hardwired in the last slot, which as argued above encrypts $u_\rho \mathbf{u}$ in H^0_ρ and $v_\rho \mathbf{v}$ in H^1_ρ as desired. In the case $i < \rho$, the derived ciphertext-piece $\mathsf{iCT}[i,\star]$ encodes values $\{\left\langle \boldsymbol{\nu}^1_i, \boldsymbol{\nu}^2_j \right\rangle\}_j$, corresponding to encrypting $v_i \mathbf{v}$; and similarly, when $i > \rho$, the ciphertext-piece $\mathsf{iCT}[i,\star]$ encrypts $u_i \mathbf{u}$ as desired. Therefore, all desiderata above are satisfied.

Now, to show the security of **qFE**, it suffices to argue that every pair of neighboring hybrids is indistinguishable. Note that the only difference between different hybrids lies in the values of the $\mathbf{X}$ vectors encoded in the ciphertexts and secret keys of **cIPE**. Observe first that in hybrids H^1_ρ and $H^0_{\rho+1}$, every pair of vectors $(\mathbf{X}^1_i, \mathbf{X}^2_j)$ produce the *same* inner products, and hence the indistinguishability of H^1_ρ and $H^0_{\rho+1}$ follows immediately from the function hiding property of **cIPE**. This is, however, not the case in hybrids H^0_ρ and H^1_ρ, where for the special index ρ, the challenge ciphertext-piece change from encrypting $u_\rho \mathbf{u}$ to $v_\rho \mathbf{v}$. Next, we show how to reduce the indistinguishability of H^0_ρ and H^1_ρ to the security of the ABCP IPE scheme, which turns out to be quite tricky.

Overcoming Challenge 2: Indistinguishability of H^0_ρ and H^1_ρ from IPE security The goal is relying on the security of **IPE** to argue that the embedded challenge ciphertext-pieces in H^0_ρ and H^1_ρ are indistinguishable, and hence so are the hybrids. But, we immediately encounter a problem: The function outputs obtained when decrypting the derived ciphertext iCT using secret keys SK_j's are different in H^0_ρ and H^1_ρ, namely

$$\left\langle v_1\mathbf{v}||\cdots||v_{\rho-1}\mathbf{v}||\underline{u_\rho\mathbf{u}}||u_{\rho+1}\mathbf{u}||\cdots||u_N\mathbf{u}, \ , \ \mathbf{c}_j\right\rangle \neq \left\langle v_1\mathbf{v}||\cdots||v_{\rho-1}\mathbf{v}||\underline{v_\rho\mathbf{v}}||u_{\rho+1}\mathbf{u}||\cdots||u_N\mathbf{u}\ , \ \mathbf{c}_j\right\rangle \ .$$

This means the hybrids are clearly distinguishable. To fix this, we modify our **qFE** scheme to build in an offset value Δ in its secret keys, which will be added to the decryption output. In the honest execution, the offsets are set to zero, whereas in hybrid H^b_ρ, they are set to $\Delta^b_j(\rho)$ in each secret key SK_j, so that, the above inner products when added with $\Delta^0_j(\rho)$ in the left hand side and $\Delta^1_j(\rho)$ in the right hand side become equal. Clearly, whether the offset values Δ are used (set to non-zero) at all and their values must be hidden, we do so by encoding it using **cIPE**, as described below.

VERSION 3 OF OUR SECRET KEY QUADRATIC FE SCHEMES **qFE**

- SETUP: A master secret key $\mathsf{msk} = (\mathbf{s}^1, \mathbf{s}^2, \mathsf{cMSK}')$ contains additionally a master secret key cMSK' of **cIPE**.
- KEY GENERATION: In the secret key $\mathsf{SK}(\mathbf{c})$, the inner product $\left\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c}\right\rangle$ is now encoded, together with an offset value Δ, using cMSK' of **cIPE** as below.

$$\mathsf{SK}(\mathbf{c}) = \left(\underline{\mathsf{cSK}'\left(\left\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c}\right\rangle ||\Delta = 0 \right)}, \ \mathbf{c} \right) \ .$$

- ENCRYPTION: In the ciphertext $\mathsf{CT}(\mathbf{x})$, the random scalar r is now encrypted, with an additional 0, using cMSK' of **cIPE** as below.

$$\mathsf{CT}(\mathbf{x}) = \left(\underline{\mathsf{cCT}'(-r||0)}, \ \left\{\mathsf{cCT}_i(\mathbf{X}^2_j)\right\}_{i\in[N]}, \ \left\{\mathsf{cSK}_j(\mathbf{X}^2_j)\right\}_{j\in[N]} \right)$$

- DECRYPTION: Decryption proceeds as before, except that now encoding Λ_1 is obtained by decrypting cCT' using cSK', which yields $\left[-r\left\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c}\right\rangle + \Delta\right]$ as desired.

With the new offset value in secret key, we can now fix our hybrids so that the function outputs always stay the same.

Set the offsets in hybrid H^b_ρ. In hybrid H^b_ρ, not only that the vectors $\mathbf{X}$'s are set differently as above, the **cIPE** ciphertext cCT' in ciphertext CT encrypts $(0||1)$ instead of $(-r||0)$ and the corresponding **cIPE** secret key cSK'_j in SK_j encodes vector $(\left\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c}\right\rangle \ || \ r\left\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c}\right\rangle + \Delta^b_j(\rho))$, instead of $(\left\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c}\right\rangle ||0)$. At decryption time, the offset $\Delta^b_j(\rho)$ is added to the inner product between $\mathbf{c}_j$ and hybrid vector underlying iCT. Setting $\Delta^b_j(\rho)$ appropriately ensures that

$$\begin{aligned}&\left\langle v_1\mathbf{v}||\cdots||v_{\rho-1}\mathbf{v}||\underline{u_\rho\mathbf{u}}||u_{\rho+1}\mathbf{u}||\cdots||u_N\mathbf{u}, \ , \ \mathbf{c}_j\right\rangle + \Delta^0_j(\rho) \\ &\qquad = \left\langle v_1\mathbf{v}||\cdots||v_{\rho-1}\mathbf{v}||\underline{v_\rho\mathbf{v}}||u_{\rho+1}\mathbf{u}||\cdots||u_N\mathbf{u}\ , \ \mathbf{c}_j\right\rangle + \Delta^1_j(\rho) = f_{\mathbf{c}}(\mathbf{u}) \ .\end{aligned}$$

Now H^0_ρ and H^1_ρ have the same function outputs. But, to formally reduce their indistinguishability to the security of **IPE**, we need a way to incorporate the offsets Δ's into the challenge **IPE** ciphertexts. We do so by viewing Δ_j's as extension of the plaintext. More specifically, we implicitly switch from encrypting $\mathbf{U} = u_\rho \mathbf{u} || \Delta_1^0(\rho) || \cdots || \Delta_L^0(\rho)$ to $\mathbf{V} = v_\rho \mathbf{v} || \Delta_1^1(\rho) || \cdots || \Delta_L^1(\rho)$ using master secret key $\mathbf{S} = s_\rho^1 \mathbf{s}^2 || t_1 || \cdots || t_L$, at the presence of secret keys for vectors $\mathbf{Y}_j = \{\mathbf{c}_j[\rho, \star] || e_j\}_j$, where L is the total number of keys, t_j's are implicitly sampled secret key elements, and e_j is the unit vector of length L with a single one at index j. Observe that from such ciphertexts and secret keys, one can extract the challenge ciphertext-piece $\mathsf{iCT}[\rho, \star]$ encrypting $u_\rho \mathbf{u}$ or $v_\rho \mathbf{v}$, and obtain an encoding of $-r \left\langle \mathbf{s}^1 \otimes \mathbf{s}^2, \mathbf{c} \right\rangle + \Delta_j^b(\rho)$ embedded in each secret key cSK'_j — these are the only parts that hybrids H_ρ^0 and H_ρ^1 differ at. Given that $\langle \mathbf{U}, \mathbf{Y}_j \rangle = \langle \mathbf{V}, \mathbf{Y}_j \rangle$ for every j, we are almost done: Apply the security of $\mathbf{IPE}$ to argue that H_ρ^0 and H_ρ^1 are indistinguishable, except that we must overcome one last hurdle — the master secret key for encrypting $u_i \mathbf{u}$ or $v_i \mathbf{v}$ is not truely random.

Pseudorandomness from SXDH The master secret key of the challenge ciphertext-piece is $s_\rho^1 \mathbf{s}^2$. It is not truely random since $\mathbf{s}^2$ is also used for generating the master secret keys of other ciphertext-pieces. But, observe that both the challenge ciphertext-piece and $\mathbf{s}^2$ are embedded in secret keys $\{\mathsf{cSK}_j\}$, and hence encoded in the same bilinear map source group. Furthermore, thanks to the fact that in H_ρ^b, the ρth ciphertext cCT_ρ encrypts the vector $(\mathbf{0} || \mathbf{0}, 1)$, the key element s_ρ^1 does not appear in the other source group. Therefore, we can apply the SXDH assumption to argue that encodings of $s_\rho^1 \mathbf{s}^2$ is indistinguishable to that of a truly random vector $\mathbf{w}$ — in other words, the master secret key $s_\rho^1 \mathbf{s}^2$ is pseudorandom, *inside encodings*. Therefore, the security of $\mathbf{IPE}$ applies, and we conclude that hybrid H_ρ^0 and H_ρ^1 are indistinguishable.

2.3 Degree-D Secret-Key FE

Generalizing from quadratic FE to degree-D secret key FE, the natural idea is again starting from the trivial $\mathbf{IPE}$-based construction that encrypts all degree-D monomials, denoted as $\otimes \mathbf{x}^{\leq D} = \otimes_{d \in [D]} \mathbf{x}$, and compressing the N^D-size ciphertext into linear size. Naturally, instead of compressing a ciphertext generated using a truly random master secret key, we will use a structured master secret key $\otimes \mathbf{s}^{\leq D} = \otimes_{d \in [D]} \mathbf{s}^i$. Thus the $\mathbf{IPE}$ ciphertext to be compressed looks like:

$$\mathsf{iCT}[0] = [-r], \qquad \mathsf{iCT}[I_1, \cdots, I_d] = \left[r s_{I_1}^1 \cdots s_{I_D}^D + x_{I_1} \cdots x_{I_D} \right]$$

The challenge is how to generate the N^D encodings $\mathsf{iCT}[I]$ from just linear-sized information?

Key Tool: High-Degree IPE We generalize IPE to the notion of high-degree IPE, or HIPE for short. More precisely, a degree-D HIPE is a *multi-input* functional encryption scheme for degree-D inner product defined as follows,

$$\left\langle \mathbf{x}^1, \cdots, \mathbf{x}^D \right\rangle = \Sigma_{i \in [N]} x_i^1 x_i^2 \cdots x_i^D$$

Introduced by [GGG+14], a multi-input functional encryption allows one to encrypt inputs at different coordinates, and generate secret keys associated with multi-input functions, so that, decryption computes the output of the function evaluated on inputs encrypted at different coordinates. In the context of HIPE, a degree-D HIPE encryption scheme $\mathbf{hIPE}$ allows one to generate a ciphertext $\mathsf{hCT}^d(\mathbf{x}^d)$ encrypting an input vector $\mathbf{x}^d$ at a coordinate $d \in [D-1]$, and a secret key $\mathsf{hSK}(\mathbf{x}^D)$ at coordinate D, so that, decryption reveals whether the degree-D inner product $\left\langle \mathbf{x}^1 \cdots \mathbf{x}^D \right\rangle$ is zero or not. Under this generalization, standard IPE is a special case of HIPE for degree $D = 2$.

In terms of security, the notion of function hiding also generalizes naturally, an HIPE scheme is function hiding, if ciphertexts and secret keys $\{\mathsf{hCT}_i^1, \cdots \mathsf{hCT}_i^{D-1}, \mathsf{hSK}_i\}_{i \in [L]}$ encrypting one set of vectors $\{\mathbf{u}_i^1, \cdots \mathbf{u}_i^{D-1}, \mathbf{u}_i^D\}_{i \in [L]}$ or another $\{\mathbf{v}_i^1, \cdots \mathbf{v}_i^{D-1}, \mathbf{v}_i^D\}_{i \in [L]}$ are indistinguishable, whenever all degree-D inner products that can be computed from them are identical, that is,

$$\forall I \in [L]^D, \langle \mathbf{u}^1_{I_1}, \cdots \mathbf{u}^D_{I_D} \rangle = \langle \mathbf{v}^1_{I_1}, \cdots \mathbf{v}^D_{I_D} \rangle$$

In this work, we give a construction of function hiding degree-D HIPE scheme from the SXDH assumption on degree-D multilinear maps. Our construction starts from a canonical function hiding IPE scheme (for $D = 2$), and inductively build degree-$(D+1)$ HIPE scheme, by composing a degree-D HIPE scheme and a special-purpose function hiding IPE scheme. Our HIPE schemes have *canonical form* (similar to the canonical form described above for standard IPE), in the sense that ciphertexts at coordinate d (or secret keys) consist of encodings in the d^{th} (or D^{th} respectively) MMap source group, and decryption simply uses degree-D pairing to produce an encoding of the degree-D inner product. That is,

$$\mathsf{hIPE.Dec}(\mathsf{hSK}(\mathbf{x}^D), \mathsf{hCT}^1(\mathbf{x}^1), \cdots, \mathsf{hCT}^D(\mathbf{x}^{D-1})) = [\langle \mathbf{x}^1, \cdots, \mathbf{x}^D \rangle]$$

From Degree-D HIPE to Degree-D FE HIPE serves perfectly for our goal of compressing the ciphertext iCT. Generalizing $\mathbf{qFE}$, our degree-D FE scheme $\mathbf{dFE}$ generates ciphertexts that look as follows:

$$\mathsf{CT}(\mathbf{x}) = \left(\mathsf{cCT}'(-r||0),\ \left\{\mathsf{cCT}^1_i(\mathbf{X}^1_i),\ \cdots,\ \mathsf{cCT}^{D-1}_i(\mathbf{X}^{D-1}_i),\ \mathsf{cSK}_i(\mathbf{X}^D_i)\right\}_{i\in[N]} \right),$$
$$\text{where } \mathbf{X}^d_i = \boldsymbol{\chi}^d_i||\mathbf{0} \text{ and } \boldsymbol{\chi}^d_i = \begin{cases} x_i||s^d_i & \text{if } d < D \\ x_i||rs^D_i & \text{if } d = D \end{cases}$$

From such a ciphertext, a decryptor can "expand" out a size-N^D $\mathbf{IPE}$ ciphertext iCT by decrypting every combination of HIPE ciphertexts and secret keys. Namely, for every $I \in [N]^D$,

$$\mathsf{hIPE.Dec}(\mathsf{cCT}^1_{I_1},\ \cdots,\ \mathsf{cCT}^{D-1}_{I_{D-1}},\ \mathsf{cSK}_{I_D}) = [\langle \mathbf{X}^1_{I_1}, \cdots, \mathbf{X}^D_{I_D} \rangle] = \left[r \prod_{d\in[D]} s^d_{I_d} + \prod_{d\in[D]} x_{I_d} \right] = \mathsf{iCT}[I]$$

where $\mathsf{iCT}[I]$ encrypts the I^{th} degree-D monomial $\prod_{d\in[D]} x_{I_d}$, using the I^{th} key element $\prod_{d\in[D]} s^d_{I_d}$.

To show security of $\mathbf{dFE}$, we, again, switch the degree-D monomials encrypted in the $\mathbf{IPE}$ ciphertext iCT in piecemeal. In each step, we can still only embed a size-N ciphertext-piece; naturally we embed $\mathsf{iCT}[\rho, \star]$ for a prefix $\rho \in [N]^{D-1}$ of length $D-1$. Thus, the N^D encrypted monomials are changed piece by piece in N^{D-1} steps, where in the ρ^{th} step, all monomials with index I smaller than ρ (*i.e.*, $I_{\le D-1} < \rho$) have already been switched to $\prod_{d\in[D]} v_{I_d}$, monomials with index I larger than ρ (*i.e.*, $I_{\le D-1} > \rho$) remain to be $\prod_{d\in[D]} u_{I_d}$, and monomials with index I that agrees with ρ (*i.e.*, $I_{\le D-1} = \rho$) are being switched from $\prod_{d\in[D]} u_{I_d}$ in H^0_ρ to $\prod_{d\in[D]} v_{I_d}$ in H^1_ρ.

Creating a sequence of hybrids that carry out these steps is more complex than the case for degree 2. First, we need more space in the ciphertext to make sure that the right monomials are encrypted for every index I; thus, the vectors $\mathbf{X}$'s are padded to length $2D-1$. Second, it becomes significantly harder to argue that the key elements $(\prod_{d\in[D-1]} s^d_{\rho_d})\mathbf{s}^{\le D}$ are pseudorandom, as the shares s^d_i's are encoded in different MMap source groups, and unlike the degree 2 case, we cannot eliminate the appearance of all shares $\{s^d_{\rho_d}\}$ since they are also used for generating the master secret keys of other ciphertext-pieces (whereas in the degree 2 case, s^1_ρ is only used for generating $s^1_\rho \mathbf{s}^2$). To resolve this, we apply the SXDH assumption iteratively to gradually replace every partial product $\prod_{d\in[d^\star]} s^d_{\rho_d}$ with an independent and random element w^d_ρ, so that, the master secret keys for other ciphertext-pieces are generated using independent w elements.

Construction of HIPE As mentioned above, we inductively construct a degree-$(D+1)$ HIPE scheme $\mathbf{hIPE}$ by composing a degree-D HIPE scheme $\mathbf{dIPE}$ with a special purpose IPE scheme $\mathbf{sIPE}$. The construction of the degree-$(D+1)$ HIPE scheme resembles that of the degree-D FE scheme. The degree-D FE scheme $\mathbf{dFE}$ uses $\mathbf{dIPE}$ to compute a ciphertext of the ABCP scheme, which allows one to control what linear function is computed on the monomials. Now, in $\mathbf{hIPE}$, we use $\mathbf{dIPE}$ to compute instead a ciphertext of the special-purpose IPE scheme $\mathbf{sIPE}$ that encrypts a degree-D (coordinate-wise) product $\mathbf{x}^1 \cdots \mathbf{x}^D$, which can be further combined with a $\mathbf{sIPE}$ secret key of vector $\mathbf{x}^{D+1}$ to compute the degree-$(D+1)$ inner product, as $\langle \mathbf{x}^1, \cdots, \mathbf{x}^D, \mathbf{x}^{D+1} \rangle = \langle (\mathbf{x}^1 \cdots \mathbf{x}^D), \mathbf{x}^{D+1} \rangle$. More precisely, the ciphertexts $\mathsf{hCT}^1(\mathbf{x}^1), \cdots, \mathsf{hCT}(\mathbf{x}^D)$ and secret key $\mathsf{hSK}(\mathbf{x}^{D+1})$ of $\mathbf{hIPE}$ look as follows:

$$\mathsf{hSK}(\mathbf{x}^{D+1}) = \mathsf{sSK}(\mathbf{x}^{D+1})$$
$$\underbrace{\mathsf{hCT}^1(\mathbf{x}^1) = \mathsf{dCT}^1(\mathbf{x}^1), \ \cdots, \ \mathsf{hCT}^{D-1}(\mathbf{x}^{D-1}) = \mathsf{dCT}^{D-1}(\mathbf{x}^{D-1}), \quad \mathsf{hCT}(\mathbf{x}^D) = \mathsf{dSK}(\mathbf{x}^D)}_{\text{decrypts to } \mathsf{sCT}(\mathbf{x}^1 \cdots \mathbf{x}^D)}$$

where sCT and sSK are respectively a ciphertext and secret key of $\mathbf{sIPE}$. The security proof of $\mathbf{hIPE}$ also resembles that of $\mathbf{qFE}$. The main difference lies in the "inner" IPE scheme; for HIPE the "inner" IPE is has a two-slot structure resembling the slotted IPE introduced in [LV16]. See section 7.2 for a more detailed overview of the construction and security proof of our HIPE schemes.

2.4 Simple Function Hiding IPE

As described above, our construction of degree-D FE crucially relies on a *canonical* function hiding IPE. However, none of the known secret-key IPE schemes [BJK15, DDM16, LV16] have the canonical form, in particular, their decryption does not produce an encoding of the output inner product $[\langle \mathbf{x}, \mathbf{y} \rangle]$, but produce the inner product masked by a scalar $[\langle \mathbf{x}, \mathbf{y} \rangle \, \theta]$ together with $[\theta]$, where the scalar θ is determined by the randomness used in key generation and encryption. In this work, we give a construction of a *canonical* function hiding IPE. Our construction is extremely simple and of independent interests. Its description and security proof fit within 2 pages, and we now summarize the idea of the construction in one paragraph.

Lin and Vaikuntanathan [LV16] give a simple transformation from IPE with weak function hiding to IPE with full function hiding. Our construction starts from the ABCP public key IPE scheme, whose secret key for a vector $\mathbf{y}$ reveals $\mathbf{y}$ and its inner product with the master secret key $\langle \mathbf{s}, \mathbf{y} \rangle$ in the clear. To achieve weak function hiding, we need to hide $\mathbf{y}$. Our idea is to simply encrypt the secret key as an input vector using the ABCP scheme itself, with an independently sampled master secret key $\mathbf{s}'$ of length $N+1$, which yields the new secret key $\mathsf{iSK}' = [r'\mathbf{s}' \ + \ (\langle \mathbf{s}, \mathbf{y} \rangle \, || \mathbf{y})]$. Recall that decryption of the ABCP scheme simply computes (homomorphically) the inner product between its secret key and ciphertext. Now that the original secret key is encrypted, we correspondingly encode the original ciphertext in a secret key using $\mathbf{s}'$, which gives the new ciphertext $\mathsf{iCT}' = [\langle \mathbf{s}' \, , \, (r\mathbf{s} + \mathbf{x}) \rangle \ || \ (r\mathbf{s} + \mathbf{x})]$. Computing the inner product of iCT' and iSK' simultaneously decrypts both "layers" of ABCP encryption, and produce exactly an encoding of the output inner product.

3 Preliminaries

Let $\mathbb{Z}$ and $\mathbb{N}$ denote the set of integers, and positive integers, respectively. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. We use $\mathcal{R}$ to denote either a ring, or an ensemble of rings $\mathcal{R} = \{\mathcal{R}_\lambda\}$, which will be clear in the context.

We denote by PPT probabilistic polynomial time Turing machines. The term *negligible* is used for denoting functions that are (asymptotically) smaller than any inverse polynomial. More precisely, a function $\nu(\star)$ from non-negative integers to reals is called *negligible* if for every constant $c > 0$ and all sufficiently large n, it holds that $\nu(n) < n^{-c}$.

We use boldface to denote vectors, for example, $\mathbf{u}, \mathbf{v}, \mathbf{c}$ etc., and use u_i, v_i, c_i to denote the i^{th} elements in the vectors.

3.1 μ-Indistinguishability

Definition 1 (μ-indistinguishability). *Let $\mu : \mathbb{N} \to [0,1]$ be a function. A pair of distribution ensembles $\{X_\lambda\}_{\lambda\in\mathbb{N}}$, $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are μ-indistinguishable if for every family of polynomial-sized distinguishers $\{D_\lambda\}_{\lambda\in\mathbb{N}}$, and every sufficiently large security parameter $\lambda \in \mathbb{N}$, it holds that*

$$|\Pr[x \xleftarrow{\$} X_\lambda : D(1^\lambda, x, z) = 1] - \Pr[y \xleftarrow{\$} Y_\lambda : D(1^\lambda, y, z) = 1]| \leq \mu(\lambda)$$

Definition 2 (Computational and Sub-exponential Indistinguishability). *A pair of distribution ensembles $\{X_\lambda\}_{\lambda\in\mathbb{N}}$, $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable if they are $1/p$-indistinguishable for every polynomial p, and are sub-exponentially indistinguishable if they are μ-indistinguishable for some sub-exponentially small $\mu(\lambda) = 2^{\lambda^\varepsilon}$ with a constant $\varepsilon > 0$.*

Note that the above definition of sub-exponential indistinguishability is weaker than standard sub-exponential hardness assumptions that consider distinguishers running in sub-exponential *time*.

Below, we provide definitions of standard cryptographic primitives using the terminology of μ-indistinguishability, which implicitly defines variants with polynomial or sub-exponential security. As a matter of convention, we will drop μ when μ is a negligible function, and say sub-exponential security when μ is a sub-exponentially small function.

3.2 Indistinguishability Obfuscation

We recall the notion of indistinguishability obfuscation for a class of circuit defined by [BGI+01b].

Definition 3 (Indistinguishability Obfuscator ($i\mathcal{O}$) for a circuit class). *A uniform* PPT *machine $i\mathcal{O}$ is an indistinguishability obfuscator for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, if the following conditions are satisfied:*

Correctness: *For all security parameters $\lambda \in \mathbb{N}$, for every $C \in \mathcal{C}_\lambda$, and every input x, we have that*

$$\Pr[C' \leftarrow i\mathcal{O}(1^\lambda, C) \ : \ C'(x) = C(x)] = 1$$

where the probability is taken over the coin-tosses of the obfuscator $i\mathcal{O}$.

μ-Indistinguishability: *For every ensemble of pairs of circuits $\{C_{0,\lambda}, C_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ satisfying that $C_{b,\lambda} \in \mathcal{C}_\lambda$, $|C_{0,\lambda}| = |C_{1,\lambda}|$, and $C_{0,\lambda}(x) = C_{1,\lambda}(x)$ for every x, the following ensembles of distributions are μ-indistinguishable:*

$$\left\{C_{1,\lambda}, C_{2,\lambda}, i\mathcal{O}(1^\lambda, C_{1,\lambda})\right\}_{\lambda\in\mathbb{N}}$$
$$\left\{C_{1,\lambda}, C_{2,\lambda}, i\mathcal{O}(1^\lambda, C_{2,\lambda})\right\}_{\lambda\in\mathbb{N}}$$

Definition 4 (IO for P/poly). *A uniform* PPT *machine $i\mathcal{O}_{\mathsf{P/poly}}(\star, \star)$ is an indistinguishability obfuscator for* P/poly *if it is an indistinguishability obfuscator for the class $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ of circuits of size at most λ.*

3.3 Pseudorandom Generator

Definition 5 (Pseudo-Random Generator (PRG)). *Let ℓ be a polynomial-bounded function. A deterministic polynomial-time uniform machine $\mathbf{PRG}$ is a $\ell(\lambda)$-stretch pseudorandom generator if the following conditions are satisfied:*

Syntax *For every $\lambda \in \mathbb{N}$ and every $r \in \{0,1\}^\lambda$, $\mathbf{PRG}(r)$ outputs $r' \in \{0,1\}^{\ell(\lambda)}$*

μ-Indistinguishability: *The following ensembles are μ-indistinguishable*

$$\left\{ r \xleftarrow{\$} \{0,1\}^\lambda : \mathbf{PRG}(r) \right\}_{\lambda \in \mathbb{N}} \approx_\mu \left\{ r' \xleftarrow{\$} \{0,1\}^{\ell(\lambda)} \right\}_{\lambda \in \mathbb{N}}$$

We defining the locality of PRGs and the degree of PRGs in a family of rings.

Definition 6 (Locality and degree of PRGs). *Let $\mathbf{PRG} : \{0,1\}^* \to \{0,1\}^*$ be an $\ell(n)$-stretch pseudorandom generator. For every $\lambda \in \mathbb{N}$, and every polynomial n, let $\mathbf{PRG}_{n(\lambda)} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(n(\lambda))}$ denote the binary function corresponding to $\mathbf{PRG}$ for $n(\lambda)$-bit inputs. We define the following parameters w.r.t. $\mathbf{PRG}$:*

- *$\mathbf{PRG}$ has locality L (for a universal constant L) if for every $\lambda \in \mathbb{N}$ and every polynomial n, every output bit of $\mathbf{PRG}_{n(\lambda)}$ depends on at most L input bits.*
- *$\mathbf{PRG}$ has $\mathcal{R}$-degree D (for a universal constant D), w.r.t. a family of rings $\mathcal{R} = \{\mathcal{R}_\lambda\}$, if for every $\lambda \in \mathbb{N}$ and every polynomial n, every output bit of $\mathbf{PRG}_{n(\lambda)}$ can be computed by a degree-D polynomial in $\mathcal{R}_\lambda$.*

Since the PRGs we consider are *binary*, mapping binary input strings to binary output strings, it holds that its locality upper bounds its degree in any ring family.

Fact 1. *For any pseudorandom generator $\mathbf{PRG} : \{0,1\}^* \to \{0,1\}^*$, and any family of rings $\mathcal{R}$, the degree of $\mathbf{PRG}$ w.r.t. $\mathcal{R}$ is no larger than its locality.*

3.4 Randomized Encodings

In this section, we recall the traditional definition of randomized encodings with simulation security [IK02, AIK06].

Definition 7 (Randomized encoding scheme for circuits). *A randomized encoding scheme $\mathbf{RE}$ consists of two PPT algorithms,*

- *$\hat{C}_x \xleftarrow{\$} \mathsf{REnc}(1^\lambda, C, x)$: On input a security parameter 1^λ, circuit C, and input x, REnc generates an encoding $\hat{C}_x$.*
- *$y = \mathsf{REval}(\hat{C}_x)$: On input $\hat{C}_x$ produced by REnc, REval outputs y.*

Correctness: *The two algorithms REnc and REval satisfy the following correctness condition: For all security parameters $\lambda \in \mathbb{N}$, circuit C, input x, it holds that,*

$$\Pr[\hat{C}_x \xleftarrow{\$} \mathsf{REnc}(1^\lambda, C, x) : \mathsf{Eval}(\hat{C}_x) = C(x)] = 1$$

μ**-Simulation Security:** *There exists a PPT algorithm* RSim*, such that, for every ensemble* $\{C_\lambda, x_\lambda\}_\lambda$ *where* $|C_\lambda|, |x_\lambda| \leq \mathrm{poly}(\lambda)$*, the following ensembles are* μ*-indistinguishable for all* $\lambda \in N$*.*

$$\left\{\hat{C}_x \xleftarrow{\$} \mathsf{REnc}(1^\lambda, C, x) : \hat{C}_x\right\}_{\lambda\in\mathbb{N}}$$
$$\left\{\hat{C}_x \xleftarrow{\$} \mathsf{RSim}(1^\lambda, C(x), 1^{|C|}, 1^{|x|}) : \hat{C}_x\right\}_{\lambda\in\mathbb{N}}$$

where $C = C_\lambda$ *and* $x = x_\lambda$*.*

Furthermore, let $\mathcal{C}$ *be a complexity class, we say that randomized encoding scheme* $\mathbf{RE}$ *is in* $\mathcal{C}$*, if the encoding algorithm* REnc *can be implemented in that complexity class.*

3.5 Functional Encryption

We provide the definition of a public-key functional encryption (FE) scheme with indistinguishability-based security which originally appeared in [BSW12, O'N10]. Below we define public key FE first, and then note the difference with secret key FE.

3.5.1 Public-Key Functional Encryption

Syntax Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles of sets. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where every function in the set $\mathcal{F}_\lambda$ maps inputs in $\mathcal{X}_\lambda$ to outputs in $\mathcal{Y}_\lambda$.

A public-key functional encryption scheme $\mathbf{FE}$ for $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of four PPT algorithms ($\mathsf{FE.Setup}$, $\mathsf{FE.KeyGen}$, $\mathsf{FE.Enc}$, $\mathsf{FE.Dec}$).

- *Setup:* $\mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ is an algorithm that on input a security parameter and some public parameter (*e.g.*, description of bilinear pairing groups) outputs a master public key and a master secret key $(\mathsf{mpk}, \mathsf{msk})$.
- *Key Generation:* $\mathsf{FE.KeyGen}(\mathsf{msk}, f)$ on input the master secret key msk and the description of a function $f \in \mathcal{F}_\lambda$, outputs a secret key SK_f.
- *Encryption:* $\mathsf{FE.Enc}(\mathsf{mpk}, x)$ on input the master public key mpk and a message $x \in \mathcal{X}_\lambda$, outputs an encryption CT of x.
- *Decryption:* $\mathsf{FE.Dec}(\mathsf{SK}, \mathsf{CT})$ on input the secret key associated with f and an encryption of x, outputs $y \in \mathcal{Y}_\lambda$.

Correctness: We define perfect correctness here. For every λ, $f \in \mathcal{F}_\lambda$, $x \in \mathcal{X}_\lambda$, it holds that,

$$\Pr\left[\begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp}) \\ \mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{mpk}, x) \\ \mathsf{SK} \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{msk}, f) \end{array} : f(x) = \mathsf{FE.Dec}(\mathsf{SK}, \mathsf{CT})\right] = 1$$

Indistinguishability Security. Indistinguishability security of a functional encryption requires that no adversary can distinguish the FE encryption of one input x_0 from that of another x_1, if the adversary only obtains secret keys for functions that yield the same outputs on x_0 and x_1, that is, for every secret key SK_f, it holds that $f(x_0) = f(x_1)$. In the adaptive setting, the two challenge inputs (x_0, x_1) and all functions f are chosen adaptively by the adversary. In the weaker selective setting, the adversary is restricted to choose (x_0, x_1) and all functions f statically.

Definition 8 (IND-security)**.** *A public-key FE scheme* $\mathbf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *for* $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* μ-IND-secure*, if for every* PPT *adversary* A*, and every sufficiently large security parameter* $\lambda \in \mathbb{N}$*, the adversary's advantage in the following games is bounded by* $\mu(\lambda)$

$$\mathsf{Advt}_A^{\mathbf{FE}} = \left|\Pr[\mathsf{IND}_A^{\mathbf{FE}}(1^\lambda, 0) = 1] - \Pr[\mathsf{IND}_A^{\mathbf{FE}}(1^\lambda, 1) = 1]\right| \le \mu(\lambda)$$

$\mathsf{IND}_A^{\mathbf{FE}}(1^\lambda, b)$ *proceeds as follows:*

1. **Key Generation.** *The challenger* CH *samples* $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ *and sends* mpk *to the adversary.*
2. **Function Queries.** *Repeat the following for an arbitrary number of times determined by* A*: Upon* A *choosing a function query* $f \in \mathcal{F}_\lambda$*,* CH *sends* A *a function key* $\mathsf{SK}_f \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{msk}, f)$*.*
3. **Message Queries.** *Upon* A *choosing a pair of messages* (x_0, x_1)*,* CH *sends* A *a ciphertext* $\mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{mpk}, x_b)$*.*
4. **Function Queries** *Repeat the second step, for an arbitrary number of times determined by* A*.*
5. *Finally* A *outputs a bit* b' *which is also the output of the experiment.*

Restriction: *Every function query* f *must satisfy that* $f(x_0) = f(x_1)$*.*

Definition 9 (Selective security)**.** *We say that* **FE** *is* μ-selectively secure *if the condition in Definition 8 holds for modified experiments* $\mathsf{SIND}_A^{\mathbf{FE}}(1^\lambda, b)$ *where the adversaries choose challenge messages* (x_0, x_1) *and all function queries* $\{f\}$ *at the beginning of the experiment.*

Definition 10 (1-key FE)**.** *We say that* **FE** *is a* μ*-secure (or* μ*-selectively secure) 1-key FE scheme if it satisfies the security requirements in Definition 8 (or, respectively, Definition 9) against adversaries that ask for at most one function key query.*

3.5.2 Secret Key Functional Encryption

A secret key FE scheme (SKFE) $\mathbf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for a class of function $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ has the same syntax and correctness as a public key FE scheme, except that, the FE.Setup algorithm outputs only the master secret key $\mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$, and the encryption algorithm encrypts using the master secret key $\mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{msk}, x)$.

In terms of security, the same (adaptive or selective) indistinguishability security is considered, with a slight modification to the definitions above for public key FE that the attacker can (adaptively or selectively) request for arbitrarily many challenge ciphertexts of messages of his/her choice. In the literature, there is also a stronger notion of security for secret key FE, called *function hiding*, which roughly speaking requires the scheme to hide both information of the encrypted inputs, as well as, the functions encoded in secret keys. We now define the notion of function hiding.

Definition 11 (Function hiding). *A secret-key FE scheme* $\mathbf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *for* $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* μ-function-hiding, *if for every* PPT *adversary* A, *and every sufficiently large security parameter* $\lambda \in \mathbb{N}$, *the adversary's advantage in the following games is bounded by* $\mu(\lambda)$

$$\mathsf{Advt}_A^{\mathbf{FE}} = \left|\Pr[\mathsf{FH}_A^{\mathbf{FE}}(1^\lambda, 0) = 1] - \Pr[\mathsf{FH}_A^{\mathbf{FE}}(1^\lambda, 1) = 1]\right| \le \mu(\lambda)$$

$\mathsf{FH}_A^{\mathbf{FE}}(1^\lambda, b)$ *proceeds as follows:*

1. **Key Generation.** *The challenger* CH *samples* $\mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$.
2. *The challenger* CH *repeats the following with* A *for an arbitrary number of times determined by* A*:*
   - **Function Queries.** *Upon* A *choosing a pair of functions* $(f_0, f_1) \in \mathcal{F}_\lambda$, CH *sends* A *a function key* $\mathsf{SK}_f \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{msk}, f_b)$.
   - **Message Queries.** *Upon* A *choosing a pair of messages* (x_0, x_1), CH *sends* A *a ciphertext* $\mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{mpk}, x_b)$.
3. *Finally* A *outputs a bit* b' *which is also the output of the experiment.*

Restriction: *Every function query* (f_0, f_1) *and message query* x_0, x_1 *must satisfy that* $f_0(x_0) = f_1(x_1)$.

We can define selective function hiding similar to Definition 9, by restricting the above security definition to a class of adversaries that choose all input queries $\{(x_0, x_1)\}$ and all the function queries $\{(f_0, f_1)\}$ at the beginning of the experiment. We can also define 1-key secret-key FE as in Definition 10.

3.5.3 FE for P/poly, NC^1 and Compactness

Definition 12 (FE schemes for families of function classes). *Let* $\mathbb{F} = \{\mathcal{F}^I\}_{I\in\mathcal{I}}$ *be a family of function classes. We say that* $\mathcal{FE} = \{\mathbf{FE}^I\}_{I\in\mathcal{I}}$ *is a family of (1-key) FE schemes for* $\mathbb{F}$ *with (selective)* μ*-IND-security or* μ*-function-hiding if for every function class* $\mathcal{F}^I = \{\mathcal{F}^I_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathbf{FE}^I$ *is a (1-key) FE scheme for* $\mathcal{F}^I$ *with (selective)* μ*-IND-secure or* μ*-function hiding.*

Moreover, define the following special cases:

- FE for P/poly *is a family of FE schemes for* $\mathbb{F} = \{\mathcal{F}^{N,D,S}\}_{N\in\mathcal{N},D\in\mathcal{D},S\in\mathcal{S}}$, *where* $\mathcal{N}, \mathcal{D}, \mathcal{S}$ *are the sets of all polynomials and* $F^{N,D,S}$ *is the class of binary functions that can be computed by circuits with* $N(\lambda)$*-bit inputs,* $S(\lambda)$ *size, and* $D(\lambda)$ *depth.*
- FE for NC^1 *is a family of FE schemes for* $\mathbb{F} = \{\mathcal{F}^{N,D,S}\}_{N\in\mathcal{N},D\in\mathcal{D},S\in\mathcal{S}}$ *as defined above but with* $\mathcal{D}$ *the set of all logarithmic functions.*

Compactness In the above definition of families of FE schemes, algorithms in scheme $\mathbf{FE}^{N,D,S}$ could run in polynomial time depending on polynomials N, D, S. In the literature, stronger efficiency requirements have been considered. In particular, the works of [AJ15, BV15] defined compact FE schemes for NC^1, which requires the encryption time to be independent of the circuit size S of the functions.

Definition 13 (Compactness of FE schemes for NC^1). *Let* $\mathcal{FE} = \{\mathbf{FE}^{N,D,S}\}$ *be a family of FE schemes for* NC^1.

Compactness: *We say that the functional encryption scheme $\mathcal{FE}$ is compact if for every logarithmic function D, there is a polynomial p, such that, for every polynomials N, S, the encryption algorithm of $\mathbf{FE}^{N,D,S}$ runs in time $p(\lambda, N(\lambda), \log S(\lambda))$.*

$(1-\varepsilon)$**-Sublinear Compactness (a.k.a.** $(1-\varepsilon)$**-Weakly Compactness):** *We say that $\mathcal{FE}$ is $(1-\varepsilon)$-sublinearly compact, if for every logarithmic function D, there is a polynomial p, such that, for every polynomials N, S, the encryption algorithm of $\mathbf{FE}^{N,D,S}$ runs in time $p(\lambda, N(\lambda)) \cdot S(\lambda)^{1-\varepsilon}$.*

3.6 Zero-Testing FE for Arithmetic Functions

For any ring $\mathcal{R}$, we refer to functions mapping from $\mathcal{R}^*$ to $\mathcal{R}^*$ as arithmetic functions in $\mathcal{R}$. Many previous works (*e.g.* [ABCP15, BJK15]) constructed FE schemes for classes of arithmetic functions in $\mathcal{R}$ with a relaxed correctness guarantee, namely, decryption does not reveal the output (in $\mathcal{R}$) entirely, but only reveals whether the output is zero or not. We refer to this relaxed correctness guarantee as *zero-testing correctness*, and FE schemes with such relaxed correctness as *zero-testing FE*. We stress that though the correctness requirement is relaxed, the security requirements, namely IND-security and function hiding, remain the same. Therefore, zero-testing FE is strictly weaker than standard FE.

Definition 14 (Zero-testing FE). *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}$ be an ensemble of rings, and $\{\mathcal{F}_\lambda\}$ a class of functions where $\mathcal{F}_\lambda$ maps from $\mathcal{X}_\lambda \subseteq \mathcal{R}_\lambda^*$ to $\mathcal{Y}_\lambda \subseteq \mathcal{R}_\lambda^*$. We say that $\mathbf{FE}$ is a (1-key) zero-testing FE scheme for $\{\mathcal{F}_\lambda\}$ with (selective) μ-IND-security or μ-function hiding, if it is a FE scheme for $\{\mathcal{F}_\lambda\}$ with the same security guarantee as in Definition 8 or 11 (or 9) respectively, and the following relaxed correctness guarantee.*

- **Zero-Testing Correctness:** *For every λ, $f \in \mathcal{F}_\lambda$, $x \in \mathcal{X}_\lambda$, it holds that,*

$$\Pr\left[\begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp}) \\ \mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{mpk}, x) \\ \mathsf{SK} \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{msk}, f) \end{array} : \mathsf{ZT}(f(x)) = \mathsf{FE.Dec}(\mathsf{SK}, \mathsf{CT})\right] = 1$$

  where ZT is a predicate that outputs 1 iff its input is the zero element in $\mathcal{R}_\lambda$, and in the case of secret key FE, $\mathsf{mpk} = \mathsf{msk}$.

Zero-Testing FE for Degree-d Polynomials and Inner Products

Definition 15 (Zero-testing FE schemes for families of arithmetic function classes). *Let $\mathbb{F} = \{\mathcal{F}^I\}_{I \in \mathcal{I}}$ be a family of arithmetic function classes. A family $\mathcal{FE} = \{\mathbf{FE}^I\}_{I \in \mathcal{I}}$ of (1-key) zero-testing FE schemes for $\mathbb{F}$ is defined identically as in Definition 12 except that every scheme $\mathbf{FE}^I$ has zero-testing correctness.*

Moreover, define the following special cases:

- **Zero-testing FE for degree-d polynomials in** $\mathcal{R}$ *is a family of zero-testing FE schemes for $\mathbb{F} = \{\mathcal{F}^N\}$ where where $\mathcal{F}^N$ is the set of degree-d polynomials mapping from $\mathcal{R}_\lambda^{N(\lambda)}$ to $\mathcal{R}_\lambda$.*

- **Zero-testing FE for inner products in** $\mathcal{R}$ *is a family of zero-testing FE schemes for $\mathbb{F} = \{\mathcal{F}^N\}$ where $\mathcal{F}^N$ is the set of functions of form $f_{\mathbf{v}}(\mathbf{x}) = \langle \mathbf{v}, \mathbf{x} \rangle$ that compute the inner product between a fixed vector $\mathbf{v}$ and an input vector $\mathbf{x}$ in $\mathcal{R}_\lambda^{N(\lambda)}$. Such a family of schemes is also called zero-testing Inner Product Functional Encryption (IPE) in $\mathcal{R}$.*

Definition 16 (Linear efficiency)**.** *Let $\mathcal{FE} = \{\mathbf{FE}^N\}$ be a family of FE schemes for degree-d polynomials or inner products in $\mathcal{R}$. We say that $\mathcal{FE}$ has* **linear efficiency** *if there exists a polynomial function p, such that, for every polynomial N, the encryption algorithm of $\mathbf{FE}^N$ runs in time $N(\lambda)\,\mathrm{poly}(\lambda)$.*

In the rest of the paper, whenever we talk about FE for arithmetic functions, in particular, IPEs and FEs for degree-d polynomials, over a family of non-binary ring $\mathcal{R}$, we mean by default a zero-testing FE.

4 Degree-D Asymmetric Multilinear Maps with SXDH Assumption

Introduced by Boneh and Silverberg [BS02], asymmetric Multilinear Maps (MMaps) naturally generalize asymmetric bilinear maps to higher degree. Let $\mathcal{G}$ denote a group generator that on input 1^λ outputs $(p, G_1, \cdots, G_D,$
$G_{D+1}, \mathbf{pair})$, where $G_1, \cdots, G_D, G_{D+1}$ are cyclic groups with order p (prime or composite). G_1 to G_D are referred to as the source groups and G_{D+1} the target group. Assume without loss of generality that the description of the source groups contain generators $g_1, \cdots, g_D$ of $G_1, \cdots, G_D$. In addition, the following properties hold.

- *Admissible:* $\mathbf{pair} : G_1 \times \cdots \times G_D \to G_{D+1}$ is efficiently computable and $g_{D+1} = \mathbf{pair}(g_1 \cdots, g_D)$ generates G_{D+1}.
- *Multilinear:* For any $a_1, \cdots, a_D \in \mathbb{Z}_p$, $\mathbf{pair}(g_1^{a_1}, \cdots, g_D^{a_D}) = \mathbf{pair}(g_1, \cdots, g_D)^{a_1 a_2 \cdots a_D} = g_{D+1}^{a_1 a_2 \cdots a_D}$.

We denote by $\mathcal{R}_\lambda = (\mathbb{Z}_p, +, \times)$ the ring corresponding to the exponent space of these multilinear pairing groups.

The Bracket Notation For clarity of notions, we use the following bracket notations to denote group elements.

$$\forall l \in [D+1], \quad [a]_l = g_l^a$$

We refer to $[a]_l$ as an encoding of a in group G_l, or with label l. Under this notation, the generator in group $l \in [D+1]$ is represented as $[1]_l = g_l$. We also use the following vector notation to represent vectors of group elements succinctly: For any $\mathbf{v} = (v_1, \cdots, v_m) \in \mathbb{Z}_p^m$, and $l \in \{0, 1, T\}$:

$$[\mathbf{v}]_l = [v_1]_l \cdots [v_m]_l$$

Homomorphic Operations Using multiplication and exponentiation in each group, we can perform addition "$\oplus$" and scalar multiplication "$\odot$" to vectors encoded in the same group l. Formally, for any $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_p^m$, and $\alpha \in \mathbb{Z}_p$,

$$[\mathbf{v}]_l \oplus [\mathbf{w}]_l := [\mathbf{v} + \mathbf{w}]_l = ([v_1 + w_1]_l \cdots [v_m + w_m]_l)$$
$$\alpha \odot [\mathbf{v}]_l := ([\mathbf{v}]_l)^\alpha = [\alpha \mathbf{v}]_l = ([\alpha v_1]_l \cdots [\alpha v_m]_l)$$

In particular, this means we can homomorphically evaluate any linear function L in $\mathbb{Z}_p$, over encoded vectors. We conveniently write

$$L([\mathbf{v}]_l) = [L(\mathbf{v})]_l$$

Using the multilinear map **pair**, we can homomorphically compute any *multilinear* polynomial p with degree $\leq D$ over encoded vectors $\{\mathbf{v}_d\}_{d \in [D]}$, where $\mathbf{v}_d$ is encoded in G_d. This is because, one can first homomorphically compute every multilinear monomial in p using **pair** and obtain an encoding of the value of the monomial in the target group. (If a monomial has exactly degree D, **pair** directly applies; otherwise, one can raise the degree to D using encodings of 1 (*i.e.*, the generators) in appropriate groups.) Next, encodings of the values of all monomials in p can be homomorphically added in the target group to produce an encoding of the output in the target group. We conveniently write

$$p([\mathbf{v}_1]_1, \cdots, [\mathbf{v}_D]_D) = [p(\mathbf{v}_1, \cdots, \mathbf{v}_D)]_{D+1}$$

The SXDH Assumption The SXDH assumption states that the standard DDH assumption holds in each of the source groups. Formally, for every source group G_l for $l \in [D]$, the following two ensembles are μ-indistinguishable.

$$\left\{ \mathsf{pp} = (p, G_1, \cdots G_D, G_{D+1}, \mathbf{pair}) \xleftarrow{\$} \mathcal{G}(1^\lambda),\ a, b \xleftarrow{\$} \mathbb{Z}_p\ :\ (\ \mathsf{pp},\ [a]_l, [b]_l, [ab]_l\ ) \right\}_\lambda$$
$$\left\{ \mathsf{pp} = (p, G_1, \cdots, G_D, G_{D+1}, \mathbf{pair}) \xleftarrow{\$} \mathcal{G}(1^\lambda),\ a, b, r \xleftarrow{\$} \mathbb{Z}_p\ :\ (\ \mathsf{pp},\ [a]_l, [b]_l, [r]_l\ ) \right\}_\lambda$$

5 IO from Locality-L PRG and Degree-L FE

In this section, we review the bootstrapping theorem by Lin and Vaikuntanathan (LV) [LV16] that IO can be bootstrapped from subexponentially secure PRG in NC^0 and FE for NC^0, which in turn is based on [BV15, AJS15]. We observe that in their bootstrapping theorem, if the PRG in NC^0 has degree D in any ring $\mathcal{R}$, then it suffices to start with a FE scheme for degree-$(3D + 2)$ polynomials in the same ring $\mathcal{R}$. (See Definition 6 for the locality and degree of a PRG.) Since the locality of a binary PRG upper bounds its degree in any ring, we have that IO can be constructed from locality-L PRG and degree-$(3L+2)$ FE. Next, we modify their bootstrapping theorem to reduce the degree of polynomials that FE needs to support from $3L + 2$ to just L, exactly the locality of the $\mathbf{PRG}$.

5.1 IO from Degree-D PRG and Degree-$(3D + 2)$ FE

The following theorem follows from the bootstrapping theorem in [LV16].

Theorem 4 ([LV16])**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}$ be any family of rings and $\varepsilon > 0$ any positive constant. Assume the existence of a sub-exponentially secure PRG with $n^{1+\varepsilon}$-stretch and $\mathcal{R}$-degree D. Then, IO for* P/poly *is implied by either of the following:*

- *any selectively sub-exponential-IND-secure* public key *(zero-testing) FE for degree-$(3D + 2)$ polynomials in $\mathcal{R}$, with linear efficiency, or*
- *any selectively sub-exponential-IND-secure* secret key *(zero-testing) FE for degree-$(3D + 2)$ polynomials in $\mathcal{R}$ with linear efficiency, and the sub-exponential hardness of LWE.*

As discussed in the Overview section (Section 2), to construct IO for P/poly, the LV bootstrapping theorem first constructs a selectively subexponential-IND-secure single-key FE scheme with $(1-\varepsilon)$-sublinear compactness for NC^1 circuits, and then invoke the result of [AJ15, BV15] to further bootstrap such a NC^1-FE scheme to IO for P/poly in the public key case, or the result of [BNPW16] in the secret key case, assuming additionally the subexponential hardness of LWE.

In Section 2.1, we give an overview of the LV construction of sublinearly-compact NC^1-FE schemes, from PRG in NC^0 and collusion resistant FE schemes for NC^0 that has linear efficiency; we also discussed there that the LV NC^1-FE schemes can also be instantiated with collusion resistant FE schemes for degree-$(3D + 2)$ polynomials in some ring $\mathcal{R}$ if the PRG has degree-D in $\mathcal{R}$.

Using Degree-D PRG and Degree-$(3D+2)$ FE We now describe formally how to instantiate the LV construction of a degree-D PRG and degree-$(3D+2)$ FE scheme. We focus on the public key case; the secret key case follows identically. Their FE scheme $\mathbf{CFE}^{N,D,S}$ for NC^1 circuits with input-length $N = N(\lambda)$, depth $D = D(\lambda)$, and size $S = S(\lambda)$, uses the following tools: Let $\mathcal{R}$ be a family of rings.

- A pseudorandom generator $\mathbf{PRG}$ with $n^{1+\alpha}$-stretch for any $\alpha > 0$ and $\mathcal{R}$-degree D.
- A weak PRF F in NC^1.
- Selectively IND-secure (collusion resistant) FE schemes for degree-$(3D+2)$ polynomials in $\mathcal{R}$, $\{\mathbf{FE}^{N'} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})\}$, with linear efficiency.
- A specific randomized encoding scheme, which is the composition of Yao's garbling scheme [Yao82, Yao86] and the AIK randomized encoding scheme in NC^0 [AIK04].

  Below, we explicitly describe how Yao's garbling and AIK RE are used, which helps us to calculate the degree later. Denote by $\hat{C}_x = \mathsf{Yao}(C, \mathbf{x};\ \mathbf{r})$ Yao's garbling algorithm that compiles a circuit C and an input $\mathbf{x}$ into a garbled circuit $\hat{C}_x$, and by $\Pi = \mathsf{AIK}(f, \mathbf{x}\ ;\ \mathbf{r})$ the AIK encoding algorithm.

The scheme $\mathbf{CFE}^{N,D,S} = (\mathsf{CFE.Setup}, \mathsf{CFE.KeyGen}, \mathsf{CFE.Enc}, \mathsf{CFE.Dec})$ is defined in Figure 1. We refer the reader to [LV16] for the correctness and security of the scheme.

Compactness The compactness of the scheme $\mathbf{CFE}$ follows from the following facts:

1. The length of the input $(\mathbf{x}, \mathbf{k}, \mathbf{s}, \mathbf{s}', 0)$ encrypted using $\mathbf{FE}$ is $O(\ell^{1/(1+\alpha)}) = S(\lambda)^{1/(1+\alpha)}\operatorname{poly}(\lambda)$.
2. $\mathbf{FE}$ has linear efficiency.

Putting them together, we have that

$$\begin{aligned}\mathsf{Time}_{\mathsf{CFE.Enc}}(\mathsf{mpk}, \mathbf{x}) &= \mathsf{Time}_{\mathsf{FE.Enc}}(\mathsf{mpk}, (\mathbf{x}, \mathbf{k}, \mathbf{s}, \mathbf{s}', 0)) \\ &= \operatorname{poly}(\lambda)|(\mathbf{x}, \mathbf{k}, \mathbf{s}, \mathbf{s}', 0)| = S(\lambda)^{1/(1+\alpha)}\operatorname{poly}(\lambda)\end{aligned}$$

which is sublinear in the function size as desired.

It remains to verify Fact 1). Recall that ℓ is the total length of the AIK randomized encodings of computations $\{h_i(\mathbf{x}, \mathbf{k})\}$, which evaluate every bit in Yao's garbled circuit of $(f, \mathbf{x})$. Since $f(\mathbf{x})$ can be computed in size $S(\lambda)$, its Yao's garbled circuit has size $S(\lambda)\operatorname{poly}(\lambda)$, and every bit i in the garbled circuit can be computed by a function h_i of a *fixed polynomial size* $\operatorname{poly}(\lambda)$. Thus, the AIK randomized encoding for each $h_i(\mathbf{x}, \mathbf{k})$ also has size $\operatorname{poly}(\lambda)$, and the total length $\ell = S(\lambda)\operatorname{poly}(\lambda)$, which concludes Fact 1).

Degree-$(3D+2)$ FE suffices we show that degree-$(3D+2)$ FE indeed suffices for the construction.

Claim 1. *If* $\mathbf{PRG}$ *has $\mathcal{R}$-degree D, then for every $\lambda \in \mathbb{N}$, every output bit of the function g described in Figure 1 can be computed by a degree-$(3D+2)$ polynomial in $\mathcal{R}_\lambda$.*

Proof. Fix any $\lambda \in \mathbb{N}$. Let $P_{\mathsf{AIK}_k(h_i,\star)}((\mathbf{x}, \mathbf{k}),\ \mathbf{r})$ be the polynomial in $\mathcal{R}_\lambda$ computing $\mathsf{AIK}_k(h_i, (\mathbf{x}, \mathbf{k}); \mathbf{r})$, the k^{th} bit in the AIK randomized encoding of $h_i(\mathbf{x}, \mathbf{r})$, $P_{\mathbf{PRG}_l}$ the polynomial that computes the l^{th} output bit of $\mathbf{PRG}$, and $P_{\mathrm{CT}_l \oplus \star}(x)$ the polynomial that computes XOR $\mathrm{CT}_l \oplus x$. For convenience, we also denote by $P_{\mathbf{PRG}[i]}$ the *multi-output* polynomial that computes the i^{th} portion of the

Single-key Compact FE Scheme CFE by [LV16]

SETUP: $\mathsf{CFE.Setup}(1^\lambda)$ samples $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda)$.

KEY GENERATION: $\mathsf{CFE.KeyGen}(\mathsf{msk}, f)$ does the following:

- Sample $\mathbf{CT} \xleftarrow{\$} \{0,1\}^\ell$, where $\ell = \ell(\lambda)$ is set below.
- Define function g as follows: On input $\mathbf{x}$ of length N, a weak PRF key $\mathbf{k}$ of length $\mathrm{poly}(\lambda)$, two PRG seeds s, s' each of length $\ell^{1/(1+\alpha)}$ and a bit b,

  $g(\mathbf{x}, \mathbf{k}, \mathrm{s}, \mathrm{s}', b)$ does the following:

  - Let $h_i(\mathbf{x}, \mathbf{k})$ denote the function that computes the i^{th} bit in Yao's garbling of $(f, \mathbf{x})$ using pseudo-randomness generated by a weak PRF

    $$\forall i \in [I], \qquad h_i(\mathbf{x}, \mathbf{k}) := \mathsf{Yao}_i(f, \mathbf{x}\ ;\ \mathbf{r} = \{r_j = \mathsf{F}(\mathbf{k}, j)\})\ ,$$

    where I is the length of Yao's garbling of $(f, \mathbf{x})$. (Note that $h \in \mathsf{NC}^1$ since Yao's garbling algorithm and the weak PRF are both computable in NC^1.)
  - If $b = 0$, for every $i \in [I]$, compute the AIK encoding $\Pi[i]$ of computation $(h_i, (\mathbf{x}, \mathbf{k})))$, using pseudo-randomness generated by a PRG

    $$\forall\ i \in [I], \qquad \Pi[i] = \mathsf{AIK}(h_i,\ (\mathbf{x}, \mathbf{k})\ ;\ \mathbf{r}[i])\ ,\ \text{where } \mathbf{r}[i] = \mathbf{PRG}[i](\mathbf{s})$$

    where $\mathbf{PRG}[i](\mathbf{s})$ denotes the i^{th} *portion* in the output of $\mathbf{PRG}$, and each portion has equal length $\mathrm{poly}(\lambda)$.
    Output $\Pi = \{\Pi[i]\}_i$.
  - If $b = 1$, output $\Pi = \mathbf{CT} \oplus \mathbf{PRG}(\mathrm{s}')$.

  For every $l \in [\ell = |\Pi|]$, let P_l denote the degree-$(3D+2)$ polynomial in $\mathcal{R}_\lambda$ that computes the l^{th} output bit of g. (We show below in Claim 1 that every output bit of g can indeed be computed by a degree-$(3D+2)$ polynomial in $\mathcal{R}_\lambda$.)
- For every $l \in [\ell]$, generate a secret key $\mathsf{SK}_l \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{msk}, P_l)$ for P_l.

Output $\mathsf{SK} = \{\mathsf{SK}_l\}_{l \in [\ell]}$.

ENCRYPTION: $\mathsf{CFE.Enc}(\mathsf{mpk}, \mathbf{x})$ samples $\mathbf{k} \xleftarrow{\$} \{0,1\}^{\mathrm{poly}(\lambda)}$ and $\mathbf{s}, \mathbf{s}' \xleftarrow{\$} \{0,1\}^{\ell^{1/(1+\alpha)}}$, and generates

$$\mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{mpk}, (\mathbf{x}, \mathbf{k}, \mathbf{s}, \mathbf{s}', 0))$$

DECRYPTION: $\mathsf{CFE.Dec}(\mathsf{SK}, \mathsf{CT})$ computes $\Pi = \{\mathsf{FE.Dec}(\mathsf{SK}_l, \mathsf{CT})\}_{l \in [\ell]}$, parses $\Pi = \{\Pi[i]\}_{i \in I}$, and decodes every $\Pi[i]$ using the AIK decoding algorithm to obtain a garbled circuit, which is further decoded to obtain the output $f(\mathbf{x})$.

Figure 1: Single-key Compact FE **CFE** by [LV16]

output of $\mathbf{PRG}$. Every output bit $l \in [\ell]$ of g corresponds to a bit, say the j^{th}, in a AIK randomized encoding for some function h_i. Then, g_l can be computed by the following polynomial P_l in $\mathcal{R}_\lambda$.

$$P_l(\mathbf{x}, \mathbf{k}, \mathbf{s}, \mathbf{s}', b) = (1-b)P_{\mathsf{AIK}_j(h_i,\star)}(\ (\mathbf{x}, \mathbf{k}),\ \ P_{\mathbf{PRG}[i]}(\mathbf{s})\ ) + bP_{\mathrm{CT}_l \oplus \star}(P_{\mathbf{PRG}_l}(\mathbf{s}')) \tag{5}$$

$P_{\mathbf{PRG}[i]}$, $P_{\mathbf{PRG}_l}$, and $P_{\mathrm{CT}_l \oplus \star}$ have respectively degree D and degree 1. AIK randomized encoding has the property that every output bit depends on at most 3 random bits and 1 input bit. Therefore, $P_{\mathsf{AIK}_j(h_i,\star)}$ has at most degree 3 in outputs of $P_{\mathbf{PRG}[i]}(\mathbf{s})$, and at most degree 1 in $(\mathbf{x}, \mathbf{k})$, and hence at most total degree $3D+1$. Therefore, the degree of P_l is bounded by $3D+2$. □

5.2 IO from Locality-L PRG and Degree-L FE

By the fact that the locality L of a PRG upper bounds the degree of a PRG in any ring $\mathcal{R}$, we have that IO can be constructed from locality-L PRG and degree-$(3L+2)$ FE. We now present modification to the LV bootstrapping theorem to reduce the degree of the FE to L.

Theorem 5 (Our Bootstrapping Theorem)**.** *Let $\mathcal{R} = \{\mathcal{R}_\lambda\}$ be any family of rings and $\varepsilon > 0$ any positive constant. Assume the existence of a sub-exponentially secure PRG with $n^{1+\varepsilon}$-stretch and locality-L. Then, IO for* P/poly *is implied by either of the following:*

- *any selectively sub-exponential-IND-secure* public key *(zero-testing) FE for degree-L polynomials in $\mathcal{R}$, with linear efficiency, or*
- *any selectively sub-exponential-IND-secure* secret key *(zero-testing) FE for degree-L polynomials in $\mathcal{R}$ with linear efficiency, and the sub-exponential hardness of LWE.*

To show the theorem, our main idea is pre-processing the input $(\mathbf{x}, \mathbf{k}, \mathrm{s}, \mathrm{s}', b)$ to be encrypted, at encryption time, in order to reduce the degree of the polynomials that FE needs to support. Observe that each polynomial P_l (in Equation 5) computed in the LV FE scheme **CFE** in Figure 1 is the sum of two polynomials, where the first term has degree $(3D+2)$ and the second has $D+1$. We start with reducing the degree of the second term from $D+1$ to D. We can write the second term T_l as a sum of monomials over b, s' as follows

$$T_l(b, \mathbf{s}') = bP_{\mathrm{CT}_l \oplus \star}(P_{\mathbf{PRG}_l}(\mathbf{s}')) = \sum_{\substack{\text{Monomial} \\ M \text{ in } T_l}} c_M M(b, \mathbf{s}')\ .$$

Since T_l is linear in b, every monomial M contained in it is also linear in b. For every monomial $M = bs'_{i_1}s'_{i_2}\cdots$ of degree d, if bs'_{i_1} is pre-computed, then M can be computed in degree $d-1$. Therefore, there exists a polynomial T'_l that on input $(1||b) \otimes (1||\mathrm{s}')$ computes $T_l(b, \mathrm{s}')$ in degree D.

$$T'_l((1||b) \otimes (1||\mathrm{s}')) := \text{ The degree } D \text{ polynomial that computes } T_l(b, \mathrm{s}') \tag{6}$$

Moreover, the length of $(1||b) \otimes (1||\mathrm{s}')$ is still $O(|\mathrm{s}'|) = S(\lambda)^{1/(1+\alpha)}\,\mathrm{poly}(\lambda)$, and hence we can pre-compute $(1||b) \otimes (1||\mathrm{s}')$ at encryption time, without losing compactness.

We now use the idea of pre-processing to reduce the degree of computing the first term in P_l. Again, we can write the first term O_l as a sum of monomials,

$$O_l(\mathbf{x}, \mathbf{k}, \mathbf{s}, b) = (1-b)P_{\mathsf{AIK}_j(h_i,\star)}(\ (\mathbf{x}, \mathbf{k})\ ;\ \mathbf{r}[i]\ ) = \sum_{\substack{\text{Monomial} \\ M \text{ in } O_l}} c_M M(\mathbf{x}, \mathbf{k}, \mathbf{r}[i], b)\ ,\ \text{where } \mathbf{r}[i] = P_{\mathbf{PRG}[i]}(\mathbf{s})\ .$$

By the property of AIK, O_l has degree 3 in $\mathbf{r}[i]$, and 1 in b and $\mathbf{x}||\mathbf{k}$. We can eliminate multiplication with b and $\mathbf{x}||\mathbf{k}$ using the same method above. The challenge lies in reducing the degree for computing degree-3 monomials on $\mathbf{r}[i]$. We cannot naively pre-compute, say, $\mathsf{s} \otimes \mathsf{s}$, as its length would exceed $S(\lambda)$. But, we do not need to. To do so, we first modify how each random bit $\mathbf{r}[i]_q$ is generated as follows:

$$\text{Let } Q = |\mathbf{r}[i]|,\ \mathsf{s} = \mathsf{s}_1, \cdots, \mathsf{s}_Q; \qquad \forall\, q \in [Q],\ \text{ set } \mathbf{r}[i]_q = \mathbf{PRG}_i(\mathsf{s}_q)$$

where Q is the maximal number of random bits needed for computing the AIK encoding of each h_i. Since every function h_i computes a single bit in Yao's garbling in a fixed polynomial time, $Q = |\mathbf{r}[i]| = \mathrm{poly}(\lambda)$. In other words, we parse s as consisting of Q seeds, and the q^{th} seed s_q is used for generating the q^{th} bit in the random tapes for computing every AIK encodings, that is, $\mathbf{PRG}(\mathsf{s}_q) = \mathbf{r}[1]_q, \cdots, \mathbf{r}[I]_q$. Since the number of AIK encodings is $I = S(\lambda)\,\mathrm{poly}(\lambda)$, the length of each seed is $|\mathsf{s}_q| = I^{1/(1+\alpha)} = S(\lambda)^{1/(1+\alpha)}\,\mathrm{poly}(\lambda)$, and the length of s is $Q|\mathsf{s}_q|$, also sublinear in $S(\lambda)$.

Now, an arbitrary degree 3 monomial on $\mathbf{r}[i]$, say $\mathbf{r}[i]_{q_1}\mathbf{r}[i]_{q_2}\mathbf{r}[i]_{q_3}$, can be written as

$$\begin{aligned}
\mathbf{r}[i]_{q_1}\mathbf{r}[i]_{q_2}\mathbf{r}[i]_{q_3} &= \mathbf{PRG}_i(\mathsf{s}_{q_1})\mathbf{PRG}_i(\mathsf{s}_{q_2})\mathbf{PRG}_i(\mathsf{s}_{q_3}) \\
&= \mathbf{PRG}_i(\{s_{q_1,\gamma}\}_{\gamma\in\Gamma(i)})\mathbf{PRG}_i(\{s_{q_2,\gamma}\}_{\gamma\in\Gamma(i)})\mathbf{PRG}_i(\{s_{q_3,\gamma}\}_{\gamma\in\Gamma(i)}) \\
&= \sum_{\substack{\text{Monomials}\\ X,Y,Z \text{ in } \mathbf{PRG}_i}} \begin{pmatrix} & X(s_{q_1,\gamma_1}, & \cdots, & s_{q_1,\gamma_L}) \\ \times & Y(s_{q_2,\gamma_1}, & \cdots, & s_{q_2,\gamma_L}) \\ \times & Z(s_{q_3,\gamma_1}, & \cdots, & s_{q_3,\gamma_L}) \end{pmatrix}, \qquad \text{where } \Gamma(i) = \{\gamma_1, \cdots, \gamma_L\}
\end{aligned}$$

and $\Gamma(i)$ is the set of indexes of the input bits that the i^{th} output bit of $\mathbf{PRG}$ depends on. Given that $\mathbf{PRG}$ has locality L, $|\Gamma(i)| \le L$. For every $\gamma \in [|\mathsf{s}_q|]$, denote by $\mathsf{s}_{\star,\gamma} = s_{1,\gamma}||\cdots||s_{Q,\gamma}$ the string consisting of the γ^{th} bit in all Q seeds. Suppose that we pre-compute for every γ, all the degree ≤ 3 monomials over $\mathsf{s}_{\star,\gamma}$, that is, $\mathbf{S}_\gamma = (1||\mathsf{s}_{\star,\gamma}) \otimes (1||\mathsf{s}_{\star,\gamma}) \otimes (1||\mathsf{s}_{\star,\gamma})$. Then, the above degree 3 monomial over $\mathbf{r}[i]$ can be computed by a polynomial of degree at most L on $\mathbf{S}_\gamma$ for $\gamma \in \Gamma(i)$. Therefore, there exists a degree $L+2$ polynomial O''_l that on input $\mathbf{x}, \mathbf{k}, b$ and these degree ≤ 3 monomials, computes O_l.

$$\begin{aligned}
O''_l(\mathbf{x}, \mathbf{k}, \mathbf{S}, b) :=\ & \text{The degree } L+2 \text{ polynomial that computes } O_l(\mathbf{x}, \mathbf{k}, \mathsf{s}, b)\ , \\
& \qquad\qquad\qquad \text{where } \mathbf{S} = \{(1||\mathsf{s}_{\star,\gamma}) \otimes (1||\mathsf{s}_{\star,\gamma}) \otimes (1||\mathsf{s}_{\star,\gamma})\}_\gamma\ .
\end{aligned}$$

In addition, since O_l and O''_l has degree 1 in $\mathbf{x}||\mathbf{k}$ and b, if we pre-compute multiplications with $\mathbf{x}||\mathbf{k}$ and b, we can further reduce the degree to L. That is, define

$$\begin{aligned}
& O'_l((1||\mathbf{x}||\mathbf{k}||b) \otimes (1||\mathbf{S}), (b(\mathbf{x}||\mathbf{k})) \otimes \mathbf{S}) \\
& \qquad\qquad := \text{ The degree } L \text{ polynomial that computes } O_l(\mathbf{x}, \mathbf{k}, \mathsf{s}, b)\ . \qquad (7)
\end{aligned}$$

Finally, we argue that the input of O'_l has length sublinear in $S(\lambda)$. It boils down to argue that $\mathbf{S}$ has length sublinear in $S(\lambda)$. For each $\gamma \in [|\mathsf{s}_q|]$, the total number of degree ≤ 3 monomials over $\mathsf{s}_{\star,\gamma}$ is bounded by $(Q+1)^3 = \mathrm{poly}(\lambda)$. Since $|\mathsf{s}_q| = S(\lambda)^{1/(1+\alpha)}\,\mathrm{poly}(\lambda)$, the length of $\mathbf{S}$ is bounded by $S(\lambda)^{1/(1+\alpha)}\,\mathrm{poly}(\lambda)$.

Combining Equation (6) and (7), we conclude that there exists a degree L polynomial P'_l, such that,

$$P'_l((1||\mathbf{x}||\mathbf{k}||b) \otimes (1||\mathbf{S}),\ (b(\mathbf{x}||\mathbf{k})) \otimes \mathbf{S},\ (1||b) \otimes (1||\mathsf{s}')) = P_l(\mathbf{x}, \mathbf{k}, \mathsf{s}, \mathsf{s}', b)$$

Single-key Compact FE Scheme CFE from locality-L PRG and degree-L FE

SETUP: $\mathsf{CFE.Setup}(1^\lambda)$ samples $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda)$.

KEY GENERATION: $\mathsf{CFE.KeyGen}(\mathsf{msk}, f)$ does the following:

- Sample $\mathbf{CT} \xleftarrow{\$} \{0,1\}^\ell$, where $\ell = \ell(\lambda)$ is set below.
- Define function g defined as follows: On input $\mathbf{x}$ of length N, a weak PRF key $\mathbf{k}$ of length $\mathrm{poly}(\lambda)$, PRG seeds s and s' of length $I^{1/(1+\alpha)} \times Q$ and $\ell^{1/1+\alpha}$ respectively, and a bit b,

  $g(\mathbf{x}, \mathbf{k}, \mathrm{s}, \mathrm{s}', b)$ does the following:

  - Let $h_i(\mathbf{x}, \mathbf{k})$ denote the function that computes the i^{th} bit in Yao's garbling of $(f, \mathbf{x})$,

    $$\forall i \in [I], \qquad h_i(\mathbf{x}, \mathbf{k}) := \mathsf{Yao}_i(f, \mathbf{x}\,;\, \mathbf{r} = \{r_j = \mathsf{F}(\mathbf{k}, j)\})\,,$$

    where I is the length of Yao's garbling of $(f, \mathbf{x})$.
  - If $b = 0$, parse s into Q strings, $\mathrm{s} = \mathrm{s}_1 || \cdots || \mathrm{s}_Q$, of equal length $I^{1/(1+\alpha)}$, and compute

    $$\forall\, i \in [I],\ \Pi[i] = \mathsf{AIK}(h_i,\ (\mathbf{x}, \mathbf{k})\ ;\ \mathbf{r}[i])\,,\ \text{where } Q = |\mathbf{r}[i]| \text{ and } \forall\, q \in [Q]\,,\ \mathbf{r}[i]_q = \mathbf{PRG}_i(\mathrm{s}_q)$$

    Output $\Pi = \{\Pi[i]\}_i$.
  - If $b = 1$, output $\Pi = \mathbf{CT} \oplus \mathbf{PRG}(\mathrm{s}')$.

  For every $l \in [\ell = |\Pi|]$, let P_l denote the degree-$(3D+2)$ polynomial in $\mathcal{R}_\lambda$ that computes the l^{th} output bit of g. Moreover, define

  $$\begin{aligned}&P'_l((1||\mathbf{x}||\mathbf{k}||b) \otimes (1||\mathbf{S}), (b(\mathbf{x}||\mathbf{k})) \otimes \mathbf{S}, (1||b) \otimes (1||\mathrm{s}'))\\ &\qquad := \text{ The degree } L \text{ polynomial that computes } P_l(\mathbf{x}, \mathbf{k}, \mathrm{s}, \mathrm{s}', b) \text{ in Figure 1}\end{aligned}$$

  where L is the locality of $\mathbf{PRG}$ and $\mathbf{S} = \{(1||\mathrm{s}_{\star,\gamma}) \otimes (1||\mathrm{s}_{\star,\gamma}) \otimes (1||\mathrm{s}_{\star,\gamma})\}_{\gamma \in [I^{1/(1+\alpha)}]}$.
- For every $l \in [\ell]$, generate a secret key $\mathsf{SK}_l \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{msk}, P'_l)$ for P'_l.

Output $\mathsf{SK} = \{\mathsf{SK}_l\}_{l \in [\ell]}$.

ENCRYPTION: $\mathsf{CFE.Enc}(\mathsf{mpk}, \mathbf{x})$ samples $\mathbf{k} \xleftarrow{\$} \{0,1\}^{\mathrm{poly}(\lambda)}$, $\mathrm{s} \xleftarrow{\$} \{0,1\}^{I^{1/(1+\alpha)} \times Q}$, and $\mathrm{s}' \xleftarrow{\$} \{0,1\}^{\ell^{1/(1+\alpha)}}$, and generates

$$\mathsf{CT} \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{mpk}, (1||\mathbf{x}||\mathbf{k}||0) \otimes (1||\mathbf{S}),\ (0(\mathbf{x}||\mathbf{k})) \otimes \mathbf{S},\ (1||0) \otimes (1||\mathrm{s}'))$$

DECRYPTION: $\mathsf{CFE.Dec}(\mathsf{SK}, \mathsf{CT})$ computes $\Pi = \{\mathsf{FE.Dec}(\mathsf{SK}_l, \mathsf{CT})\}_{l \in [\ell]}$, parse $\Pi = \{\Pi[i]\}_{i \in I}$, and decodes every $\Pi[i]$ using the AIK decoding algorithm to obtain a garbled circuit, which is further decoded to obtain the output $f(\mathbf{x})$.

Figure 2: Single-key Compact FE **CFE** from locality-L PRG and degree-L FE

Therefore, we modify the LV compact FE scheme to encrypt the input of these polynomials P'_l's, and generate secret keys for them. Since P'_l has only degree L, it suffices to use a degree-L FE scheme. The resulting new compact FE scheme **CFE** is described in Figure 2 (with the difference from the LV scheme highlighted). The compactness of the new scheme follows directly from the fact that the encrypted input, that is, the input of P'_l's, is sublinear in $S(\lambda)$, and that the degree-L **FE** scheme has linear efficiency. Moreover, its correctness and security follows from the same proof as that in [LV16], which concludes Theorem 5.

6 Inner Product Encryption

In this section, we construct families of (*zero-testing* by default) secret key IPE schemes (Definition 15), with different properties.

- In Section 6.3, we give a new construction of *weakly function hiding* IPE schemes based on the SXDH assumption on bilinear maps. Our construction builds upon the public key IPE schemes by [ABCP15] (ABCP) in a simple and modular way.
- Lin and Vaikuntanathan [LV16] presented a simple and generic transformation from any weakly function hiding IPE schemes to fully function hiding IPE schemes. In Section 6.4, we apply their transformation to our weakly function hiding IPE schemes to obtain fully function hiding IPE schemes that have certain canonical form. (The canonical form is not satisfied by previous constructions [BJK15, LV16, DDM16]).
- In Section 6.5, we further build function hiding IPEs with special *two-slot* structures and special security properties, namely *partial weakly-function-hiding* and *strong IND-security*. The special structures and properties will be important for our construction of FE for degree-d polynomials later.

Before constructing the above schemes, we first review the definition of weak function hiding [LV16] in Section 6.1, and the ABCP public key IPE scheme in Section 6.2.

6.1 Definition of Weak Function Hiding

Let $\{\mathbf{skIPE}^N\}$ be a family of secret key IPE schemes for inner products, where $\mathbf{skIPE}^N$ consisting algorithms $(\mathsf{skIPE.Setup}, \mathsf{skIPE.Enc}, \mathsf{skIPE.KeyGen}, \mathsf{skIPE.Dec})$ is a an IPE scheme for computing inner products of length $N(\lambda)$ vectors in $\mathcal{R}$, with the following syntax:

- *Setup:* $\mathsf{skIPE.Setup}(1^\lambda, \mathsf{pp})$ outputs a master secret key msk. (The public parameter pp used in our constructions will be the description of bilinear pairing groups).
- *Key Generation:* $\mathsf{skIPE.KeyGen}(\mathsf{msk}, \mathbf{y})$ outputs a secret key SK encoding a vector $\mathbf{y} \in \mathcal{R}_\lambda^N$.
- *Encryption:* $\mathsf{skIPE.Enc}(\mathsf{msk}, \mathbf{x})$ outputs an encryption CT encrypting a vector $\mathbf{x} \in \mathcal{R}_\lambda^N$.
- *Decryption:* $\mathsf{skIPE.Dec}(\mathsf{SK}, \mathsf{CT})$ computes $\mathsf{ZT}(\langle \mathbf{x}, \mathbf{y} \rangle)$, that is, whether the inner product is zero in $\mathcal{R}_\lambda$.

The *function hiding property* as defined in Definition 11 requires that secret keys and ciphertexts for one set of vectors $\{\mathbf{y}_j^0\}$ and $\{\mathbf{x}_i^0\}$ are indistinguishable from that for another vectors $\{\mathbf{y}_j^1\}$ and $\{\mathbf{x}_i^1\}$, as long as all their inner products satisfy the following constraint:

$$\textbf{Constraint R:} \quad \forall i, j, \quad \langle \mathbf{x}_i^0, \mathbf{y}_j^0 \rangle = \langle \mathbf{x}_i^1, \mathbf{y}_j^1 \rangle \ .$$

The *weak function hiding property* weakens function hiding, by relaxing the above constraint $\mathbf{R}$ to the following

$$\textbf{Constraint R':} \quad \forall i, j, \quad \langle \mathbf{x}_i^0, \mathbf{y}_j^0 \rangle = \underline{\left\langle \mathbf{x}_i^0, \mathbf{y}_j^1 \right\rangle} = \langle \mathbf{x}_i^1, \mathbf{y}_j^1 \rangle \ .$$

For completeness, we provide the formal definition below.

Definition 17 (Weak Function Hiding for Secret Key IPE)**.** *We say that a secret key IPE scheme* $\mathbf{skIPE}^N$ *for computing length-N inner products in $\mathcal{R}$ is μ-*weak function hiding, *if for every* PPT *adversary A, and every sufficiently large security parameter $\lambda \in \mathbb{N}$, the adversary's advantage in the following games is bounded by $\mu(\lambda)$:*

$$\mathsf{Advt}_A^{\mathbf{skIPE}^N} = \left| \Pr[\mathsf{wFH}_A^{\mathbf{skIPE}^N}(1^\lambda, 0) = 1] - \Pr[\mathsf{wFH}_A^{\mathbf{skIPE}^N}(1^\lambda, 1) = 1] \right| \leq \mu(\lambda)$$

The game $\mathsf{wFH}_A^{\mathbf{skIPE}^N}(1^\lambda, b)$ *proceeds as follows:*

- **Key Generation.** *The challenger generates a master secret key* $\mathsf{msk} \xleftarrow{\$} \mathsf{skIPE.Setup}(1^\lambda, \mathsf{pp})$.
- **Function and Input Queries.** *Repeat the following for an arbitrary number of times decided by A:*
  - *Upon A choosing a pair of challenge functions $\mathbf{y}_i^0, \mathbf{y}_i^1 \in \mathcal{R}_\lambda^{N(\lambda)}$, CH sends A a function key* $\mathsf{SK}_i \xleftarrow{\$} \mathsf{skIPE.KeyGen}(\mathsf{msk}, \mathbf{y}_i^b)$.
  - *Upon A choosing a pair of challenge messages $\mathbf{x}_i^0, \mathbf{x}_i^1 \in \mathcal{R}_\lambda^{N(\lambda)}$, CH sends A a ciphertext* $\mathsf{CT}_i \xleftarrow{\$} \mathsf{skIPE.Enc}(\mathsf{msk}, \mathbf{x}_i^b)$.
- *Finally A outputs a bit b'.*

Restriction R$'$: *Every message query $\mathbf{x}_i^0, \mathbf{x}_i^1$ and every function query $\mathbf{y}_j^0, \mathbf{y}_j^1$ must satisfy that* $\left\langle \mathbf{x}_i^0, \mathbf{y}_j^0 \right\rangle = \underline{\left\langle \mathbf{x}_i^1, \mathbf{y}_j^0 \right\rangle} = \left\langle \mathbf{x}_i^1, \mathbf{y}_j^1 \right\rangle$.

6.2 Review of the ABCP Public Key IPE

In [ABCP15], Abdalla, Bourse, De Caro, and Pointcheval constructed public key IPE schemes with IND-security, based on a variety assumptions. We recall their scheme based on the DDH assumption, which is very similar to the ElGamal encryption scheme.

The Decisional Diffie-Hellman (DDH) Assumption Let $\mathcal{G}$ denote a group generator that on input 1^λ outputs (p, G), where G is a group of order p, and g is a generator of G contained in its description. As above, we use the bracket notation to denote elements in G: $[v] = g^v$ and $[\mathbf{x}] = g^{\mathbf{x}} = g^{x_1}, \cdots, g^{x_m}$ for $v \in \mathbb{Z}_p, \mathbf{x} \in \mathbb{Z}_p^m$. The DDH assumption states that the following two ensembles are μ-indistinguishable:

$$\left\{(p, G) \xleftarrow{\$} \mathcal{G}(1^\lambda),\ a, b \xleftarrow{\$} \mathbb{Z}_p \ : \ (p, G),\ [a], [b], [ab]\right\}_{\lambda \in \mathbb{N}}$$
$$\left\{(p, G) \xleftarrow{\$} \mathcal{G}(1^\lambda),\ a, b, r \xleftarrow{\$} \mathbb{Z}_p \ : \ (p, G),\ [a], [b], [r]\right\}_{\lambda \in \mathbb{N}}$$

Overview of the ABCP Scheme Recall that the basic ElGamal encryption scheme for message space $\mathbb{Z}_p$ is as follows:

$$\mathsf{msk} \xleftarrow{\$} \mathbb{Z}_p, \qquad \mathsf{mpk} = g^{\mathsf{msk}}, \qquad \mathsf{CT} = (g^r,\ \mathsf{mpk}^r g^x) = (g^r,\ g^{(r\ \mathsf{msk}+x)}) \text{ for } r \xleftarrow{\$} \mathbb{Z}_p$$

Note that in this scheme, decryption can be done when x is small.

Under our bracket notation this is written as:

$$\mathsf{msk} \xleftarrow{\$} \mathbb{Z}_p, \qquad \mathsf{mpk} = [\mathsf{msk}], \qquad \mathsf{CT} = [r],\ (r \odot \mathsf{mpk}) \oplus [x] = [r \mid\mid (r\ \mathsf{msk} + x)]$$

(Recall that "$\odot$" and "$\oplus$" are respectively the homomorphic scalar multiplication and addition operations over encodings.) The ElGamal encryption can be easily modified to encrypt vectors $\mathbf{x} \in \mathbb{Z}_p^N$, *while sharing the random scalar* r, and maintaining security under the same DDH assumption.

$$\mathsf{msk} \xleftarrow{\$} \mathbb{Z}_p^N, \quad \mathsf{mpk} = [\mathsf{msk}], \quad \mathsf{CT} = [-r],\ (r \odot \mathsf{mpk}) \oplus [\mathbf{x}] = [(-r \mid\mid r\ \mathsf{msk} + \mathbf{x})]$$

(We encode $-r$ instead of r for convenience.) To turn the above scheme into an IPE scheme, observe that given a vector $\mathbf{y} \in \mathbb{Z}_p^N$ and the inner product $\langle \mathbf{y}, \mathsf{msk} \rangle$ in the clear, one can homomorphically evaluate,

$$\begin{aligned}\langle\ (\langle \mathbf{y}, \mathsf{msk}\rangle\ \mid\mid \mathbf{y}), \mathsf{CT}\ \rangle &= \big\langle\ \big(\ \langle \mathbf{y}, \mathsf{msk}\rangle\ \mid\mid \mathbf{y}\big), [-r \mid\mid (r(\mathsf{msk} + \mathbf{x}))\ ]\ \big\rangle \\ &= [\ \langle -r\mathbf{y}, \mathsf{msk}\rangle + \langle r\mathbf{y}, \mathsf{msk}\rangle + \langle \mathbf{x}, \mathbf{y}\rangle] = [\langle \mathbf{x}, \mathbf{y}\rangle\ ]\ .\end{aligned}$$

Therefore, it suffice to release $\langle \mathbf{y}, \mathsf{msk}\rangle\ \mid\mid \mathbf{y}$ as the secret key for computing the inner product.

The ABCP Scheme We now formally describe the ABCP public key IPE scheme $\mathbf{pkIPE}^N$. Let $\mathsf{pp} = (p, G) \xleftarrow{\$} \mathcal{G}(1^\lambda)$ be a public parameter that describes a group G of order p; the inner product is computed over $\mathbb{Z}_p^N$.

- $\mathsf{pkIPE.Setup}(1^\lambda, \mathsf{pp})$ samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^N$, and outputs master public key $\mathsf{mpk} = [\mathbf{s}]$ and master secret key $\mathsf{msk} = \mathbf{s}$.
- $\mathsf{pkIPE.KeyGen}(\mathsf{msk}, \mathbf{y})$ on input the master secret key $\mathsf{msk} = \mathbf{s}$ and vector $\mathbf{y}$ both in $\mathbb{Z}_p^N$, simply outputs $\mathsf{SK} = \mathbf{sk} = \langle \mathbf{y}, \mathbf{s}\rangle\ \mid\mid \mathbf{y}$.
- $\mathsf{pkIPE.Enc}(\mathsf{mpk}, \mathbf{x})$ on input the master public key $\mathsf{mpk} = [\mathbf{s}]$ and vector $\mathbf{x} \in \mathbb{Z}_p^N$, samples a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs
$$\mathsf{CT} = [-r] \mid\mid (r \odot \mathsf{mpk}) \oplus [\mathbf{x}] = [-r \mid\mid r\mathbf{s} + \mathbf{x}] = [\mathbf{ct}]$$
- $\mathsf{pkIPE.Dec}(\mathsf{SK}, \mathsf{CT})$ on input $\mathsf{SK} = \mathbf{sk}$ and $\mathsf{CT} = [\mathbf{ct}]$ homomorphically computes the inner product between them.
$$\langle \mathsf{SK}, \mathsf{CT}\rangle = [\langle \mathbf{sk}, \mathbf{ct}\rangle] = \big[\big\langle\ \big(\ \langle \mathbf{y}, \mathbf{s}\rangle\ \mid\mid \mathbf{y}\big),\ \big(-r \mid\mid r\mathbf{s} + \mathbf{x}\big)\ \big\rangle\big] = [\langle \mathbf{x}, \mathbf{y}\rangle]$$
Output 1 iff the output encoding encodes zero. [6]

Correctness of the scheme is easy to see. The security proof is, however, non-trivial. Abdalla et al. [ABCP15] showed that the scheme in fact satisfies simulation-based security, which implies the notion of IND-security considered in this work.

Lemma 1. *Assume that the DDH assumption holds in the group* (p, G)*. Then, for any polynomial* N*, the ABCP public key IPE scheme* $\mathbf{pkIPE}^N$ *is IND-secure.*

[6]More generally, if the output value z falls into any polynomial-sized range $\Gamma \subseteq R$, it can be extracted by trying all possible values $i \in [\Gamma]$, and outputting the value i satisfying $[\langle \mathbf{x}, \mathbf{y}\rangle] = i \odot [1]$.

6.3 Our New Weakly Function Hiding IPE

We construct weakly function hiding IPE schemes $\{\mathbf{wIPE}^N\}$, from the SXDH assumption on bilinear pairing groups. Our construction uses the ABCP public key IPE schemes $\{\mathbf{pkIPE}^N\}$ described above as a building block in a modular way. At a very high-level, to make the ABCP IPE scheme weakly function hiding, we treat its secret key as a plaintext vector, and its ciphertext as a key vector, and use an "outer instance" of the ABCP IPE scheme itself to "encrypt" the secret key in an "outer ciphertext" and "encode" the ciphertext in an "outer secret key". Since decryption essentially performs inner product, decrypting the outer instance effectively decrypts also the inner instance and yields the desired output. Furthermore, since now the secret key is encrypted, the IND-security of the ABCP IPE scheme provides some hiding guarantees for the key vector, based on which we can argue that weak-function hiding holds.

Let $\mathsf{pp} = (p, G_1, G_2, G_3,\ \mathbf{pair})$ be a public parameter that describes bilinear pairing groups with order p; let $\mathcal{R} = \mathbb{Z}_p$. Algorithms of the scheme $\mathbf{wIPE}^N$ proceed as follows:

- $\mathsf{skIPE.Setup}(1^\lambda, \mathsf{pp})$ samples $\mathbf{s}_1, \mathbf{s}_2 \xleftarrow{\$} \mathcal{R}^N$, and outputs master secret key $\mathsf{wMSK} = (\mathbf{s}_1, \mathbf{s}_2)$.
- $\mathsf{skIPE.KeyGen}(\mathsf{wMSK}, \mathbf{y})$ on input the master secret key $\mathsf{wMSK} = (\mathbf{s}_1, \mathbf{s}_2)$ and vector $\mathbf{y} \in \mathcal{R}^N$, samples a random scalar $r_2 \xleftarrow{\$} \mathcal{R}$ and outputs SK computed as follows.

$$\begin{aligned} \mathsf{SK} = \mathsf{pkIPE.KeyGen}(\mathbf{s}_1, \mathbf{y}) &= \mathbf{sk} = \langle \mathbf{y}, \mathbf{s}_1 \rangle \ ||\ \mathbf{y} \\ \mathsf{wSK} = \mathsf{pkIPE.Enc}(\mathbf{s}_2,\ \mathbf{sk};\ r_2) &= [-r_2 \ ||\ (r_2\mathbf{s}_2 + \mathbf{sk})]_2 \end{aligned} \tag{8}$$

  Basically, wSK is an ABCP encryption (with key $\mathbf{s}_2$ and randomness r_2) of the ABCP secret key $\mathbf{sk}$ of the vector $\mathbf{y}$ (with key $\mathbf{s}_1$) in group G_2.

- $\mathsf{skIPE.Enc}(\mathsf{wMSK}, \mathbf{x})$ on input the master secret key $\mathsf{wMSK} = (\mathbf{s}_1, \mathbf{s}_2)$ and vector $\mathbf{x} \in \mathcal{R}^N$, samples a random scalar $r_1 \xleftarrow{\$} \mathcal{R}$ and outputs wCT computed as follows.

$$\begin{aligned} \mathsf{CT} &= \mathsf{pkIPE.Enc}(\mathbf{s}_1, \mathbf{x};\ r_1) = [\mathbf{ct}]_1,\ \text{where } \mathbf{ct} = (-r_1 \ ||\ r_1\mathbf{s}_1 + \mathbf{x}) \qquad (9) \\ \mathsf{wCT} &= \mathsf{pkIPE.KeyGen}(\mathbf{s}_2, \mathsf{CT}) = [\langle \mathbf{s}_2, \mathbf{ct} \rangle \ ||\ \mathbf{ct}]_1 \end{aligned}$$

  Basically, wCT can be viewed as the ABCP secret key (with key $\mathbf{s}_2$) of an ABCP ciphertext of the vector $\mathbf{x}$ (with key $\mathbf{s}_1$ and randomness r_1).

- $\mathsf{skIPE.Dec}(\mathsf{wSK}, \mathsf{wCT})$ on input wSK and wCT homomorphically computes their inner product using pairing, which gives an encoding of $\langle \mathbf{sk}, \mathbf{ct} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle$ in the target group G_3.

$$\langle \mathsf{wSK}, \mathsf{wCT} \rangle = [\langle \mathbf{sk}, \mathbf{ct} \rangle]_3 = [\langle \mathbf{x}, \mathbf{y} \rangle]_3$$

  Output 1 iff the obtained encoding encodes zero.

Correctness of the scheme $\mathbf{wIPE}$ is easy to see. We next show that it is weakly function hiding, based on the fact that the ABCP scheme $\mathbf{pkIPE}$ is IND-secure.

Lemma 2. *Assume that SXDH holds in bilinear pairing groups. For every polynomial N, the above secret key IPE scheme $\mathbf{wIPE}^N$ is weakly function hiding.*

Proof. We want to show that for every PPT adversary A, its view in games $\mathsf{Exp}_0 = \mathsf{wFH}_A^{\mathbf{wIPE}^N}(1^\lambda, 0)$ and $\mathsf{Exp}_1 = \mathsf{wFH}_A^{\mathbf{wIPE}^N}(1^\lambda, 1)$ (Definition 18) are indistinguishable.

To show this, we consider an intermediate hybrid Hyb:

- **Hybrid** Hyb proceeds identically as Exp_0, except that, upon A choosing a pair of challenge messages $\mathbf{x}_i^0, \mathbf{x}_i^1$, the challenger returns a ciphertext wCT_i encrypting $\mathbf{x}_i^1$ as opposed to $\mathbf{x}_i^0$.

Observe that the only difference between Exp_0 and Hyb is that in Exp_0, vectors $\{\mathbf{x}_i^0\}$ are encrypted, whereas in Hyb, vectors $\{\mathbf{x}_i^1\}$ are encrypted (and both games encode vectors $\{\mathbf{y}_i^0\}$ in the secret keys). On the other hand, the only difference between Hyb and Exp_1 is that in the former, vectors $\{\mathbf{y}_i^0\}$ are encoded in the secret keys, whereas in the latter, $\{\mathbf{y}_i^1\}$ are encoded (and both games encrypt vectors $\{\mathbf{x}_i^1\}$ in the ciphertexts).

Recall that the weak-function-hiding games $\mathsf{Exp}_0, \mathsf{Exp}_1$ have the constraint that every message query $\mathbf{x}_i^0, \mathbf{x}_i^1$ and every function query $\mathbf{y}_j^0, \mathbf{y}_j^1$ satisfy that

$$\langle \mathbf{x}_i^0, \mathbf{y}_i^0 \rangle = \langle \mathbf{x}_i^1, \mathbf{y}_i^0 \rangle = \langle \mathbf{x}_i^1, \mathbf{y}_i^1 \rangle \ .$$

Therefore, in all three games $\mathsf{Exp}_0, \mathsf{Hyb}, \mathsf{Exp}_1$, the inner products of the vectors encrypted and encoded in ciphertexts and secret keys are identical.

To see that Exp_0 is indistinguishable from Hyb, recall that the encryption algorithm of $\mathbf{wIPE}^N$ first encrypts a vector $\mathbf{x}$ using $\mathbf{pkIPE}$ and master secret key s_1 to produce a ciphertext $\mathsf{CT} = [\mathbf{ct}]_1$ (Line (9)) and then homomorphically generates a secret key of the encoded vector $\mathbf{ct}$ using an independently sampled master secret key s_2. Therefore, it follows directly from the IND-security of the "inner" $\mathbf{pkIPE}$ instance with master secret key s_1 that switching from encrypting $\mathbf{x}_i^0$ in Exp_0 to encrypting $\mathbf{x}_i^1$ in Hyb is indistinguishable.

Similarly, to see that Hyb and Exp_1 are indistinguishable, recall that the key generation algorithm of $\mathbf{wIPE}^N$ first generates a secret key $\mathsf{SK} = \mathbf{sk}$ of $\mathbf{y}$ using $\mathbf{pkIPE}$ and master secret key s_1, and then encrypts vector $\mathbf{sk}$ using $\mathbf{pkIPE}$ and master secret key s_2 (Line (8)). Therefore, it follows directly from the IND-security of the "outer" $\mathbf{pkIPE}$ instance with master secret key s_2 that switching from encoding $\mathbf{y}_i^0$ in Hyb to encoding $\mathbf{y}_i^1$ in Exp_1 is indistinguishable. □

6.4 Our New Function Hiding IPE

Lin and Vaikuntanathan [LV16] showed that any IPE scheme with weak function hiding can be generically "lifted" to an IPE scheme with full function hiding. Applying their technique to our weak-function hiding IPE schemes $\{\mathbf{wIPE}^N\}$ in Section 6.3 immediately gives a family of function function IPE schemes, denoted as $\{\mathbf{tIPE}^N\}$.

Corollary 2. *Assume that SXDH holds in bilinear pairing groups over ring $\mathcal{R}$. There is a family of function-hiding secret-key IPE schemes for computing inner products in $\mathcal{R}$.*

The [LV16] transformation is extremely simple: To generate a key or a ciphertext for a vector $\mathbf{v}$, $\mathbf{tIPE}^N$ simply uses the weak function hiding IPE scheme $\mathbf{wIPE}^{2N}$ to generate a key or a ciphertext for the vector $\mathbf{v}||\mathbf{0}$ padded with zeros upto to length $2N$. (The setup and decryption algorithms are identical to that of $\mathbf{wIPE}^{2N}$.) That is,

$$\begin{aligned} \mathsf{tIPE.Enc}(\mathsf{msk}, \mathbf{x}) \ &: \qquad \mathsf{tCT} \xleftarrow{\$} \mathsf{wIPE.Enc}(\mathsf{msk}, \mathbf{x}||\mathbf{0}) \ , \\ \mathsf{tIPE.KeyGen}(\mathsf{msk}, \mathbf{y}) \ &: \qquad \mathsf{tSK} \xleftarrow{\$} \mathsf{wIPE.KeyGen}(\mathsf{msk}, \mathbf{y}||\mathbf{0}) \ . \end{aligned}$$

Since the transformation is so simple, $\mathbf{tIPE}^N$ inherits many nice properties of $\mathbf{wIPE}^{2N}$ that will be instrumental for our construction of FE schemes later. Jumping ahead, we remark here that $\mathbf{tIPE}^N$ has the so-called canonical form (defined in Section 7.2).

Remark 1. $\mathbf{tIPE}^N$ *has canonical form, that is, it satisfies the following three properties (as inherited from* $\mathbf{wIPE}^{2N}$*).*

1. *Its ciphertext or secret key consist of only encodings in group G_1 or G_2 respectively of ring elements that depend linearly in the encoded vector* $\mathbf{v}$.
2. *The setup, key generation, and encryption algorithms do not use pairing nor the target group G_3.*
3. *The decryption algorithm homomorphically evaluates a degree 2 polynomial (namely inner product), on the encodings in the secret key and ciphertext, and then zero-tests the output encoding.*

6.5 Special-Purpose Two-Slot IPE

We construct a family of special-purpose secret key IPE schemes with the following special structure: We view the vectors $\mathbf{x} = \mathbf{x}_1||\mathbf{x}_2$ and $\mathbf{y} = \mathbf{y}_1||\mathbf{y}_2$ encoded in the ciphertext and secret key as consisting of two parts, referred to as the first- and second-slot vectors. A master secret key of the schemes contains three parts, a shared key $\mathbf{s}$ and two specific keys $\mathbf{k}'_1, \mathbf{k}'_2$, so that, encrypting a vector of form $\mathbf{u}_1||\mathsf{null}$ uses only $(\mathbf{s}, \mathbf{k}'_1)$ while encrypting $\mathsf{null}||\mathbf{u}_2$ uses only $(\mathbf{s}, \mathbf{k}'_2)$. Moreover, the scheme also satisfies several special properties, including *strong IND-security* and *partial weak-function-hiding*. Roughly speaking, the former states that the IND-security of the schemes hold as long as the shared key $\mathbf{s}$ is hidden (even when the slot keys are revealed), and the latter states that the schemes are weakly function hiding w.r.t. *individual slot*, even when the keys for encrypting to the other slot are published.

We call such IPE schemes, *two-slot IPE schemes*. Below we first formally describe their syntax and define partial weak-function-hiding. Then, we construct two-slot IPE schemes by modularly combined the ABCP public-key IPE scheme and the function hiding secret-key IPE scheme constructed in previous sections.

Syntax

- $\mathsf{sIPE.Setup}(1^\lambda, \mathsf{pp})$ outputs a master secret key msk consisting of a shared key $\mathbf{s}$ and two specific keys $\mathbf{k}'_1, \mathbf{k}'_2$. We denote by $\mathbf{k}_1 = (\mathbf{s}, \mathbf{k}'_1)$ the first-slot key, and $\mathbf{k}_2 = (\mathbf{s}, \mathbf{k}'_2)$ the second-slot key. For convenience, we write $\mathsf{msk} = (\mathbf{k}_1, \mathbf{k}_2)$ below.
- $\mathsf{sIPE.KeyGen}(\mathsf{msk}, \mathbf{y}_1, \mathbf{y}_2)$ on input msk and first- and second-slot vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ in $\mathcal{R}^N$, outputs a secret key sSK associated with $(\mathbf{y}_1, \mathbf{y}_2)$.
- $\mathsf{sIPE.Enc}(\mathsf{msk}, \mathbf{x}_1, \mathbf{x}_2)$ on input msk and first- and second-slot vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\mathcal{R}^N$, outputs a ciphertext sCT associated with $(\mathbf{x}_1, \mathbf{x}_2)$.
- $\mathsf{sIPE.Dec}(\mathsf{sSK}, \mathsf{sCT})$ on input a secret key sSK associated with $(\mathbf{y}_1, \mathbf{y}_2)$ and a ciphertext sCT associated with $(\mathbf{x}_1, \mathbf{x}_2)$, outputs whether $\langle \mathbf{x}_1||\mathbf{x}_2, \mathbf{y}_1||\mathbf{y}_2 \rangle$ is zero or not.

In addition, there is a new partial encryption algorithm $\mathsf{sIPE.PEnc}$ that uses either the first- or second-slot key to encrypt to only the first or second slot respectively.

- **Partial Encryption:** For $\beta \in [2]$, $\mathsf{sIPE.PEnc}(\beta, \mathbf{k}_\beta, \mathbf{x}_\beta)$, on input the β-slot key $\mathbf{k}_\beta$ and a vector $\mathbf{x}_\beta$, outputs a ciphertext sCT associated with $(\mathbf{x}_1, \mathsf{null})$ if $\beta = 1$ and $(\mathsf{null}, \mathbf{x}_2)$ if $\beta = 2$. When decrypting such a ciphertext with a secret key sSK associated with $(\mathbf{y}_1, \mathbf{y}_2)$, $\mathsf{sIPE.Dec}(\mathsf{sSK}, \mathsf{sCT})$ outputs whether $\langle \mathbf{y}_\beta, \mathbf{x}_\beta \rangle$ is zero.

Partial Weak-Function-Hiding This property states that weak function hiding holds w.r.t. the first (or the second) slot, even when the second-slot key (or the first-slot key respectively) are revealed. Formally,

Definition 18 (Partial Weak Function Hiding). *A two-slot IPE scheme* $\mathbf{sIPE}^N$ *in* $\mathcal{R}$ *is* μ-partial weak-function-hiding, *if for every* $\beta \in [2]$*, every* PPT *adversary* A*, and every sufficiently large security parameter* $\lambda \in \mathbb{N}$*, the adversary's advantage in the following games is bounded by* $\mu(\lambda)$*:*

$$\mathsf{Advt}_A^{\mathbf{sIPE}^N} = \left|\Pr[\mathsf{pwFH}_A^{\mathbf{sIPE}^N}(1^\lambda, 0) = 1] - \Pr[\mathsf{pwFH}_A^{\mathbf{sIPE}^N}(1^\lambda, 1) = 1]\right| \le \mu(\lambda)$$

The game $\mathsf{pwFH}_A^{\mathbf{sIPE}^N}(1^\lambda, b, \beta)$ *for* $\beta = 1$ *proceeds as follows:*

- **Key Generation.** *The challenger generates a master secret key* $\mathsf{msk} = (\mathbf{k}_1, \mathbf{k}_2) \xleftarrow{\$} \mathsf{sIPE.Setup}(1^\lambda, \mathsf{pp})$ *and sends* A *the second slot-key* $\mathbf{k}_2$*.*
- **Function and Input Queries.** *Repeat the following for an arbitrary number of times decided by* A*:*
  - *Upon* A *choosing challenge vectors* $\mathbf{y}_{1,i}^0, \mathbf{y}_{1,i}^1 \mathbf{y}_{2,i} \in \mathcal{R}^{N(\lambda)}$*,* CH *sends* A *a function key* $\mathsf{sSK}_i \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{msk}, \mathbf{y}_{1,i}^b, \mathbf{y}_{2,i})$*.*
  - *Upon* A *choosing a pair of challenge messages* $\mathbf{x}_{1,i}^0, \mathbf{x}_{1,i}^1, \mathbf{x}_{2,i} \in \mathcal{R}^{N(\lambda)}$*,* CH *sends* A *a ciphertext* $\mathsf{sCT}_i \xleftarrow{\$} \mathsf{sIPE.Enc}(\mathsf{msk}, \mathbf{x}_{1,i}^b, \mathbf{x}_{2,i})$*.*
- *Finally* A *outputs a bit* b'*.*

Restriction R′: *Every message query* $(\mathbf{x}_{1,i}^0, \mathbf{x}_{1,i}^1, \mathbf{x}_{2,i})$ *and every function query* $(\mathbf{y}_{1,j}^0 \mathbf{y}_{2,j}^1, \mathbf{y}_{2,j})$ *must satisfy that* $\left\langle \mathbf{x}_{1,i}^0, \mathbf{y}_{1,j}^0 \right\rangle = \left\langle \mathbf{x}_{1,i}^0, \mathbf{y}_{1,j}^1 \right\rangle = \left\langle \mathbf{x}_{1,i}^1, \mathbf{y}_{1,j}^1 \right\rangle$*.*

For $\beta = 2$*, the game proceeds identically except that the second-slot challenge vectors differ, instead of the first-slot vectors.*

Construction Let $\mathsf{pp} = (p, G_1, G_2, G_3, \mathbf{pair})$ be a public parameter that describes bilinear pairing groups with order p; let $\mathcal{R} = \mathbb{Z}_p$. Let **pkIPE** be the ABCP public key IPE scheme and **wIPE** the weakly function hiding IPE scheme constructed in Section 6.3. Our two-slot IPE scheme combines these two schemes in a modular way as follows.

- $\mathsf{sIPE.Setup}(1^\lambda, \mathsf{pp})$ on input 1^λ and public parameter $\mathsf{pp} = (p, G_1, G_2, G_3, \mathbf{pair})$ generates:

  $$(\mathbf{s}, [\mathbf{s}]_1) = \mathsf{pkIPE.Setup}(1^\lambda, (p, G_1)) \quad \text{and} \quad \forall \beta \in [2],\ \mathsf{wMSK}_\beta = \mathsf{wIPE.Setup}(1^\lambda, \mathsf{pp})$$

  It outputs $\mathsf{msk} = (\mathbf{k}_1, \mathbf{k}_2)$ where $\mathbf{k}_\beta = (\mathbf{s}, \mathsf{wMSK}_\beta)$ for $\beta \in [2]$. $\mathbf{s}$ is the shared key and $\mathbf{k}_\beta$ is the β-slot key.

- $\mathsf{sIPE.KeyGen}(\mathsf{msk}, \mathbf{y}_1, \mathbf{y}_2)$ on input msk and first- and second-slot vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ in $\mathcal{R}^N$, first generates a **wIPE** secret key for each vector $\mathbf{y}_\beta$ to obtain

  $$\forall \beta \in [2],\ \mathsf{wSK}_\beta \xleftarrow{\$} \mathsf{wIPE.KeyGen}(\mathsf{wMSK}_\beta, \mathbf{y}_\beta)\ .$$

  Recall that $\mathsf{wSK}_\beta = [\mathbf{wsk}_\beta]_2$ for some vector $\mathbf{wsk}_\beta$. It then homomorphically computes a **pkIPE** secret key of the concatenation $\mathbf{wsk}_1 || \mathbf{wsk}_2$.

  $$\mathsf{sSK} = \mathsf{pkIPE.KeyGen}(\mathbf{s}, (\mathsf{wSK}_1 || \mathsf{wSK}_2)) = [\langle \mathbf{s},\ (\mathbf{wsk}_1 || \mathbf{wsk}_2) \rangle \ ||\ (\mathbf{wsk}_1 || \mathbf{wsk}_2)]_2\ .$$

  It outputs sSK.

- $\mathsf{sIPE.Enc}(\mathsf{msk}, \mathbf{x}_1, \mathbf{x}_2)$ on input msk and first- and second-slot vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\mathcal{R}^N$, first encrypts each $\mathbf{x}_\beta$ using **wIPE**, to obtain

$$\forall \beta \in [2],\ \mathsf{wCT}_\beta \xleftarrow{\$} \mathsf{wIPE.Enc}(\mathsf{wMSK}_\beta, \mathbf{x}_\beta)\ .$$

  Recall that $\mathsf{wCT}_\beta = [\mathbf{wct}_\beta]_1$ for some vector $\mathbf{wct}_\beta$. It then homomorphically computes a **pkIPE** ciphertext of the concatenation $\mathbf{wct}_1||\mathbf{wct}_2$,

$$\mathsf{sCT} = \mathsf{pkIPE.KeyGen}([\mathbf{s}]_1, (\mathsf{wCT}_1||\mathsf{wCT}_2)) = [r \mid\mid r\mathbf{s} + (\mathbf{wct}_1||\mathbf{wct}_2)]_1\ .$$

  (Note that homomorphic evaluation of $\mathsf{pkIPE.KeyGen}$ can be done using only homomorphic addition and scalar multiplication in G_2, as the algorithm does not need to multiply $\mathbf{s}$, $\mathbf{wct}_1$ and $\mathbf{wct}_2$.)

  It outputs sCT.

- **Partial Encryption:** For $\beta \in [2]$, $\mathsf{sIPE.PEnc}(\beta, \mathbf{k}_\beta, \mathbf{x}_\beta)$, on input the β-slot key $\mathbf{k}_\beta$ and β-slot vector $\mathbf{x}_\beta$, proceeds identically as the normal encryption algorithm $\mathsf{sIPE.Enc}$ above except that it does not generate the ciphertext $\mathsf{wCT}_{3-\beta}$ as it does not have $\mathsf{wMSK}_{3-\beta}$, and instead homomorphically computes a **pkIPE** ciphertext of $\mathbf{wct}_1||\mathbf{0}$ if $\beta = 1$ or a ciphertext of $\mathbf{0}||\mathbf{wct}_2$ if $\beta = 2$.

$$\begin{aligned}
&\text{If } \beta = 1, && \mathsf{sCT} = \mathsf{pkIPE.KeyGen}([\mathbf{s}]_1, \underline{(\mathsf{wCT}_1||[\mathbf{0}]_1)}) = \big[r \mid\mid r\mathbf{s} + \underline{(\mathbf{wct}_1||\mathbf{0})}\big]_1\\
&\text{If } \beta = 2, && \mathsf{sCT} = \mathsf{pkIPE.KeyGen}([\mathbf{s}]_1, \underline{([\mathbf{0}]_1||\mathsf{wCT}_2)}) = \big[r \mid\mid r\mathbf{s} + \underline{(\mathbf{0}||\mathbf{wct}_2)}\big]_1
\end{aligned}$$

- $\mathsf{sIPE.Dec}(\mathsf{sSK}, \mathsf{sCT})$ simply homomorphically evaluates the inner product between sCT and sSK. Since the decryption algorithms of **pkIPE** and **wIPE** involve only homomorphically evaluating inner product, this effectively performs two layers of decryption.

$$\mathsf{pkIPE.Dec}(\mathsf{sCT}, \mathsf{sSK}) = \langle \mathsf{sCT}, \mathsf{sSK} \rangle = [\langle \mathbf{wsk}_1||\mathbf{wsk}_2, \mathbf{wct}_1||\mathbf{wct}_2 \rangle]_3$$

  Output 1 iff the obtained encoding encodes zero.

  To see correctness, let sSK *be associated with* $(\mathbf{y}_1, \mathbf{y}_2)$*, and consider the following three cases for* sCT*:*

  - *If* sCT *is associated with* $(\mathbf{x}_1, \mathbf{x}_2)$, $\langle \mathbf{wsk}_1||\mathbf{wsk}_2, \mathbf{wct}_1||\mathbf{wct}_2 \rangle = \langle \mathbf{x}_1||\mathbf{x}_2, \mathbf{y}_1||\mathbf{y}_2 \rangle$.
  - *If* sCT *is associated with* $(\mathbf{x}_1, \mathsf{null})$, $\mathbf{wct}_2$ *is set to* $\mathbf{0}$, $\langle \mathbf{wsk}_1||\mathbf{wsk}_2, \mathbf{wct}_1||\mathbf{wct}_2 \rangle = \langle \mathbf{wsk}_1, \mathbf{wct}_1 \rangle = \langle \mathbf{x}_1||\mathbf{y}_1 \rangle$.
  - *If* sCT *is associated with* $(\mathsf{null}, \mathbf{x}_2)$, $\mathbf{wct}_1$ *is set to* $\mathbf{0}$, $\langle \mathbf{wsk}_1||\mathbf{wsk}_2, \mathbf{wct}_1||\mathbf{wct}_2 \rangle = \langle \mathbf{wsk}_2, \mathbf{wct}_2 \rangle = \langle \mathbf{x}_2||\mathbf{y}_2 \rangle$.

  Therefore, in all three cases, the decryption outputs whether the correct inner product is zero or not.

6.5.1 Special Properties of Our Two-Slot IPE

Our two-slot IPE scheme $\mathbf{sIPE}^N$ has the following special properties:

- <u>LINEARITY IN INPUT AND FUNCTION VECTORS</u> A secret key of **sIPE** encoding vectors $(\mathbf{y}_1, \mathbf{y}_2)$ consists of only encodings in group G_2 of elements that depend linearly in $\mathbf{y}_1$ and $\mathbf{y}_2$. Similarly, a ciphertext of **sIPE** encrypting vectors $(\mathbf{x}_1, \mathbf{x}_2)$ (or $(\mathbf{x}_1, \mathsf{null})$, or $(\mathsf{null}, \mathbf{x}_2)$) consists of only encodings in group G_1 of elements that depend linearly in $\mathbf{x}_1$ and/or $\mathbf{x}_2$.

- LINEARITY IN SHARED KEY Recall that the β-slot key $\mathbf{k}_\beta$ consists of a shared key s and a specific key $\mathbf{k}'_\beta = \mathsf{wMSK}_\beta$. The ciphertext of **sIPE** produced by the partial encryption algorithm sIPE.PEnc using $\mathbf{k}_\beta$ consists of only encodings in group G_1 of elements that depend linearly in the shared key s.

  In particular, this means, given encoding $[\mathrm{s}]_1$ of s in group G_1, one can homomorphically compute a ciphertext $\mathsf{sCT} = \mathsf{sIPE.PEnc}(\beta, \mathbf{k}_\beta, \mathbf{x}_\beta;\ \mathbf{r})$ with knowledge of β, $\mathbf{k}'_\beta$, $\mathbf{x}_\beta$, and $\mathbf{r}$.

- STRONG IND-SECURITY. **sIPE** is IND-secure even when encodings $[\mathrm{s}]_1$ of the shared key s in group G_1 (the group to which ciphertext encodings belong) and the specific keys $(\mathbf{k}'_0, \mathbf{k}'_1)$ are published. This follows directly from the IND-security of **pkIPE**, and the fact that encodings $[\mathrm{s}]_1$ of s in G_1 is exactly the public key of **pkIPE**, and that $(\mathbf{k}'_0, \mathbf{k}'_1)$ only affects the input vectors encrypted using **pkIPE**.

  Lemma 3. *Assume that DDH holds in G_1. For every polynomial N, the above secret key IPE scheme $\mathbf{sIPE}^N$ satisfies IND-security even when encodings of the shared key $[\mathrm{s}]_1$ (in the group where ciphertexts are generated) and the specific keys $(\mathbf{k}'_0, \mathbf{k}'_1)$ are published.*

  (Note that DDH in G_1 is implied by SXDH on the bilinear pairing groups.)

- PARTIAL WEAK-FUNCTION-HIDING **sIPE** satisfies partial weak-function-hiding.

  Lemma 4. *Assume that SXDH holds in bilinear pairing groups. For every polynomial N, the above secret key IPE scheme $\mathbf{sIPE}^N$ satisfies partial weak-function-hiding.*

Proof of Lemma 4. We prove partial weak-function hiding w.r.t. the first slot, that is, the case of $\beta = 1$. Partial weak-function hiding w.r.t. the second slot, that is, the case of $\beta = 2$, follows from the same proof.

To show this, it suffices to show that there exists a simulator S, such that, for every PPT adversary A, its view in game $\mathsf{pwFH}_A^{\mathbf{sIPE}^N}(1^\lambda, b, \beta = 1)$ can be simulated by S^A in the weak-function hiding game $\mathsf{wFH}_S^{\mathbf{wIPE}^N}(1^\lambda, b)$. Therefore, if A can violate the partial weak-function-hiding property w.r.t. the first slot of $\mathbf{sIPE}^N$, S can violate the weak-function-hiding property of $\mathbf{wIPE}^N$, which rules out the existence of such attackers A.

The simulator S^A proceeds as follows:

- It internally samples s, and a master secret key wMSK_2 of $\mathbf{wIPE}^N$, and sends A $\mathbf{k}_2 = (\mathrm{s}, \mathbf{k}'_2 = \mathsf{wMSK}_2)$.

- Upon A choosing challenge vectors $\mathbf{y}^0_{1,i}, \mathbf{y}^1_{1,i}\mathbf{y}_{2,i} \in \mathcal{R}^{N(\lambda)}$, S sends $\mathbf{y}^0_{1,i}, \mathbf{y}^1_{1,i}$ to its challenger as its function query, and receives a secret key wSK_1 for $\mathbf{y}^b_{1,i}$. It then emulates a secret key sSK for A as follows:

  - Generate a secret key wSK_2 for $\mathbf{y}_{2,i}$ using wMSK_2.
  - Homomorphically evaluate a **pkIPE** secret key of $\mathbf{wsk}_1||\mathbf{wsk}_2$ encoded in $\mathsf{wSK}_1||\mathsf{wSK}_2$ as the key generation algorithm sIPE.KeyGen does. This can be done since S knows the master secret key s of **pkIPE**.

  It sends the produced secret key sSK to A.

- Upon A choosing a pair of challenge messages $\mathbf{x}^0_{1,i}, \mathbf{x}^1_{1,i}, \mathbf{x}_{2,i} \in \mathcal{R}^{N(\lambda)}$, S sends $\mathbf{x}^0_{1,i}, \mathbf{x}^1_{1,i}$ to its challenger as its input query, and receives a ciphertext wCT_1 for $\mathbf{x}^b_{1,i}$. It then emulates a ciphertext sCT for A as follows:

  - Generate a ciphertext wCT_2 for $\mathbf{x}_{2,i}$ using wMSK_2.
  - Homomorphically evaluate a $\mathbf{pkIPE}$ ciphertext of $\mathbf{wct}_1||\mathbf{wct}_2$ encoded in $\mathsf{wCT}_1||\mathsf{wCT}_2$ as the encryption algorithm $\mathsf{sIPE.Enc}$ does. Again, this can be done since S knows the master secret key $\mathbf{s}$ of $\mathbf{pkIPE}$.

  It sends the produced ciphertext sCT to A.

- Upon A outputting b', S outputs the same bit.

It is easy to verify that S simulates the view of A perfectly. Therefore, it follows from the weak-function-hiding property of $\mathbf{wIPE}^N$ that $\mathbf{sIPE}^N$ satisfies partial weak-function-hiding. $\square$

7 High-Degree IPE

In this section, we define and construct High-degree IPE (HIPE) schemes, which are multi-input functional encryption for computing high-degree inner products (defined shortly). HIPEs are key tools for constructing collusion reisstant FE for polynomials later. We start with formalizing the notion of HIPE and then construct degree-D HIPE schemes from SXDH on degree-D multilinear pairing groups.

7.1 Definition of HIPE

Degree-D Inner Product: Operation $\langle \mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^D \rangle$, on input D vectors in $\mathcal{R}$, computes

$$\langle \mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^D \rangle = \Sigma_{i\in[N]} \left(\prod_{d\in[D]} \mathbf{x}_i^d \right) \quad \text{in } \mathcal{R} \ .$$

Syntax A family of high-degree IPE schemes $\{\mathbf{hIPE}^{D,N}\}$ in $\mathcal{R}$ consists of for every constant D and polynomial N, a D-ary multi-input functional encryption scheme $\mathbf{hIPE}^{D,N}$ for computing degree-D inner products of length-N vectors in $\mathcal{R}$. The scheme $\mathbf{hIPE}^{D,N}$ has the following syntax.

- *Setup:* $\mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp})$ outputs a master secret key msk.
- *Key Generation:* $\mathsf{hIPE.KeyGen}(\mathsf{msk}, \mathbf{x}^D)$ outputs a secret key hSK encoding a vector $\mathbf{x}^D \in \mathcal{R}^N$.
- *Encryption:* For every $d \in [D-1]$, $\mathsf{hIPE.Enc}^d(\mathsf{msk}, \mathbf{x}^d)$ outputs an encryption hCT^d encrypting a vector $\mathbf{x}^d \in \mathcal{R}^N$.
- *Decryption:* $\mathsf{hIPE.Dec}(\mathsf{hSK}, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^{D-1})$ computes $\mathsf{ZT}(\langle \mathbf{x}^1, \cdots, \mathbf{x}^D \rangle)$, that is, whether the degree-D inner product is zero in $\mathcal{R}$.

Function Hiding of HIPE We define only the selective function-hiding property of HIPE, which is what we achieve and sufficient for constructing selectively IND-secure FE for degree-d polynomials later.

Definition 19 (Selective Function Hiding for HIPE). *We say that a HIPE scheme $\mathbf{hIPE}^{D,N}$ in $\mathcal{R}$ is* selectively μ-function hiding, *if the following holds: For every polynomial Γ and every two ensembles of* *sets of vectors* $\{\ \{\mathbf{u}^1_\gamma, \cdots, \mathbf{u}^D_\gamma\}_{\gamma\in[\Gamma(\lambda)]}\ \}_{\lambda\in\mathbb{N}}$ *and* $\{\ \{\mathbf{v}^1_\gamma, \cdots, \mathbf{v}^D_\gamma\}_{\gamma\in[\Gamma(\lambda)]}\ \}_{\lambda\in\mathbb{N}}$ *satisfying* $\mathbf{u}^d_\gamma, \mathbf{v}^d_\gamma \in \mathcal{R}^{N(\lambda)}$*, and the constraint that*

$$\forall\, I \in [\Gamma]^D, \quad \left\langle \mathbf{u}^1_{I_1}, \cdots, \mathbf{u}^D_{I_D} \right\rangle = \left\langle \mathbf{v}^1_{I_1}, \cdots, \mathbf{v}^D_{I_D} \right\rangle \ ,$$

the two ensembles of distributions $\{\mathcal{D}_0(\lambda)\}_\lambda$ *and* $\{\mathcal{D}_1(\lambda)\}_\lambda$ *defined below are* μ*-indistinguishable.*

$$\{\mathcal{D}_b(\lambda)\}_\lambda = \left\{ \begin{array}{l} \mathsf{msk} \xleftarrow{\$} \mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp}) \\ \left\{\mathsf{CT}^d_\gamma \xleftarrow{\$} \mathsf{hIPE.Enc}^d(\mathsf{msk}, \mathbf{x}^d_\gamma)\right\}_{\gamma\in[\Gamma], d\in[D-1]} \\ \left\{\mathsf{SK}_\gamma \xleftarrow{\$} \mathsf{hIPE.KeyGen}(\mathsf{msk}, \mathbf{x}^D_\gamma)\right\}_{\gamma\in[\Gamma]} \end{array} : \quad \left\{\mathsf{SK}_\gamma,\ \mathsf{CT}^1_\gamma, \cdots, \mathsf{CT}^{D-1}_\gamma\right\}_{\gamma\in[\Gamma]} \right\}_{\lambda\in\mathbb{N}}$$

where $\mathbf{x}^d_\gamma = \mathbf{u}^d_\gamma$ *for every* $d \in [D]$ *when* $b = 0$*, and* $\mathbf{x}^d_\gamma = \mathbf{v}^d_\gamma$ *when* $b = 1$*.*

7.2 Degree-D HIPE from Degree-D MMaps

In this section, we construct a family of function hiding HIPE schemes $\{\mathbf{hIPE}^{D,N}\}$ in $\mathcal{R}$; Every scheme $\mathbf{hIPE}^{D,N}$ for computing degree-D inner product (for a universal constant D) of length-N vectors is built from degree-D MMaps, and has the following canonical form.

Canonical Form We say that a HIPE scheme $\mathbf{hIPE}^{D,N}$ based on degree-D MMaps has canonical form if it satisfies the following properties:

1. For every $d \in [D-1]$, every ciphertext hCT^d in the support of its encryption algorithm $\mathsf{hIPE.Enc}^d(\star, \mathbf{x}^d)$ consists of only encodings in group G_d of ring elements that depend linearly in the encrypted vector $\mathbf{x}$. Moreover, every secret key hSK in the support of its key generation algorithm $\mathsf{hIPE.KeyGen}(\star, \mathbf{y})$ consists of only encodings in group G_D of ring elements that depend linearly in the encoded vector $\mathbf{y}$.

2. Second, the setup, key generation, and encryption algorithms do not use pairing nor the target group G_{D+1}.

3. Third, the decryption algorithm $\mathsf{hIPE.Dec}(\mathsf{hSK}, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^{D-1})$ homomorphically evaluates a polynomial p of degree $\leq D$ on the encodings in the secret key hSK and the ciphertexts hCT^d using degree-D MMaps, and then test whether the output encoding encodes zero. More specifically,

$$\begin{aligned} \mathsf{hIPE.Dec}(\mathsf{hSK}, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^{D-1}) &= \mathsf{ZT}(\ p(\ \mathsf{hSK}, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^d\ )\ ) \\ &= \mathsf{ZT}(\ \left[\langle x^1, \cdots x^D \rangle\right]_{D+1}\ ) \end{aligned}$$

We call HIPE schemes in such canonical form, canonical HIPE schemes.

For simplicity of notation, below we will omit ZT in the decryption equation, and write

$$\begin{aligned} \mathsf{hIPE.Dec}(\mathsf{hSK}, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^{D-1}) &= p(\ \mathsf{hSK}, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^d\ ) \\ &= \left[\langle x^1, \cdots x^D \rangle\right]_{D+1} \end{aligned}$$

to mean that decryption homomorphically evaluates polynomial p and yields encoding of the degree-$(D+1)$ inner product.

7.2.1 Overview of Construction

We construct function hiding canonical HIPE schemes $\{\mathbf{hIPE}^{D,N}\}$, by induction in the degree D.

- **For the base case of** $D = 2$, function hiding degree-2 HIPE is identical to function hiding IPE. In Section 6.3, we constructed such schemes $\{\mathbf{tIPE}^{N}\}$ from SXDH on bilinear maps. It is easy to check that the construction in Section 6.3 indeed has canonical form; see Remark 1.
- **For the induction step**, we show that for any $D \geq 2$, if there exist *canonical* degree-D HIPE schemes $\{\mathbf{hIPE}^{D,N}\}$ with function hiding from SXDH on degree-D MMaps, there exist *canonical* degree-$(D+1)$ HIPE schemes $\{\mathbf{hIPE}^{D+1,N}\}$ with function hiding from SXDH degree-$(D+1)$ MMap.

Below, we describe high-level ideas for the induction step. For simplicity of exposition, we do not explicitly specify the lengths of vectors discussed below, nor the groups in which they are encoded in this high-level description; see details in the formal construction below.

We construct a canonical degree-$(D+1)$ HIPE scheme, denoted by $\mathbf{hIPE}$, by combining a canonical degree-D HIPE scheme, denoted by $\mathbf{dIPE}$, with a two-slot IPE scheme $\mathbf{sIPE}$ constructed in Section 6.5. We will use the following notations for different schemes:

- We use notations $(\mathsf{hCT}^1, \cdots, \mathsf{hCT}^D)$ and hSK to represent the ciphertexts and secret key of the degree-$(D+1)$ HIPE scheme $\mathbf{hIPE}$ we construct.
- For $\mathbf{dIPE}$, we denote by $\mathsf{dCT}^1, \cdots, \mathsf{dCT}^{D-1}$ and dSK its the ciphertexts (at different coordinates) and secret key. Recall that If they encode vectors $(\boldsymbol{\chi}^1, \cdots, \boldsymbol{\chi}^{D-1})$ and $\boldsymbol{\chi}^D$ respectively, decryption produces an encoding of the degree-D inner product.

$$\mathsf{dIPE.Dec}(\mathsf{dSK}, \mathsf{dCT}^1, \cdots, \mathsf{dCT}^{D-1}) = \left[\langle \boldsymbol{\chi}^1, \cdots, \boldsymbol{\chi}^{D-1}, \boldsymbol{\chi}^D \rangle\right]$$

- For $\mathbf{sIPE}$, we denote by sCT and sSK its ciphertext and secret key, and their decryption produces an encoding of the inner product of the encoded vectors $\mathbf{v}^1$ and $\mathbf{v}^2$.

$$\mathsf{sIPE.Dec}(\mathsf{sSK}, \mathsf{sCT}) = \left[\mathbf{v}^1, \mathbf{v}^2\right]$$

  Recall that in the two-slot scheme the encoded vectors $\mathbf{v}^1, \mathbf{v}^2$ are viewed as consisting of two slots $\mathbf{v}^\beta = \mathbf{v}^\beta_1 || \mathbf{v}^\beta_2$, and the scheme satisfies several special properties. As we will see later, the sepcial two-slot structure and properties of $\mathbf{sIPE}$ are crucial for the security proof. In the high-level description of the construction below, it will, however, be convenient to temporarily ignore these special features, and simply think of $\mathbf{sIPE}$ as a normal function hiding IPE scheme.

Overview of Our Degree-$(D+1)$ HIPE Scheme. To achieve functionality, we need to specify how to generate ciphertexts and secret key for input vectors $\mathbf{x}^1, \cdots, \mathbf{x}^D$ and $\mathbf{x}^{D+1}$, so that,

$$\mathsf{hIPE.Dec}(\mathsf{hSK}, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^D) = \left[\langle \mathbf{x}^1, \cdots, \mathbf{x}^D, \mathbf{x}^{D+1} \rangle\right] .$$

Observe that a degree-$(D+1)$ inner product of $\mathbf{x}^1, \cdots \mathbf{x}^{D+1}$, can be computed as the inner product between $\mathbf{x}^{D+1}$ and the coordinate-wise product of the first D vectors $\prod_{d \in [D]} \mathbf{x}^d$, denoted as $\mathbf{x}^{\leq D}$, that is,

$$y = \langle \mathbf{x}^1, \cdots \mathbf{x}^{D+1} \rangle = \left\langle \prod_{d \in [D]} \mathbf{x}^d, \mathbf{x}^{D+1} \right\rangle = \langle \mathbf{x}^{\leq D}, \mathbf{x}^{D+1} \rangle$$

Therefore, if the decryptor obtains a pair of **sIPE** ciphertext and secret key $(\mathsf{sCT}, \mathsf{sSK})$ for $(\mathbf{x}^{\leq D}, \mathbf{x}^{D+1})$, he/she can decrypt to obtain $[y]$. To do so, our new scheme **hIPE** simply publishes sSK as its secret key,

$$\textbf{Secret key of hIPE:} \qquad \mathsf{hSK} = \mathsf{sSK} \leftarrow \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{x}^{D+1}) \ .$$

However, it cannot directly publish a ciphertext of $\mathbf{x}^{\leq D}$, as $\mathbf{x}^{\leq D}$ is the product of D input vectors, but each encryption algorithm $\mathsf{hIPE.Enc}^d$ receives only a single vector $\mathbf{x}^d$ as input and cannot compute $\mathbf{x}^{\leq D}$. The idea is to include in the D ciphertexts $\mathsf{hCT}^1, \cdots, \mathsf{hCT}^D$ of **hIPE**, ciphertexts and secret keys of the degree-D scheme, so that the decryptor can combine them to generate a ciphertext sCT of $\mathbf{x}^{\leq D}$.

Towards this end, recall that every **sIPE** ciphertext sCT consists of many encodings $\{\mathsf{sCT}_l\}_{l\in[L]}$. If the element encoded sct_l in every encoding sCT_l can be expressed as the inner product of D vectors

$$\textbf{Condition C:} \qquad \mathrm{sct}_l = \left\langle \chi_l^1, \cdots \chi_l^D \right\rangle , \text{ and each } \chi_l^d \text{ depends only on } \mathbf{x}^d \ ,$$

it suffices to encode these vectors in a tuple $(\mathsf{dCT}_l^1, \cdots \mathsf{dCT}_l^{D-1}, \mathsf{dSK}_l)$ of $D-1$ ciphertexts and one secret key of **dIPE** using an independently sampled master secret key dMSK_l, from which the decryptor can obtain exactly sCT_l. Thus, the D ciphertexts $\mathsf{hCT}^1, \cdots, \mathsf{hCT}^D$ of our new scheme **hIPE** consists of exactly one such tuple ($\mathsf{dCT}_l^1, \cdots \mathsf{dCT}_l^{D-1}, \mathsf{dSK}_l$ for every l), namely,

$$\textbf{Ciphertext of hIPE:} \qquad \mathsf{hCT}^d = \begin{cases} \left\{\mathsf{dCT}_l^d \leftarrow \mathsf{dIPE.Enc}(\mathsf{dMSK}_l, \chi_l^d)\right\}_{l\in[L]} & \text{if } d \leq D \\ \left\{\mathsf{dSK}_l \leftarrow \mathsf{dIPE.KeyGen}(\mathsf{dMSK}_l, \chi_l^D)\right\}_{l\in[L]} & \text{if } d = D \end{cases} .$$

Given $(\mathsf{hCT}^1, \cdots, \mathsf{hCT}^D)$ and hSK as specified above, the decryptor proceeds in two steps:

1. First, decrypt for every l, the tuple $(\mathsf{dCT}_l^1, \cdots \mathsf{dCT}_l^{D-1}, \mathsf{dSK}_l)$ using the decryption algorithm of **dIPE** to obtain sCT_l; put them together to get a ciphertext sCT of $\mathbf{x}^{\leq D}$.

2. Then, decrypt the obtained ciphertext sCT using the decryption algorithm of **sIPE** and secret key $\mathsf{hSK} = \mathsf{sSK}$ of $\mathbf{x}^{D+1}$ to obtain an encoding of the final inner product y, as illustrated below.

$$\underbrace{\mathsf{hCT}^1 = \{\mathsf{dCT}_l^1\}_l\ , \quad \cdots\ , \quad \mathsf{hCT}^{D-1} = \{\mathsf{dCT}_l^{D-1}\}_l\ , \quad \mathsf{hCT}^D = \{\mathsf{dSK}_l\}_l}_{\displaystyle \underbrace{\text{Decrypt to } \mathsf{sCT} \qquad\qquad \mathsf{hSK} = \mathsf{sSK}}_{\text{Decrypt to } [y]}}$$

Setting Condition C – A First Attempt We now argue that **Condition C** above indeed holds. This crucially relies on the fact that the two-slot IPE scheme **sIPE** has the special property that the elements $\{\mathrm{sct}_l\}$ encoded in its ciphertext sCT, depends *linearly* in the encrypted vector $\mathbf{x}^{\leq D}$ and randomness $\mathbf{r}$ of encryption. More specifically, when the master secret key sMSK is fixed, each element sct_l is the output of a linear function $h_l^{(\mathsf{sMSK})}$ on input $(\mathbf{x}^{\leq D}, \mathbf{r})$,

$$\mathsf{sCT} = \mathsf{sIPE.Enc}(\mathsf{sMSK},\ \mathbf{x}^{\leq D}\ ;\ \mathbf{r}) = \{[\mathrm{sct}_l]\}_l\ ,$$
$$\text{with } \mathrm{sct}_l = h_l^{(\mathsf{sMSK})}(\mathbf{x}^{\leq D},\ \mathbf{r}) = \left\langle \mathbf{c}_l^{(\mathsf{sMSK})}, (\mathbf{x}^{\leq D}||\mathbf{r}) \right\rangle\ ,$$

where $\mathbf{c}_l^{(\mathsf{sMSK})}$ is the coefficient vector of $h_l^{(\mathsf{sMSK})}$. Then, since $\mathbf{x}^{\leq D} = \mathbf{x}^1 \cdots \mathbf{x}^D$, we can represent sct_l as the inner product of D vectors $\chi_l^1, \cdots, \chi_l^D$, each depending on only one input vector $\mathbf{x}^D$, as follows:

$$\mathrm{sct}_l = \langle \chi_l^1, \chi_l^2, \cdots \chi_l^D \rangle \qquad \chi_l^d = \begin{cases} \mathbf{x}^1 || \underline{\mathbf{r}} & \text{if } d = 1 \\ \mathbf{x}^d || \mathbf{1} & \text{if } 1 < d < D \\ (\mathbf{x}^D || \mathbf{1})\underline{(\mathbf{c}_l^{(\mathsf{sMSK})})} & \text{if } d = D \end{cases} .$$

Therefore, as discussed above, encrypting the vectors $\{\chi_l^d\}$ in the ciphertexts of **hIPE** guarantees that the decryptor can obtain sCT from the ciphertexts, and decrypting the ciphertext sCT further produces an encoding of the correct output y.

A Security Issue The above way of setting the vectors $\{\chi_l^d\}_{d,l}$ achieves functionality, but, does not guarantee security. A security issue stems from the fact that the randomness $\mathbf{r}$ used for generating the ciphertext sCT is hardcoded entirely in the input vectors $\{\chi_l^1\}_l$ encrypted at the first coordinate. Consider a simple scenario where a single ciphertext of **hIPE** at the first coordinate, two ciphertexts at each other coordinate, and a single secret key, are published:

$$\begin{array}{ccccc} \mathsf{hCT}^1, & \mathsf{hCT}_0^2, & \cdots, & \mathsf{hCT}_0^D, & \mathsf{hSK} \\ & \mathsf{hCT}_1^2, & \cdots, & \mathsf{hCT}_1^D & \end{array}$$

Since the randomness $\mathbf{r}$ is embedded in hCT^1, different combinations of ciphertexts, say hCT^1 and $\mathsf{hCT}_{b_2}^2 \cdots \mathsf{hCT}_{b_D}^D$, produce **sIPE** ciphertexts encrypting different vectors, $\mathbf{x}^1\mathbf{x}_{b_2}^2 \cdots \mathbf{x}_{b_D}^D$, but using the same random coins $\mathbf{r}$. The security of **sIPE** does not hold when attackers can observe ciphertexts with shared randomness, and in particular, information of the encrypted vector $\mathbf{x}^1\mathbf{x}_{b_2}^2 \cdots \mathbf{x}_{b_D}^D$ may be revealed. On the other hand, the function hiding property requires that only the final degree-$(D+1)$ inner products $\mathbf{x}^1\mathbf{x}_{b_2}^2 \cdots \mathbf{x}_{b_D}^D\mathbf{x}^{D+1}$ are revealed, and nothing else.

Setting Condition C, Right To address this security issue, we need to ensure that ciphertexts sCT produced by different combinations of ciphertexts of **hIPE** correspond to (at the very least) distinct randomness. To do so, we embed fresh randomness $\mathbf{r}^d$ in ciphertexts at every coordinate by modifying the encrypted vectors χ_l^d to the following:

$$\chi_l^d = \begin{cases} \mathbf{x}^d || \underline{\mathbf{r}^d} & \text{if } d < D \\ (\mathbf{x}^D || \underline{\mathbf{r}^D})(\mathbf{c}_l^{(\mathsf{sMSK})}) & \text{if } d = D \end{cases}$$

Note that the inner products of these vectors correspond exactly to a ciphertext sCT generated using random coins $\mathbf{r}^{\leq D} = \prod_{d \in [D]} \mathbf{r}^d$, aggregating the shares of randomness embedded at all coordinates. That is,

$$\langle \chi_1, \cdots, \chi_D \rangle = \left\langle \mathbf{c}_l^{(\mathsf{sMSK})}, (\mathbf{x}^{\leq D} || \mathbf{r}^{\leq D}) \right\rangle = h_l^{(\mathsf{sMSK})}(\mathbf{x}^{\leq D},\ \mathbf{r}^{\leq D}) = \mathrm{sct}_l ,$$
$$\mathsf{sCT} = \{[\mathrm{sct}_l]\}_l = \mathsf{sIPE.Enc}(\mathsf{sMSK},\ \mathbf{x}^{\leq D}\ ;\ \mathbf{r}^{\leq D}) .$$

In the simple scenario above, combining $\mathsf{hCT}^1, \mathsf{hCT}_{b_2}^2 \cdots, \mathsf{hCT}_{b_D}^D$ now produces sCT with randomness $\mathbf{r}^1\mathbf{r}_{b_2}^2 \cdots \mathbf{r}_{b_D}^D$, which is distinct for each combination.

Having distinct randomness is still not enough for applying the security of **sIPE**, which requires independently and uniformly sampled randomness. The security analysis of the above scheme turns out to be quite complicated, and in fact for security to hold, the scheme needs to further pad the vectors χ_l^d with zeros, serving as redundant space for hardwiring information in different hybrids in the security proof. Below, we first describe our construction formally, and then move to describe ideas of the security proof.

7.2.2 Construction — The Induction Step

Fix any polynomial N. We construct a canonical degree-$(D+1)$ HIPE scheme $\mathbf{hIPE} = \mathbf{hIPE}^{D+1,N}$ for input length N with algorithms $(\mathsf{hIPE.Setup}, \mathsf{hIPE.Enc}, \mathsf{hIPE.KeyGen}, \mathsf{hIPE.Dec})$ relying on the following building blocks:

- A canonical degree-D HIPE scheme $\mathbf{dIPE} = \mathbf{hIPE}^{D,M} = (\mathsf{dIPE.Setup}, \mathsf{dIPE.Enc}, \mathsf{dIPE.KeyGen}, \mathsf{dIPE.Dec})$ from SXDH on degree-D MMaps, for a specific input length M specified below.
- The two-slot IPE scheme $\mathbf{sIPE} = \mathbf{sIPE}^N = (\mathsf{sIPE.Setup}, \mathsf{sIPE.Enc}, \mathsf{sIPE.KeyGen}, \mathsf{sIPE.Dec})$ from SXDH on bilinear maps constructed in Section 6.5. Let $L = L(\lambda)$ denote the length of ciphertexts of the scheme $\mathbf{sIPE}^N$.
- Degree-$D+1$ multi-linear pairing groups described by $\mathsf{pp} = (p, G_1, \cdots, G_{D+1}, G_{D+2}, \mathbf{pair})$.

  It would be convenient to assume that the multilinear pairing groups support a slightly richer interface. Namely, one can pair encodings in the first D groups to obtain an encoding in an intermediate target group denoted as $G_{\leq D}$, which can further be paired with encodings in group G_{D+1} to yield encodings in the actual target group G_{D+2}. The presentation of our scheme becomes simpler using this interface. As we discuss later, since our scheme has canonical form, this interface is not necessary, since decryption simply homomorphically evaluate a polynomial of degree $\leq D+1$, which can be done using degree-$(D+1)$ MMaps with standard interface. (See discussion on canonical form and Remark 2.)

The scheme $\mathbf{hIPE}$ proceeds as follows. We inline analysis of correctness in italic font in the description of the construction below.

- $\mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp})$ samples a master secret key of $\mathbf{sIPE}$ and L independent master secret keys of $\mathbf{dIPE}$.

$$\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2) \xleftarrow{\$} \mathsf{sIPE.Setup}(1^\lambda, (p, G_{\leq D}, G_{D+1}, G_{D+2}))$$
$$\left\{\mathsf{dMSK}_l \xleftarrow{\$} \mathsf{dIPE.Setup}(1^\lambda, (p, G_1, \cdots, G_D, G_{\leq D}))\right\}_{l\in[L]}$$

  Output master secret key $\mathsf{hMSK} = (\mathsf{sMSK}, \{\mathsf{dMSK}_l\}_{l\in[L]})$.

- $\mathsf{hIPE.KeyGen}(\mathsf{hMSK}, \mathbf{x}^{D+1})$ generates a secret key of the $\mathbf{sIPE}$ scheme using sMSK, encoding $\mathbf{x}^{D+1}$ in the first slot, and the zero vector $\mathbf{0}$ in the second slot.

$$\mathsf{sSK} \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{x}^{D+1}, \mathbf{0})$$

  Output $\mathsf{hSK} = \mathsf{sSK}$. (Note that the second-slot vector encoded in sSK is set to zero.)

- $\mathsf{hIPE.Enc}^d(\mathsf{hMSK}, \mathbf{x}^d)$ for $d \in [D]$ proceeds in the following steps:

  1. Sample $\mathbf{r}^d \leftarrow \mathcal{R}^5$.
     (The encryption algorithm of $\mathbf{sIPE}$ *samples only 5 random elements.)*

2. Prepare vectors $\{\chi_l^d\}_{l\in[L]}$ as follows.
   By construction, the partial encryption algorithm sIPE.PEnc of **sIPE** produces a set of L encodings
   $$\mathsf{sCT} = [\mathrm{sct}_1, \cdots \mathrm{sct}_L]_{\leq D} = \mathsf{sIPE.PEnc}(1, \mathbf{k}_1, \mathbf{u};\ \mathbf{w})\ ,$$
   where each encoded element sct_l depends linearly on $\mathbf{u}$ and $\mathbf{w}$. Let $\mathrm{sct}_l = h_l^{(\mathbf{k}_1)}(\mathbf{u}, \mathbf{w})$ denote the linear function that computes the l^{th} encoded element, and $\mathbf{c}_l^{(\mathbf{k}_1)}$ its coefficient vector, that is, $\mathrm{sct}_l = \left\langle \mathbf{c}_l^{(\mathbf{k}_1)}, \mathbf{u}||\mathbf{w} \right\rangle$.
   Then, set the vector χ_l^d as follows.
   $$\chi_l^d = \begin{cases} \mathbf{x}^d \mid\mid \underline{\mathbf{r}^d} & \text{if } d < D \\ (\mathbf{x}^D || \underline{\mathbf{r}^D})(\mathbf{c}_l^{(\mathbf{k}_1)}) & \text{if } d = D \end{cases}$$
   The length of χ_l^d is $M' = |\chi_l^d| = N + 5$.
   Note that encodings of the inner products of vectors $\chi_l^d, \cdots \chi_l^D$, for every l, is a **sIPE** *ciphertext of the vector $\mathbf{x}^{\leq D} = \prod_{d\in[D]} \mathbf{x}^d$ (in the first slot), generated using the first slot key $\mathbf{k}_1$ and random elements $\mathbf{r}^{\leq D} = \prod_{d\in[D]} \mathbf{r}^d$. That is,*
   $$\left\{ [\langle \chi_1^1 \cdots, \chi_l^D \rangle]_{\leq D} = \left[h_l^{(\mathbf{k}_1)}(\mathbf{x}^{\leq D}, \mathbf{r}^{\leq D}) \right]_{\leq D} \right\}_{l\in[L]}$$
   $$= \mathsf{sCT} = \mathsf{sIPE.PEnc}(0, \mathbf{k}_1, \mathbf{x}^{\leq D};\ \mathbf{r}^{\leq D})$$

3. Pad the above vectors with zeros to get $\{\mathbf{X}_l^d\}_{l\in[L]}$.
   $$\mathbf{X}_l^d = \chi_l^d || \mathbf{0}, \qquad \text{where } M = |\mathbf{X}_l^d| = 2(D-1)|\chi_l^d| + 1 = 2(D-1)(N+5) + 1 = \Theta(DN)$$
   Since padding with zero does not change inner products, we still have
   $$\left\{ [\langle \mathbf{X}_1^1 \cdots, \mathbf{X}_l^D \rangle]_{\leq D} \right\}_{l\in[L]} = \mathsf{sCT} = \mathsf{sIPE.PEnc}(0, \mathbf{k}_1, \mathbf{x}^{\leq D};\ \mathbf{r}^{\leq D})\ .$$

4. Encrypt $\{\mathbf{X}_l^d\}_l$ in one of the following two ways, depending on whether $d = D$: If $d < D$, it encrypts every $\mathbf{X}_l^d$ at the d^{th} coordinate, using **dIPE** and master secret key dMSK_l. Otherwise, if $d = D$, it encodes every $\mathbf{X}_l^D$ as a function using **dIPE** and master secret key dMSK_l. Formally,
   $$\mathsf{hCT}^d = \begin{cases} \left\{ \mathsf{dCT}_l^d \xleftarrow{\$} \mathsf{dIPE.Enc}(\mathsf{dMSK}_l, \mathbf{X}_l^d) \right\}_{l\in[L]} & \text{if } d < D \\ \left\{ \mathsf{dSK}_l \xleftarrow{\$} \mathsf{dIPE.KeyGen}(\mathsf{dMSK}_l, \mathbf{X}_l^D) \right\}_{l\in[L]} & \text{if } d = D \end{cases}$$

Finally, output hCT^d.

- $\mathsf{hIPE.Dec}(\mathsf{hSK}, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^D)$ parses $\mathsf{hCT}^d = \{\mathsf{dCT}_l^d\}_l$ for $d \leq D$ and $\mathsf{hCT}^D = \{\mathsf{dSK}_l\}_l$ as ciphertexts and secret keys of **dIPE**, and $\mathsf{hSK} = \mathsf{sSK}$ as a secret key of **sIPE**. Decryption proceeds in two steps

1. Decrypt, for every $l \in [L]$, the tuple $(\mathsf{dSK}_l, \mathsf{dCT}_l^1, \cdots, \mathsf{dCT}_l^{D-1})$ using $\mathbf{dIPE}$, obtaining a ciphertext of $\mathbf{sIPE}$.

$$\mathsf{sCT} = \left\{\mathsf{sCT}_l = \mathsf{dIPE.Dec}(\mathsf{dSK}_l, \mathsf{dCT}_l^1, \cdots, \mathsf{dCT}_l^{D-1})\right\}_{l\in[L]}$$

2. Decrypt sCT using sSK,

$$[y]_{D+2} = \mathsf{sIPE.Dec}(\mathsf{sSK}, \mathsf{sCT})$$

Output 1 iff the obtained encoding $[y]_{D+1}$ encodes zero.

For correctness, we argue that y equals to $\langle \mathbf{x}^1, \cdots, \mathbf{x}^D \rangle$.

By construction, the tuple $(\mathsf{dCT}_l^1, \cdots, \mathsf{dCT}_l^{D-1}, \mathsf{dSK}_l)$ encodes vectors $(\mathbf{X}_l^1, \cdots, \mathbf{X}_l^D)$. Therefore, it follows from the correctness of $\mathbf{dIPE}$,

$$\mathsf{sCT} = \left\{\mathsf{sCT}_l = \mathsf{dIPE.Dec}(\mathsf{dSK}_l, \mathsf{dCT}_l^1, \cdots, \mathsf{dCT}_l^{D-1})\right\}_{l\in[L]} = \left\{[\langle \mathbf{X}_l^1, \cdots \mathbf{X}_l^D\rangle]_{\le D}\right\}_{l\in[L]}$$

which as analyzed above is exactly the output of $\mathsf{sIPE.PEnc}(0, \mathbf{k}_1, \mathbf{x}^{\le D};\ \mathbf{r}^{\le D})$.

Then, in the second step, by the correctness of $\mathbf{sIPE}$*, decrypting* sCT *with* sSK *encoding vector* $\mathbf{x}^{D+1}$ *produces*

$$\mathsf{sIPE.Dec}(\mathsf{sSK}, \mathsf{sCT}) = \left[\langle \mathbf{x}^{\le D},\ \mathbf{x}^{D+1}\rangle\right]_{D+2} = \left[\langle \mathbf{x}^1, \cdots, \mathbf{x}^{D+1}\rangle\right]_{D+2} = [y]_{D+2}\,.$$

This concludes the correctness of the scheme $\mathbf{hIPE}^{D,N}$.

hIPE Has Canonical Form We verify the following three properties.

- First, we show that for every $d \in [D]$, every ciphertext hCT^d consists of only encodings in group G_d of elements that depend linearly in the encrypted vector $\mathbf{x}^d$, and every secret key hSK consists of encodings in group G_{D+1} of elements linear in $\mathbf{x}^{D+1}$. By construction, every ciphertext hCT^d of $\mathbf{x}^d$ consists of a set of ciphertexts $\{\mathsf{dCT}_l^d\}$ at coordinate d (if $d < D$) or secret key $\{\mathsf{dSK}_l\}$ (if $d = D$) of $\mathbf{dIPE}$ encoding vectors $\{\mathbf{X}_l^d\}$ derived from $\mathbf{x}^d$. Note that by definition, vector $\mathbf{X}_l^d$ for every d, l is linear in $\mathbf{x}^d$. Thus, by the induction hypothesis that $\mathbf{dIPE}$ has canonical form, every hCT_l^d consists of only encodings in group G_d of elements linear in $\mathbf{X}_l^d$, which in turn are linear in $\mathbf{x}^d$. Moreover, every secret key hSK is simply a secret key sSK of $\mathbf{sIPE}$ encoding the same vector $\mathbf{x}^{D+1}$. By the fact that $\mathbf{sIPE}$ secret keys consist of only encodings in G_{D+1} of elements linear in $\mathbf{x}^{D+1}$, so are secret keys of $\mathbf{hIPE}$.
- Second, since $\mathbf{dIPE}$ satisfies that its setup, key generation, and encryption algorithms do not use pairing nor the target group $G_{\le D}$, it is easy to verify that $\mathbf{hIPE}$'s setup, key generation and encryption algorithms also do not use pairing, nor the target group G_{D+2}.
- Third, we argue that the decryption algorithm of $\mathbf{hIPE}$ can be carried out by homomorphically evaluating a polynomial q of degree $\le D+1$. By the fact that $\mathbf{dIPE}$ has canonical form, which means that its decryption homomorphically evaluates a polynomial p of degree $\le D$ over the encodings contained in the ciphertexts and secret key under decryption. Therefore, in the first decryption step of of $\mathbf{hIPE}$, the decryptor does

$$\mathsf{sCT} = \left\{\mathsf{dIPE.Dec}(\mathsf{dSK}_l, \mathsf{dCT}_l^1, \cdots, \mathsf{dCT}_l^{D-1})\right\}_{l\in[L]} = \left\{p(\mathsf{dSK}_l, \mathsf{dCT}_l^1, \cdots, \mathsf{dCT}_l^{D-1})\right\}_{l\in[L]}$$

Furthermore, by the fact that decryption of **sIPE** simply homomorphically evaluate inner product, we have

$$\begin{aligned}[y]_{D+2} = \mathsf{sIPE.Dec}(\mathsf{sSK}, \mathsf{sCT}) &= \langle \mathsf{sSK},\ \mathsf{sCT}\rangle \\ &= \left\langle \mathsf{sSK},\ \left\{p(\mathsf{dSK}_l, \mathsf{dCT}_l^1, \cdots, \mathsf{dCT}_l^{D-1})\right\}_{l,\in[L]}\right\rangle \\ &= q(\mathsf{hSK}, \mathsf{hCT}^D, \mathsf{hCT}^1, \cdots, \mathsf{hCT}^{D-1}),\end{aligned}$$

where q is the polynomial corresponding to the composition of p and $\langle \bullet, \bullet\rangle$, up to appropriate re-arrangement of input variables. Clearly the degree of q is no larger than $D+1$.

Remark 2 (Standard MMaps Suffices)**.** *In the above construction, we used a slightly richer interface of degree-$(D+1)$ MMaps, where the first D groups can be paired together into an intermediate target group $G_{\leq D}$, which can further be paired with G_{D+1} into target group G_{D+2}. We now argue that this richer interface is not necessary, and our construction can be instantiated with standard MMaps supporting only pairing all groups together to the target group G_{D+2}.*

First, it follows from the fact that **dIPE** *has canonical form, that the setup, key generation, and encryption algorithms of* **hIPE** *actually do not use the intermediate pairing, nor the intermediate target group $G_{\leq D}$. Second, by that* **hIPE** *has canonical form, its decryption algorithm only involves homomorphically evaluating a degree $\leq D+1$ polynomial over the encodings contained in the ciphertexts and secret keys of* **hIPE***, which can be done using* standard *degree-$(D+1)$ MMaps.*

7.2.3 Efficiency

We analyze the maximum time $\mathsf{Time}^{D+1}(N)$ the key generation and encryption algorithm of $\mathbf{hIPE} = \mathbf{hIPE}^{D+1,N}$ runs. In the base case, when $D+1=2$, the **hIPE** scheme is simply a standard **IPE** scheme; our construction in Section 6 has $\mathsf{Time}^2(N) = \mathrm{poly}(\lambda)N$.

In the induction step from degree D to $D+1$, **hIPE** is constructed from the degree-D HIPE scheme $\mathbf{dIPE}^{D,M}$ with efficiency $\mathsf{Time}^D(M)$ and the $\mathbf{sIPE}^N$ scheme whose encryption and key generation time is $\Theta(N)$. By construction, the key generation time is much smaller than the encryption time. Thus, we focus on analyzing the latter. The encryption algorithm $\mathsf{hIPE.Enc}$ generates **dIPE** ciphertexts or secret keys of vectors $\{\mathbf{X}_l^d\}$ each of length $M = \Theta(DN)$, and l is the length of ciphertexts of $\mathbf{sIPE}^N$, which is $\Theta(N)$. Let c be a sufficiently large universal constant.

$$\begin{aligned}\mathsf{Time}^{D+1}(N) = &\ \Theta(l \times \mathsf{Time}^D(M)) = \Theta(N) \times \mathsf{Time}^D(\Theta(DN)) \\ \leq &\ cN\ \mathsf{Time}^D(cDN) \\ \leq &\ cN\ \left(c^2DN \times \mathsf{Time}^{D-1}(c(D-1)\ (cDN))\right) \leq c^3DN^2\ \mathsf{Time}^{D-1}(c^2D^2N) \\ &\quad \cdots \\ \leq &\ c^{\tilde{\Theta}(D^2)}N^{D-1}\ \mathsf{Time}^2((cD)^{D-1}N) \\ \leq &\ N^D\,\mathrm{poly}(\lambda)\end{aligned}$$

where the third line used the fact that Time^d for any d is a non-decreasing function.

The decryption algorithm of **hIPE** scheme simply homomorphically evaluates a degree $\leq D+1$ polynomial over all encodings contained in ciphertexts and secret keys, followed by a zero test. Thus, its runtime is at most $((D+1) \times T^{D+1}(N))^{D+1}\,\mathrm{poly}(\lambda) = N^{\Theta(D^2)}\,\mathrm{poly}(\lambda)$.

7.3 Security Proof

In this section, we prove that **hIPE** is selectively function hiding.

Proposition 1. *Assume SXDH on degree-$(D+1)$ multilinear pairing groups, and that* **dIPE** *is selectively function hiding. The scheme* **hIPE** *described above is also selectively function hiding.*

Fix any polynomial Γ and any two ensembles of sets of vectors $\{\{\mathbf{u}^1_\gamma, \cdots, \mathbf{u}^{D+1}_\gamma\}_{\gamma\in[\Gamma(\lambda)]}\}_{\lambda\in\mathbb{N}}$ and $\{\{\mathbf{v}^1_\gamma, \cdots, \mathbf{v}^{D+1}_\gamma\}_{\gamma\in[\Gamma(\lambda)]}\}_{\lambda\in\mathbb{N}}$, such that, $\mathbf{u}^d_\gamma, \mathbf{v}^d_\gamma \in \mathcal{R}^{N(\lambda)}$ and the following holds.

$$\forall\, I \in [\Gamma]^{D+1}, \quad \left\langle \mathbf{u}^1_{I_1}, \cdots, \mathbf{u}^{D+1}_{I_D} \right\rangle = \left\langle \mathbf{v}^1_{I_1}, \cdots, \mathbf{v}^{D+1}_{I_D} \right\rangle$$

We need to show the indistinguishability of ensembles of distributions $\{\mathcal{D}_0(\lambda)\}_\lambda$ and $\{\mathcal{D}_1(\lambda)\}_\lambda$ defined below.

$$\{\mathcal{D}_b(\lambda)\}_\lambda = \left\{ \begin{array}{l} \mathsf{hMSK} \xleftarrow{\$} \mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp}) \\ \left\{\mathsf{hCT}^d_\gamma \xleftarrow{\$} \mathsf{hIPE.Enc}^d(\mathsf{hMSK}, \mathbf{x}^d_\gamma)\right\}_{\gamma\in[\Gamma], d\in[D]} \\ \left\{\mathsf{hSK}_\gamma \xleftarrow{\$} \mathsf{hIPE.KeyGen}(\mathsf{msk}, \mathbf{x}^{D+1}_\gamma)\right\}_{\gamma\in[\Gamma]} \end{array} : \mathsf{pp},\ \left\{\mathsf{hSK}_\gamma,\ \mathsf{hCT}^1_\gamma, \cdots, \mathsf{hCT}^{D+1}_\gamma\right\}_{\gamma\in[\Gamma]} \right\}_{\lambda\in\mathbb{N}}$$

where $\mathbf{x}^d_\gamma = \mathbf{u}^d_\gamma$ when $b = 0$, and $\mathbf{x}^d_\gamma = \mathbf{v}^d_\gamma$ when $b = 1$.

Below, we first describe the high-level ideals for the security proof and then provide the formal proof.

7.3.1 Overview of Security Proof

We reduce function hiding of **hIPE** to function hiding of **dIPE** and the special security properties of **sIPE**. Consider proving the indistinguishability of distributions $\mathcal{D}_0 = \mathcal{D}_0(\lambda)$ and $\mathcal{D}_1 = \mathcal{D}_1(\lambda)$ above, where up to Γ ciphertexts are published at every coordinate d. By construction, any combination of ciphertexts $(\mathsf{hCT}^1_{I_1}, \cdots, \mathsf{hCT}^D_{I_D})$ at different coordinates indexed by $I \in [\Gamma]^{D+1}$, yields a **sIPE** ciphertext denoted by sCT_I, satisfying that

$$\mathsf{sCT}_I = \mathsf{sIPE.PEnc}(0, \mathbf{k}_1, \mathbf{x}^{\le D}_I;\ \mathbf{r}^{\le D}_I), \qquad \text{where } \mathbf{x}^{\le D}_I = \prod_{d\in[D]} \mathbf{x}^d_{I_d},\ \mathbf{r}^{\le D}_I = \prod_{d\in[D]} \mathbf{r}^d_{I_d}$$

and $\mathbf{k}_1$ is the first slot key of **sIPE** contained in hMSK. This ciphertext sCT_I can then be decrypted with any secret key $\mathsf{hSK}_{I_{D+1}} = \mathsf{sSK}_{I_{D+1}}$ to obtain an encoding of the output inner product

$$[y_I]_{D+1} = \left[\left\langle \mathbf{x}^1_{I_1}, \cdots \mathbf{x}^{D+1}_{I_{D+1}} \right\rangle\right]_{D+1} = \mathsf{sIPE.Dec}(\mathsf{sCT}_I, \mathsf{sSK}_{I_{D+1}})\,.$$

A natural first idea for proving the security of **hIPE** is using the security of **dIPE** to argue that the set of ciphertexts in distribution $\mathcal{D}_b$ reveals nothing except from the set of **sIPE** ciphertexts $\{\mathsf{sCT}_I\}_I$ that can be possibly computed, and then reduce the indistinguishability of $\mathcal{D}^0$ and $\mathcal{D}^1$ to the indistinguishability of $\{\mathsf{sCT}_I\}_I$ at the presence of the secret keys $\{\mathsf{sSK}_{I_{D+1}}\}_I$ included these distributions. The latter indistinguishability seems to follow from the function hiding property of **sIPE**, since the inner products of vectors encoded in $\{\mathsf{sCT}_I, \mathsf{sSK}_{I_{D+1}}\}_I$ are identical in $\mathcal{D}_0$ and $\mathcal{D}_1$, that is,

$$\left\{\left\langle \mathbf{u}^{\le D}_I, \mathbf{u}^{D+1}_{I_{D+1}} \right\rangle\right\}_I = \left\{\left\langle \mathbf{v}^{\le D}_I, \mathbf{v}^{D+1}_{I_{D+1}} \right\rangle\right\}_I\,.$$

For the above idea to go through, the two building blocks **dIPE** and **sIPE** need to satisfy very strong (potentially impossible) security properties:

- **dIPE** have simulation security, in the sense that its ciphertexts and secret keys can be simulated from the set of possible output encodings. This means that $\mathcal{D}_b$ can be simulated from the set of derived ciphertexts $\{\mathsf{sCT}_I\}_I$, together with $\{\mathsf{sSK}_{I_{D+1}}\}_I$. Then, the indistinguishability of $\mathcal{D}_0$ and $\mathcal{D}_1$ reduces in a black-box way to that of $\{\mathsf{sCT}_I, \mathsf{sSK}_{I_{D+1}}\}_I$.

- The indistinguishability of $\{\mathsf{sCT}_I, \mathsf{sSK}_{I_{D+1}}\}_I$ has to rely on the security of **sIPE**. This, however, requires the security of **sIPE** to hold even when the ciphertexts are generated using correlated randomness of certain specific form — namely, for different I, sCT_I is generated with random elements $\mathbf{r}_I^{\leq D}$.

Unfortunately, the above strong security properties do not hold, and proving security without them are the main technical challenges.

- **Challenge 1 — Relying only on indistinguishability-based Function Hiding of dIPE.** **dIPE** satisfies only indistinguishability-based function hiding property, which means $\mathcal{D}_b$ cannot be simulated using the **sIPE** ciphertexts and secret keys $\{\mathsf{sCT}_I, \mathsf{sSK}_{I_D}\}_i$. Then, how can we reduce to the security of **sIPE**? To do so in a black-box way, typically, the security proof moves to a hybrid distribution where the challenge ciphertexts of **sIPE** can be embedded directly into the hybrid distributions. [7] Unfortunately, given that the total number of **sIPE** ciphertexts that can be derived from $\mathcal{D}_b$ is Γ^D, but the total size of **hIPE** ciphertexts in $\mathcal{D}_b$ is way smaller than that (there are $(D+1)\Gamma$ of them, each of size independent of Γ), there is not enough space to embed all **sIPE** ciphertexts.

  To resolve this problem, instead of attempting to embed all ciphertexts $\{\mathsf{sCT}_I\}$ in one shot, we hardwire them in "piecemeal", through a long sequence of Γ^{D-1} steps. In each step, we hardwire only Γ ciphertexts $\{\mathsf{sCT}_I\}$ that are indexed by a fixed prefix ρ, $I = \rho || I_D$. When the Γ ciphertexts are hardwired, we rely on the security of **sIPE** to argue that switching the vector encrypted inside from $\mathbf{u}_I^{\leq D}$ to $\mathbf{v}_I^{\leq D}$ is indistinguishable. After Γ^{D-1} steps, all encrypted vectors are switched, and by a hybrid argument, we conclude that $\mathcal{D}_0$ and $\mathcal{D}_1$ are indistinguishable.

- **Challenge 2 — Relying on the Security of sIPE.** To argue the indistinguishability of the Γ hardwired ciphertexts $\{\mathsf{sCT}_{\rho||I_D}\}$, we still need to overcome two issues. First, ciphertexts of $\{\mathbf{u}_{\rho||I_D}^{\leq D}\}$ and $\{\mathbf{v}_{\rho||I_D}^{\leq D}\}$ are indistinguishable only if vectors encoded in the secret keys are simultaneously switched from $\{\mathbf{u}_\gamma^{D+1}\}_\gamma$ to $\{\mathbf{v}_\gamma^{D+1}\}_\gamma$ (as otherwise, the inner products differ). But, switching the vectors encoded in secret keys would affect the inner products obtained when decrypting other **sIPE** ciphertexts with prefix different from ρ. To resolve this problem, we leverage the two-slot structure and *partial weak-function-hiding* of **sIPE**.

  Another issue is that the hardwired ciphertexts are generated with structured randomness $\mathbf{r}_{\rho||I_D} = \mathbf{r}_{\rho_1}^1 \cdots \mathbf{r}_{\rho_{D-1}}^{D-1} \mathbf{r}_{I_D}^D$. The hope is that given that each randomness corresponds to a unique combination I of random shares, we can try to apply the the SXDH assumption on MMaps to argue that their product is pseudorandom inside MMap encodings. Then, by the *strong IND-security* of **sIPE**, the hardwired ciphertexts are indistinguishable.

Hardwiring dIPE Ciphertext in Piecemeal To hardwire ciphertexts $\{\mathsf{sCT}_I\}$ in piecemeal, we build a sequence of $2\times\Gamma^{D-1}$ hybrid distributions $\{H_\rho^0, H_\rho^1\}_{\rho\in[\Gamma]^{D-1}}$, where in the ρth pair of hybrids

[7]One can also resolve to non-black-box security reduction, but is is unclear to us how to design non-black-box reductions here.

(H^0_ρ, H^b_ρ), the ciphertexts $\{\mathsf{sCT}_{\rho||I_D}\}$ indexed by prefix ρ are hardwired, while all other ciphertexts sCT_I (indexed by prefix different from ρ) are computed in ways different from that in the honest distributions $\mathcal{D}_b$, in order to satisfy the following desiderata .

Desiderata For every I, combining $\mathsf{hCT}^1_{I_1}, \cdots, \mathsf{hCT}^D_{I_D}$ produces sCT_I, such that,

- if I has a prefix smaller than ρ (*i.e.*, $I_{\leq D-1} < \rho$), sCT_I is a ciphertext of vector $\mathbf{u}^{\leq D}_I$,
- if I has a prefix greater than ρ (*i.e.*, $I_{\leq D-1} > \rho$), sCT_I is a ciphertext of $\mathbf{v}^{\leq D}_I$, and
- if I has exactly prefix ρ (*i.e.*, $I_{\leq D-1} = \rho$), sCT_I is hardwired, and is a ciphertext of $\mathbf{u}^{\leq D}_I$ in H_0 and a ciphertext of $\mathbf{v}^{\leq D}_I$ in H_1.

The only difference between H^0_ρ and H^1_ρ is that the vectors encrypted in the Γ hardwired ciphertexts are switched from $\mathbf{u}^{\leq D}_I$ to $\mathbf{v}^{\leq D}_I$ — we refer to them as the **sIPE** *challenge* ciphertexts below. Our idea is to 1) rely on the security of **sIPE** to show the indistinguishability of H^0_ρ and H^1_ρ, and 2) rely on the security of **dIPE** to show that of H^1_ρ and $H^0_{\rho+1}$. Following the sequence of hybrids allows us to step by step switch the encrypted vectors from $\mathbf{u}$'s to $\mathbf{v}$'s, corresponding to moving from $\mathcal{D}_0$ to $\mathcal{D}_1$ in an indistinguishable way. Below, we first show that we can indeed build hybrids satisfying the above desiderata, and then describe ideas for showing the indistinguishability of neighboring hybrids.

Our Hybrids $\{H^b_\rho\}$ The hybrid distribution H^b_ρ is generated like the honest distribution $\mathcal{D}_b$, except that, the set of vectors $\{\mathbf{X}^d_{\gamma,l}\}_{\gamma,l}$ encoded in the **dIPE** ciphertexts $\{\mathsf{hCT}^d_\gamma = \{\mathsf{dCT}^d_{\gamma,l}\}_l\}_\gamma$ and secret keys $\{\mathsf{hCT}^D_\gamma = \{\mathsf{dSK}_{\gamma,l}\}_l\}_\gamma$ are "engineered" carefully to fulfill the desiderata . Recall that for any combination I, combining $\{\mathsf{hCT}^d_{I_d}\}$ produces

$$\left\{[\langle \mathbf{X}^1_{I_1,l} \cdots, \mathbf{X}^D_{I_D,l}\rangle]_{\leq D}\right\}_{l\in[L]} = \{\mathsf{sCT}_{I,l}\}_{l\in L} = \mathsf{sCT}_I$$

In an honest distribution $\mathcal{D}_b$, vectors $\{\mathbf{X}^d_{\gamma,l}\}$ have much "redundant space" filled with zeros, and their inner products are determined by their non-zero prefixes, $\boldsymbol{\mu}$'s and $\boldsymbol{\nu}$'s below, derived from the actual input vectors $\mathbf{u}$'s and $\mathbf{v}$'s.

$$\text{In } \mathcal{D}_0, \mathbf{X}^d_{I_d,l} = \boldsymbol{\mu}^d_{I_d,l}||\mathbf{0} \quad \text{s.t.} \quad \left\{[\langle \boldsymbol{\mu}^1_{I_1,l} \cdots, \boldsymbol{\mu}^D_{I_D,l}\rangle]_{\leq D}\right\}_l = \mathsf{sCT}_I = \mathsf{sIPE.Enc}(1, \mathbf{k}_1, \mathbf{u}^{\leq D}_I; \mathbf{r}^{\leq D}_I) \quad (10)$$

$$\text{In } \mathcal{D}_1, \mathbf{X}^d_{I_d,l} = \boldsymbol{\nu}^d_{I_d,l}||\mathbf{0} \quad \text{s.t.} \quad \left\{[\langle \boldsymbol{\nu}^1_{I_1,l} \cdots, \boldsymbol{\nu}^D_{I_D,l}\rangle]_{\leq D}\right\}_l = \mathsf{sCT}_I = \mathsf{sIPE.Enc}(1, \mathbf{k}_1, \mathbf{v}^{\leq D}_I; \mathbf{r}^{\leq D}_I) \quad (11)$$

In a hybrid distribution H^b_ρ, we will use the redundant space in vectors $\mathbf{X}^d_{\gamma,l}$ for two purposes: First, for hardwiring the Γ challenge ciphertexts $\{\mathsf{sCT}_{\rho||I_D}\}$, and second, for differentiating how ciphertexts indexed with prefix different from ρ. To do so, we parse $\mathbf{X}^d_{\gamma,l}$ as containing D slots — each of the first $D-1$ slots fits two vectors of length $|\boldsymbol{\chi}^d_{\gamma,l}|$, and the last slot fits 1 element. Under this parsing, $\mathbf{X}^d_{\gamma,l}$ in an honest distribution can be written as:

$$\mathbf{X}^d_{\gamma,l} = \quad \underbrace{\boldsymbol{\chi}^d_{\gamma,l}||\mathbf{0}}_{\text{slot } 1} \quad \underbrace{\mathbf{0}||\mathbf{0}}_{\text{slot } 2} \quad \cdots \quad \underbrace{\mathbf{0}||\mathbf{0},}_{\text{slot } D-1} \quad \underbrace{0}_{\text{slot } D}$$

In $\mathcal{D}_0$, $\boldsymbol{\chi}^d_{\gamma,l} = \boldsymbol{\mu}^d_{\gamma,l}$, whereas In $\mathcal{D}_1$, $\boldsymbol{\chi}^d_{\gamma,l} = \boldsymbol{\nu}^d_{\gamma,l}$.

Setting the vectors $\{\mathbf{X}^d_{\gamma,l}\}$ *in hybrid* H^b_ρ Consider two cases depending on d.

- If $d = D$, we hardwire the Γ challenge ciphertexts $\{\mathsf{sCT}_{\rho||\gamma}\}_\gamma$ indexed with prefix ρ in $\{\mathbf{X}^D_{\gamma,l}\}$,

$$\mathbf{X}^D_{\gamma,l} = \underbrace{\boldsymbol{\mu}^D_{\gamma,l}||\boldsymbol{\nu}^D_{\gamma,l}}_{\text{slot } 1} \quad \underbrace{\boldsymbol{\mu}^D_{\gamma,l}||\boldsymbol{\nu}^D_{\gamma,l}}_{\text{slot } 2} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^D_{\gamma,l}||\boldsymbol{\nu}^D_{\gamma,l}}_{\text{slot } D-1} \quad \underbrace{\begin{cases} \left\langle \boldsymbol{\mu}^{<D}_{\rho,l}, \boldsymbol{\mu}^D_{\gamma,l} \right\rangle & \text{in } H^0_\rho \\ \left\langle \boldsymbol{\nu}^{<D}_{\rho,l}, \boldsymbol{\nu}^D_{\gamma,l} \right\rangle & \text{in } H^1_\rho \end{cases}}_{\text{slot } D}$$

where $\boldsymbol{\chi}^{<D}_{\rho,l} = \prod_{d<D} \boldsymbol{\chi}^d_{\rho_d,l}$ for $\boldsymbol{\chi} \in \{\boldsymbol{\mu}, \boldsymbol{\nu}\}$. Note that in the last slot, the inner product hardwired equals exactly to the element in the l^{th} encoding of ciphertext $\mathsf{sCT}_{\rho||\gamma}$, which encrypts $\mathbf{u}^{\leq D}_{\rho||\gamma}$ in H^0_ρ and $\mathbf{v}^{\leq D}_{\rho||\gamma}$ in H^1_ρ. (See equation (10) and (11).)

- If $d < D$, we use the d^{th} slot to "differentiate" what ciphertexts to produce for combinations I with prefix $\rho_{<d}$, depending on whether i) $I_d < \rho_d$ or ii) $I_d > \rho_d$ or iii) $I_d = \rho_d$. In case i) sCT_I should encrypt $\mathbf{v}^{\leq D}_I$ (*i.e.*, the encrypted vector has already been switched in previous hybrids), in case ii) sCT_I should encrypt $\mathbf{u}^{\leq D}_I$ (*i.e.*, the encrypted vector will be switched in later hybrids), and in case iii) the vector encrypted in sCT_I depends on the suffix $I_{>d}$ and will be "differentiated" by vectors $\mathbf{X}^{d'}_{\gamma,l}$ for larger coordinates $d' > d$.

$$\mathbf{X}^d_{\gamma,l} = \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } 1} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } d-1} \quad \underbrace{\begin{cases} \mathbf{0} \,||\, \boldsymbol{\nu}^{<d}_{\rho,l}\boldsymbol{\nu}^d_{\gamma,l} & \text{i) if } \gamma < \rho_d \\ \boldsymbol{\mu}^{<d}_{\rho,l}\boldsymbol{\mu}^d_{\gamma,l} \,||\, \mathbf{0} & \text{ii) if } \gamma > \rho_d \\ \mathbf{0} \;\;||\;\; \mathbf{0} & \text{iii) if } \gamma = \rho_d \end{cases}}_{\text{slot } d} \quad \underbrace{\begin{cases} \mathbf{0} & \text{i) if } \gamma < \rho_d \\ \mathbf{0} & \text{ii) if } \gamma > \rho_d \\ \mathbf{1} & \text{iii) if } \gamma = \rho_d \end{cases}}_{\text{slot } > d}$$

Desiderata are satisfied We now verify that setting $\{\mathbf{X}^d_{\gamma,l}\}$ as above indeed satisfies our desiderata. First consider any combination $I = \rho||I_D$ that starts with ρ. For every $d < D$, the corresponding vector $\mathbf{X}^d_{\rho_d}$ has zeros in the d^{th} slot, and thus their product $\mathbf{X}^{\leq D-1}_{\rho,l}$ has all zeros in the first $D-1$ slots and ones in the D^{th} slot. Hence, the inner product is determined by the values in the last slot of $\mathbf{X}^D_{I_D,l}$, which are exactly elements encoded in $\mathsf{sCT}_{I,l}$. And the ciphertext sCT_I encrypts $\mathbf{u}^{\leq D}_I$ in H^0_ρ or $\mathbf{v}^{\leq D}_I$ in H^1_ρ.

Second, consider any other combination I that agrees with ρ at the first $d^\star < D-1$ coordinates, $I_{\leq d^\star} = \rho_{\leq d^\star}$, and is, *say*, smaller than ρ at coordinate $d^\star + 1$, $I_{d^\star+1} < \rho_{d^\star+1}$. In this case we want the inner product of $\{\mathbf{X}^d_{I_d,l}\}_d$ to produce the l^{th} element in a ciphertext of $\mathbf{v}^{\leq D}_I$; this follows from the following observations.

- The product of vectors at the first $d^\star$ coordinates $\mathbf{X}^{\leq d^\star}_{I,l} = \prod_{d \leq d^\star} \mathbf{X}^d_{I_d,l}$ have zeros in the first $d^\star$ slots, and ones in the rest slots.

- At coordinate $d^\star + 1$, since $I_{d^\star} < \rho_{d^\star}$, slot $d^\star + 1$ of $\mathbf{X}^{d^\star}_{I_{d^\star},l}$ contains $\mathbf{0}||\boldsymbol{\nu}^{<d^\star+1}_{\rho,l}\boldsymbol{\nu}^{d^\star+1}_{I_{d^\star+1},l} = \mathbf{0}||\boldsymbol{\nu}^{\leq d^\star+1}_{I,l}$ and zeros in following slots.

- At larger coordinates $d' \geq d^\star + 2$, slot $d^\star + 1$ are set to $\boldsymbol{\mu}^{d'}_{I_{d'},l}||\boldsymbol{\nu}^{d'}_{I_{d'},l}$.

Therefore, inner product $\left\langle \mathbf{X}^1_{I_1,l}, \cdots, \mathbf{X}^D_{I_D,l} \right\rangle$ is exactly $\left\langle \boldsymbol{\nu}^{\leq d^\star+1}_{I,l}, \boldsymbol{\nu}^{d^\star+2}_{I_{d^\star+2},l}, \cdots, \boldsymbol{\nu}^D_{I_D,l} \right\rangle$, which corresponds to an encryption of $\mathbf{v}^{\leq D}_I$ as desired. The other case where $I_{d^\star+1} > \rho_{d^\star+1}$ can be verified similarly.

Indistinguishability of H^1_ρ and $H^0_{\rho+1}$ Given that the desiderata is satisfied, it is easy to see that in H^1_ρ and $H^0_{\rho+1}$, inner products of all combination of vectors $\{\mathbf{X}^d_{\gamma,l}\}$ are identical. By construction, these vectors are encoded in **dIPE** ciphertexts and secret keys, and hence by function hiding of **dIPE**, H^1_ρ and $H^0_{\rho+1}$ are indistinguishable. (Jumping ahead, later, we need to further modify the hybrids H^b_ρ, which would make the argument for the indistinguishability between H^1_ρ and $H^0_{\rho+1}$ more compliciated.)

Indistinguishability of H^0_ρ and H^1_ρ The only difference between H^0_ρ and H^1_ρ is that the Γ hardwired challenge ciphertexts $\{\mathsf{sCT}_{\rho||\gamma}\}_\gamma$ encrypt $\{\mathbf{u}^{\le D}_{\rho||\gamma}\}$ in H^0_ρ, and $\{\mathbf{v}^{\le D}_{\rho||\gamma}\}$ in H^1_ρ. Hence, we want to apply function hiding of **sIPE** to argue that H^0_ρ and H^1_ρ are indistinguishable. To do so, however, we need to simultaneously switch the vectors encoded in the secret keys $\{\mathsf{hSK}_\gamma = \mathsf{sSK}_\gamma\}_\gamma$ from $\{\mathbf{u}^{D+1}_\gamma\}$ to $\{\mathbf{v}^{D+1}_\gamma\}$. This would ensure the output decrypted from the challenge ciphertexts remain the same in the two hybrids, but would change the outputs decrypted from other ciphertexts sCT_I indexed with prefix different from ρ (whose encrypted vectors remain the same), making the hybrids easily distinguishable.

To address this problem, we rely on the special structure and properties of **sIPE**. First, **sIPE** has *two slots*, and so far we only used the first slot. Instead, whenever we want to switch an encrypted vector from $\mathbf{u}^{\le D}_I$ to $\mathbf{v}^{\le D}_I$, we actually switch from encrypting $\mathbf{u}^{\le D}_I$ in the first slot, to encrypting $\mathbf{v}^{\le D}_I$ in the *second* slot — more precisely, switching from encrypting $\mathbf{u}^{\le D}_I||\mathsf{null}$ to $\mathsf{null}||\mathbf{v}^{\le D}_I$. This can be done by modifying the values of vectros $\boldsymbol{\nu}$'s, such that,

$$\left\{[\langle \boldsymbol{\mu}^1_{I_1,l}, \cdots \boldsymbol{\mu}^D_{I_D,l}\rangle]_{\le D}\right\}_l = \mathsf{sCT}_I = \mathsf{sIPE.PEnc}(\ 1\ , \mathbf{k}_1, \mathbf{u}^{\le D}_I\ ;\ \mathbf{r}^{\le D}_I)\ , \text{ and}$$
$$\left\{[\langle \boldsymbol{\nu}^1_{I_1,l}, \cdots \boldsymbol{\nu}^D_{I_D,l}\rangle]_{\le D}\right\}_l = \mathsf{sCT}_I = \mathsf{sIPE.PEnc}(\underline{\ 2\ }, \mathbf{k}_2, \mathbf{v}^{\le D}_I\ ;\ \mathbf{r}^{\le D}_I)\ .$$

Suppose that the secret keys encode vectors $\{\mathbf{u}^{D+1}_\gamma||\mathbf{v}^{D+1}_\gamma\}_\gamma$ (in the first and second slots respectively). Then, we can rely on the *strong IND-security* of **sIPE** to show that switching from encrypting $\mathbf{u}^{\le D}_I||\mathsf{null}$ to $\mathsf{null}||\mathbf{v}^{\le D}_I$ in the **sIPE** challenge ciphertexts is indistinguishable, since

$$\forall I, \qquad \left\langle \mathbf{u}^{\le D}_I||\mathsf{null},\ \mathbf{u}^{D+1}_I||\mathbf{v}^D_{I_D}\right\rangle = \left\langle \mathsf{null}||\mathbf{v}^{\le D}_I,\ \mathbf{u}^{D+1}_I||\mathbf{v}^D_{I_D}\right\rangle\ .$$

This step requires overcoming the issue of correlated randomness. Before addressing this, we first complete the steps in the proof.

Putting Pieces Together Starting from the honest distribution $\mathcal{D}_0$, where the secret keys encode vectors $\{\mathbf{u}^{D+1}_\gamma||\mathbf{0}\}_\gamma$, we first move to an initial hybrid distribution $\mathbf{Init}^0$ where secret keys encode vectors $\{\mathbf{u}^{D+1}_\gamma||\mathbf{v}^{D+1}_\gamma\}_\gamma$. Since all the ciphertexts in $\mathcal{D}_0$ are generated using only the first-slot key $\mathbf{k}_1$ of **sIPE**, by the *partial weak-function-hiding w.r.t. the second slot* of **sIPE**, changing the second-slot vectors in secret keys is indistinguishable. Next, starting from $\mathbf{Init}^0$, we follow the above sequence of hybrid $\{H^b_\rho\}$ all the way to hybrid $H^1_{\Gamma^{D-1}}$, in which all derived ciphertext $\{\mathsf{sCT}_I\}$ encrypt vectors $\{\mathsf{null}||\mathbf{v}^{\le D}_I\}_I$. Now, since only the second-slots in these ciphertexts are active, the distribution can be generated using only the second-slot key $\mathbf{k}_2$ of **sIPE**. Therefore, by the *partial weak-function-hiding w.r.t. the first slot* of **sIPE**, we can change the vectors encoded in the secret keys from $\{\mathbf{u}^{D+1}_\gamma||\mathbf{v}^{D+1}_\gamma\}_\gamma$ to $\{\mathbf{0}||\mathbf{v}^{D+1}_\gamma\}$ — call the resulting distribution **Mid**. **Mid** is almost identical to $\mathcal{D}_1$, except that in **Mid**, all vectors $\{\mathbf{v}^{\le D}_I, \mathbf{v}^{D+1}_\gamma\}$ are encoded in the second slot of **sIPE** ciphertexts and secret keys, but in $\mathcal{D}_1$, the same vectors are encoded in the first slot. Nevertheless, it follows from a sequence of syntactically identical hybrids that $\mathcal{D}_1$ is also indistinguishable from **Mid**, which implies that $\mathcal{D}_0$ and $\mathcal{D}_1$ are indistinguishable.

Overcoming Correlated Randomness Finally, we come to address the issue of correlated randomness of the Γ challenge ciphertexts $\{\mathsf{sCT}_{\rho||\gamma}\}_\gamma$ hardwired in H^0_ρ and H^1_ρ. Recall that the randomness of the challenge ciphertexts has form $\mathbf{r}^{\leq D}_{\rho||\gamma} = \mathbf{r}^1_{\rho_1} \cdots \mathbf{r}^{D-1}_{\rho_{D-1}} \mathbf{r}^D_\gamma$, and the problem is that the shares are encoded at different coordinates for generating other ciphertexts. More specifically, in H^b_ρ, each $\mathbf{r}^D_\gamma$ is encoded in hCT^D_γ at coordinate D, and each partial product $\mathbf{r}^{\leq d}_\rho = \prod_{i \leq d} \mathbf{r}^i_{\rho_i}$ is encoded in $\{\mathsf{hCT}^{d+1}_\gamma\}_\gamma$ at coordinate $d+1$. Thus, the following encodings are embedded in H^b_ρ.

$$[\mathbf{r}^D_1]_D, \cdots [\mathbf{r}^D_\Gamma]_D, \quad [\mathbf{r}^{\leq 1}_\rho]_2, \cdots \left[\mathbf{r}^{\leq d}_\rho\right]_{d+1}, \cdots [\mathbf{r}^{\leq D-1}_\rho]_D \quad \left\{[\mathbf{r}^{\leq D-1}_\rho \mathbf{r}^D_\gamma]_D\right\}_\gamma$$

Suppose that the multilinear pairing groups were ideal (*i.e.*, the only ways to interact with encodings are through the honest interface). The above distribution would be indistinguishable to the following, where the correlated randomness is replaced with truly random elements $\mathbf{w}^D_{\rho||\gamma}$.

$$[\mathbf{r}^D_1]_D, \cdots [\mathbf{r}^D_\Gamma]_D, \quad [\mathbf{r}^{\leq 1}_\rho]_2, \cdots \left[\mathbf{r}^{\leq d}_\rho\right]_{d+1}, \cdots [\mathbf{r}^{\leq D-1}_\rho]_D \quad \left\{\underline{[\mathbf{w}^D_{\rho||\gamma}]_D}\right\}_\gamma$$

This means that the correlated randomness are pseudorandom, *under encodings*, and hence we can apply the security of **sIPE**.

But, in this work, we assume only the SXDH assumption over MMaps, which does not imply the above indistinguishability. Instead, we further change the above-described hybrids H^b_ρ, so that, every partial product $\mathbf{r}^{\leq d}_\rho$ is replaced with an *independently and randomly* sampled vector $\mathbf{w}^d_\rho$. In particular, now every $\mathsf{sCT}_{\rho||\gamma}$ is generated using randomness $\mathbf{w}^{D-1}_\rho \mathbf{r}^D_\gamma$ (instead of $\mathbf{r}^{\leq D-1}_\rho \mathbf{r}^D_\gamma$), and $\mathbf{w}^d_\rho$ (instead of $\mathbf{r}^{\leq d}_\rho$) is encoded at coordinate $d+1$. Thus, the set of encodings embedded in H^b_ρ becomes

$$[\mathbf{r}^D_1]_D, \cdots [\mathbf{r}^D_\Gamma]_D, \quad \underline{[\mathbf{w}^1_\rho]}_2, \cdots \underline{[\mathbf{w}^d_\rho]}_{d+1}, \cdots \underline{[\mathbf{w}^{D-1}_\rho]}_D \quad \left\{[\underline{\mathbf{w}^{D-1}_\rho} \mathbf{r}^D_\gamma]_D\right\}_\gamma ,$$

which by SXDH is indistinguishable to

$$[\mathbf{r}^D_1]_D, \cdots [\mathbf{r}^D_\Gamma]_D, \quad [\mathbf{w}^1_\rho]_2, \cdots \left[\mathbf{w}^d_\rho\right]_{d+1}, \cdots [\mathbf{w}^{D-1}_\rho]_D \quad \left\{\underline{[\mathbf{w}^D_{\rho||\gamma}]_D}\right\}_\gamma .$$

Changing the hybrids as such allows us to argue that the correlated randomness are pseudorandom, and makes it easy to show the indistinguishability of H^0_ρ and H^1_ρ. However, it brings new technical challenges when proving the indistinguishability of H^1_ρ to $H^0_{\rho+1}$, as it is no longer true that the inner products of all combination of vectors $\{\mathbf{X}^d_{\gamma,d}\}$ are identical. In particular, their inner products correspond to **sIPE** encryption of the same vectors, but with different randomness. Hence we cannot apply the security of **dIPE** directly. Nevertheless, we are able to use additional hybrids to show the indistinguishability of H^1_ρ to $H^0_{\rho+1}$. At a very high-level, the (overly simplified) idea is iteratively applying the SXDH assumption to switch the random elements $\mathbf{w}^d_\rho$ back to the form of partial products $\mathbf{r}^{\leq d}_\rho$ one by one, till we can again apply the security of **dIPE**.

7.3.2 Proof of Proposition 2

We want to show the indistinguishability of ensembles $\{\mathcal{D}_b(\lambda)\}_\lambda$, for $b = 0$ or 1,

$$\{\mathcal{D}_b(\lambda)\}_\lambda = \left\{ \begin{array}{l} \mathsf{hMSK} \xleftarrow{\$} \mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp}) \\ \left\{\mathsf{hCT}^d_\gamma \xleftarrow{\$} \mathsf{hIPE.Enc}^d(\mathsf{hMSK}, \mathbf{x}^d_\gamma)\right\}_{\gamma\in[\Gamma], d\in[D]} \\ \left\{\mathsf{hSK}_\gamma \xleftarrow{\$} \mathsf{hIPE.KeyGen}(\mathsf{msk}, \mathbf{x}^{D+1}_\gamma)\right\}_{\gamma\in[\Gamma]} \end{array} : \mathsf{pp}, \left\{\mathsf{hSK}_\gamma,\ \mathsf{hCT}^1_\gamma, \cdots, \mathsf{hCT}^{D+1}_\gamma\right\}_{\gamma\in[\Gamma]} \right\}_{\lambda\in\mathbb{N}}$$

where $\mathbf{x}_\gamma^d = \mathbf{u}_\gamma^d$ when $b = 0$ and $\mathbf{x}_\gamma^d = \mathbf{v}_\gamma^d$ when $b = 1$, such that,

$$\forall\, I \in [\Gamma]^{D+1}, \quad \left\langle \mathbf{u}_{I_1}^1, \cdots, \mathbf{u}_{I_D}^{D+1} \right\rangle = \left\langle \mathbf{v}_{I_1}^1, \cdots, \mathbf{v}_{I_D}^{D+1} \right\rangle .$$

Fix a λ and $\Gamma = \Gamma(\lambda)$, we construct an intermediate hybrid **Mid**, and show that both $\mathcal{D}^0$ and $\mathcal{D}^1$ are indistinguishable to **Mid** and hence are indistinguishable. To show the indistinguishability between $\mathcal{D}^0$ and **Mid**, we construct a sequence of $2\Gamma^{D-1} + 2$ hybrid distributions **Init**, $\{H_\rho^b\}_{b\in\{0,1\},\rho\in[\Gamma]^{D-1}}$, **Mid**$'$, and show that $\mathcal{D}^0$ is indistinguishable to **Init**, **Mid**$'$ is indistinguishable to **Mid**, and all neighboring hybrids are indistinguishable; then by a hybrid argument, $\mathcal{D}^0$ is indistinguishable to **Mid**. It follows from syntactically the same proof that $\mathcal{D}^1$ is also indistinguishable to **Mid**. Below we focus on proving the former and note in the end why the indistinguishability of $\mathcal{D}^1$ and **Mid** follows from the same the proof.

Hybrid Init(λ) is generated identically as $\mathcal{D}_0$ except that instead of generating the secret keys hSK_γ as the **sIPE** key sSK_γ encoding vector $\mathbf{u}_\gamma^{D+1}$ in the first slot, **Init** generates sSK_γ encoding both vectors $\mathbf{u}_\gamma^{D+1}, \mathbf{v}_\gamma^{D+1}$ in the first and second slot respectively. See figure 3 for a precise description. (The difference from distribution $\mathcal{D}_b(\lambda)$ is underlined.)

Hybrid distribution Init(λ)

Generate the following

- $\mathsf{hMSK} \xleftarrow{\$} \mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp})$ and parse $\mathsf{hMSK} = (\mathsf{sMSK}, \{\mathsf{dMSK}_l\}_{l\in[L]})$ and $\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2)$.
- For every $\gamma \in [\Gamma]$ and $d \in [D]$, generate $\mathsf{hCT}_\gamma^d \xleftarrow{\$} \mathsf{hIPE.Enc}^d(\mathsf{hMSK}, \mathbf{u}_\gamma^d)$.
- For every $\gamma \in [\Gamma]$, generate $\mathsf{hSK}_\gamma = \underline{\mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{u}_\gamma^{D+1}, \mathbf{v}_\gamma^{D+1})}$.

Output $\left\{\mathsf{hSK}_\gamma,\ \mathsf{hCT}_\gamma^1, \cdots, \mathsf{hCT}_\gamma^{D+1}\right\}_{\gamma\in[\Gamma]}$

Figure 3: Initial Hybrid Distribution **Init**(λ)

We show that $\mathcal{D}_0$ and **Init** are indistinguishable, relying on the partial weak-function-hiding property of **sIPE** w.r.t. the second slot.

Lemma 5. *The ensembles $\{\mathcal{D}_0(\lambda)\}_\lambda$ and $\{\mathbf{Init}(\lambda)\}_\lambda$ are indistinguishable.*

Proof. The only difference between $\mathcal{D}_0$ and **Init** lies in the second-slot vectors encoded in the secret keys, $\mathbf{0}$ in the former and $\mathbf{v}_\gamma^{D+1}$ in the latter. Note that by construction of **hIPE**, its encryption algorithm $\mathsf{hIPE.Enc}$ uses only the first slot key $\mathbf{k}_1$. Therefore, distributions $\mathcal{D}_0$ and **Init** can be emulated perfectly given just $(\mathbf{k}_1, \{\mathsf{sSK}_\gamma\}_\gamma)$, where the latter encode $\mathbf{0}$ or $\mathbf{v}_\gamma^{D+1}$ in the second slot respectively. More precisely, $\mathcal{D}_0$ can be emulated from $\widetilde{\mathcal{D}}_0$ and **Init** from $\widetilde{\mathcal{D}}_1$ defined below.

$$\widetilde{\mathcal{D}}_b = \left\{ \begin{array}{l} \mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2) \xleftarrow{\$} \mathsf{sIPE.Setup}(1^\lambda, (p, G_{\le D}, G_{D+1}, G_{D+2})) \\ \left\{ \mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}\left(\mathsf{sMSK}, \mathbf{u}_\gamma^{D+1}, \begin{cases} \underline{\mathbf{0}} & \text{if } b = 0 \\ \underline{\mathbf{v}_\gamma^{D+1}} & \text{if } b = 1 \end{cases} \right) \right\}_\gamma \end{array} \ :\ \mathbf{k}_1,\ \{\mathsf{sSK}_\gamma\}_\gamma \right\}$$

It follows directly from the partial weak-function-hiding w.r.t. the second slot of **sIPE** that $\widetilde{\mathcal{D}}_0$ and $\widetilde{\mathcal{D}}_0$ are indistinguishable, and hence so are $\mathcal{D}_0$ and **Init**. □

Hybrid $H^b_\rho(\lambda)$ Hybrid H^b_ρ for any $\rho \in [\Gamma]^{D-1}$ and $b \in \{0,1\}$ is identical to **Init**, except that every ciphertext hCT^d_γ encode a set of vectors $\{\widetilde{\mathbf{X}}^d_{\gamma,l}\}_l$ different from that in **Init**. Recall that in **Init**,

$$\mathbf{X}^d_{\gamma,l} = \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\mathbf{0}}_{\text{slot } 1} \quad \underbrace{\mathbf{0}||\mathbf{0}}_{\text{slot } 2} \quad \underset{\cdots}{\cdots} \quad \underbrace{\mathbf{0}||\mathbf{0},}_{\text{slot } D-1} \quad \underbrace{0}_{\text{slot } D}$$

The vectors $\boldsymbol{\mu}$'s are set as follows:

$$\boldsymbol{\mu}^d_{\gamma,l} = \begin{cases} \mathbf{u}^d_\gamma \,||\, \mathbf{r}^d_\gamma & \text{if } d < D \\ (\mathbf{u}^D_\gamma || \,\mathbf{r}^D_\gamma)(\mathbf{c}^{(\mathbf{k}_1)}_l) & \text{if } d = D \end{cases}, \text{ s.t.}$$
$$\forall\, I \in [\Gamma^D], \quad \Big\{ [\langle \boldsymbol{\mu}^1_{I_1,l} \cdots, \boldsymbol{\mu}^D_{I_D,l} \rangle]_{\leq D} \Big\}_l = \mathsf{sIPE.PEnc}(1, \mathbf{k}_1, \mathbf{u}^{\leq D}_I;\ \mathbf{r}^{\leq D}_I)$$

where $\mathbf{c}^{(\mathbf{k}_1)}_l$ is the coefficient vector of the linear function that computes the l^{th} output element of $\mathsf{sIPE.PEnc}(1, \mathbf{k}_1, \star;\ \star)$.

In addition to vectors $\boldsymbol{\mu}^d_{\gamma,l}$, hybrid H^b_ρ generates vectors $\boldsymbol{\nu}^d_{\gamma,l}$ as follows:

$$\boldsymbol{\nu}^d_{\gamma,l} = \begin{cases} \underline{\mathbf{v}^d_\gamma} \,||\, \mathbf{r}^d_\gamma & \text{if } d < D \\ (\underline{\mathbf{v}^D_\gamma} || \,\mathbf{r}^D_\gamma)(\underline{\widetilde{\mathbf{c}}^{(\mathbf{k}_2)}_l}) & \text{if } d = D \end{cases}, \text{ s.t.}$$
$$\forall\, I \in [\Gamma^D], \quad \Big\{ [\langle \boldsymbol{\nu}^1_{I_1,l} \cdots, \boldsymbol{\nu}^D_{I_D,l} \rangle]_{\leq D} \Big\}_l = \mathsf{sIPE.PEnc}(\underline{2, \mathbf{k}_2, \mathbf{v}^{\leq D}_I};\ \mathbf{r}^{\leq D}_I)$$

where $\widetilde{\mathbf{c}}^{(\mathbf{k}_2)}_l$ is the coefficient vector of the linear function that computes the l^{th} output element of $\mathsf{sIPE.PEnc}(2, \mathbf{k}_2, \star;\ \star)$.

H^b_ρ also generates vectors $\widetilde{\boldsymbol{\mu}}^d_{\rho_{\leq d-1}||\gamma,l}, \widetilde{\boldsymbol{\nu}}^d_{\rho_{\leq d-1}||\gamma,l}$ for every prefix of form $\rho_{\leq d-1}||\gamma$ (of length d) as follows. These vectors are derived from the partial products associated with the prefix $\rho_{\leq d-1}||\gamma$. Take a partial product of $\boldsymbol{\mu}$'s for example,

$$\boldsymbol{\mu}^{\leq d}_{\rho_{\leq d-1}||\gamma,l} = \left(\prod_{i \leq d-1} \boldsymbol{\mu}^i_{\rho_i,l} \right) \boldsymbol{\mu}^d_{\gamma,l} = \begin{cases} \mathbf{u}^{\leq d}_{\rho_{\leq d-1}||\gamma} \,||\, \mathbf{r}^{\leq d}_{\rho_{\leq d-1}||\gamma} & \text{if } d < D \\ (\mathbf{u}^{\leq D}_{\rho_{\leq d-1}||\gamma} \,||\, \mathbf{r}^{\leq D}_{\rho_{\leq d-1}||\gamma})\mathbf{c}^{(\mathbf{k}_1)}_l & \text{if } d = D \end{cases}$$

Then, $\widetilde{\boldsymbol{\mu}}^d_{\rho_{\leq d-1}||\gamma,l}$ is derived by replacing the partial product of random shares $\mathbf{r}^{\leq d}_{\rho_{\leq d-1}||\gamma}$ for $d > 1$ in it, with an independently and randomly sampled vector $\mathbf{w}^d_{\rho_{\leq d-1}||\gamma} \xleftarrow{\$} \mathcal{R}^5$. Vector $\widetilde{\boldsymbol{\nu}}^{\leq d}_{\rho_{\leq d-1}||\gamma,l}$ is derived similarly from the corresponding partial product of $\boldsymbol{\nu}$'s. More precisely,

$$\widetilde{\boldsymbol{\mu}}^d_{\rho_{\leq d-1}||\gamma,l} = \begin{cases} \mathbf{u}^{\leq d}_{\rho_{\leq d-1}||\gamma} \,||\, \underline{\mathbf{w}^d_{\rho_{\leq d-1}||\gamma}} & \text{if } d < D \\ (\mathbf{u}^{\leq D}_{\rho||\gamma} \,||\, \underline{\mathbf{w}^D_{\rho||\gamma}})\mathbf{c}^{(\mathbf{k}_1)}_l & \text{if } d = D \end{cases} \qquad \widetilde{\boldsymbol{\nu}}^d_{\rho_{\leq d-1}||\gamma,l} = \begin{cases} \mathbf{v}^{\leq d}_{\rho_{\leq d-1}||\gamma} \,||\, \underline{\mathbf{w}^d_{\rho_{\leq d-1}||\gamma}} & \text{if } d < D \\ (\mathbf{v}^{\leq D}_{\rho||\gamma} \,||\, \underline{\mathbf{w}^D_{\rho||\gamma}})\widetilde{\mathbf{c}}^{(\mathbf{k}_2)}_l & \text{if } d = D \end{cases}$$

where for any $\gamma \in [\Gamma]$, $\mathbf{w}^1_\gamma = \mathbf{r}^1_\gamma$, and $\mathbf{w}^d_{\rho_{\leq d-1}||\gamma} \xleftarrow{\$} \mathcal{R}$ for $d > 1$.

Fact 2. *Observe that since $\mathbf{w}^1_\gamma = \mathbf{r}^1_\gamma$, $\widetilde{\boldsymbol{\mu}}^1_{\gamma,l} = \boldsymbol{\mu}_{\gamma,l}$, and $\widetilde{\boldsymbol{\nu}}^1_{\gamma,l} = \boldsymbol{\nu}_{\gamma,l}$.*

H^b_ρ encodes in every ciphertext hCT^d_γ a set of vectors $\{\widetilde{\mathbf{X}}^d_{\gamma,l}\}_l$ depending on $\{\boldsymbol{\mu}^d_{\gamma,l}, \boldsymbol{\nu}^d_{\gamma,l}\}$ and $\{\widetilde{\boldsymbol{\mu}}^d_{\rho_{\leq d-1}||\gamma,l}, \widetilde{\boldsymbol{\nu}}^d_{\rho_{\leq d-1}||\gamma,l}\}_{d,\gamma}$, as described in Figure 4. (The difference from distribution **Init**(λ) is underlined.)

Hybrid distribution $H^b_\rho(\lambda)$ for $\rho \in [\Gamma]^{D-1}$

Generate the following:

- $\mathsf{hMSK} \xleftarrow{\$} \mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp})$ and parse $\mathsf{hMSK} = (\mathsf{sMSK}, \{\mathsf{dMSK}_l\}_{l\in[L]})$ and $\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2)$.
- For every $\gamma \in [\Gamma]$ and $d \in [D]$, generate

$$\mathsf{hCT}^d_\gamma = \begin{cases} \left\{\mathsf{dCT}^d_{\gamma,l} \xleftarrow{\$} \mathsf{dIPE.Enc}^d(\mathsf{dMSK}_l, \widetilde{\mathbf{X}}^d_{\gamma,l})\right\}_{l\in[L]} & \text{if } d < D \\ \left\{\mathsf{dSK}_{\gamma,l} \xleftarrow{\$} \mathsf{dIPE.KeyGen}(\mathsf{dMSK}_l, \widetilde{\mathbf{X}}^D_{\gamma,l})\right\}_{l\in[L]} & \text{if } d = D \end{cases}$$

where the vectors $\widetilde{\mathbf{X}}^d_{\gamma,l}$ are set as follows.

$$\widetilde{\mathbf{X}}^D_{\gamma,l} = \underbrace{\boldsymbol{\mu}^D_{\gamma,l}||\boldsymbol{\nu}^D_{\gamma,l}}_{\text{slot } 1} \quad \underbrace{\boldsymbol{\mu}^D_{\gamma,l}||\boldsymbol{\nu}^D_{\gamma,l}}_{\text{slot } 2} \quad \cdots \quad \underbrace{\boldsymbol{\mu}^D_{\gamma,l}||\boldsymbol{\nu}^D_{\gamma,l}}_{\text{slot } D-1} \quad \underbrace{\begin{cases} \left\langle \widetilde{\boldsymbol{\mu}}^D_{\rho||\gamma,l}, \mathbf{1}\right\rangle & \text{in } H^0_\rho \\ \left\langle \widetilde{\boldsymbol{\nu}}^D_{\rho||\gamma,l}, \mathbf{1}\right\rangle & \text{in } H^1_\rho \end{cases}}_{\text{slot } D}$$

$$\widetilde{\mathbf{X}}^d_{\gamma,l} = \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } 1} \quad \cdots \quad \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } d-1} \quad \underbrace{\begin{cases} \mathbf{0} \,||\, \widetilde{\boldsymbol{\nu}}^d_{\rho_{\le d-1}||\gamma,l} & \text{if } \gamma < \rho_d \\ \widetilde{\boldsymbol{\mu}}^d_{\rho_{\le d-1}||\gamma,l} \,||\, \mathbf{0} & \text{if } \gamma > \rho_d \\ \mathbf{0} \,||\, \mathbf{0} & \text{if } \gamma = \rho_d \end{cases}}_{\text{slot } d} \quad \underbrace{\begin{cases} \mathbf{0} & \text{if } \gamma < \rho_d \\ \mathbf{0} & \text{if } \gamma > \rho_d \\ \mathbf{1} & \text{if } \gamma = \rho_d \end{cases}}_{\text{slot } > d}$$

- For every $\gamma \in [\Gamma]$, generate $\mathsf{hSK}_\gamma = \mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{u}^{D+1}_\gamma, \mathbf{v}^{D+1}_\gamma)$.

Output $\left\{\mathsf{hSK}_\gamma, \; \mathsf{hCT}^1_\gamma, \cdots, \mathsf{hCT}^{D+1}_\gamma\right\}_{\gamma\in[\Gamma]}$

Figure 4: Hybrid $H^b_\rho(\lambda)$ for $\rho \in [\Gamma]^{D-1}$

We show that for every $\rho \in [\Gamma]^{D-1}$, moving from H^0_ρ to H^1_ρ is indistinguishable.

Lemma 6. *For every $\rho \in [\Gamma]^{D-1}$, the ensembles $\{H^0_\rho(\lambda)\}_\lambda$ and $\{H^1_\rho(\lambda)\}_\lambda$ are indistinguishable.*

At a high-level, the only difference between these two hybrids lies in the values in slot D of vectors $\{\widetilde{\mathbf{X}}^D_{\gamma,l}\}$. By definition of the vectors $\widetilde{\boldsymbol{\mu}}$'s and $\widetilde{\boldsymbol{\nu}}$'s, the values in slot D satisfy that,

$$\left\{\left[\left\langle \widetilde{\boldsymbol{\mu}}^D_{\rho||\gamma,l}, \; \mathbf{1}\right\rangle\right]_D\right\}_l = \left\{\left[\left\langle \mathbf{u}^{\le D}_{\rho||\gamma} \,||\, \mathbf{w}^D_{\rho||\gamma}, \; \mathbf{c}^{(\mathbf{k}_1)}_l\right\rangle\right]_D\right\}_l = \mathsf{sIPE.PEnc}(\underline{1}, \mathbf{k}_1, \mathbf{u}^{\le D}_{\rho||\gamma}; \; \mathbf{w}^D_{\rho||\gamma})$$
$$\left\{\left[\left\langle \widetilde{\boldsymbol{\nu}}^D_{\rho||\gamma,l}, \; \mathbf{1}\right\rangle\right]_D\right\}_l = \left\{\left[\left\langle \mathbf{v}^{\le D}_{\rho||\gamma} \,||\, \mathbf{w}^D_{\rho||\gamma}, \; \widetilde{\mathbf{c}}^{(\mathbf{k}_2)}_l\right\rangle\right]_D\right\}_l = \mathsf{sIPE.PEnc}(\underline{2}, \mathbf{k}_2, \mathbf{v}^{\le D}_{\rho||\gamma}; \; \mathbf{w}^D_{\rho||\gamma})$$

The former are hardwired in H^0_ρ, and correspond to **sIPE** ciphertexts encrypting $\{\mathbf{u}^{\le D}_{\rho||\gamma}||\mathsf{null}\}_\gamma$, whereas the latter are hardwired in H^1_ρ and correspond to ciphertexts encrypting $\{\mathsf{null}||\mathbf{v}^{\le D}_{\rho||\gamma}\}_\gamma$ in the second slot. Importantly, all these ciphertexts are generated using *fresh and random* elements $\mathbf{w}^D_{\rho||\gamma}$. Moreover, since in H^b_ρ the secret keys encode $\{(\mathbf{u}^{D+1}_\gamma, \mathbf{v}^{D+1}_\gamma)\}$ in the first and second slot respectively, the inner products are identical.

$$\left\langle \mathbf{u}^{\le D}_{\rho||\gamma}||\mathsf{null},\ \mathbf{u}^{D+1}_{\gamma}, \mathbf{v}^{D+1}_{\gamma} \right\rangle = \left\langle \mathsf{null}||\mathbf{u}^{\le D}_{\rho||\gamma},\ \mathbf{u}^{D+1}_{\gamma}, \mathbf{v}^{D+1}_{\gamma} \right\rangle$$

Then, we show that by the strong IND-security of **sIPE**, hybrid H^0_ρ and H^1_ρ are indistinguishable. The strong IND-security guarantees that **sIPE** remains IND-secure even when $\mathbf{k}'_1, \mathbf{k}'_2$ and $[\mathrm{s}]_D$ are revealed (as long as the shared key s is hidden). Using $\mathbf{k}'_1, \mathbf{k}'_2, [\mathrm{s}]_D$, we can emulate the distributions H^0_ρ or H^1_ρ from the **sIPE** challenge ciphertexts and secret keys, and hence their indistinguishability follows from the strong IND-security of **sIPE**. A formal proof can be found below.

We also show that moving from H^1_ρ to $H^1_{\rho+1}$ are indistinguishable.

Lemma 7. *For every $\rho \in [\Gamma]^{D-1} \setminus \{\Gamma^N\}$, the ensembles $\{H^1_\rho(\lambda)\}_\lambda$ and $\{H^0_{\rho+1}(\lambda)\}_\lambda$ are indistinguishable, where $\rho + 1$ denote the member in $[\Gamma]^{D-1}$ following immediately after ρ in increasing numerical order.*

At a high-level, the difference between H^1_ρ and $H^0_{\rho+1}$ lies in the values of $\{\widetilde{\mathbf{X}}^d_{\gamma,l}\}_{d,\gamma,l}$. These vectors are encoded in **hIPE** ciphertexts, which, by construction, consist of ciphertexts and secret key of different instances of **dIPE** with different master secret key. It turns out that, all **sIPE** ciphertexts derived from these ciphertexts and secret keys encrypt the same vectors in H^1_ρ and $H^0_{\rho+1}$, but with different randomness. Therefore, one cannot directly apply the security of **dIPE** to argue that H^1_ρ and $H^0_{\rho+1}$ are indistinguishable, because the output **sIPE** ciphertexts are not identical. Nevertheless, by relying on the SXDH assumption on MMaps, (and additional hybrids), we can show that H^1_ρ and $H^0_{\rho+1}$ are respectively indistinguishable to two other hybrid distributions, in which the output ciphertexts are identical and hence the security of **dIPE** applies. A formal proof can be found below.

It follows from similar proof that **Init** and $H^0_{1^{D-1}}$ are also indistinguishable.

Lemma 8. *The ensembles $\{\mathbf{Init}(\lambda)\}_\lambda$ and $\{H^0_{1^{D-1}}(\lambda)\}_\lambda$ are indistinguishable.*

Hybrid $\mathbf{Mid}'(\lambda)$ is generated identically as **Init** except that every ciphertext hCT^d_γ encode a set of vectors $\{\bar{\mathbf{X}}^d_{\gamma,l}\}_l$ different from that in **Init**, where instead of having vectors $\boldsymbol{\mu}$'s in the first half of slot 1, we have vectors $\boldsymbol{\nu}$'s in the second half of slot 1 (and zeros elsewhere). See figure 5 for a precise description.

We show that moving from the last hybrid $H^1_{\Gamma^{D-1}}$ to $\mathbf{Mid}'$ is indistinguishable.

Lemma 9. *The ensembles $\{H^1_{\Gamma^{D-1}}(\lambda)\}_\lambda$ and $\{\mathbf{Mid}'(\lambda)\}_\lambda$ are indistinguishable.*

At a high-level, the proof of this lemma is similar to that of Lemma 7, as the only difference between $H^1_{\Gamma^{D-1}}$ and $\mathbf{Mid}'$ lies in the vectors being encrypted, $\{\widetilde{\mathbf{X}}^d_{\gamma,l}\}$ and $\{\bar{\mathbf{X}}^d_{\gamma,l}\}$ respectively, and they all produce **sIPE** ciphertexts of vectors $\{\mathbf{v}^{\le D}_I\}_I$, modulo using different randomness. Thus again, we use the SXDH assumption to bridge the difference in randomness, and use the function hiding property of **dIPE** to argue the indistinguishability of the two hybrids. A formal proof is provided below.

Hybrid $\mathbf{Mid}(\lambda)$ proceeds identically to $\mathbf{Mid}'$ except that every secret key $\mathsf{hSK}_\gamma = \mathsf{sSK}_\gamma$ encodes vector $(\mathbf{0}, \mathbf{v}^{D+1}_\gamma)$ as opposed to $(\mathbf{u}^{D+1}_\gamma, \mathbf{v}^{D+1}_\gamma)$.

Hybrid distribution $\mathbf{Mid}'(\lambda)$

Generate the following

- $\mathsf{hMSK} \xleftarrow{\$} \mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp})$; parse $\mathsf{hMSK} = (\mathsf{sMSK}, \{\mathsf{dMSK}_l\}_{l\in[L]})$ and $\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2)$.
- For every $\gamma \in [\Gamma]$ and $d \in [D]$, generate

$$\mathsf{hCT}^d_\gamma = \begin{cases} \left\{\mathsf{dCT}^d_{\gamma,l} \xleftarrow{\$} \mathsf{dIPE.Enc}^d(\mathsf{dMSK}_l, \bar{\mathbf{X}}^d_{\gamma,l})\right\}_{l\in[L]} & \text{if } d < D \\ \left\{\mathsf{dSK}_{\gamma,l} \xleftarrow{\$} \mathsf{dIPE.KeyGen}(\mathsf{dMSK}_l, \bar{\mathbf{X}}^D_{\gamma,l})\right\}_{l\in[L]} & \text{if } d = D \end{cases}$$

  where the vectors $\bar{\mathbf{X}}^d_{\gamma,l}$ are set as follows.

$$\bar{\mathbf{X}}^d_{\gamma,l} = \underbrace{\mathbf{0}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } 1} \quad \underbrace{\mathbf{0}||\mathbf{0}}_{\text{slot } 2} \quad \cdots \quad \underbrace{\mathbf{0}||\mathbf{0},}_{\text{slot } D-1} \quad \underbrace{0}_{\text{slot } D}$$

- For every $\gamma \in [\Gamma]$, generate $\mathsf{hSK}_\gamma = \mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{u}^{D+1}_\gamma, \mathbf{v}^{D+1}_\gamma)$.

Output $\left\{\mathsf{hSK}_\gamma,\ \mathsf{hCT}^1_\gamma, \cdots, \mathsf{hCT}^{D+1}_\gamma\right\}_{\gamma\in[\Gamma]}$

Figure 5: Middle Hybrid Distribution $\mathbf{Mid}'(\lambda)$

Lemma 10. *The ensembles* $\{\mathbf{Mid}'(\lambda)\}_\lambda$ *and* $\{\mathbf{Mid}(\lambda)\}_\lambda$ *are indistinguishable.*

This lemma follows from essentially the same proof as Lemma 5, relying on the partial weak-function-hiding property of **sIPE** w.r.t. the *first slot*.

Proof. The only difference between **Mid** and **Mid**$'$ lies in the first-slot vectors encoded in the secret keys, $\mathbf{0}$ in the former and $\mathbf{u}^{D+1}_\gamma$ in the latter. Note that in **Mid**$'$ all the vectors $\{\bar{\mathbf{X}}^d_{\gamma,l}\}_{d,\gamma,l}$ encrypted rely only on the second slot key $\mathbf{k}_2$ of **sIPE** (since they depend on vectors $\{\boldsymbol{\nu}^d_{\gamma,l}\}$, which in turn depends only on $\mathbf{k}_2$.) Therefore, distribution **Mid** and **Mid**$'$ can be emulated perfectly given just $(\mathbf{k}_2, \{\mathsf{sSK}_\gamma\}_\gamma)$ encoding $\mathbf{0}$ or $\mathbf{u}^{D+1}_\gamma$ in the first slot respectively. Thus, it follows from the partial weak-function-hiding w.r.t. the first slot of **sIPE** that **Mid** and **Mid**$'$ are indistinguishable. □

Combining Lemma 5 to 10, by a hybrid argument, we have that the honest distribution $\mathcal{D}_0$ is indistinguishable to the middle hybrid distribution **Mid**. It follows from syntactically identical proof, by replacing the input vectors $\mathbf{u}$'s with vectors $\mathbf{v}$'s, that $\mathcal{D}_1$ is also indistinguishable to **Mid**. Therefore, the honest distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are indistinguishable.

We proceed to prove Lemma 6, 7, 8, and 9 in the next sections.

7.3.3 Proof Lemma 6

Proof of Lemma 6. Hybrid H^0_ρ and H^1_ρ differ only in the values of vectors $\{\widetilde{\mathbf{X}}^d_{\gamma,l}\}$. Fix any b, we argue that H^b_ρ can be emulated from the following distribution

$$\widetilde{\mathcal{D}}_b = \left\{ \left\{ \left[\widetilde{\mathbf{X}}^D_{\gamma,l}\right]_D \right\}_{\gamma,l},\ \left\{\mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{u}^{D+1}_\gamma, \mathbf{v}^{D+1}_\gamma)\right\}_\gamma \right\},$$

where $\widetilde{\mathbf{X}}^{D}_{\gamma,l}$ are the vectors encoded in H^{b}_{ρ}. It suffices to describe how the ciphertexts $\{\mathsf{hCT}^{d}_{\gamma}\}$ are generated. Recall that $\{\mathsf{hCT}^{D}_{\gamma}\}$ consists of secret keys of **dIPE** of vectors $\widetilde{\mathbf{X}}^{D}_{\gamma,l}$. By that **dIPE** has canonical form, each $\mathsf{hCT}^{D}_{\gamma}$ consists of encodings in G_D of values that depend linearly in $\widetilde{\mathbf{X}}^{D}_{\gamma,l}$; thus, they can be generated from encodings of $\widetilde{\mathbf{X}}^{D}_{\gamma,l}$ by internally sampling $\{\mathsf{dMSK}_l\}$ and relying on the linear homomorphism of G_D. At other coordinates $d < D$, ciphertexts $\{\mathsf{hCT}^{d}_{\gamma}\}_{d<D}$ can be generated using $\{\mathsf{dMSK}_l\}$, and the input vectors $\mathbf{u}$'s, $\mathbf{v}$'s and internally sampled randomness $\{\mathbf{r}^{d}_{\gamma}\}_{d<D,\gamma}$, $\{\mathbf{w}^{d}_{\rho_{\le d-1}||\gamma}\}_{d<D,\gamma}$ (which together determine $\{\widetilde{\mathbf{X}}^{d}_{\gamma,l}\}_{d<D}$). Therefore, it suffices to prove that distributions $\widetilde{\mathcal{D}}_0$ and $\widetilde{\mathcal{D}}_1$ are indistinguishable.

We further argue that $\widetilde{\mathcal{D}}_b$ can be emulated from the following distributions. Recall that $\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2)$ and $\mathbf{k}_\beta = (\mathbf{s}, \mathbf{k}'_\beta)$ contains a shared key $\mathbf{s}$ and a specific key $\mathbf{k}'_\beta$.

$$\mathcal{D}'_b = \Bigg\{ [\mathbf{s}]_D,\ (\mathbf{k}'_0, \mathbf{k}'_1),\ \Big\{ \Big[\text{element in slot-}D \text{ of } \widetilde{\mathbf{X}}^{D}_{\gamma,l}\Big]_D \Big\}_{\gamma,l},$$
$$\Big\{ \mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{u}^{D+1}_{\gamma}, \mathbf{v}^{D+1}_{\gamma}) \Big\}_{\gamma} \Bigg\},$$

This follows because encodings of $\widetilde{\mathbf{X}}^{D}_{\gamma,l}$ consist of encodings of vectors in its D slots. The vectors in the first $D-1$ slots depend linearly in the coefficients $\{\mathbf{c}^{(\mathbf{k}_1)}_{l}, \widetilde{\mathbf{c}}^{(\mathbf{k}_2)}_{l}\}_l$, which by the special property of *linearity in shared key* of **sIPE** are linear in $\mathbf{s}$ (See Section 6.5.1). Therefore, their encodings in group G_D can be emulated from $[\mathbf{s}]_D$ and $(\mathbf{k}'_0, \mathbf{k}'_1)$, with additional knowledge of $\mathbf{u}$'s, $\mathbf{v}$'s, and internally sampled randomness $\{\mathbf{r}^{D}_{\gamma}\}$. Therefore, it suffices to show the indistinguishability of $\mathcal{D}'_0$ and $\mathcal{D}'_1$.

For every γ, let us analyze the elements in slot-D of $\{\widetilde{\mathbf{X}}^{D}_{\gamma,l}\}_l$ in $\mathcal{D}'_0$ and $\mathcal{D}'_1$. In the former, the elements equal to the inner products

$$\Big\{\Big[\Big\langle \widetilde{\boldsymbol{\mu}}^{D}_{\rho||\gamma,l},\ \mathbf{1}\Big\rangle\Big]_D\Big\}_l = \Big\{\Big[\Big\langle \mathbf{u}^{\le D}_{\rho||\gamma} \ ||\ \mathbf{w}^{D}_{\rho||\gamma},\ \mathbf{c}^{(\mathbf{k}_1)}_{l}\Big\rangle\Big]_D\Big\}_l = \mathsf{sIPE.PEnc}(1, \mathbf{k}_1, \mathbf{u}^{\le D}_{\rho||\gamma};\ \mathbf{w}^{D}_{\rho||\gamma})$$

whereas in the latter, it equals to

$$\Big\{\Big[\Big\langle \widetilde{\boldsymbol{\nu}}^{D}_{\rho||\gamma,l},\ \mathbf{1}\Big\rangle\Big]_D\Big\}_l = \Big\{\Big[\Big\langle \mathbf{v}^{\le D}_{\rho||\gamma} \ ||\ \mathbf{w}^{D}_{\rho||\gamma},\ \widetilde{\mathbf{c}}^{(\mathbf{k}_2)}_{l}\Big\rangle\Big]_D\Big\}_l = \mathsf{sIPE.PEnc}(2, \mathbf{k}_2, \mathbf{v}^{\le D}_{\rho||\gamma};\ \mathbf{w}^{D}_{\rho||\gamma})$$

Therefore, we can re-write distributions $\mathcal{D}'_0$ and $\mathcal{D}'_1$ as,

$$\mathcal{D}'_0 = \Bigg\{ [\mathbf{s}]_D,\ (\mathbf{k}'_0, \mathbf{k}'_1),\ \Big\{ \mathsf{sIPE.PEnc}(1, \mathbf{k}_1, \mathbf{u}^{\le D}_{\rho||\gamma};\ \mathbf{w}_\gamma) \Big\}_\gamma,\ \Big\{ \mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{u}^{D+1}_{\gamma}, \mathbf{v}^{D+1}_{\gamma}) \Big\}_\gamma \Bigg\}$$
$$\mathcal{D}'_1 = \Bigg\{ [\mathbf{s}]_D,\ (\mathbf{k}'_0, \mathbf{k}'_1),\ \Big\{ \mathsf{sIPE.PEnc}(2, \mathbf{k}_2, \mathbf{v}^{\le D}_{\rho||\gamma};\ \mathbf{w}_\gamma) \Big\}_\gamma,\ \Big\{ \mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{u}^{D+1}_{\gamma}, \mathbf{v}^{D+1}_{\gamma}) \Big\}_\gamma \Bigg\}$$

For any combination of ciphertext and secret key, their output inner product are identical in these two distributions.

$$\forall\ \gamma, \gamma',\ \Big\langle \mathbf{u}^{\le D}_{\rho||\gamma},\ \mathbf{u}^{D+1}_{\gamma'} \Big\rangle = \Big\langle \mathbf{v}^{\le D}_{\rho||\gamma},\ \mathbf{v}^{D+1}_{\gamma'} \Big\rangle$$

Then, it follows from the fact that **sIPE** is strong IND-secure (see Lemma 3), that $\mathcal{D}'_0$ and $\mathcal{D}'_1$ are indistinguishable and hence so are H^{0}_{ρ} and H^{1}_{ρ}. □

7.3.4 Proofs of Lemma 7, 8 and 9

In order to prove Lemma 7, 8 and 9. We construct additional hybrid distributions G^b_ρ for prefixes $\rho \in [\Gamma]^{d^\star}$ of any length $d^\star$ from 1 to $D-1$, and use these G hybrids "in between" hybrids **Init**, $\{H^b_\rho\}$, **Mid** to "glue" them together. To do so, we show the following lemma.

Lemma 11. *There exist hybrids $\{G^b_\rho(\lambda)\}$ for $b \in \{0,1\}$ and $\rho \in [\Gamma]^{d^\star}$ where $d^\star \in [D-1]$, such that, the following holds.*

Rule 1: *Ensembles $\{G^0_1(\lambda)\}$ and $\{\mathbf{Init}(\lambda)\}$ are indistinguishable.*

Rule 2: *Ensembles $\{G^1_\Gamma(\lambda)\}$ and $\{\mathbf{Mid}'(\lambda)\}$ are indistinguishable*

Rule 3: *For every $\rho \in [\Gamma]^{d^\star}$ with $1 \le d^\star < D-1$, ensembles $\{G^0_{\rho||1}(\lambda)\}$ and $\{G^0_\rho(\lambda)\}$, are indistinguishable.*

Rule 4: *For every $\rho \in [\Gamma]^{d^\star}$ with $1 \le d^\star < D-1$, ensembles $\{G^1_{\rho||\Gamma}(\lambda)\}$ and $\{G^1_\rho(\lambda)\}$ are indistinguishable.*

Rule 5: *For every $\rho \in [\Gamma]^{D-1}$ and every b, ensembles $\{G^b_\rho(\lambda)\}$ and $\{H^b_\rho(\lambda)\}$ are indistinguishable.*

Rule 6: *For every $\rho \in [\Gamma]^{d^\star}$ with $1 \le d^\star \le D-1$, such that, $\rho_{d^\star} \neq \Gamma$, ensembles $\{G^1_\rho(\lambda)\}$ and $\{G^0_{\rho+1}(\lambda)\}$ are identical.*

Before describing the G hybrids and proving the above lemma, we first show that Lemma 7, 8 and 9 follow easily from the above lemma.

Proof of Lemma 7. To show that for every $\rho \in [\Gamma]^{D-1}$, hybrid H^1_ρ is indistinguishable from $H^0_{\rho+1}$, by Rule 5 (of Lemma 11), it suffices to prove that G^1_ρ and $G^0_{\rho+1}$ are indistinguishable. Consider two cases:

- <u>*Case 1:*</u> The last letter of ρ is not Γ, that is, $\rho_{D-1} \neq \Gamma$, then it follows immediately from Rule 6 that G^1_ρ and $G^0_{\rho+1}$ are indistinguishable.

- <u>*Case 2:*</u> $\rho = \rho_{\le d^\star}||\Gamma\cdots\Gamma$, where the last k letters of ρ are Γ and the $k+1$th last letter is not Γ, $\rho_{d^\star} \neq \Gamma$ for $d^\star = D-1-k$. In this case, the member $\rho+1$ following ρ must be $(\rho_{\le d^\star}+1)||1\cdots 1$. By iteratively applying Rule 4, G^1_ρ is indistinguishable from $G^1_{\rho_{\le d^\star}}$, and similarly by iteratively applying Rule 3, $G^0_{\rho+1}$ is indistinguishable from $G^0_{\rho_{\le d^\star}+1}$. Finally by Rule 6 $G^1_{\rho_{\le d^\star}}$ and $G^0_{\rho_{\le d^\star}+1}$ are *identical*, which concludes that G^1_ρ and $G^0_{\rho+1}$ are indistinguishable.

□

Proof of Lemma 8. We want to show that the initial hybrid **Init** and $H^0_{1^{D-1}}$ are indistinguishable. First, by Rule 5 (of Lemma 11), $H^0_{1^{D-1}}$ is indistinguishable to $G^0_{1^{D-1}}$. Then, by Rule 3, $G^0_{1^{D-1}}$ is indistinguishable to G^0_1. Finally, by Rule 1, G^0_1 is indistinguishable to **Init**. This concludes that **Init** and $H^0_{1^{D-1}}$ are indistinguishable. □

Proof of Lemma 9. We want to show that the last H hybrid $H^1_{\Gamma^{D-1}}$ is indistinguishable from the middle hybrid **Mid**$'$. First, by Rule 5 (of Lemma 11), $H^1_{\Gamma^{D-1}}$ is indistinguishable to $G^1_{\Gamma^{D-1}}$. Then, by Rule 4, $G^1_{\Gamma^{D-1}}$ is indistinguishable to G^1_Γ. Finally, by Rule 2, G^1_Γ is indistinguishable to **Mid**$'$. This concludes that **Init** and $H^1_{1^{D-1}}$ are indistinguishable. □

Proof of Lemma 11. We first formally describe the G hybrid distributions.

Hybrid $G^b_\rho(\lambda)$: For $d^\star \in [D-1]$ and $\rho \in [\Gamma]^{d^\star}$, distribution $G^b_\rho(\lambda)$ is identical to the initial hybrid distribution **Init**, except that the ciphertexts $\{\mathsf{hCT}^d_\gamma\}$ encode vectors $\{\hat{\mathbf{X}}^d_{\gamma,l}\}$ different from that encoded in **Init**. A formal description is provided in Figure 6.

Hybrid distribution $G^b_\rho(\lambda)$ for $d^\star \in [D-1]$ and $\rho \in [\Gamma]^{d^\star}$

Generate the following:

- $\mathsf{hMSK} \xleftarrow{\$} \mathsf{hIPE.Setup}(1^\lambda, \mathsf{pp})$ and parse $\mathsf{hMSK} = (\mathsf{sMSK}, \{\mathsf{dMSK}_l\}_{l\in[L]})$ and $\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2)$.
- For every $\gamma \in [\Gamma]$ and $d \in [D]$, generate

$$\mathsf{hCT}^d_\gamma = \begin{cases} \left\{\mathsf{dCT}^d_{\gamma,l} \xleftarrow{\$} \mathsf{dIPE.Enc}^d(\mathsf{dMSK}_l, \underline{\hat{\mathbf{X}}^d_{\gamma,l}})\right\}_{l\in[L]} & \text{if } d < D \\ \left\{\mathsf{dSK}_{\gamma,l} \xleftarrow{\$} \mathsf{dIPE.KeyGen}(\mathsf{dMSK}_l, \underline{\hat{\mathbf{X}}^D_{\gamma,l}})\right\}_{l\in[L]} & \text{if } d = D \end{cases}$$

  where the vectors $\hat{\mathbf{X}}^d_{\gamma,l}$ are set as follows.
  Case 1: $d > d^\star$.

$$\hat{\mathbf{X}}^d_{\gamma,l} = \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } 1} \quad \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } 2} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } d^\star} \quad \underbrace{\underline{\mathbf{0}}}_{\text{slot } > d^\star}$$

  Case 2: $d = d^\star$

$$\hat{\mathbf{X}}^{d^\star}_{\gamma,l} = \underbrace{\boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l}}_{\text{slot } 1} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l}}_{\text{slot } d^\star - 1} \quad \underbrace{\begin{cases} \mathbf{0} \,||\, \widetilde{\boldsymbol{\nu}}^{d^\star}_{\rho_{\le d^\star-1}||\gamma,l} & \text{if } \gamma < \rho_{d^\star} \\ \widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho_{\le d^\star-1}||\gamma,l} \,||\, \mathbf{0} & \text{if } \gamma > \rho_{d^\star} \\ \underline{\mathbf{0} \quad || \quad \widetilde{\boldsymbol{\nu}}^{d^\star}_{\rho,l}} & \underline{\text{if } \gamma = \rho_{d^\star} \text{ and } b = 1} \\ \underline{\widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho,l} \quad || \quad \mathbf{0}} & \underline{\text{if } \gamma = \rho_{d^\star} \text{ and } b = 0} \end{cases}}_{\text{slot } d^\star} \quad \underbrace{\underline{\mathbf{0}}}_{\text{slot } > d^\star}$$

  Case 3: $d < d^\star$. (In this case $\hat{\mathbf{X}}^d_{\gamma,l} = \widetilde{\mathbf{X}}^d_{\gamma,l}$ in hybrid H^b_ρ.)

$$\hat{\mathbf{X}}^d_{\gamma,l} = \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } 1} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot } d-1} \quad \underbrace{\begin{cases} \mathbf{0} \,||\, \widetilde{\boldsymbol{\nu}}^d_{\rho_{\le d-1}||\gamma,l} & \text{if } \gamma < \rho_d \\ \widetilde{\boldsymbol{\mu}}^d_{\rho_{\le d-1}||\gamma,l} \,||\, \mathbf{0} & \text{if } \gamma > \rho_d \\ \mathbf{0} \quad || \quad \mathbf{0} & \text{if } \gamma = \rho_d \end{cases}}_{\text{slot } d} \quad \underbrace{\begin{cases} \mathbf{0} & \text{if } \gamma < \rho_d \\ \mathbf{0} & \text{if } \gamma > \rho_d \\ \mathbf{1} & \text{if } \gamma = \rho_d \end{cases}}_{\text{slot } > d}$$

- For every $\gamma \in [\Gamma]$, generate $\mathsf{hSK}_\gamma = \mathsf{sSK}_\gamma \xleftarrow{\$} \mathsf{sIPE.KeyGen}(\mathsf{sMSK}, \mathbf{u}^{D+1}_\gamma, \mathbf{v}^{D+1}_\gamma)$.

Output $\left\{\mathsf{hSK}_\gamma,\ \mathsf{hCT}^1_\gamma, \cdots, \mathsf{hCT}^{D+1}_\gamma\right\}_{\gamma\in[\Gamma]}$

Figure 6: Hybrid $G^b_\rho(\lambda)$ for $d^\star \in [D-1]$ and $\rho \in [\Gamma]^{d^\star}$

Next, we prove each of the rules.

Proof of Rule 1: $G_1^0 \approx$ **Init.** The only difference between these two hybrids are the vectors encrypted in the ciphertexts and secret keys of **dIPE** contained in $\{\mathsf{hCT}_\gamma^d\}$. In G_1^0 the following vectors are encrypted:

$$\hat{\mathbf{X}}^1_{\gamma,l} = \underbrace{\widetilde{\boldsymbol{\mu}}^1_{\gamma,l}||\mathbf{0}}_{\text{slot 1}} \;\Big|\Big|\; \mathbf{0} \qquad \forall d>1 \;\; \hat{\mathbf{X}}^d_{\gamma,l} = \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot 1}} \;\Big|\Big|\; \mathbf{0}$$

In **Init**, the following vectors are encrypted

$$\forall d \quad \mathbf{X}^d_{\gamma,l} = \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\mathbf{0}}_{\text{slot 1}} \;\Big|\Big|\; \mathbf{0}$$

By definition, $\widetilde{\boldsymbol{\mu}}^1_{\gamma,l} = \boldsymbol{\mu}^1_{\gamma,l}$ (see Fact 2). Thus, for any combination, the inner products of vector $\hat{\mathbf{X}}$'s in G_1^0 and that of vector $\mathbf{X}$'s in **Init** are identical. Thus, it follows from the security of **dIPE** that G_1^0 and **Init** are indistinguishable.

Proof of Rule 2: $G_\Gamma^1 \approx$ **Mid$'$.** This follows essentially from the same proof as that for Rule 1. In G_Γ^1, the vectors $\hat{\mathbf{X}}^d_{\gamma,l}$ encrypted are

$$\hat{\mathbf{X}}^1_{\gamma,l} = \underbrace{\mathbf{0}||\,\widetilde{\boldsymbol{\nu}}^1_{\gamma,l}}_{\text{slot 1}} \;\Big|\Big|\; \mathbf{0} \qquad \forall d>1 \;\; \hat{\mathbf{X}}^d_{\gamma,l} = \underbrace{\boldsymbol{\mu}^d_{\gamma,l}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot 1}} \;\Big|\Big|\; \mathbf{0}$$

whereas in **Mid$'$**, the following vectors are encrypted

$$\forall d \quad \mathbf{X}^d_{\gamma,l} = \underbrace{\mathbf{0}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot 1}} \;\Big|\Big|\; \mathbf{0}$$

By definition, $\widetilde{\boldsymbol{\nu}}^1_{\gamma,l} = \boldsymbol{\nu}^1_{\gamma,l}$ (see Fact 2). Thus, for any combination, the inner products of vector $\hat{\mathbf{X}}$'s in G_Γ^1 and that of vector $\mathbf{X}$'s in **Init** are identical. Thus, it follows from the security of **dIPE** that G_Γ^1 and **Mid$'$** are indistinguishable.

Proof of Rule 3: $G^0_{\rho||1} \approx G^0_\rho$, for every $\rho \in [\Gamma]^{d^\star}$ with $1 \le d^\star < D-1$. Fix one such ρ and $d^\star$. The only difference between $G^0_{\rho||1}, G^0_\rho$ lies in the values of $\hat{\mathbf{X}}^d_{\gamma,l}$ for $d \ge d^\star$.

In $G^0_{\rho||1}$, these vectors have values,

$$\begin{array}{cccccc}
\hat{\mathbf{X}}^{d^\star}_{\gamma,l} & = & \boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l} & \begin{cases} \mathbf{0}\;||\;\widetilde{\boldsymbol{\nu}}^{d^\star}_{\rho_{\le d^\star-1}||\gamma,l} & \text{if } \gamma < \rho_{d^\star} \\ \widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho_{\le d^\star-1}||\gamma,l}\;||\;\mathbf{0} & \text{if } \gamma > \rho_{d^\star} \\ \mathbf{0} \;\;||\;\; \mathbf{0} & \text{if } \gamma = \rho_{d^\star} \end{cases} & \begin{cases} \mathbf{0} & \text{if } \gamma < \rho_{d^\star} \\ \mathbf{0} & \text{if } \gamma > \rho_{d^\star} \\ \mathbf{1} & \text{if } \gamma = \rho_{d^\star} \end{cases} \\
\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l} & = & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \widetilde{\boldsymbol{\mu}}^{d^\star+1}_{\rho||\gamma,l}\;||\;\mathbf{0}\;||\;\mathbf{0} \\
\hat{\mathbf{X}}^{d}_{\gamma,l} & = & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \boldsymbol{\mu}^{d}_{\gamma,l}\;||\;\boldsymbol{\nu}^{d}_{\gamma,l}||\;\mathbf{0} \\
 & & \text{slot } 1 & \cdots & \text{slot } d^\star-1 & \text{slot } d^\star & \text{slot} \ge d^\star+1
\end{array}$$

(where the last line is for $d > d^\star + 1$.) We first show that it is indistinguishable to switch to value of the vectors $\{\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l}\}$ to the following vectors (while keeping the rest the same) – call the resulting distribution $\widetilde{G}^0_{\rho||1}$.

$$\begin{array}{cccccc} \hat{\mathbf{X}}^{d^\star+1}_{\gamma,l} = & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \underline{\widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho,l}\boldsymbol{\mu}^{d^\star+1}_{\gamma,l}} \,||\, \mathbf{0} \,||\, \mathbf{0} \\ & \text{slot } 1 & \cdots & \text{slot } d^\star - 1 & \text{slot } d^\star & \text{slot} \geq d^\star + 1 \end{array}$$

Compare the values encoded in slot $d^\star + 1$ of vectors $\{\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l}\}$.

$$\text{In } G^0_\rho, \qquad \widetilde{\boldsymbol{\mu}}^{d^\star+1}_{\rho||\gamma,l} = \begin{cases} \mathbf{u}^{\leq d^\star+1}_{\rho||\gamma} \,||\, \underline{\mathbf{w}^{d^\star+1}_{\rho||\gamma}} & \text{if } d^\star + 1 < D \\ (\mathbf{u}^{\leq D}_{\rho||\gamma} \,||\, \underline{\mathbf{w}^{D}_{\rho||\gamma}})\mathbf{c}^{(\mathbf{k}_1)}_l & \text{if } d^\star + 1 = D \end{cases}$$

$$\text{In } \widetilde{G}^0_\rho, \qquad \widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho,l}\boldsymbol{\mu}^{d^\star+1}_{\gamma,l} = \begin{cases} \mathbf{u}^{\leq d^\star+1}_{\rho||\gamma} \,||\, \underline{\mathbf{w}^{d^\star}_{\rho}\mathbf{r}^{d^\star+1}_{\gamma}} & \text{if } d^\star + 1 < D \\ (\mathbf{u}^{\leq D}_{\rho||\gamma} \,||\, \underline{\mathbf{w}^{D-1}_{\rho}\mathbf{r}^{d^\star+1}_{\gamma}})\mathbf{c}^{(\mathbf{k}_1)}_l & \text{if } d^\star + 1 = D \end{cases}$$

The only difference lies in how the random elements are generated. It follows from the facts that $\{\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l}\}$ are encoded in either ciphertexts (if $d^\star + 1 < D$) or secret keys (if $d^\star + 1 = D$) of **dIPE**. By the fact that **dIPE** is canonical, its ciphertexts or secret keys contain encodings in group $G_{d^\star+1}$ of elements that depend linearly in $\widetilde{\mathbf{X}}^{d^\star+1}_{\gamma,l}$. Therefore, $G^0_{\rho||1}$ and $\widetilde{G}^0_{\rho||1}$ can be generated from the following distributions respectively,

$$\left\{ \left[\mathbf{w}^{d^\star}_{\rho}\right]_{d^\star+1}, \left\{\left[\mathbf{r}^{d^\star+1}_{\gamma}\right]_{d^\star+1}\right\}_\gamma, \left\{\left[\mathbf{w}^{d^\star+1}_{\rho||\gamma}\right]_{d^\star+1}\right\}_\gamma \right\}$$
$$\left\{ \left[\mathbf{w}^{d^\star}_{\rho}\right]_{d^\star+1}, \left\{\left[\mathbf{r}^{d^\star+1}_{\gamma}\right]_{d^\star+1}\right\}_\gamma, \left\{\left[\mathbf{w}^{d^\star}_{\rho}\mathbf{r}^{d^\star+1}_{\gamma}\right]_{d^\star+1}\right\}_\gamma \right\}$$

This is because, from the above encodings, one can generate the encodings of $\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l}$ with knowledge of values of $\mathbf{u}$'s, $\mathbf{v}$'s, $\mathbf{k}_1$, $\mathbf{k}_2$, and from encodings of $\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l}$, one can emulate $G^0_{\rho+1}$ or $\widetilde{G}^0_\rho$ with additional knowledge of $\{\mathsf{dMSK}_l\}$.

Finally, the indistinguishability of the above two distributions follow directly from the SXDH assumption on group $G_{d^\star+1}$, which concludes the indistinguishability of $G^0_{\rho||1}$ and $\widetilde{G}^0_{\rho||1}$.

It remains to show that $\widetilde{G}^0_{\rho||1}$ is indistinguishable from G^0_ρ, which encrypts the following vectors (the difference from the vectors encrypted in $\widetilde{G}^0_{\rho||1}$ is underlined).

$$\begin{array}{cccccc} \hat{\mathbf{X}}^{d^\star}_{\gamma,l} = & \boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l} & \begin{cases} \mathbf{0} \,||\, \widetilde{\boldsymbol{\nu}}^{d^\star}_{\rho_{\leq d^\star-1}||\gamma,l} & \text{if } \gamma < \rho_{d^\star} \\ \widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho_{\leq d^\star-1}||\gamma,l} \,||\, \mathbf{0} & \text{if } \gamma > \rho_{d^\star} \\ \underline{\widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho,l} \,||\, \mathbf{0}} & \text{if } \gamma = \rho_{d^\star} \end{cases} & \underline{\mathbf{0}} \\ \hat{\mathbf{X}}^{d^\star+1}_{\gamma,l} = & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \underline{\mathbf{0}} \\ \hat{\mathbf{X}}^{d}_{\gamma,l} = & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \underline{\mathbf{0}} \\ & \text{slot } 1 & \cdots & \text{slot } d^\star - 1 & \text{slot } d^\star & \text{slot} \geq d^\star + 1 \end{array}$$

Examine the different values of $\{\hat{\mathbf{X}}^d_{\gamma,l}\}$ for $d \geq d^\star$ in $\widetilde{G}^0_{\rho||1}$ and G^0_ρ, and the values of $\{\hat{\mathbf{X}}^d_{\gamma,l}\}$ for $d < d^\star$ that are the same in these two hybrids as described in Case 3 of Figure 6. They satisfy that for every combination $I \in [\Gamma]^D$ and l, the inner product of $\{\hat{\mathbf{X}}^d_{I_d,l}\}_d$ in in $\widetilde{G}^0_{\rho||1}$ and G^0_ρ is identical. Therefore, it follows from the function hiding of **dIPE** that these two hybrids are indistinguishable.

Proof of Rule 4: $G^1_{\rho||\Gamma} \approx G^1_\rho$, for every $\rho \in [\Gamma]^{d^\star}$ with $1 \leq d^\star < D - 1$. Fix one such ρ and $d^\star$. This rule follows from syntactically the same proof for Rule 3. We sketch the proof below. Hybrid $G^1_{\rho||\Gamma}$ and G^1_ρ encrypt the same set of vectors $\{\hat{\mathbf{X}}^d_{\gamma,l}\}$ for $d < d^\star$, but different vectors for $d \geq d^\star$. In $G^1_{\rho||\Gamma}$, these vectors have values (the difference from the vectors encrypted in $G^0_{\rho||1}$ in Rule 3 is underlined).

$$\begin{array}{lccccc}
\hat{\mathbf{X}}^{d^\star}_{\gamma,l} & = & \boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l} & \begin{cases} \mathbf{0} \;||\; \widetilde{\boldsymbol{\nu}}^{d^\star}_{\rho_{\leq d^\star-1}||\gamma,l} & \text{if } \gamma < \rho_{d^\star} \\ \widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho_{\leq d^\star-1}||\gamma,l} \;||\; \mathbf{0} & \text{if } \gamma > \rho_{d^\star} \\ \mathbf{0} \;||\; \mathbf{0} & \text{if } \gamma = \rho_{d^\star} \end{cases} & \begin{cases} \mathbf{0} & \text{if } \gamma < \rho_{d^\star} \\ \mathbf{0} & \text{if } \gamma > \rho_{d^\star} \\ \mathbf{1} & \text{if } \gamma = \rho_{d^\star} \end{cases} \\
\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l} & = & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \underline{\mathbf{0} \;||\; \widetilde{\boldsymbol{\nu}}^{d^\star+1}_{\rho||\gamma,l}} \;||\; \mathbf{0} \\
\hat{\mathbf{X}}^{d}_{\gamma,l} & = & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \boldsymbol{\mu}^{d}_{\gamma,l} \;||\; \boldsymbol{\nu}^{d}_{\gamma,l}||\; \mathbf{0} \\
 & & \text{slot } 1 & \cdots & \text{slot } d^\star - 1 & \text{slot } d^\star & \text{slot} \geq d^\star + 1
\end{array}$$

We first rely on the SXDH assumption w.r.t. group $G_{d^\star+1}$ to show that $G^1_{\rho||\Gamma}$ is indistinguishable from $\widetilde{G}^1_{\rho||\Gamma}$, where the vectors $\{\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l}\}$ are replaced with

$$\begin{array}{lcccccc}
\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l} & = & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \underline{\mathbf{0} \;||\; \widetilde{\boldsymbol{\nu}}^{d^\star}_{\rho,l}\boldsymbol{\nu}^{d^\star+1}_{\gamma,l}} \;||\; \mathbf{0} \\
 & & \text{slot } 1 & \cdots & \text{slot } d^\star - 1 & \text{slot } d^\star & \text{slot} \geq d^\star + 1
\end{array}$$

Finally, the vectors encoded in G^1_ρ are

$$\begin{array}{lcccccc}
\hat{\mathbf{X}}^{d^\star}_{\gamma,l} & = & \boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star}_{\gamma,l}||\boldsymbol{\nu}^{d^\star}_{\gamma,l} & \begin{cases} \mathbf{0} \;||\; \widetilde{\boldsymbol{\nu}}^{d^\star}_{\rho_{\leq d^\star-1}||\gamma,l} & \text{if } \gamma < \rho_{d^\star} \\ \widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho_{\leq d^\star-1}||\gamma,l} \;||\; \mathbf{0} & \text{if } \gamma > \rho_{d^\star} \\ \underline{\mathbf{0}||\; \widetilde{\boldsymbol{\nu}}^{d^\star}_{\rho,l}} & \text{if } \gamma = \rho_{d^\star} \end{cases} & \mathbf{0} \\
\hat{\mathbf{X}}^{d^\star+1}_{\gamma,l} & = & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \boldsymbol{\mu}^{d^\star+1}_{\gamma,l}||\boldsymbol{\nu}^{d^\star+1}_{\gamma,l} & \mathbf{0} \\
\hat{\mathbf{X}}^{d}_{\gamma,l} & = & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \boldsymbol{\mu}^{d}_{\gamma,l}||\boldsymbol{\nu}^{d}_{\gamma,l} & \mathbf{0} \\
 & & \text{slot } 1 & \cdots & \text{slot } d^\star - 1 & \text{slot } d^\star & \text{slot} \geq d^\star + 1
\end{array}$$

It holds that inner products of all combination of vectors $\hat{\mathbf{X}}$'s in $\widetilde{G}^1_{\rho||\Gamma}$ and G^1_ρ are identical. Thus, it follows from the function hiding of **dIPE** that these two hybrids are indistinguishable.

Proof of Rule 5: $G^b_\rho \approx H^b_\rho$, for every $\rho \in [\Gamma]^{D-1}$ and every b. Fix one such ρ. The proof of this rule is again very similar to that of Rule 3 and 4. We here sketch the proof. The only difference between G^b_ρ and H^b_ρ lies in the values of vectors $\{\hat{\mathbf{X}}^d_{\gamma,l}\}$ and $\{\widetilde{\mathbf{X}}^d_{\gamma,l}\}$ for $d = D-1$ and D.

In hybrid H^b_ρ, these vectors have values.

$$\begin{array}{ccccccc}
\widetilde{\mathbf{X}}^{D-1}_{\gamma,l} & = & \boldsymbol{\mu}^{D-1}_{\gamma,l}||\boldsymbol{\nu}^{D-1}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{D-1}_{\gamma,l}||\boldsymbol{\nu}^{D-1}_{\gamma,l} & \begin{cases} \mathbf{0}\,||\,\widetilde{\boldsymbol{\nu}}^{D-1}_{\rho_{\le D-2}||\gamma,l} & \text{if } \gamma < \rho_{D-1} \\ \widetilde{\boldsymbol{\mu}}^{D-1}_{\rho_{\le D-2}||\gamma,l}\,||\,\mathbf{0} & \text{if } \gamma > \rho_{D-1} \\ \mathbf{0}\,||\,\mathbf{0} & \text{if } \gamma = \rho_{D-1} \end{cases} & 0 \\
\widetilde{\mathbf{X}}^{D}_{\gamma,l} & = & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \begin{cases} \left\langle \widetilde{\boldsymbol{\mu}}^{D}_{\rho||\gamma,l}, \mathbf{1} \right\rangle & \text{if } b=0 \\ \left\langle \widetilde{\boldsymbol{\nu}}^{D}_{\rho||\gamma,l}, \mathbf{1} \right\rangle & \text{in } b=1 \end{cases} \\
 & & \text{slot } 1 & \cdots & \text{slot } D-2 & \text{slot } D-1 & \text{slot } D
\end{array}$$

It follows from the SXDH assumption w.r.t. group G_D that H^b_ρ is indistinguishable to $\widetilde{H}^b_\rho$, where the vectors $\{\widetilde{\mathbf{X}}^D_{\gamma,l}\}$ are replaced with

$$\begin{array}{ccccccc}
\widetilde{\mathbf{X}}^{D}_{\gamma,l} & = & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \begin{cases} \left\langle \underline{\widetilde{\boldsymbol{\mu}}^{D-1}_{\rho,l}\boldsymbol{\mu}^{D}_{\gamma,l}}, \mathbf{1} \right\rangle & \text{if } b=0 \\ \left\langle \underline{\widetilde{\boldsymbol{\nu}}^{D-1}_{\rho,l}\boldsymbol{\nu}^{D}_{\gamma,l}}, \mathbf{1} \right\rangle & \text{in } b=1 \end{cases} \\
 & & \text{slot } 1 & \cdots & \text{slot } D-2 & \text{slot } D-1 & \text{slot } D
\end{array}$$

It now remains to show that $\widetilde{H}^b_\rho$ is indistinguishable to G^b_ρ, which encrypts the following vectors $\{\hat{\mathbf{X}}^{D-1}_{l,n}, \hat{\mathbf{X}}^{D}_{l,n}\}_n$.

$$\begin{array}{ccccccc}
\hat{\mathbf{X}}^{D-1}_{\gamma,l} & = & \boldsymbol{\mu}^{D-1}_{\gamma,l}||\boldsymbol{\nu}^{D-1}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{D-1}_{\gamma,l}||\boldsymbol{\nu}^{D-1}_{\gamma,l} & \begin{cases} \mathbf{0}\,||\,\widetilde{\boldsymbol{\nu}}^{D-1}_{\rho_{\le D-2}||\gamma,l} & \text{if } \gamma < \rho_{D-1} \\ \widetilde{\boldsymbol{\mu}}^{D-1}_{\rho_{\le D-2}||\gamma,l}\,||\,\mathbf{0} & \text{if } \gamma > \rho_{D-1} \\ \underline{\mathbf{0}\,||\,\widetilde{\boldsymbol{\nu}}^{D-1}_{\rho,l}} & \text{if } \gamma = \rho_{D-1} \text{ and } b=1 \\ \underline{\widetilde{\boldsymbol{\mu}}^{D-1}_{\rho,l}\,||\,\mathbf{0}} & \text{if } \gamma = \rho_{D-1} \text{ and } b=0 \end{cases} & \underline{0} \\
\hat{\mathbf{X}}^{D}_{\gamma,l} & = & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \cdots & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \boldsymbol{\mu}^{D}_{\gamma,l}||\boldsymbol{\nu}^{D}_{\gamma,l} & \underline{0} \\
 & & \text{slot } 1 & \cdots & \text{slot } D-2 & \text{slot } D-1 & \text{slot } D
\end{array}$$

For all other coordinates $d < D-1$, $\hat{\mathbf{X}}^d_{l,n} = \widetilde{\mathbf{X}}^d_{l,n}$ in $\widetilde{H}^b_\rho$. Observe that the inner products of all combination of vectors $\widetilde{\mathbf{X}}$'s in $\widetilde{H}^b_\rho$ and $\hat{\mathbf{X}}$'s in G^b_ρ are identical. Thus, it follows from the security of **dIPE** that these two hybrids are indistinguishable.

Proof of Rule 6: $G^1_\rho = G^0_{\rho+1}$ for every $\rho \in [\Gamma]^{d^\star}$ with $1 \le d^\star \le D-1$, such that, $\rho_{d^\star} \ne \Gamma$. Fix such a ρ and $d^\star$. Since $\rho_{d^\star} \ne \Gamma$, $\rho+1$ has form $\rho_{<d^\star}||(\rho_{d^\star}+1)$, where $\rho_{d^\star}+1$ is the letter that follows immediately after $\rho_{d^\star}$ in the alphabet $[\Gamma]$. Thus the vectors $\{\hat{\mathbf{X}}^d_{\gamma,l}\}$ for $d \ne d^\star$ are set identically in these two hybrids. For $d = d^\star$, $\{\hat{\mathbf{X}}^d_{\gamma,l}\}$ are again set identically for all $\gamma \ne \rho_{d^\star}$ and $\ne \rho_{d^\star}+1$. For $d = d^\star$ and $\gamma = \rho_{d^\star}$, in G^1_ρ, $\{\hat{\mathbf{X}}^{d^\star}_{\gamma,l}\}_l$ are set according to the third line in Case 2 of Figure 6,

whereas in $G^0_{\rho+1}$, they are set according to the first line in Case 2; but, the values of the vectors are still identical. For $d = d^\star$ and $\gamma = \rho_{d^\star} + 1$, $\{\hat{\mathbf{X}}^{d^\star}_{\gamma,l}\}_l$ are set according to the second line in Case 2 of Figure 6, whereas in $G^0_{\rho+1}$, they are set according to the fourth line in Case 2; again, the values of the vectors are still identical.

Since all other random variables are sampled identically in G^0_ρ and G^1_ρ. These two distributions are identical. $\square$

8 FE for Degree-D Polynomials from Degree-D MMaps

In this section, we construct FE schemes $\{\mathbf{FE}^{D,N}\}$ for degree D polynomials in $\mathcal{R}$, from the SXDH assumption over degree-D MMaps in $\mathcal{R}$. Importantly, our FE scheme has linear efficiency, that is, encrypting a length-N input vector takes time $N\,\mathrm{poly}(\lambda)$. An overview of the scheme is presented in Section 2; below, we present the formal construction.

8.1 Construction

Fix any polynomial N. We construct a secret key FE scheme $\mathbf{FE} = \mathbf{FE}^{D,N}$ for computing degree D polynomials over inputs of length N over ring $\mathcal{R}$, using the following building blocks:

- The canonical degree-D HIPE scheme $\mathbf{dIPE} = \mathbf{hIPE}^{D,M} = (\mathsf{dIPE.Setup}, \mathsf{dIPE.Enc}, \mathsf{dIPE.KeyGen}, \mathsf{dIPE.Dec})$ from SXDH on degree-D MMaps in Section 7, for a specific *constant* input length $M = O(D)$ specified below.
- The ABCP public-key IPE scheme $\mathbf{IPE} = \mathbf{IPE}^L = (\mathsf{IPE.Setup}, \mathsf{IPE.Enc}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Dec})$ from DDH groups by [ABCP15] (reviewed in Section 6.2), for inputs of length $L = O(N^D)$ specified below.
- The canonical degree-2 IPE scheme $\mathbf{tIPE} = \mathbf{tIPE}^2 = (\mathsf{tIPE.Setup}, \mathsf{tIPE.Enc}, \mathsf{tIPE.KeyGen}, \mathsf{tIPE.Dec})$ for inputs of length 2 from SXDH on bilinear maps in Section 6.4.
- Degree-D multilinear pairing groups described by $\mathsf{pp} = (p, G_1, \cdots, G_D, G_{D+1}, \mathbf{pair})$.

Below, we will use $\otimes \mathbf{s}^{\leq d}$ to denote the tensor product of d vectors $\mathbf{s}^1, \cdots, \mathbf{s}^d$, and for any index $I = (I_1, \cdots, I_d)$, we denote the I^{th} elements in the tensor product as $s^{\leq d}_I$.

$$\otimes \mathbf{s}^{\leq d} = \mathbf{s}^1 \otimes \cdots \otimes \mathbf{s}^d \qquad \otimes \mathbf{s}^{\leq d}_I = s^{\leq d}_I = \prod_{i \leq d} s^i_{I_i}$$

For notational convenience, we overload the notations $\otimes \mathbf{x}^{\leq d}$ and $x^{\leq d}_I$ to also mean the tensor product of the same vector $\mathbf{x}$ for d times and the I^{th} element in the tensor product.

$$\otimes \mathbf{x}^{\leq d} = \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{d \text{ times}} \qquad \otimes \mathbf{x}^{\leq d}_I = x^{\leq d}_I = \prod_{i \leq d} x_{I_i}$$

Whether the notations denote the former or latter depends on whether there exist different vectors $\mathbf{s}^1, \cdots, \mathbf{s}^d$ or only a single vector $\mathbf{x}$, which is clear in the context below.

Our FE scheme $\mathbf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec})$ proceeds as follows. We inline analysis of correctness in italic font in the description of the construction below.

- $\mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ does the following
  - Sample D vectors $\mathbf{s}^1, \cdots \mathbf{s}^D \xleftarrow{\$} \mathcal{R}^N$.
    Note: The tensor product of the vectors $\otimes\mathbf{s}^{\leq D}$ serves as the secret key $\mathsf{iMSK} = \otimes\mathbf{s}^{\leq D}$ of **IPE**.
  - Sample a master secret key of $\mathbf{tIPE}^2$, $\mathsf{tMSK} \xleftarrow{\$} \mathsf{tIPE.Setup}(1^\lambda, (p, G_{D-1}, G_D, \mathsf{null}, \mathsf{null}))$ with source group G_{D-1} and G_D. (Since $\mathbf{tIPE}^2$ has canonical form, its setup algorithm does not depend on the target group, nor the pairing function, which can be set to null.)

  Output $\mathsf{msk} = (\mathbf{s}^1, \cdots, \mathbf{s}^D, \mathsf{tMSK}, \mathsf{pp})$.

- $\mathsf{FE.KeyGen}(\mathsf{msk}, \mathbf{c})$ on input the length-N^D vector $\mathbf{c}$ listing the coefficients of a degree D polynomial $f_{\mathbf{c}}(\mathbf{x}) = \langle \mathbf{c}, \otimes\mathbf{x}^{\leq d} \rangle$, samples the following:
  - Generates a secret key of **IPE** for vector $\mathbf{c}$, using $\otimes\mathbf{s}^{\leq D}$ as the secret key,
$$\mathsf{iSK} = \left(\left\langle \otimes\mathbf{s}^{\leq D}, \mathbf{c} \right\rangle, \mathbf{c}\right) = \mathsf{IPE.KeyGen}\left(\otimes\mathbf{s}^{\leq D}, \mathbf{c}\right) .$$
  - Generates a secret key of **tIPE** for vector $\left\langle \otimes\mathbf{s}^{\leq D}, \mathbf{c} \right\rangle || 0$,
$$\mathsf{tSK} \xleftarrow{\$} \mathsf{tIPE.KeyGen}\left(\mathsf{tMSK}, \left(\left\langle \otimes\mathbf{s}^{\leq D}, \mathbf{c} \right\rangle || 0\right)\right) .$$

  Output secret key $\mathsf{SK} = (\mathbf{c}, \mathsf{tSK})$.

  Note: SK *is almost the same as the* **IPE** *secret key of* $\mathbf{c}$*, except that* $\left\langle \otimes\mathbf{s}^{\leq D}, \mathbf{c} \right\rangle$ *is not revealed in the plaintext, but encoded in a secret key of* **tIPE**.

- $\mathsf{FE.Enc}(\mathsf{msk}, \mathbf{x})$ samples the following
  - Sample a random element $r \xleftarrow{\$} \mathcal{R}$.
    Note: r serves as the randomness for **IPE** *encryption.*
  - Encrypt $-r||0$ using **tIPE**, $\mathsf{tCT} \xleftarrow{\$} \mathsf{tIPE.Enc}(\mathsf{tMSK}, (-r||0))$.
  - Generate a master secret key of **dIPE**, $\mathsf{dMSK} \xleftarrow{\$} \mathsf{dIPE.Setup}(1^\lambda, \mathsf{pp})$.
  - Prepare the following vectors $\{\boldsymbol{\chi}_n^d\}_{d\in[D], n\in[N]}$
$$\boldsymbol{\chi}_n^d = \begin{cases} x_n \,||\, s_n^d & \text{if } d < D \\ x_n \,||\, r s_n^D & \text{if } d = D \end{cases}$$
  - Pad the above vectors with zeros to get $\{\mathbf{X}_n^d\}_{d\in[D], n\in[N]}$,
$$\mathbf{X}_n^d = \boldsymbol{\chi} \,||\, \mathbf{0} \qquad \text{where } M = |\mathbf{X}_n^d| = 2(D-1)|\boldsymbol{\chi}_l^d| + 1 = 4D - 3 = \Theta(D) .$$
  - Generate the following **dIPE** ciphertexts and secret keys.
$$\left\{\mathsf{dCT}_n^d \xleftarrow{\$} \mathsf{dIPE.Enc}(\mathsf{dMSK}, \mathbf{X}_n^d)\right\}_{d<D, n\in[N]}$$
$$\left\{\mathsf{dSK}_n \xleftarrow{\$} \mathsf{dIPE.KeyGen}(\mathsf{dMSK}, \mathbf{X}_n^D)\right\}_{n\in[N]}$$

Output $\mathsf{CT} = (\mathsf{tCT}, \{\mathsf{dCT}^d_n\}_{d<D, n\in[N]}, \{\mathsf{dSK}_n\}_{n\in[N]})$.

Note: The ciphertext CT *implicitly encodes an* **IPE** *ciphertext* iCT *of the monomials* $\otimes\mathbf{x}^{\leq d}$ *under secret key* $\otimes\mathbf{s}^{\leq D}$ *and public key* $\left[\otimes\mathbf{s}^{\leq D}\right]_{D+1}$*. Such a ciphertext looks as follows,*

$$\mathsf{IPE.Enc}(\left[\otimes\mathbf{s}^{\leq D}\right]_{D+1}, \otimes\mathbf{x}^{\leq d};\ r) = \left[-r\right]_{D+1},\ \left[r(\otimes\mathbf{s}^{\leq D}) + \otimes\mathbf{x}^{\leq d}\right]_{D+1} = \mathsf{iCT}\ .$$

For convenience, we denote by iCT_0 *the encoding* $[r]_{D+1}$*, and* iCT_I *the encoding of the* I^{th} *element* $\mathrm{ict}_I = rs^{\leq D}_I + x^D_I$ *in the second part.*

The random element r *encoded in* iCT_0 *is encrypted in the* **tIPE** *ciphertext* tCT. *On the other hand,* iCT_I *can be computed by decrypting the combination of secret key and ciphertexts* $(\mathsf{dSK}_{I_D}, \mathsf{dCT}^1_{I_1}, \cdots, \mathsf{dCT}^{D-1}_{I_{D-1}})$ *of* **dIPE***.*

$$\begin{aligned}\mathsf{dIPE.Dec}(\mathsf{dSK}_{I_D}, \mathsf{dCT}^1_{I_1}, \cdots, \mathsf{dCT}^{D-1}_{I_{D-1}}) &= \left[\langle \mathbf{X}^1_{I_1}, \cdots, \mathbf{X}^D_{I_D}\rangle\right]_{D+1} \\ &= \left[\langle \boldsymbol{\chi}^1_{I_1}, \cdots, \boldsymbol{\chi}^D_{I_D}\rangle\right]_{D+1} = \left[rs^{\leq D}_I + x^D_I\right]_{D+1} = \mathsf{iCT}_I\end{aligned}$$

- $\mathsf{FE.Dec}(\mathsf{SK}, \mathsf{CT})$ does the following

  1. Decrypt $\mathsf{tSK}, \mathsf{tCT}$ using **tIPE** to obtain $\Lambda_1 = \mathsf{tIPE.Dec}(\mathsf{tSK}, \mathsf{tCT})$.

     (Note that since **tIPE** is canonical, its decryption algorithm involves evaluating a quadratic function over encodings in its secret key and ciphertext in groups G_D and G_{D-1} respectively. Using the degree-D multilinear map and the generators of other groups $G_1, \cdots, G_{D-2}$, one can obtain an encoding of the output of the quadratic function in the target group G_{D+1}.)

     Note: Recall that decrypting an **IPE** *ciphertext* $\mathsf{iCT} = [-r]_{D+1}, \{\mathsf{iCT}_I\}_I$ *with a secret key* $\mathsf{iSK} = (\langle\otimes\mathbf{s}^{\leq D},\ \mathbf{y}\rangle,\ \mathbf{y})$ *involves homomorphically evaluating the inner product between the ciphertext and the secret key. That is,*

     $$\mathsf{IPE.Dec}(\mathsf{iSK}, \mathsf{iCT}) = [-r]_{D+1}\langle\otimes\mathbf{s}^{\leq D},\ \mathbf{y}\rangle\ +\ \langle\mathbf{y}\ ,\ \{\mathsf{iCT}_I\}\rangle = \Lambda_1 + \Lambda_2$$

     The above step 1 computes exactly the first term Λ_1*,*

     $$\mathsf{tIPE.Dec}(\mathsf{tSK}, \mathsf{tCT}) = \left[\langle\ \langle\otimes\mathbf{s}^{\leq D}, \mathbf{c}\rangle\,||0\ \ ,\ \ -r||0\ \rangle\right]_{D+1} = \left[-r\,\langle\otimes\mathbf{s}^{\leq D}, \mathbf{c}\rangle\right]_{D+1} = \Lambda_1$$

  2. For every $I \in [N]^D$, decrypt $\mathsf{dSK}_{I_D}, \mathsf{dCT}^1_{I_1}, \cdots, \mathsf{dCT}^{D-1}_{I_{D-1}}$ using **dIPE** to obtain $\mathsf{iCT}_I = \mathsf{dIPE.Dec}(\mathsf{dSK}_{I_D}, \mathsf{dCT}^1_{I_1}, \cdots, \mathsf{dCT}^{D-1}_{I_{D-1}})$.

     Note: iCT_I *is the* I^{th} *element in the* **IPE** *ciphertext of* $\otimes\mathbf{x}^{\leq d}$ *under secret key* $\otimes\mathbf{s}^{\leq D}$

     $$\mathsf{dIPE.Dec}(\mathsf{dSK}_{I_D}, \mathsf{dCT}^1_{I_1}, \cdots, \mathsf{dCT}^{D-1}_{I_{D-1}}) = \left[\langle \mathbf{X}^1_{I_1}, \cdots, \mathbf{X}^D_{I_D}\rangle\right]_{D+1} = \left[rs^{\leq D}_I + x^D_I\right]_{D+1} = \mathsf{iCT}_I$$

     Their concatenation can be written as

     $$\mathsf{iCT} = \left[r\otimes\mathbf{s}^{\leq D} + \otimes\mathbf{x}^{\leq d}\right]_{D+1}.$$

3. Homomorphically evaluate $\langle \mathsf{iCT}, \mathbf{c}\rangle$ to obtain encoding Λ_2 in G_{D+1}, followed by homomorphically adding Λ_1 and Λ_2 to obtain $[y]_{D+1}$. Output 1 iff the output encoding encodes zero.

   Note: The above two steps 2 and 3 correspond to computing the second term Λ_2 in the decryption equation of **IPE**

   $$\langle \mathsf{iCT}, \mathbf{c}\rangle = \left[\left\langle \left(r \otimes \mathbf{s}^{\leq D} + \otimes \mathbf{x}^{\leq d}\right),\, \mathbf{c}\right\rangle\right]_{D+1} = \left[r\left\langle \otimes \mathbf{s}^{\leq D},\, \mathbf{c}\right\rangle \,+\, \left\langle \otimes \mathbf{x}^{\leq d},\, \mathbf{c}\right\rangle\right]_{D+1} = \Lambda_2$$

   followed by obtaining an encoding of the output,

   $$\Lambda_1 + \Lambda_2 = \left[\left\langle \otimes \mathbf{x}^{\leq d},\, \mathbf{c}\right\rangle\right]_{D+1} = \left[f_{\mathbf{c}}(\mathbf{x})\right]_{D+1}.$$

   Therefore the scheme **FE** *is correct.*

Efficiency It is easy to see that the key generation $\mathsf{FE.KeyGen}$ runs in time $N^D \operatorname{poly}(\lambda)$. To analyze the run time of the encryption and decryption algorithm, recall that both the encryption $\mathsf{dIPE.Enc}$ and key generation $\mathsf{dIPE.KeyGen}$ algorithms of the HIPE scheme $\mathbf{dIPE}^{D,M}$ run in time $\mathsf{Time}^D(M) = M^D \operatorname{poly}(\lambda)$ (see Section 7.2.3). Since $M = O(D)$ and D is a constant, $\mathsf{Time}^D(M) = \operatorname{poly}(\lambda)$ is bounded by a fixed polynomial in the security parameter. The encryption algorithm $\mathsf{FE.Enc}$ of **FE** involves generating one ciphertext of $\mathbf{tIPE}^2$ and N ciphertexts of **dIPE** at every coordinate in $[D-1]$ and N secret keys of **dIPE**. The total time is thus is bounded by

$$\mathsf{Time}_{\mathsf{FE.Enc}}(\lambda, D, N) = \operatorname{poly}(\lambda) + DN \operatorname{poly}(\lambda) = N \operatorname{poly}(\lambda)$$

In other words, encryption time is linear in the input length. On the other hand, the decryption time of $\mathbf{dIPE}^{D,M}$ is bounded by $M^{\Theta(D^2)} \operatorname{poly}(\lambda)$ (see Section 7.2.3), which in turn is again bounded by a fixed polynomial. The decryption algorithm $\mathsf{FE.Dec}$ involves decrypting N^D combination of **dIPE** secret key and ciphertexts, and decrypting one pair of secret key and ciphertext of **tIPE**. Thus, decryption time is bounded by.

$$\mathsf{Time}_{\mathsf{FE.Dec}}(\lambda, D, N) = N^D \operatorname{poly}(\lambda) + \operatorname{poly}(\lambda) = N^D \operatorname{poly}(\lambda)$$

8.2 Security Proof

We prove that $\mathbf{FE}^{D,N}$ is IND-secure.

Proposition 2. *Assume SXDH on degree-D multilinear pairing groups. The scheme* $\mathbf{FE}^{D,N}$ *described above is selectively IND-secure.*

Proof. Fix any polynomial L and any ensembles of sets of vectors $\{\{\mathbf{x}_l^0, \mathbf{x}_l^1, \mathbf{c}_l\}_{l \in [L(\lambda)]}\}_{\lambda \in \mathbb{N}}$, such that, $\mathbf{x}_l^0, \mathbf{x}_l^1 \in \mathcal{R}^{N(\lambda)}$, $\mathbf{c}_\gamma \in \mathcal{R}^{N(\lambda)^D}$ and the following holds.

$$\forall\, l,\ j \in [L], \quad f_{\mathbf{c}_j}(\mathbf{x}_l^0) = \left\langle \otimes (\mathbf{x}_l^0)^{\leq D}, \mathbf{c}_j \right\rangle = \left\langle \otimes (\mathbf{x}_l^1)^{\leq D}, \mathbf{c}_j \right\rangle = f_{\mathbf{c}_j}(\mathbf{x}_l^1)$$

We need to show the indistinguishability of two ensembles of distributions $\{\mathcal{D}_0(\lambda)\}_\lambda$ and $\{\mathcal{D}_1(\lambda)\}_\lambda$ defined below.

$$\{\mathcal{D}_b(\lambda)\}_\lambda = \left\{ \begin{array}{l} \mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp}) \\ \left\{\mathsf{CT}_l \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{msk}, \mathbf{x}_l^b)\right\}_{l \in [L]} \\ \left\{\mathsf{SK}_l \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{msk}, \mathbf{c}_l)\right\}_{l \in [L]} \end{array} \quad : \quad \mathsf{pp},\ \{\mathsf{SK}_l,\ \mathsf{CT}_l\}_{l \in [L]} \right\}_{\lambda \in \mathbb{N}}$$

To show the indistinguishability of ciphertexts of $\{\mathbf{x}_l^0\}$ from that of $\{\mathbf{x}_l^1\}$, we go through a sequence of hybrid in which the encrypted vectors are exchanged one by one.

Hybrid Hyb_ℓ for $0 \le \ell \le L$ is generated identically as $\mathcal{D}_0$ except that the first ℓ ciphertexts encrypts $\mathbf{x}_l^1$ as opposed to $\mathbf{x}_l^0$. Formally,

$$\{\mathsf{Hyb}_\ell(\lambda)\}_\lambda = \left\{ \begin{array}{l} \mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp}) \\ \left\{ \mathsf{CT}_l \xleftarrow{\$} \mathsf{FE.Enc}\left(\mathsf{msk}, \begin{cases} \mathbf{x}_l^1 & \text{if } l \le \ell \\ \mathbf{x}_l^0 & \text{if } l > \ell \end{cases} \right) \right\}_{1 \le l \le \ell} \\ \left\{ \mathsf{SK}_l \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{msk}, \mathbf{c}_l) \right\}_{l \in [L]} \end{array} \quad : \quad \mathsf{pp},\ \{\mathsf{SK}_l,\ \mathsf{CT}_l\}_{l \in [L]} \right\}_{\lambda \in \mathbb{N}}$$

It is easy to see that $\mathsf{Hyb}_0 = \mathcal{D}_0$ and $\mathsf{Hyb}_L = \mathcal{D}_1$. Then the the proposition follows immediately from the following lemma.

Lemma 12. *For every $\ell \in [L]$, hybrids $\mathsf{Hyb}_{\ell-1}(\lambda)$ and $\mathsf{Hyb}_\ell(\lambda)$ are indistinguishable.*

□

Indistinguishability of $\mathsf{Hyb}_{\ell-1}$ and Hyb_ℓ

Proof of Lemma 12. The only difference in $\mathsf{Hyb}_{\ell-1}$ and Hyb_ℓ lies in the vector encrypted in the ℓ^{th} ciphertext, in the former, it is $\mathbf{x}_\ell^0$ and in the latter it is $\mathbf{x}_\ell^1$. For convenience, we will denote the list of vectors encrypted in $\mathsf{Hyb}_{\ell-1}$ as $\mathbf{u}_1, \cdots, \mathbf{u}_L$ and that encrypted in Hyb_ℓ as $\mathbf{v}_1, \cdots, \mathbf{v}_L$.

$$\forall l < \ell,\ \mathbf{u}_l = \mathbf{v}_l = \mathbf{x}_l^1, \qquad \mathbf{u}_\ell = \mathbf{x}_\ell^0,\ \mathbf{v}_\ell = \mathbf{x}_\ell^1, \qquad \forall l > \ell,\ \mathbf{u}_l = \mathbf{v}_l = \mathbf{x}_l^0 \tag{12}$$

(Using these notations helps us to "align" parts of this proof with that of the proof of Proposition 2 which share the same arguments.)

We show that both $\mathsf{Hyb}_{\ell-1}$ and Hyb_ℓ are indistinguishable to an intermediate hybrid **Mid**. To show the indistinguishability between $\mathsf{Hyb}_{\ell-1}^0$ and **Mid**, we construct a sequence of $2N^{D-1} + 2$ hybrid distributions **Init**, $\{H_\rho^b\}_{b \in \{0,1\}, \rho \in [N]^{D-1}}$, **Mid**′, and show that $\mathsf{Hyb}_{\ell-1}$ is indistinguishable to **Init**, **Mid**′ is indistinguishable to **Mid**, and all neighboring hybrids are indistinguishable; then by a hybrid argument, $\mathsf{Hyb}_{\ell-1}$ is indistinguishable to **Mid**. Below we describe all the hybrids and show their indistinguishability. After that, we note that it follows from syntactically the same proof that Hyb_ℓ is also indistinguishable to **Mid**.

Hybrid Init(λ) is generated identically as $\mathsf{Hyb}_{\ell-1}$ except that the **tIPE** ciphertext tCT_ℓ contained in the ℓ^{th} ciphertext CT_ℓ, and the **tIPE** secret keys $\{\mathsf{tSK}_l\}$ contained in secret keys $\{\mathsf{SK}_l\}$ are generated differently:

- tCT_ℓ encrypts vector $(0||1)$ as opposed to $(-r_\ell||0)$, and
- for every $l \in [L]$, tSK_l encodes vector $(\langle \otimes \mathbf{s}^{\le D}, \mathbf{c}_l \rangle \,||\, -r_\ell \langle \otimes \mathbf{s}^{\le D}, \mathbf{c}_l \rangle)$ as opposed to $(\langle \otimes \mathbf{s}^{\le D}, \mathbf{c}_l \rangle \,||0)$.

See figure 7 for a formal description. (The difference from distribution $\mathsf{Hyb}_{\ell-1}$ is underlined.)

Lemma 13. *The ensembles $\{\mathsf{Hyb}_{\ell-1}(\lambda)\}_\lambda$ and $\{\mathbf{Init}(\lambda)\}_\lambda$ are indistinguishable.*

Hybrid Distribution Init(λ)

Generate the following

- $\mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ and parse $\mathsf{msk} = (\mathbf{s}^1, \cdots, \mathbf{s}^D, \mathsf{tMSK}, \mathsf{pp})$.
- For every $l \in [L]$, generate $\mathsf{CT}_l \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{msk}, \mathbf{u}_l)$.
  Let r_ℓ be the random element sampled when generating CT_ℓ.
  Parse $\mathsf{CT}_\ell = (\mathsf{tCT}_\ell, \{\mathsf{dCT}^d_{\ell,n}\}_{d<D,n\in[N]}, \{\mathsf{dSK}_{\ell,n}\}_{n\in[N]})$.
  Replace tCT_ℓ with $\mathsf{tCT}_\ell \xleftarrow{\$} \mathsf{tIPE.Enc}(\mathsf{tMSK}, \underline{(0||1)})$.
- For every $j \in [L]$, generate $\mathsf{SK}_j = (\mathbf{c}_j, \mathsf{tSK}_j)$ for coefficient vector $\mathbf{c}_j$ with

$$\mathsf{tSK}_j \xleftarrow{\$} \mathsf{tIPE.KeyGen}\left(\mathsf{tMSK}, \left\langle \otimes \mathbf{s}^{\leq D}, \mathbf{c}_j \right\rangle || \underline{-r_\ell \left\langle \otimes \mathbf{s}^{\leq D}, \mathbf{c}_j \right\rangle}\right) .$$

Output $\{\mathsf{SK}_l,\ \mathsf{CT}_l\}_{l\in[L]}$

Figure 7: Initial hybrid distribution **Init**(λ) for proving security of $\mathbf{FE}^{D,N}$

Proof. The only difference between **Init** and $\mathsf{Hyb}_{\ell-1}$ lies in the ℓ^{th} **tIPE** ciphertext tCT_ℓ and all **tIPE** secret keys $\{\mathsf{tSK}_l\}$. But, for every $l \in [L]$, the output inner product obtained when decrypting tCT_ℓ with tSK_l is identical, namely $-r_\ell \left\langle \otimes \mathbf{s}^{\leq D}, \mathbf{c}_l \right\rangle$, in $\mathsf{Hyb}_{\ell-1}$ and **Init**. Moreover, all other **tIPE** ciphertexts tCT_l for $l \neq \ell$ encrypt the same vector in both hybrids. Therefore, by the function hiding property of **tIPE**, we have that $\mathsf{Hyb}_{\ell-1}$ and **Init** are indistinguishable. □

Hybrid $H^b_\rho(\lambda)$ Hybrid H^b_ρ for any $\rho \in [N]^{D-1}$ and $b \in \{0,1\}$ is identical to **Init**, except that the set of vectors $\{\widetilde{\mathbf{X}}^d_{l,n}\}_{d\leq D,n\in[N]}$ encoded in the **dIPE** ciphertexts and secret keys $\{\mathsf{dCT}^d_{l,n}, \mathsf{dSK}_{l,n}\}_{d<D,n\in[N]}$ contained in $\{\mathsf{CT}_l\}_l$ are different, as well as the vectors $\{\mathbf{t}_l\}_l$ encoded in the **tIPE** secret keys $\{\mathsf{tSK}_l\}_l$ contained in $\{\mathsf{SK}_l\}_l$. Next, we describe how the $\widetilde{\mathbf{X}}$ and $\mathbf{t}$ vectors are set; a formal description of the hybrid can be found in Figure 8.

<u>SET THE $\widetilde{\mathbf{X}}$ VECTORS.</u> Recall that in **Init** (the same as in Hyb_ℓ)

$$\mathbf{X}^d_{l,n} = \boldsymbol{\mu}^d_{l,n} \,||\, \mathbf{0} \qquad \text{where } \boldsymbol{\mu}^d_{l,n} = \begin{cases} u_{l,n} \,||\, s^d_n & \text{if } d < D \\ u_{l,n} \,||\, r_l s^D_n & \text{if } d = D \end{cases}$$

Since $\mathbf{X}^d$ has length $4(D-1)+1$, we can parse it as containing D slots, where the first $D-1$ slots can hold 2 vectors of length-2 each, and the last slot can hold a single ring element, that is,

$$\mathbf{X}^d_{l,n} = \underbrace{\boldsymbol{\mu}^d_{l,n}||\mathbf{0}}_{\text{slot } 1} \quad \underbrace{\mathbf{0}||\mathbf{0}}_{\text{slot } 2} \quad \underset{\cdots}{\cdots} \quad \underbrace{\mathbf{0}||\mathbf{0}}_{\text{slot } D-1}, \quad \underbrace{0}_{\text{slot } D}$$

In addition to vectors $\boldsymbol{\mu}^d_{l,n}$, hybrid H^b_ρ generates vectors $\boldsymbol{\nu}^d_{l,n}$ corresponding to input vectors $\mathbf{v}_l$ as follows:

$$\boldsymbol{\nu}^d_{l,n} = \begin{cases} v_{l,n} \,||\, s^d_n & \text{if } d < D \\ v_{l,n} \,||\, r_l s^D_n & \text{if } d = D \end{cases}$$

Hybrid Distribution $H^b_\rho(\lambda)$ for $\rho \in [N]^{D-1}$

Generate the following:

- $\mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ and parse $\mathsf{msk} = (\mathsf{sMSK}, \{\mathsf{dMSK}_l\}_{l\in[L]})$ and $\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2)$.
- For every $l \in [L]$, sample $r_l \xleftarrow{\$} \mathcal{R}$ and $\mathsf{dMSK}_l \xleftarrow{\$} \mathsf{dIPE.Setup}(1^\lambda, \mathsf{pp})$.
  Generate $\mathsf{CT}_l = (\mathsf{tCT}_l, \{\mathsf{dCT}^d_{l,n}, \mathsf{dSK}_{l,n}\}_{d,n})$ as follows:

$$\mathsf{tCT}_l \xleftarrow{\$} \mathsf{tIPE.Enc}\left(\mathsf{tMSK}, \begin{cases}(-r_l||0) & \text{if } l \neq \ell \\ (0||1) & \text{if } l = \ell\end{cases}\right) \quad \begin{matrix}\left\{\mathsf{dCT}^d_{l,n} \xleftarrow{\$} \mathsf{dIPE.Enc}(\mathsf{dMSK}_l, \underline{\widetilde{\mathbf{X}}^d_{l,n}})\right\}_{d<D, n\in[N]} \\ \left\{\mathsf{dSK}_{l,n} \xleftarrow{\$} \mathsf{dIPE.KeyGen}(\mathsf{dMSK}_l, \underline{\widetilde{\mathbf{X}}^D_{l,n}})\right\}_{n\in[N]}\end{matrix}$$

  where the vectors $\{\widetilde{\mathbf{X}}^d_{l,n}\}_{d\in[D],n\in[N]}$ are set as follows

$$\widetilde{\mathbf{X}}^D_{l,n} = \underbrace{\boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n}}_{\text{slot } 1} \quad \underbrace{\boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n}}_{\text{slot } 2} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n}}_{\text{slot } D-1} \quad \underbrace{\begin{cases}\left\langle \widetilde{\boldsymbol{\mu}}^D_{l,\rho||n}, \mathbf{1}\right\rangle & \text{in } H^0_\rho \\ \left\langle \widetilde{\boldsymbol{\nu}}^D_{l,\rho||n}, \mathbf{1}\right\rangle & \text{in } H^1_\rho\end{cases}}_{\text{slot } D}$$

$$\widetilde{\mathbf{X}}^d_{l,n} = \underbrace{\boldsymbol{\mu}^d_{l,n}||\boldsymbol{\nu}^d_{l,n}}_{\text{slot } 1} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^d_{l,n}||\boldsymbol{\nu}^d_{l,n}}_{\text{slot } d-1} \quad \underbrace{\begin{cases}\mathbf{0} \,||\, \widetilde{\boldsymbol{\nu}}^d_{l,\rho_{\le d-1}||n} & \text{if } n < \rho_d \\ \widetilde{\boldsymbol{\mu}}^d_{l,\rho_{\le d-1}||n} \,||\, \mathbf{0} & \text{if } n > \rho_d \\ \mathbf{0} \;\;||\;\; \mathbf{0} & \text{if } n = \rho_d\end{cases}}_{\text{slot } d} \quad \underbrace{\begin{cases}\mathbf{0} & \text{if } n < \rho_d \\ \mathbf{0} & \text{if } n > \rho_d \\ \mathbf{1} & \text{if } n = \rho_d\end{cases}}_{\text{slot } > d}$$

- For every $j \in [L]$, generate $\mathsf{SK}_j = (\mathbf{c}_j, \mathsf{tSK}_j)$ for coefficient vector $\mathbf{c}_j$ with

$$\mathsf{tSK}_j \xleftarrow{\$} \mathsf{tIPE.KeyGen}\left(\mathsf{tMSK}, \left\langle \underline{S(\rho)}, \mathbf{c}_j \right\rangle || -r_\ell \left\langle \underline{S(\rho)}, \mathbf{c}_j\right\rangle + \underline{\Delta^b_j(\rho)}\right),$$

  where $S(\rho)$ and $\Delta^b_j(\rho)$ defined as follows:

  - $\forall I \in [N^D]$, $(S(\rho))_I = w^d_{\rho_{\le d-1}||I_d} s^{d+1}_{I_{d+1}} \cdots s^D_{I_D}$ if the longest prefix I share with ρ is $\rho_{\le d-1}$ (see Equation 13), and
  - $\Delta^b_j(\rho) = \begin{cases}\sum_{\rho'<\rho} \delta_j(\rho') & \text{in } H^0_\rho \\ \sum_{\rho'\le\rho} \delta_j(\rho') & \text{in } H^1_\rho\end{cases}$ with $\delta_j(\rho') = \sum_{\substack{I=\rho'||n \\ n\in[N]}} \mathbf{c}_{j,I}\left(u^{\le D}_{\ell,I} - v^{\le D}_{\ell,I}\right)$

Output $\{\mathsf{SK}_l,\ \mathsf{CT}_l\}_{l\in[L]}$

Figure 8: Hybrid $H^b_\rho(\lambda)$ for $\rho \in [N]^{D-1}$ for proving security of $\mathbf{FE}^{D,N}$

H^b_ρ also generates vectors $\widetilde{\boldsymbol{\mu}}^d_{l,\rho_{\le d-1}||n}, \widetilde{\boldsymbol{\nu}}^d_{l,\rho_{\le d-1}||n}$ for every prefix of form $\rho_{\le d-1}||n$ of length d as follows. These vectors are derived from the partial products associated with the prefix $\rho_{\le d-1}||n$. Take a partial product of $\boldsymbol{\mu}$'s for example,

$$\boldsymbol{\mu}^{\le d}_{l,\rho_{\le d-1}||n} = \left(\prod_{i\le d-1} \boldsymbol{\mu}^i_{l,\rho_i}\right) \boldsymbol{\mu}^d_{l,n} = \begin{cases} u^{\le d}_{l,\rho_{\le d-1}||n} \;\Big|\Big|\; s^{\le d}_{\rho_{\le d-1}||n} & \text{if } d < D \\ u^{\le D}_{l,\rho||n} \;\Big|\Big|\; r_l s^{\le D}_{\rho||n} & \text{if } d = D \end{cases}$$

where $s^{\le d}_I = \prod_{i\le d} s^i_{I_i}$ and $u^{\le d}_I = \prod_{i \le d} u_{I_i}$. Then, $\widetilde{\boldsymbol{\mu}}^{\le d}_{l,\rho_{\le d-1}||n}$ is derived by replacing the partial product of key elements $s^{\le d}_{\rho_{\le d-1}||n}$ in it, with an independently and randomly sampled element $w^d_{\rho_{\le d-1}||n} \xleftarrow{\$} \mathcal{R}$. Vector $\widetilde{\boldsymbol{\nu}}^{\le d}_{l,\rho_{\le d-1}||n}$ is derived similarly from the $\boldsymbol{\nu}$ vectors. More precisely,

$$\widetilde{\boldsymbol{\mu}}^d_{l,\rho_{\le d-1}||n} = \begin{cases} u^{\le d}_{l,\rho_{\le d-1}||n} \;\Big|\Big|\; w^d_{\rho_{\le d-1}||n} & \text{if } d < D \\ u^{\le D}_{l,\rho||n} \;\Big|\Big|\; r_l w^D_{\rho||n} & \text{if } d = D \end{cases} \qquad \widetilde{\boldsymbol{\nu}}^d_{\rho_{\le d-1}||n,l} = \begin{cases} v^{\le d}_{l,\rho_{\le d-1}||n} \;\Big|\Big|\; w^d_{\rho_{\le d-1}||n} & \text{if } d < D \\ v^{\le D}_{l,\rho||n} \;\Big|\Big|\; r_l w^D_{\rho||n} & \text{if } d = D \end{cases}$$

where for any $n \in [N]$, $w^1_n = s^1_n$, and $w^d_{\rho_{\le d-1}||n} \xleftarrow{\$} \mathcal{R}$ for $d > 1$.

Fact 3. *Observe that since $w^1_n = s^1_n$, $\widetilde{\boldsymbol{\mu}}^1_{l,n} = \boldsymbol{\mu}_{l,n}$, and $\widetilde{\boldsymbol{\nu}}^1_{l,n} = \boldsymbol{\nu}_{l,n}$.*

H^b_ρ encodes in every ciphertext CT_l a set of vectors $\{\widetilde{\mathbf{X}}^d_{l,n}\}_{d,n}$ depending on $\{\boldsymbol{\mu}^d_{l,n}, \boldsymbol{\nu}^d_{l,n}\}_{d,n}$ and $\{\widetilde{\boldsymbol{\mu}}^d_{l,\rho_{\le d-1}||n}, \widetilde{\boldsymbol{\nu}}^d_{l,\rho_{\le d-1}||n}\}_{d,n}$, as described in Figure 8.

EVERY CIPHERTEXT CT_l IMPLICITLY ENCODES AN **IPE** CIPHERTEXT iCT_l. When the $\widetilde{\mathbf{X}}$ vectors are set as such, the set of high-degree inner products that can be computed from the **dIPE** ciphertexts and secret keys satisfy the following. Recall that for different l, each ciphertext CT_l samples its own **dIPE** master secret key; therefore, inner products can be only be computed among $\widetilde{\mathbf{X}}$ vectors with the same index l. Consider a fixed l. For every combination $I \in [N]^D$, if the the longest prefix that I share with ρ is $\rho_{\le d-1}$, that is, $I = \rho_{\le d-1}||I_{\ge d}$, then,

$$\left[\left\langle\, \widetilde{\mathbf{X}}^1_{l,I_1}, \cdots, \widetilde{\mathbf{X}}^D_{l,I_D} \,\right\rangle\right]_{D+1} = \left[r_l\, w^d_{\rho_{\le d-1}||I_d} s^{d+1}_{I_{d+1}} \cdots s^D_{I_D} + \begin{cases} v^{\le D}_{l,I} & \text{if } d < D \text{ and } I_d < \rho_d \\ u^{\le D}_{l,I} & \text{if } d < D \text{ and } I_d > \rho_d \\ u^{\le D}_{l,I} & \text{if } d = D \text{ and in } H^0_\rho \\ v^{\le D}_{l,I} & \text{if } d = D \text{ and in } H^1_\rho \end{cases} \right]_{D+1}$$

Define $V^b_l(\rho)$ to be the length-N^D vector where its I^{th} element is exactly the second term in addition, and $S(\rho)$ the length-N^D vector where $(S(\rho))_I$ is the first term, that is,

$$\forall \rho \in [N]^{D-1},$$
$$(S(\rho))_I = w^d_{\rho_{\le d-1}||I_d} s^{d+1}_{I_{d+1}} \cdots s^D_{I_D} \quad \text{if the longest prefix } I \text{ share with } \rho \text{ is } \rho_{\le d-1}\,. \tag{13}$$

(Recall that $w^1_n = s^1_n$, and $w^d_{\rho_{\le d-1}||n}$ for $d > 1$ is a random element in $\mathcal{R}$.) Then, we can rewrite the above encoding as

$$\left[\left\langle\, \widetilde{\mathbf{X}}^1_{l,I_1}, \cdots, \widetilde{\mathbf{X}}^D_{l,I_D} \,\right\rangle\right]_{D+1} = \Big[\quad r_l\,(S(\rho))_I \quad + \quad (V^b_l(\rho))_I \quad \Big]_{D+1} = \mathsf{iCT}_{l,I} \tag{14}$$

which is exactly the I^{th} encoding in an **IPE** ciphertext iCT_l of vector $V_l^b(\rho)$ with master secret key $S(\rho)$, and random element r_l. Notably, in H_ρ^0, the encrypted vector $V_l^0(\rho)$ satisfy that every element indexed by I with a prefix $< \rho$ is the I^{th} monomial of $\mathbf{v}_l$, and every element I with a prefix $\geq \rho$ is the I^{th} monomial of $\mathbf{u}_l$. In H_ρ^1, $V_l^1(\rho)$ is the same as $V_l^0(\rho)$ except that elements indexed with exactly prefix ρ are switched from monomials of $\mathbf{u}_l$ to monomials of $\mathbf{v}_l$.

Fact 4. *By the way vectors $\mathbf{u}_l$ and $\mathbf{v}_l$ are set in Equation (12). For every $l \neq \ell$, $\mathbf{u}_l = \mathbf{v}_l$. Only the vector $V_\ell^b(\rho)$ encrypted in CT_ℓ changes from H_ρ^0 to H_ρ^1; more specifically, only the values of N elements indexed with prefix ρ change. Furthermore, observe that $V_\ell^1(\rho) = V_\ell^0(\rho+1)$.*

<u>SET THE VECTOR ENCODED IN tSK_l.</u> For every secret key $\mathsf{SK}_j = (\mathsf{tSK}_j, \mathbf{c}_j)$ with $j \in [L]$, we set the vector encoded in tSK_j in the following way

$$\mathsf{tSK}_j \xleftarrow{\$} \mathsf{tIPE.KeyGen}\left(\mathsf{tMSK},\ \left(\langle S(\rho)\ ,\ \mathbf{c}_j\rangle \,\middle|\middle|\, - r_\ell \langle S(\rho)\ ,\ \mathbf{c}_j\rangle + \Delta_j^b(\rho)\right)\right)\ ,$$

where the values of $\{\Delta_j^b(\rho)\}$ are specified below. Roughly speaking, the first encoded element $\langle S(\rho)\ ,\ \mathbf{c}_j\rangle$ is the first element in an **IPE** secret key $\mathsf{iSK} = (\langle S(\rho)\ ,\ \mathbf{c}_j\rangle, \mathbf{c}_j)$ for vector $\mathbf{c}_j$ under master secret key $S(\rho)$. The second encoded element $-r_\ell \langle S(\rho)\ ,\ \mathbf{c}_j\rangle + \Delta_j^b(\rho)$ is there specially for the ℓ^{th} ciphertext CT_ℓ, whose **tIPE** ciphertext tCT_ℓ encodes the unique vector $(0||1)$ (whereas all other ciphertexts CT_l contain **tIPE** ciphertexts tCT_l of $(-r_l||0)$.)

To see why they are generated as such, it is best to run through the process of decrypting a ciphertext CT_l with SK_j. The first step decrypts tCT_l with tSK_j to obtain,

$$\Lambda_1 = \mathsf{tIPE.Dec}(\mathsf{tSK}_j, \mathsf{tCT}_l) = \begin{cases} [-r_l \langle S(\rho)\ ,\ \mathbf{c}_j\rangle]_{D+1} & \text{if } l \neq \ell \\ \left[-r_\ell \langle S(\rho)\ ,\ \mathbf{c}_j\rangle + \Delta_j^b(\rho)\ \right]_{D+1} & \text{if } l = \ell \end{cases}$$

The next step, decrypts the **dIPE** ciphertexts and secret keys contained in CT_l to obtain all encodings $\{\mathsf{iCT}_I\}_I$ as described above in Equation (14), it then homomorphically computes the inner product of $\{\mathsf{iCT}_I\}_I$ and $\mathbf{c}_j$ to obtain,

$$\Lambda_2 = \langle \{\mathsf{iCT}_I\}_I, \mathbf{c}_j\rangle = \left[\left\langle r_l\ S(\rho) + V_l^b(\rho)\ ,\ \mathbf{c}_j\right\rangle\right]_{D+1}$$

In the last step, it computes

$$[y_{l,j}]_{D+1} = \Lambda_1 + \Lambda_2 = \begin{cases} \left[\langle V_l^b(\rho)\ ,\ \mathbf{c}_j\rangle\right]_{D+1} & \text{if } l \neq \ell \\ \left[\langle V_\ell^b(\rho)\ ,\ \mathbf{c}_j\rangle + \Delta_j^b(\rho)\right]_{D+1} & \text{if } l = \ell \end{cases}$$

Note that, for all hybrids $\{H_\rho^b\}$ to be indistinguishable to each other, the decryption outputs $\{y_{l,j}\}_{l,j}$ in them must be identical. As observed in Fact 4, for all $l \neq \ell$, $V_l^b(\rho)$ stay the same in all H_ρ^b and thus so are the outputs $\{y_{l,j}\}_{l \neq \ell, j}$. But, $V_\ell^b(\rho)$ changes with ρ and hence the additional term $\Delta_j^b(\rho)$ is needed to ensure that the outputs $\{y_{\ell,j}\}_j$ stay the same. Since V_ρ^0 and V_ρ^1 differ only at the N indexes with prefix ρ, when switching from V_ρ^0 to V_ρ^1, the inner product changes by

$$\delta_j(\rho) = \sum_{\substack{I=\rho||n \\ n \in [N]}} \mathbf{c}_{j,I}\left(u_{\ell,I}^{\leq D} - v_{\ell,I}^{\leq D}\right)\ .$$

By setting $\Delta_j^b(\rho)$ to the cumulative sum of $\delta_j(\rho')$ with $\rho' < \rho$ if $b = 0$, and the sum of $\delta_j(\rho')$ with $\rho' \leq \rho$ if $b = 1$,

$$\Delta_j^0(\rho) = \sum_{\rho' < \rho} \delta_j(\rho) \quad \text{and} \quad \Delta_j^1(\rho) = \sum_{\rho' \leq \rho} \delta_j(\rho) \; ,$$

we ensure that the outputs $\{y_{\ell,j}\}_j$ of decrypting CT_ℓ with every SK_j are identical in all hybrids H_ρ^b.

INDISTINGUISHABILITY BETWEEN H_ρ^0 AND H_ρ^1. We show that for every $\rho \in [N]^{D-1}$, moving from H_ρ^0 to H_ρ^1 is indistinguishable.

Lemma 14. *For every $\rho \in [N]^{D-1}$, the ensembles $\{H_\rho^0(\lambda)\}_\lambda$ and $\{H_\rho^1(\lambda)\}_\lambda$ are indistinguishable.*

At a very high-level, the only difference between these two hybrids lies in the values of the following elements:

- the elements in slot D of vectors $\{\widetilde{\mathbf{X}}_{\ell,n}^D\}_n$ encoded in $\{\mathsf{dSK}_{\ell,n}\}_n$ (note that for all other $l \neq \ell$, $\widetilde{\mathbf{X}}_{l,n}^D$ stays the same because $\mathbf{u}_l = \mathbf{v}_l$), and
- the second elements of vectors $\{\mathbf{t}_j\}_j$ encoded in $\{\mathsf{tSK}_j\}_j$.

By the fact that **dIPE** and **tIPE** are canonical, these vectors $\widetilde{\mathbf{X}}^D$'s and $\mathbf{t}$'s are encoded in group G_D, and H_ρ^0 and H_ρ^1 can essentially be emulated from encodings of these elements in G_D. Furthermore, these encodings correspond to an **IPE** encryption of either vector $\mathbf{x}_0' = \{u_{\ell,\rho||n}^{\leq D}\}_n||\{\Delta_j^0(\rho)\}_j$ or $\mathbf{x}_1' = \{v_{\ell,\rho||n}^{\leq D}\}_n||\{\Delta_j^1(\rho)\}_j$ (of length $N+1$), under a truly random master secret key $\mathbf{s}' = \{w_{\rho||n}^D\}_n||\{t_j\}_j$ (for random t_j's), at the presence of a set of **IPE** secret keys for vectors $\mathbf{y}_j' = \{c_{j,\rho||n}\}_n||\mathbf{e}_j$ for every $j \in [L]$ ($\mathbf{e}_j$ is the unit vector of length L with a single 1 at index j). The way that the values of $\{\Delta_j^b(\rho)\}_{b,j}$ are set ensures that the inner product of $\mathbf{x}_0'$ and $\mathbf{y}_j'$ is identical to that of $\mathbf{x}_1'$ and $\mathbf{y}_j'$. Therefore, it follows from the IND-security of **IPE** that the two hybrids are indistinguishable. A formal proof can be found later.

INDISTINGUISHABILITY BETWEEN H_ρ^1 AND $H_{\rho+1}^0$. We also show that moving from H_ρ^1 to $H_{\rho+1}^1$ are indistinguishable.

Lemma 15. *For every $\rho \in [N]^{D-1} \setminus \{n^N\}$, the ensembles $\{H_\rho^1(\lambda)\}_\lambda$ and $\{H_{\rho+1}^0(\lambda)\}_\lambda$ are indistinguishable, where $\rho + 1$ denote the member in $[N]^{D-1}$ following immediately after ρ, in increasing numerical order.*

At a high-level, the difference between H_ρ^1 and $H_{\rho+1}^0$ lies in the values of $\{\widetilde{\mathbf{X}}_{l,n}^d\}_{d,n,l}$, as well as that of $\{\mathbf{t}_j\}$ encoded in $\{\mathsf{tSK}_j\}$. Note that the way that vectors $\widetilde{\mathbf{X}}_{l,n}^d$ are set in different hybrids H_ρ^b (as well as how vectors $\widetilde{\boldsymbol{\mu}}$'s and $\widetilde{\boldsymbol{\nu}}$'s are derived from $\boldsymbol{\mu}$'s and $\boldsymbol{\nu}$'s) are identical to that in the security proof of the degree-$(D+1)$ HIPE scheme in Section 7.3 (Proof of Proposition 2). There, we already showed in the proof of Lemma 7 that changing the values of $\widetilde{\mathbf{X}}$'s from H_ρ^1 to $H_{\rho+1}^0$ is indistinguishable, relying on the security of **dIPE** and the SXDH assumption on MMaps. Here, the proof is almost the same, except that besides from the difference in the values of $\widetilde{\mathbf{X}}$'s, the values of $\mathbf{t}$'s also change, from being generated using $S(\rho)$ in H_ρ^1 to using $S(\rho+1)$ in $H^0_{\rho+1}$. We observe that $S(\rho)$ and $S(\rho+1)$ can be uniformly derived from the random elements embedded in $\widetilde{\mathbf{X}}$'s in H^0_ρ and $H^1_{\rho+1}$ (see equation 13). Thus, it suffices to slightly modify the proof of Lemma 7, to make sure that whenever the SXDH assumption is applied to change the random elements in $\widetilde{\mathbf{X}}$, we change the value of S correspondingly. With the modified proof, we can show that hybrids H^1_ρ and $H^0_{\rho+1}$ are indistinguishable.

Furthermore, it follows from similar proof that **Init** and $H^0_{1^{D-1}}$ are also indistinguishable.

Lemma 16. *The ensembles* $\{\mathbf{Init}(\lambda)\}_\lambda$ *and* $\{H^0_{1^{D-1}}(\lambda)\}_\lambda$ *are indistinguishable.*

Hybrid Mid$'(\lambda)$ is generated identically as **Init** except that every ciphertext CT_l encode a set of vectors $\{\bar{\mathbf{X}}^d_{l,n}\}_l$ different from that in **Init**, where instead of having vectors $\boldsymbol{\mu}$'s in the first half of slot 1, we have vectors $\boldsymbol{\nu}$'s in the second half of slot 1 (and zeros elsewhere). See figure 9 for a precise description.

Hybrid Distribution Mid$'(\lambda)$

Generate the following:

- $\mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ and parse $\mathsf{msk} = (\mathsf{sMSK}, \{\mathsf{dMSK}_l\}_{l\in[L]})$ and $\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2)$.
- For every $l \in [L]$, sample $r_l \xleftarrow{\$} \mathcal{R}$ and $\mathsf{dMSK}_l \xleftarrow{\$} \mathsf{dIPE.Setup}(1^\lambda, \mathsf{pp})$.
  Generate $\mathsf{CT}_l = (\mathsf{tCT}_l, \{\mathsf{dCT}^d_{l,n}, \mathsf{dSK}_{l,n}\}_{d,n})$ as follows:

$$\mathsf{tCT}_l \xleftarrow{\$} \mathsf{tIPE.Enc}\left(\mathsf{tMSK}, \begin{cases} (-r_l||0) & \text{if } l \neq \ell \\ (0||1) & \text{if } l = \ell \end{cases}\right) \quad \begin{array}{l} \left\{\mathsf{dCT}^d_{l,n} \xleftarrow{\$} \mathsf{dIPE.Enc}(\mathsf{dMSK}_l, \underline{\bar{\mathbf{X}}^d_{l,n}})\right\}_{d<D, n\in[N]} \\ \left\{\mathsf{dSK}_{l,n} \xleftarrow{\$} \mathsf{dIPE.KeyGen}(\mathsf{dMSK}_l, \underline{\bar{\mathbf{X}}^D_{l,n}})\right\}_{n\in[N]} \end{array}$$

  where the vectors $\{\bar{\mathbf{X}}^d_{l,n}\}_{d\in[D],n\in[N]}$ are set as follows

$$\bar{\mathbf{X}}^d_{\gamma,l} = \underbrace{\mathbf{0}||\boldsymbol{\nu}^d_{\gamma,l}}_{\text{slot }1} \quad \underbrace{\mathbf{0}||\mathbf{0}}_{\text{slot }2} \quad \underset{\cdots}{\cdots} \quad \underbrace{\mathbf{0}||\mathbf{0},}_{\text{slot }D-1} \quad \underbrace{0}_{\text{slot }D}$$

- For every $j \in [L]$, generate $\mathsf{SK}_j = (\mathbf{c}_j, \mathsf{tSK}_j)$ for coefficient vector $\mathbf{c}_j$ with

$$\mathsf{tSK}_j \xleftarrow{\$} \mathsf{tIPE.KeyGen}\left(\mathsf{tMSK}, \underline{\langle \otimes \mathbf{s}^{\leq D}, \mathbf{c}_j\rangle \,||\, - r_\ell \langle \otimes \mathbf{s}^{\leq D}, \mathbf{c}_j\rangle}\right) \,.$$

Output $\{\mathsf{SK}_l,\ \mathsf{CT}_l\}_{l\in[L]}$

Figure 9: Hybrid **Mid**$'(\lambda)$ for proving security of $\mathbf{FE}^{D,N}$

We show that moving from the last hybrid $H^1_{N^{D-1}}$ to **Mid**$'$ is indistinguishable.

Lemma 17. *The ensembles* $\{H^1_{N^{D-1}}(\lambda)\}_\lambda$ *and* $\{\mathbf{Mid}'(\lambda)\}_\lambda$ *are indistinguishable.*

A fomal proof of this lemma, similar to that of Lemma 15, is provided below.

Hybrid Mid(λ) is generated identically to **Mid**$'$ except that every secret key tSK_j encodes vector $(\langle \otimes \mathbf{s}^{\leq D}, \mathbf{c}_l\rangle\, ||0)$ as opposed to $(\langle \otimes \mathbf{s}^{\leq D}, \mathbf{c}_l\rangle \,||\, - r_\ell \langle \otimes \mathbf{s}^{\leq D}, \mathbf{c}_l\rangle)$, and the ℓ^{th} ciphertext CT_ℓ contains tCT_ℓ encrypting vector $(-r_\ell||0)$ as opposed to $(0||1)$.

As in Lemma 13, it follows directly from the function hiding of **tIPE** that it is indistinguishable to move from $\mathbf{Mid}'$ to $\mathbf{Mid}$.

Lemma 18. *The ensembles* $\{\mathbf{Mid}'(\lambda)\}_\lambda$ *and* $\{\mathbf{Mid}(\lambda)\}_\lambda$ *are indistinguishable.*

By a hybrid argument, we have that the honest distribution Hyb_ℓ is indistinguishable to the middle hybrid distribution $\mathbf{Mid}$. It follows syntactically from the same proof (replacing the input vectors $\mathbf{u}$'s with vectors $\mathbf{v}$'s) that $\mathsf{Hyb}_{\ell+1}$ is also indistinguishable to $\mathbf{Mid}$. Therefore, Hyb_ℓ and $\mathsf{Hyb}_{\ell+1}$ are indistinguishable, which concludes that $\mathcal{D}_0$ and $\mathcal{D}_1$ are indistinguishable and hence $\mathbf{FE}^{D,N}$ is selectively IND-secure. $\square$

8.2.1 Proof of Lemma 14

Proof of Lemma 14. The only difference between these hybrids H^0_ρ and H^1_ρ lies in the elements in slot D of vectors $\{\widetilde{\mathbf{X}}^D_{\ell,n}\}_{\ell,n}$ encoded in $\{\mathsf{dSK}_{\ell,n}\}_{n\in[N]}$ (Note that for all other $l \neq \ell$, $\widetilde{\mathbf{X}}^D_{l,n}$ stays the same because $\mathbf{u}_l = \mathbf{v}_l$), and the second elements of vectors $\{\mathbf{t}_j\}_j$ encoded in $\{\mathsf{tSK}_j\}_{j\in[N]}$.

By definition of the vectors $\widetilde{\boldsymbol{\mu}}$'s and $\widetilde{\boldsymbol{\nu}}$'s, the elements in slot D of $\widetilde{\mathbf{X}}^D_{\ell,n}$ satisfy the following.

$$\text{in } H^0_\rho, \qquad \left\langle \widetilde{\boldsymbol{\mu}}^D_{\ell,\rho||n},\ \mathbf{1} \right\rangle = \left\langle\ u^{\leq D}_{\ell,\rho||n} \ \middle|\middle|\ r_\ell w^D_{\rho||n},\ \right\rangle = r_\ell w^D_{\rho||n} + u^{\leq D}_{\ell,\rho||n}$$

$$\text{in } H^1_\rho, \qquad \left\langle \widetilde{\boldsymbol{\nu}}^D_{\ell,\rho||n},\ \mathbf{1} \right\rangle = \left\langle v^{\leq D}_{\ell,\rho||n} \ \middle|\middle|\ r_\ell w^D_{\rho||n},\ \right\rangle = r_\ell w^D_{\rho||n} + v^{\leq D}_{\ell,\rho||n}$$

The second element of the vector $\mathbf{t}_j$ encoded in tSK_j in H^b_ρ is

$$\begin{aligned} -r_\ell \langle S(\rho)\ ,\ \mathbf{c}_j\rangle + \Delta^b_j(\rho) =\ & \big(-r_\ell \big\langle\ \{S(\rho)_I\}_{I_{\leq D-1}\neq\rho}\ ,\ \{c_{j,I}\}_{I_{\leq D-1}\neq\rho}\big\rangle\big) \\ & +\ \Big(-r_\ell \big\langle\ \{S(\rho)_I\}_{I_{\leq D-1}=\rho}\ ,\ \{c_{j,I}\}_{I_{\leq D-1}=\rho}\big\rangle\ +\ \Delta^b_j(\rho)\Big) \\ =\ & \text{term 1}\ +\ \Big(-r_\ell \Big\langle\ \{w^D_{\rho||n}\}_{n\in[N]}\ ,\ \{c_{j,\rho||n}\}_{n\in[N]}\Big\rangle\ +\ \Delta^b_j(\rho)\Big)\ , \end{aligned}$$

where variables are sampled as in H^b_ρ. We claim that the hybrid H^b_ρ can be emulated from the following distribution $\widetilde{\mathcal{D}}^b_\rho$

$$\begin{aligned} \widetilde{\mathcal{D}}^0_\rho = \Big\{\ & \Big\{\Big[w^D_{\rho||n}\Big]_D\Big\}_n\ ,\Big\{\Big[-r_\ell\Big\langle\ \{w^D_{\rho||n}\}_{n\in[N]}\ ,\ \{c_{\rho||n}\}_{n\in[N]}\Big\rangle\ +\ \underline{\Delta^0_j(\rho)}\Big]_D\Big\}_{j\in[L]}\ , \\ & \Big\{\Big[r_\ell w^D_{\rho||n} + \underline{u^{\leq D}_{\ell,\rho||n}}\Big]_D\Big\}_n\ \Big\} \\ \widetilde{\mathcal{D}}^1_\rho = \Big\{\ & \Big\{\Big[w^D_{\rho||n}\Big]_D\Big\}_n\ ,\Big\{\Big[-r_\ell\Big\langle\ \{w^D_{\rho||n}\}_{n\in[N]}\ ,\ \Big\{c_{j,\rho||n}\Big\}_{n\in[N]}\Big\rangle\ +\ \underline{\Delta^1_j(\rho)}\Big]_D\Big\}_{j\in[L]}\ , \\ & \Big\{\Big[r_\ell w^D_{\rho||n} + \underline{v^{\leq D}_{\ell,\rho||n}}\Big]_D\Big\}_n\ \Big\} \end{aligned}$$

To see this, observe that

- Every secret key SK_j can be emulated as follows: From the first two terms in the distribution, one can emulate encodings of vectors $\mathbf{t}_j$ in group G_D, with knowledge of $\mathbf{c}_j$, vectors $\{\mathbf{s}^d\}_{d\in[D]}$, and all the w random elements $\{w^D_{\rho_{\leq d}||n}\}_{d<D-1,n\in[N]}$ that do not appear in the above distribution (*i.e.*, except the ones with indexes $\rho||n$). Since **tIPE** is canonical, its secret keys tSK_j consists of encodings in group G_D of values linear in the encoded vector $\mathbf{t}_j$. Therefore, one can emulate tSK_j from encodings of $\mathbf{t}_j$, with knowledge of tMSK (relying on the linear homomorphism of group G_D).

- The ciphertext CT_ℓ can be emulated as follows: From the last term in the distribution, one can emulate encodings of $\{\widetilde{\mathbf{X}}^D_{\ell,n}\}_n$ in G_D, with knowledge of $\mathbf{u}_\ell, \mathbf{v}_\ell$ and $\{\mathbf{s}^d\}_{d\in[D]}$. Since **tIPE** is canonical, from these encodings, one can emulate $\{\mathsf{dSK}_{\ell,n}\}_n$ with knowledge of dMSK_ℓ. Moreover, since vectors $\{\widetilde{\mathbf{X}}^d_{\ell,n}\}_{d<D,n}$ at other coordinates $d < D$ depend only on $\mathbf{u}_\ell, \mathbf{v}_\ell$, $\{\mathbf{s}^d\}_{d\in[D]}$ and the w random elements $\{w^D_{\rho_{\leq d}||n}\}_{d<D-1,n\in[N]}$ that do not appear in the above distribution, one can generate $\{\mathsf{dCT}^d_{\ell,n}\}_{d<D,n}$ honestly. Similarly, one can generate tCT_ℓ encrypting $(0||1)$ honestly.
- The ciphertext CT_l for $l \neq \ell$ can be emulated as follows: From the first term in the distribution, one can emulate encodings of $\{\widetilde{\mathbf{X}}^D_{l,n}\}_n$ in G_D, with knowledge of $\mathbf{u}_l = \mathbf{v}_l$ and $\{\mathbf{s}^d\}_{d\in[D]}$, and r_l. Then, like above, from these encodings, one can emulate $\{\mathsf{dSK}_{l,n}\}_n$ knowing dMSK_l, and can generate $\{\mathsf{dCT}^d_{l,n}\}_{d<D,n}$ and tCT_l encrypting $(-r_l||0)$ honestly.

Thus, it remains to show that distribution $\widetilde{\mathcal{D}}^0_\rho$ and $\widetilde{\mathcal{D}}^1_\rho$ are indistinguishable. To show this, we argue that $\widetilde{\mathcal{D}}^b_\rho$ can be generated from the following distributions.

$$\bar{\mathcal{D}}^b_\rho = \left\{ \mathsf{pk}' = [\mathbf{s}']_D,\ \{\mathsf{iSK}'_j = (\langle\, \mathbf{s}'\ ,\ \mathbf{y}'_j \rangle\ ,\ \mathbf{y}'_j)\}_{j\in[L]}\ ,\ \mathsf{iCT}' = ([-r_\ell]_D,\ [r_\ell \mathbf{s}' + \mathbf{x}'_b]_D) \right\}, \text{ where}$$

$$\mathbf{s}' = \left(\left\{w^D_{\rho||n}\right\}_n \Big|\Big| \left\{t_j\right\}_{j\in[L]}\right), \quad \mathbf{y}'_j = \left(\left\{c_{j,\rho||n}\right\}_n \Big|\Big| \mathbf{e}_j\right), \quad \mathbf{x}'_b = \begin{cases} \left\{u^{\leq D}_{\ell,\rho||n}\right\}_n \Big|\Big| \left\{\Delta^0_j(\rho)\right\}_{j\in[N]} & \text{if } b = 0 \\ \left\{v^{\leq D}_{\ell,\rho||n}\right\}_n \Big|\Big| \left\{\Delta^1_j(\rho)\right\}_{j\in[N]} & \text{if } b = 1 \end{cases},$$

$t_j \xleftarrow{\$} \mathcal{R}$, and $\mathbf{e}_j$ is the unit vector of length L with a single 1 at index j.

In particular, the first term in $\widetilde{\mathcal{D}}^b_\rho$ can be derived from pk', the second term can be derived by computing for every $j \in [L]$,

$$\begin{aligned}\langle \mathbf{s}', \mathbf{y}'_j\rangle [-r_\ell]_D + \left[r_\ell t_j + \Delta^b_j(\rho)\right]_D &= \left[-r_\ell \left\langle \left\{w^D_{\rho||n}\right\}_n, \left\{c_{j,\rho||n}\right\}_n \right\rangle - r_\ell t_j\right]_D + \left[r_\ell t_j + \Delta^b_j(\rho)\right]_D \\ &= \left[-r_\ell \left\langle \left\{w^D_{\rho||n}\right\}_n, \left\{c_{j,\rho||n}\right\}_n \right\rangle + \Delta^b_j(\rho)\right]_D\end{aligned}$$

where the second operand in the left hand side is the $N + j + 1^{\text{th}}$ encoding of iCT', and finally the last term can be derived from iCT'.

Since for every $j \in [L]$, $\left\langle \mathbf{y}'_j, \mathbf{x}'_0 \right\rangle = \left\langle \mathbf{y}'_j, \mathbf{x}'_1 \right\rangle$, it follows directly from the IND-security of **IPE** that $\bar{\mathcal{D}}^0_\rho$ and $\bar{\mathcal{D}}^1_\rho$ are indistinguishable. Therefore, so are $\widetilde{\mathcal{D}}^0_\rho$ and $\widetilde{\mathcal{D}}^1_\rho$, which concludes the indistinguishability of H^0_ρ and H^1_ρ. □

8.2.2 Proof of Lemma 15 to 17

The proof for Lemma 15, 16, and 17, is almost identical to the proofs of Lemma 7, 8 and 9, up to one modification: In every pair of hybrids we want to prove indistinguishability of changing the values of vectors $\mathbf{X}$'s, and the values of $\mathbf{t}$'s encoded in tSK's. As discussed above, the changes in $\mathbf{t}$ is "induced" by the changes in the random elements (namely, the s's and w's) used in the $\mathbf{X}$ vectors, thus, we simply need to ensure that whenever we apply the SXDH assumption to switch the random elements, $\mathbf{X}$'s and $\mathbf{t}$'s are changed together accordingly. There is only one technicality that we need to be careful with. That is, the $\mathbf{X}$ vectors encrypted and encoded in **dIPE** ciphertexts and secret keys are encoded in different groups $\{G_d\}_d$, whereas $\mathbf{t}$'s are encoded using **tIPE** and always in group G_D. Suppose we want to switch, say one random element w to a product sw' of two random elements, in some vectors $\mathbf{X}^d_{l,n}$ and $\mathbf{t}_j$ simultaneously. The SXDH assumption does not hold in G_d and G_D simultaneously since the two groups can be paired together. To resolve this issue, we will add an additional hybrid step, to first *swap* all vectors encrypted at coordinate d with vectors encrypted at coordinate D in the l^{th} ciphertext CT_l, that is, we encrypt vectors $\{\mathbf{X}^d_{l,n}\}_n$ at coordinate D, and $\{\mathbf{X}^D_{l,n}\}_n$ at coordinate d. By the function hiding property of **dIPE**, "swapping coordinates" is indistinguishable since all inner products stay the same. Now, both $\mathbf{X}^d_{l,n}$ and $\mathbf{t}_j$ are encoded in group G_D, and the SXDH assumption can be applied to switch random element w to $w's$ in them simultaneously. Finally, "swap" coordinate D with d again to arrive at the distribution wanted.

Since most parts of the proofs are identical to that in Section 7.3.4, below we sketch the proofs, emphasizing on the modification.

As in Section 7.3.4, we construct additional hybrid distributions G^b_ρ for prefixes $\rho \in [N]^{d^\star}$ of any length $d^\star$ from 1 to $D-1$, and show the following lemma, from which Lemma 15, 16, and 17 follow, using the same arguments as in Section 7.3.4 (*i.e.*, proof of Lemma 7, 8, and 9).

Lemma 19. *There exist hybrids $\{G^b_\rho(\lambda)\}$ for $b \in \{0,1\}$ and $\rho \in [N]^{d^\star}$ where $d^\star \in [D-1]$, such that, the following holds.*

Rule 1: *Ensembles $\{G^0_1(\lambda)\}$ and $\{\mathbf{Init}(\lambda)\}$ are indistinguishable.*

Rule 2: *Ensembles $\{G^1_N(\lambda)\}$ and $\{\mathbf{Mid}'(\lambda)\}$ are indistinguishable*

Rule 3: *For every $\rho \in [N]^{d^\star}$ with $1 \le d^\star < D-1$, ensembles $\{G^0_{\rho||1}(\lambda)\}$ and $\{G^0_\rho(\lambda)\}$, are indistinguishable.*

Rule 4: *For every $\rho \in [N]^{d^\star}$ with $1 \le d^\star < D-1$, ensembles $\{G^1_{\rho||N}(\lambda)\}$ and $\{G^1_\rho(\lambda)\}$ are indistinguishable.*

Rule 5: *For every $\rho \in [N]^{D-1}$ and every b, ensembles $\{G^b_\rho(\lambda)\}$ and $\{H^b_\rho(\lambda)\}$ are indistinguishable.*

Rule 6: *For every $\rho \in [N]^{d^\star}$ with $1 \le d^\star \le D-1$, such that, $\rho_{d^\star} \ne N$, ensembles $\{G^1_\rho(\lambda)\}$ and $\{G^0_{\rho+1}(\lambda)\}$ are identical.*

Proof. We first formally describe the G hybrid distributions.

Hybrid $G^b_\rho(\lambda)$: For $1 \le d^\star \le D-1$ and $\rho \in [N]^{d^\star}$, distribution $G^b_\rho(\lambda)$ is identical to the initial hybrid distribution **Init**, except that the ciphertexts $\{\mathsf{CT}_l\}$ encode vectors $\{\hat{\mathbf{X}}^d_{l,n}\}$, and the secret keys $\{\mathsf{tSK}_j\}$ encode vectors $\{\mathbf{t}_j\}$ different from that in **Init**. The values of $\hat{\mathbf{X}}^d_{l,n}$ are described in Figure 6, and vectors $\mathbf{t}_j$'s depend on $\hat{S}(\rho)$ defined below

$$\forall \rho \in [N]^{d^\star},\ d^\star \in [D-1],\ I \in [N]^D, \qquad \text{let } \rho_{\le d-1} \text{ be the longest prefix that } I \text{ shares with } \rho$$

$$(\hat{S}(\rho))_I = \begin{cases} w^{d^\star}_\rho s^{d^\star+1}_{I_{d^\star+1}} \cdots s^D_{I_D} & \text{if } d-1 = d^\star \\ w^d_{\rho_{\le d-1}||I_d} s^{d+1}_{I_{d+1}} \cdots s^D_{I_D} & \text{if } d-1 < d^\star\ . \end{cases} \tag{15}$$

Note that the vector $\hat{S}$ is defined similarly as vector S in H^b_ρ, except that it handles prefixes of any length no longer than $D-1$, as opposed to length exactly $D-1$. In addition, the value of $(\hat{S}(\rho))_I$ for indexes that contain prefix ρ entirely is differently: $(S(\rho))_I = w^D_{\rho||I_D}$, whereas $(\hat{S}(\rho))_I = w^{D-1}_\rho s^D_{I_D}$ as defined in the first case above. (This matches the way how $\bar{\mathbf{X}}^{d^\star}_{l,n}$ are set.)

Hybrid distribution $G^b_\rho(\lambda)$ for $1 \le d^\star \le D-1$ and $\rho \in [N]^{d^\star}$

Generate the following:

- $\mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, \mathsf{pp})$ and parse $\mathsf{msk} = (\mathsf{sMSK}, \{\mathsf{dMSK}_l\}_{l\in[L]})$ and $\mathsf{sMSK} = (\mathbf{k}_1, \mathbf{k}_2)$.
- For every $l \in [L]$, sample $r_l \xleftarrow{\$} \mathcal{R}$ and $\mathsf{dMSK}_l \xleftarrow{\$} \mathsf{dIPE.Setup}(1^\lambda, \mathsf{pp})$.
  Generate $\mathsf{CT}_l = (\mathsf{tCT}_l, \{\mathsf{dCT}^d_{l,n}, \mathsf{dSK}_{l,n}\}_{d,n})$ as follows:

$$\mathsf{tCT}_l \xleftarrow{\$} \mathsf{tIPE.Enc}\left(\mathsf{tMSK}, \begin{cases}(-r_l\|0) & \text{if } l \neq \ell\\ (0\|1) & \text{if } l = \ell\end{cases}\right) \qquad \begin{matrix}\left\{\mathsf{dCT}^d_{l,n} \xleftarrow{\$} \mathsf{dIPE.Enc}(\mathsf{dMSK}_l, \underline{\hat{\mathbf{X}}^d_{l,n}})\right\}_{d<D, n\in[N]}\\ \left\{\mathsf{dSK}_{l,n} \xleftarrow{\$} \mathsf{dIPE.KeyGen}(\mathsf{dMSK}_l, \underline{\hat{\mathbf{X}}^D_{l,n}})\right\}_{n\in[N]}\end{matrix}$$

  where the vectors $\{\hat{\mathbf{X}}^d_{l,n}\}_{d\in[D],n\in[N]}$ are set as follows
  Case 1: $d > d^\star$.

$$\hat{\mathbf{X}}^d_{l,n} = \underbrace{\boldsymbol{\mu}^d_{l,n}\|\boldsymbol{\nu}^d_{l,n}}_{\text{slot } 1} \quad \underbrace{\boldsymbol{\mu}^d_{l,n}\|\boldsymbol{\nu}^d_{l,n}}_{\text{slot } 2} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^d_{l,n}\|\boldsymbol{\nu}^d_{l,n}}_{\text{slot } d^\star} \quad \underbrace{\underline{\mathbf{0}}}_{\text{slot } > d^\star}$$

  Case 2: $d = d^\star$

$$\hat{\mathbf{X}}^{d^\star}_{l,n} = \underbrace{\boldsymbol{\mu}^{d^\star}_{l,n}\|\boldsymbol{\nu}^{d^\star}_{l,n}}_{\text{slot } 1} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^{d^\star}_{l,n}\|\boldsymbol{\nu}^{d^\star}_{l,n}}_{\text{slot } d^\star-1} \quad \underbrace{\begin{cases}\mathbf{0} \,\|\, \widetilde{\boldsymbol{\nu}}^{d^\star}_{l,\rho_{\le d^\star-1}\|n} & \text{if } n < \rho_{d^\star}\\ \widetilde{\boldsymbol{\mu}}^{d^\star}_{l,\rho_{\le d^\star-1}\|n} \,\|\, \mathbf{0} & \text{if } n > \rho_{d^\star}\\ \underline{\mathbf{0} \;\|\; \widetilde{\boldsymbol{\nu}}^{d^\star}_{l,\rho}} & \underline{\text{if } n = \rho_{d^\star} \text{ and in } G^1_\rho}\\ \underline{\widetilde{\boldsymbol{\mu}}^{d^\star}_{l,\rho} \;\|\; \mathbf{0}} & \underline{\text{if } n = \rho_{d^\star} \text{ and in } G^0_\rho}\end{cases}}_{\text{slot } d^\star} \quad \underbrace{\underline{\mathbf{0}}}_{\text{slot } > d^\star}$$

  Case 3: $d < d^\star$. (In this case $\hat{\mathbf{X}}^d_{l,n} = \widetilde{\mathbf{X}}^d_{l,n}$ in hybrid H^b_ρ.)

$$\hat{\mathbf{X}}^d_{l,n} = \underbrace{\boldsymbol{\mu}^d_{l,n}\|\boldsymbol{\nu}^d_{l,n}}_{\text{slot } 1} \quad \underset{\cdots}{\cdots} \quad \underbrace{\boldsymbol{\mu}^d_{l,n}\|\boldsymbol{\nu}^d_{l,n}}_{\text{slot } d-1} \quad \underbrace{\begin{cases}\mathbf{0} \,\|\, \widetilde{\boldsymbol{\nu}}^d_{l,\rho_{\le d-1}\|n} & \text{if } n < \rho_d\\ \widetilde{\boldsymbol{\mu}}^d_{l,\rho_{\le d-1}\|n} \,\|\, \mathbf{0} & \text{if } n > \rho_d\\ \mathbf{0} \;\|\; \mathbf{0} & \text{if } n = \rho_d\end{cases}}_{\text{slot } d} \quad \underbrace{\begin{cases}\mathbf{0} & \text{if } n < \rho_d\\ \mathbf{0} & \text{if } n > \rho_d\\ \mathbf{1} & \text{if } n = \rho_d\end{cases}}_{\text{slot } > d}$$

- For every $j \in [L]$, generate $\mathsf{SK}_j = (\mathbf{c}_j, \mathsf{tSK}_j)$ for coefficient vector $\mathbf{c}_j$ with

$$\mathsf{tSK}_j \xleftarrow{\$} \mathsf{tIPE.KeyGen}\left(\mathsf{tMSK}, \left\langle \underline{\hat{S}(\rho)}, \mathbf{c}_j \right\rangle \| -r_\ell \left\langle \underline{\hat{S}(\rho)}, \mathbf{c}_j \right\rangle + \Delta^b_j(\rho)\right),$$

  where $\hat{S}(\rho)$ is defined in Equation 15 and,

$$\Delta^b_j(\rho) = \begin{cases}\sum_{\rho' < \rho} \delta_j(\rho') & \text{in } H^0_\rho\\ \sum_{\rho' \le \rho} \delta_j(\rho') & \text{in } H^1_\rho\end{cases} \quad \text{with} \quad \delta_j(\rho') = \sum_{\substack{I = \rho'\|n\\ n\in[N]}} \mathbf{c}_{j,I}\left(u^{\le D}_{\ell,I} - v^{\le D}_{\ell,I}\right)$$

Output $\{\mathsf{SK}_l,\ \mathsf{CT}_l\}_{l\in[L]}$

Figure 10: Hybrid $G^b_\rho(\lambda)$ for $1 \le d^\star \le D-1$ and $\rho \in [N]^{d^\star}$ for proving security of $\mathbf{FE}^{D,N}$

Observe that when $d^\star = 1$ and $\rho \in [N]$, $(\hat{S}(\rho))_I = w^1_{I_1} s^2_{I_2} \cdots s^D_{I_D}$ for all I. By Fact 3, $w^1_n = s^1_n$, we have that $\hat{S}(\rho) = \otimes \mathbf{s}^{\leq D}$, the tensor product of $\{\mathbf{s}^d\}_d$.

Next, we prove each of the rules.

Proof of Rule 1: $G^0_1 \approx$ **Init.** Since $\rho = 1$ has length 1, as observed above, $\hat{S}(\rho) = \otimes \mathbf{s}^{\leq D}$. Thus the vectors $\{\mathbf{t}_j\}$ encoded in $\{\mathsf{tSK}_j\}$ in G^0_1 are identical to that in **Init**. Therefore, the only difference between these two hybrids are the vectors encrypted in ciphertexts $\{\mathsf{CT}_l\}$, which consists of ciphertexts and secret keys of **dIPE**. In G^0_1 the following vectors are encrypted:

$$\hat{\mathbf{X}}^1_{l,n} = \underbrace{\widetilde{\boldsymbol{\mu}}^1_{l,n} || \mathbf{0}}_{\text{slot 1}} \; \Big|\Big| \; \mathbf{0} \qquad \forall\, d > 1, \;\; \hat{\mathbf{X}}^d_{l,n} = \underbrace{\boldsymbol{\mu}^d_{l,n} || \boldsymbol{\nu}^d_{l,n}}_{\text{slot 1}} \; \Big|\Big| \; \mathbf{0}$$

In **Init**, the following vectors are encrypted

$$\mathbf{X}^d_{l,n} = \underbrace{\boldsymbol{\mu}^d_{l,n} || \mathbf{0}}_{\text{slot 1}} \; \Big|\Big| \; \mathbf{0}$$

By Fact 3, $\widetilde{\boldsymbol{\mu}}^1_{l,n} = \boldsymbol{\mu}^1_{l,n}$. Thus, for every l and every combination I, the inner product of vectors $(\hat{\mathbf{X}}^1_{l,I_1}, \cdots, \hat{\mathbf{X}}^D_{l,I_D})$ in G^0_1 and that of vectors $(\mathbf{X}^1_{l,I_1}, \cdots, \mathbf{X}^D_{l,I_D})$ in **Init** are identical. Thus, it follows from the security of **dIPE** that G^0_1 and **Init** are indistinguishable.

Proof of Rule 2: $G^1_\Gamma \approx$ **Mid′.** This rule follows syntactically the same proof for Rule 1.

Proof of Rule 3: $G^0_{\rho||1} \approx G^0_\rho$, for every $\rho \in [N]^{d^\star}$ with $1 \leq d^\star < D - 1$. Fix one such ρ and $d^\star$. Hybrids $G^0_{\rho||1}$ and G^0_ρ differ in the values of $\{\hat{\mathbf{X}}^d_{l,n}\}$ for $d \geq d^\star$, as well as the $\hat{S}$ vector from which $\{\mathbf{t}_j\}$ encoded in $\{\mathsf{tSK}_j\}$ are derived form.
In $G^0_{\rho||1}$, the $\hat{X}$ vectors have values,

$$\begin{array}{ccccccc}
\hat{\mathbf{X}}^{d^\star}_{l,n} & = & \boldsymbol{\mu}^{d^\star}_{l,n} || \boldsymbol{\nu}^{d^\star}_{l,n} & \cdots & \boldsymbol{\mu}^{d^\star}_{l,n} || \boldsymbol{\nu}^{d^\star}_{l,n} & \begin{cases} \mathbf{0} \,||\, \widetilde{\boldsymbol{\nu}}^{d^\star}_{l,\rho_{\leq d^\star-1}||n} & \text{if } n < \rho_{d^\star} \\ \widetilde{\boldsymbol{\mu}}^{d^\star}_{l,\rho_{\leq d^\star-1}||n} \,||\, \mathbf{0} & \text{if } n > \rho_{d^\star} \\ \mathbf{0} \;\; || \;\; \mathbf{0} & \text{if } n = \rho_{d^\star} \end{cases} & \begin{cases} \mathbf{0} & \text{if } n < \rho_{d^\star} \\ \mathbf{0} & \text{if } n > \rho_{d^\star} \\ \mathbf{1} & \text{if } n = \rho_{d^\star} \end{cases} \\
\hat{\mathbf{X}}^{d^\star+1}_{l,n} & = & \boldsymbol{\mu}^{d^\star+1}_{l,n} || \boldsymbol{\nu}^{d^\star+1}_{l,n} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{l,n} || \boldsymbol{\nu}^{d^\star+1}_{l,n} & \boldsymbol{\mu}^{d^\star+1}_{l,n} || \boldsymbol{\nu}^{d^\star+1}_{l,n} & \widetilde{\boldsymbol{\mu}}^{d^\star+1}_{l,\rho||n} \,||\, \mathbf{0} \,||\, \mathbf{0} \\
\hat{\mathbf{X}}^{d}_{l,n} & = & \boldsymbol{\mu}^{d}_{l,n} || \boldsymbol{\nu}^{d}_{l,n} & \cdots & \boldsymbol{\mu}^{d}_{l,n} || \boldsymbol{\nu}^{d}_{l,n} & \boldsymbol{\mu}^{d}_{l,n} || \boldsymbol{\nu}^{d}_{l,n} & \boldsymbol{\mu}^{d}_{l,n} \,||\, \boldsymbol{\nu}^{d}_{l,n} || \, \mathbf{0} \\
 & & \text{slot } 1 & \cdots & \text{slot } d^\star - 1 & \text{slot } d^\star & \text{slot} \geq d^\star + 1
\end{array}$$

where the last line is for $d > d^\star + 1$. In addition, $\hat{S}(\rho||1)$ is define as

$$\forall\, I \in [N]^D, \quad \text{let } d-1 \text{ be the length of the longest prefix that } I \text{ shares with } \rho||1$$

$$(\hat{S}(\rho||1))_I = \begin{cases} w^{d^\star+1}_{\rho||1} s^{d^\star+2}_{I_{d^\star+2}} \cdots s^D_{I_D} & \text{if } d-1 = d^\star + 1 \\ w^d_{\rho_{\leq d-1}||I_d} s^{d+1}_{I_{d+1}} \cdots s^D_{I_D} & \text{if } d - 1 < d^\star + 1\,. \end{cases}$$

Note that the vectors $\hat{\mathbf{X}}$'s and $\hat{S}$ depend on different w elements. In particular, the N random element $\{w^{d^\star+1}_{\rho||n}\}_{n\in[N]}$ appears in $(\hat{S}(\rho||1))_I$ for every I that starts with ρ, as well as in vectors $\{\bar{\mathbf{X}}^{d^\star+1}_{l,n}\}_{l,n}$, whose slot $d^\star+1$ contains

$$\widetilde{\boldsymbol{\mu}}^{d^\star+1}_{l,\rho||n} = \begin{cases} u^{\leq d^\star+1}_{\rho||n} \mid\mid \underline{w^{d^\star+1}_{\rho||n}} & \text{if } d^\star+1 < D \\ u^{\leq D}_{\rho||n} \mid\mid r_l \underline{w^{D}_{\rho||n}} & \text{if } d^\star+1 = D \end{cases}$$

Moving to $\widetilde{G}^0_{\rho+1}$ We first show that it is indistinguishable to switch every appearance of the random element $w^{d^\star+1}_{\rho||n}$ with the product $w^{d^\star}_{\rho} s^{d^\star+1}_{n}$ (the rest stays the same) – call the resulting distribution $\widetilde{G}^0_{\rho||1}$. More precisely, the vectors $\{\hat{\mathbf{X}}^{d^\star+1}_{l,n}\}_{l,n}$ now becomes the following

$$\begin{array}{rcccccc} \hat{\mathbf{X}}^{d^\star+1}_{l,n} & = & \boldsymbol{\mu}^{d^\star+1}_{l,n}||\boldsymbol{\nu}^{d^\star+1}_{l,n} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{l,n}||\boldsymbol{\nu}^{d^\star+1}_{l,n} & \boldsymbol{\mu}^{d^\star+1}_{l,n}||\boldsymbol{\nu}^{d^\star+1}_{l,n} & \underline{\widetilde{\boldsymbol{\mu}}^{d^\star}_{l,\rho}\boldsymbol{\mu}^{d^\star+1}_{l,n}} \mid\mid \mathbf{0} \mid\mid \mathbf{0} \\ & & \text{slot } 1 & \cdots & \text{slot } d^\star-1 & \text{slot } d^\star & \text{slot} \geq d^\star+1 \end{array},$$

where the last slot contains value

$$\widetilde{\boldsymbol{\mu}}^{d^\star}_{\rho,l}\boldsymbol{\mu}^{d^\star+1}_{l,n} = \begin{cases} u^{\leq d^\star+1}_{\rho||n} \mid\mid \underline{w^{d^\star}_{\rho} s^{d^\star+1}_{n}} & \text{if } d^\star+1 < D \\ u^{\leq D}_{\rho||n} \mid\mid r_l \underline{w^{D-1}_{\rho} s^{d^\star+1}_{n}} & \text{if } d^\star+1 = D \end{cases}.$$

Correspondingly, in $\hat{S}(\rho||1)$, for every I with prefix ρ, that is, $I = \rho||I_{\geq d^\star+1}$, the I^{th} element is replaced with the corresponding element in $\hat{S}(\rho)_I$,

$$(\hat{S}(\rho))_I = \underline{w^{d^\star}_{\rho} s^{d^\star+1}_{I_{d^\star+1}}} s^{d^\star+2}_{I_{d^\star+2}} \cdots s^{D}_{I_D}$$

Note that by definition of $\hat{S}(\star)$, $\hat{S}(\rho)$ and $\hat{S}(\rho||1)$ differ only at indexes that start with ρ. Therefore, in $\widetilde{G}^b_{\rho||1}$, the vectors $\mathbf{t}_j$'s encoded in tSK_j's are derived from $\hat{S}(\rho)$ as opposed to $\hat{S}(\rho||1)$.

Claim 2. *Hybrids $G^0_{\rho||1}(\lambda)$ and $\widetilde{G}^0_{\rho||1}(\lambda)$ are indistinguishable.*

Proof. To show the claim, we consider yet another two hybrids, $L^0_{\rho||1}$ and $\widetilde{L}^0_{\rho||1}$, which are identical to $G^0_{\rho||1}(\lambda)$ and $\widetilde{G}^0_{\rho||1}(\lambda)$ respectively, except that for every l, n, the vector $\hat{\mathbf{X}}^{d^\star+1}_{l,n}$ is encoded in $\mathsf{dSK}_{l,n}$ contained in CT_l, whereas the vector $\hat{\mathbf{X}}^{D}_{l,n}$ is encrypted in $\mathsf{dCT}^{d^\star+1}_{l,n}$ in CT_l. In short, the vectors encrypted at coordinate $d^\star+1$ are swapped with the vectors encoded at coordinate D. Since swapping does not change the set of inner product outputs that can be computed from $\{\mathsf{dSK}_{l,n}, \mathsf{dCT}^{d}_{l,n}\}_{d<D,n}$ for every l (secret keys and ciphertexts with different l indexes cannot be decrypted together since they are generated using independently sampled master secret keys), by the security of **dIPE**, $L^0_{\rho||1}$ and $G^0_{\rho||1}$, as well as $\widetilde{L}^0_{\rho||1}$ and $\widetilde{G}^0_{\rho||1}$ are indistinguishable.

Thus, it remains to show that $L^0_{\rho||1}$ and $\widetilde{L}^0_{\rho||1}$ are indistinguishable. The only difference between them is that in the former random element $\{w^{d^\star+1}_{\rho||n}\}_n$ are used, whereas in the latter, $\{w^{d^\star}_{\rho} s^{d^\star+1}_{n}\}_n$ are used. Recall that these elements only appear in the $\hat{S}$ and $\{\hat{\mathbf{X}}^{d^\star+1}_{l,n}\}_{l,n}$ vectors, encoded in $\{\mathsf{tSK}_j\}_j$ and $\{\mathsf{dSK}_{l,n}\}$ respectively. By the fact that **tIPE** and **dIPE** are canonical, $G^0_{\rho||1}$ and $\widetilde{G}^0_{\rho||1}$ can be generated from the following distributions respectively,

$$\left\{ \left[w^{d^\star}_{\rho}\right]_D, \left\{ \left[s^{d^\star+1}_{n}\right]_D \right\}_n, \left\{ \left[w^{d^\star+1}_{\rho||n}\right]_D \right\}_n \right\}$$
$$\left\{ \left[w^{d^\star}_{\rho}\right]_D, \left\{ \left[s^{d^\star+1}_{n}\right]_D \right\}_n, \left\{ \left[w^{d^\star}_{\rho} s^{d^\star+1}_{n}\right]_D \right\}_n \right\}$$

This is because, from the above encodings, one can generate the encodings of $\hat{\mathbf{X}}^{d^\star+1}_{n,l}$ in G_D with knowledge of values of $\mathbf{u}$'s, $\mathbf{v}$'s, and r_l, as well as encodings of $\hat{S}(\rho||1)$ or $\hat{S}(\rho)$ in G_D, with knowledge of all the s and w elements that do not appear in the above distributions. From these encodings, one can emulate every $\mathsf{dSK}_{l,n}$ and tSK_j with knowledge of dMSK_l and tMSK_j, and further hybrids $G^0_{\rho+1}$ or $\widetilde{G}^0_\rho$.

Finally, the indistinguishability of the above two distributions follow directly from the SXDH assumption on group G_D, which concludes the indistinguishability of $G^0_{\rho||1}$ and $\widetilde{G}^0_{\rho||1}$. □

Moving to G^0_ρ It remains to show that $\widetilde{G}^0_{\rho||1}$ is indistinguishable from G^0_ρ, we have already argued above that these two hybrids depend on the same $\hat{S}(\rho)$ vector. Thus, their only difference lies in the $\hat{\mathbf{X}}$ vectors. In G^0_ρ, the value of these vectors are (the difference from that in $\widetilde{G}^0_{\rho||1}$ is underlined).

$$\begin{array}{ccccccc}
\hat{\mathbf{X}}^{d^\star}_{l,n} & = & \boldsymbol{\mu}^{d^\star}_{l,n}||\boldsymbol{\nu}^{d^\star}_{l,n} & \cdots & \boldsymbol{\mu}^{d^\star}_{l,n}||\boldsymbol{\nu}^{d^\star}_{l,n} & \begin{cases} \mathbf{0} \;||\; \widetilde{\boldsymbol{\nu}}^{d^\star}_{l,\rho_{\le d^\star-1}||n} & \text{if } n < \rho_{d^\star} \\ \widetilde{\boldsymbol{\mu}}^{d^\star}_{l,\rho_{\le d^\star-1}||n} \;||\; \mathbf{0} & \text{if } n > \rho_{d^\star} \\ \underline{\widetilde{\boldsymbol{\mu}}^{d^\star}_{l,\rho} \;||\; \mathbf{0}} & \text{if } n = \rho_{d^\star} \end{cases} & \underline{\mathbf{0}} \\
\hat{\mathbf{X}}^{d^\star+1}_{l,n} & = & \boldsymbol{\mu}^{d^\star+1}_{l,n}||\boldsymbol{\nu}^{d^\star+1}_{l,n} & \cdots & \boldsymbol{\mu}^{d^\star+1}_{l,n}||\boldsymbol{\nu}^{d^\star+1}_{l,n} & \boldsymbol{\mu}^{d^\star+1}_{l,n}||\boldsymbol{\nu}^{d^\star+1}_{l,n} & \underline{\mathbf{0}} \\
\hat{\mathbf{X}}^{d}_{l,n} & = & \boldsymbol{\mu}^{d}_{l,n}||\boldsymbol{\nu}^{d}_{l,n} & \cdots & \boldsymbol{\mu}^{d}_{l,n}||\boldsymbol{\nu}^{d}_{l,n} & \boldsymbol{\mu}^{d}_{l,n}||\boldsymbol{\nu}^{d}_{l,n} & \underline{\mathbf{0}} \\
 & & \text{slot } 1 & \cdots & \text{slot } d^\star - 1 & \text{slot } d^\star & \text{slot} \ge d^\star + 1
\end{array}$$

Examine the different values of $\{\hat{\mathbf{X}}^d_{l,n}\}$ for $d \ge d^\star$ in $\widetilde{G}^0_{\rho||1}$ and G^0_ρ, and the values of $\{\hat{\mathbf{X}}^d_{l,n}\}$ for $d < d^\star$ that are the same in these two hybrids (as described in Case 3 of Figure 10). They satisfy that for every combination $I \in [N]^D$ and l, the inner product of $\{\hat{\mathbf{X}}^d_{I_d,l}\}_d$ in $\widetilde{G}^0_{\rho||1}$ and G^0_ρ is identical. Therefore, by the function hiding of **dIPE**, these two hybrids are indistinguishable.

Proof of Rule 4: $G^1_{\rho||\Gamma} \approx G^1_\rho$, for every $\rho \in [N]^{d^\star}$ with $1 \le d^\star < D-1$. This rule follows syntactically from the same proof for Rule 3.

Proof of Rule 5: $G^b_\rho \approx H^b_\rho$, for every $\rho \in [N]^{D-1}$ and every b. The proof of this rule is again very similar to that of Rule 3 and 4. We here sketch the proof. The difference between G^b_ρ and H^b_ρ lies in the values of vectors $\{\hat{\mathbf{X}}^d_{l,n}\}$ and $\{\widetilde{\mathbf{X}}^d_{l,n}\}$ for $d = D-1$ and D, as well as in the values of $\hat{S}(\rho)_I$ and $S(\rho)_I$ for these indexes $I = \rho||n$ that start with ρ.

In hybrid H^b_ρ, the vectors $\{\widetilde{\mathbf{X}}^{D-1}_{l,n}, \widetilde{\mathbf{X}}^D_{l,n}\}_n$ are set to the following.

$$\begin{array}{ccccccc}
\widetilde{\mathbf{X}}^{D-1}_{l,n} & = & \boldsymbol{\mu}^{D-1}_{l,n}||\boldsymbol{\nu}^{D-1}_{l,n} & \cdots & \boldsymbol{\mu}^{D-1}_{l,n}||\boldsymbol{\nu}^{D-1}_{l,n} & \begin{cases} \mathbf{0} \;||\; \widetilde{\boldsymbol{\nu}}^{D-1}_{l,\rho_{\le D-2}||n} & \text{if } n < \rho_{D-1} \\ \widetilde{\boldsymbol{\mu}}^{D-1}_{l,\rho_{\le D-2}||n} \;||\; \mathbf{0} & \text{if } n > \rho_{D-1} \\ \mathbf{0} \;||\; \mathbf{0} & \text{if } n = \rho_{D-1} \end{cases} & 0 \\
\widetilde{\mathbf{X}}^{D}_{l,n} & = & \boldsymbol{\mu}^{D}_{l,n}||\boldsymbol{\nu}^{D}_{l,n} & \cdots & \boldsymbol{\mu}^{D}_{l,n}||\boldsymbol{\nu}^{D}_{l,n} & \boldsymbol{\mu}^{D}_{l,n}||\boldsymbol{\nu}^{D}_{l,n} & \begin{cases} \left\langle \widetilde{\boldsymbol{\mu}}^{D}_{l,\rho||n}, \mathbf{1} \right\rangle & \text{if in } H^0_\rho \\ \left\langle \widetilde{\boldsymbol{\nu}}^{D}_{l,\rho||n}, \mathbf{1} \right\rangle & \text{if in } H^1_\rho \end{cases} \\
 & & \text{slot } 1 & \cdots & \text{slot } D-2 & \text{slot } D-1 & \text{slot } D
\end{array}$$

Moreover, for every index $I = \rho||n$,

$$(S(\rho))_{\rho||n} = w^D_{\rho||n}$$

It follows from the SXDH assumption w.r.t. group G_D that H^b_ρ is indistinguishable to $\widetilde{H}^b_\rho$, where every random element $\{w^D_{\rho||n}\}_n$ is replaced with $\{w^{D-1}_\rho s^D_n\}_n$. This changes the values of vectors $\{\widetilde{\mathbf{X}}^D_{l,n}\}$ to the following

$$\begin{array}{ccccccc} \widetilde{\mathbf{X}}^D_{l,n} & = & \boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n} & \cdots & \boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n} & \boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n} & \begin{cases} \left\langle \underline{\widetilde{\boldsymbol{\mu}}^{D-1}_{l,\rho}\boldsymbol{\mu}^D_{l,n}}, \mathbf{1} \right\rangle & \text{if in } \widetilde{H}^0_\rho \\ \left\langle \underline{\widetilde{\boldsymbol{\nu}}^{D-1}_{l,\rho}\boldsymbol{\nu}^D_{l,n}}, \mathbf{1} \right\rangle & \text{if in } \widetilde{H}^1_\rho \end{cases} \\ & & \text{slot } 1 & \cdots & \text{slot } D-2 & \text{slot } D-1 & \text{slot } D \end{array}$$

Moreover, it changes the value of the S vector for every index $I = \rho||n$ to

$$(\hat{S}(\rho))_{\rho||n} = w^{D-1}_\rho s^D_n$$

Since $I = \rho||n$ are the only indexes where $S(\rho)$ and $\hat{S}(\rho)$ differ, we have that $\widetilde{H}^b_\rho$ uses $\hat{S}(\rho)$.
It now remains to show that $\widetilde{H}^b_\rho$ is indistinguishable to G^b_ρ. They both use vector $\hat{S}(\rho)$, and their only difference is that G^b_ρ encodes the following vectors $\{\hat{\mathbf{X}}^{D-1}_{l,n}, \hat{\mathbf{X}}^D_{l,n}\}_n$.

$$\begin{array}{ccccccc} \hat{\mathbf{X}}^{D-1}_{l,n} & = & \boldsymbol{\mu}^{D-1}_{l,n}||\boldsymbol{\nu}^{D-1}_{l,n} & \cdots & \boldsymbol{\mu}^{D-1}_{l,n}||\boldsymbol{\nu}^{D-1}_{l,n} & \begin{cases} \mathbf{0} \,||\, \widetilde{\boldsymbol{\nu}}^{D-1}_{l,\rho_{\le D-2}||n} & \text{if } n < \rho_{D-1} \\ \widetilde{\boldsymbol{\mu}}^{D-1}_{l,\rho_{\le D-2}||n} \,||\, \mathbf{0} & \text{if } n > \rho_{D-1} \\ \underline{\mathbf{0} \,||\, \widetilde{\boldsymbol{\nu}}^{D-1}_{l,\rho}} & \text{if } n = \rho_{D-1} \text{ and in } G^1_\rho \\ \underline{\widetilde{\boldsymbol{\mu}}^{D-1}_{l,\rho} \,||\, \mathbf{0}} & \text{if } n = \rho_{D-1} \text{ and in } G^0_\rho \end{cases} & \underline{0} \\ \hat{\mathbf{X}}^D_{l,n} & = & \boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n} & \cdots & \boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n} & \boldsymbol{\mu}^D_{l,n}||\boldsymbol{\nu}^D_{l,n} & \underline{0} \\ & & \text{slot } 1 & \cdots & \text{slot } D-2 & \text{slot } D-1 & \text{slot } D \end{array}$$

For all other coordinates $d < D-1$, $\hat{\mathbf{X}}^d_{l,n} = \widetilde{\mathbf{X}}^d_{l,n}$ in $\widetilde{H}^b_\rho$. Observe that the inner products of all combination of vectors $\widetilde{\mathbf{X}}$'s in $\widetilde{H}^b_\rho$ and $\hat{\mathbf{X}}$'s in G^b_ρ are identical. Thus, it follows from the security of **dIPE** that these two hybrids are indistinguishable.

Proof of Rule 6: $G^1_\rho = G^0_{\rho+1}$ for every $\rho \in [N]^{d^\star}$ with $1 \le d^\star \le D-1$, such that, $\rho_{d^\star} \ne N$. Fix such a ρ and $d^\star$. Since $\rho_{d^\star} \ne N$, $\rho + 1$ has form $\rho_{<d^\star}||(\rho_{d^\star} + 1)$, where $\rho_{d^\star} + 1$ is the letter that follows immediately after $\rho_{d^\star}$ in the alphabet $[N]$. In this case, vectors $\{\hat{\mathbf{X}}^d_{l,n}\}_{d,l,n}$ are identical the two hybrids. (See the argument for Rule 6 in proof of Lemma 11). Moreover, observe that by definition in Equation 15, for ρ and $\rho+1$ that have the same length and differ only at the last letter, $\hat{S}(\rho) = \hat{S}(\rho + 1)$. Therefore, G^0_ρ and G^1_ρ are identical. $\square$

Acknowledgements. The author thanks Nir Bitansky, Stefano Tessaro, and Vinod Vaikuntanathan for many helpful discussions and insights, and Benny Applebaum for sharing his expertise.
Huijia Lin was partially supported by NSF grants CNS-1528178 and CNS-1514526.

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 528–556, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[ABCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, 2015.

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 646–658, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. In *FOCS*, pages 166–175, 2004.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[AIK08] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc^0. *Computational Complexity*, 17(1):38–69, 2008.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. *IACR Cryptology ePrint Archive*, 2015:730, 2015.

[AL16] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 1087–1100. ACM, 2016.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307. IEEE Computer Society, 2003.

[App12] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 805–816. ACM, 2012.

[BGI$^+$01a] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology CRYPTO 2001*, pages 1–18. Springer, 2001.

[BGI$^+$01b] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BGJ$^+$16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 345–356. ACM, 2016.

[BGK$^+$14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[BGL$^+$15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 439–448. ACM, 2015.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 470–491. Springer, 2015.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 391–418, 2016.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In Guruswami [Gur15], pages 1480–1498.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 474–502, 2016.

[BQ12] Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. *Computational Complexity*, 21(1):83–127, 2012.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of

LNCS, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *IACR Cryptology ePrint Archive*, 2002:80, 2002.

[BSW12] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 171–190, 2015.

[CEMT09] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In *TCC*, pages 521–538, 2009.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 429–437. ACM, 2015.

[CHN$^+$15] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *IACR Cryptology ePrint Archive*, 2015:1096, 2015. To Appear in ACM STOC 2016.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 267–286, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[CM01] M. Cryan and P. B. Miltersen. On pseudorandom generators in nc0. In Proc. 26th MFCS, 2001.

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 164–195. Springer, 2016.

[DN15] Yevgeniy Dodis and Jesper Buus Nielsen, editors. *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*. Springer, 2015.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Lindell [Lin14], pages 74–94.

[GGHZ16] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 480–511. Springer, 2016.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476. ACM, 2013.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private RAM computation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 404–413. IEEE Computer Society, 2014.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Guruswami [Gur15], pages 151–170.

[GMS16] Sanjam Garg, Pratyay Mukherjee, and Akshayaram Srinivasan. Obfuscation without the vulnerabilities of multilinear maps. *IACR Cryptology ePrint Archive*, 2016:390, 2016.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.

[GP15] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Dodis and Nielsen [DN15], pages 614–637.

[Gur15] Venkatesan Guruswami, editor. *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. IEEE Computer Society, 2015.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428, Portland, OR, USA, June 14–17, 2015. ACM Press.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, pages 146–162, 2008.

[Lin14] Yehuda Lindell, editor. *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*. Springer, 2014.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes, 2016. To Appear in Eurocrypt'16.

[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, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. Gghlite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256. Springer, 2014.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, New Brunswick, NJ, USA, 9-11 October, 2016*, 2016.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 136–145, 2003.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. *IACR Cryptology ePrint Archive*, 2016:147, 2016.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/.

[OW14] Ryan O'Donnell and David Witmer. Goldreich's PRG: evidence for near-optimal polynomial stretch. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 1–12, 2014.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 500–517, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 439–467, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.