(Compact) Adaptively Secure FE for Attribute-Weighted Sums from k-Lin[⋆]

Pratish Datta[1] and Tapas Pal[1,2]

[1] NTT Research, Sunnyvale, CA 94085, USA
pratish.datta@ntt-research.com,
[2] IIT Kharagpur, West Bengal, India-721302
tapas.pal@iitkgp.ac.in,

September 28, 2021

Abstract

This paper presents the *first adaptively simulation secure* functional encryption (FE) schemes for attribute-weighted sums. In such an FE scheme, encryption takes as input N pairs of attribute $\{(x_i, z_i)\}_{i\in[N]}$ for some $N \in \mathbb{N}$ where the attributes $\{x_i\}_{i\in[N]}$ are public while the attributes $\{z_i\}_{i\in[N]}$ are private. The indices $i \in [N]$ are referred to as the slots. A secret key corresponds to some weight function f, and decryption recovers the weighted sum $\sum_{i=1}^{N} f(x_i)z_i$. This is an important functionality with a wide range of potential real life applications. In the proposed FE schemes attributes are viewed as vectors and weight functions are arithmetic branching programs (ABP). We present two schemes with varying parameters and levels of adaptive security.

(a) We first present a one-slot scheme that achieves adaptive security in the simulation-based security model against a bounded number of ciphertext queries and an arbitrary polynomial number of secret key queries both before and after the ciphertext queries. This is the best possible level of security one can achieve in the adaptive simulation-based framework. From the relations between the simulation-based and indistinguishability-based security frameworks for FE, it follows that the proposed FE scheme also achieves indistinguishability-based adaptive security against an a-priori unbounded number of ciphertext queries and an arbitrary polynomial number of secret key queries both before and after the ciphertext queries. Moreover, the scheme enjoys *compact* ciphertexts that do not grow with the number of appearances of the attributes within the weight functions.

(b) Next, bootstrapping from the one-slot scheme, we present an unbounded-slot scheme that achieves simulation-based adaptive security against a bounded number of ciphertext and pre-ciphertext secret key queries while supporting an a-priori unbounded number of post-ciphertext secret key queries. The scheme achieves public parameters and secret key sizes independent of the number of slots N and a secret key can decrypt a ciphertext for any a-priori unbounded N. Further, just like the one-slot scheme, this scheme also has the ciphertext size independent of the number of appearances of the attributes within the weight functions. However, all the parameters of the scheme, namely, the master public key, ciphertexts, and secret keys scale linearly with the bound on the number of pre-ciphertext secret key queries.

Our schemes are built upon asymmetric bilinear groups of prime order and the security is derived under the standard (bilateral) k-Linear (k-Lin) assumption. Our work *resolves an open problem* posed by Abdalla, Gong, and Wee in CRYPTO 2020, where they presented an unbounded-slot FE scheme for attribute-weighted sum achieving only semi-adaptive simulation security. At a technical level, our work extends the recent adaptive security framework of Lin and Luo [EUROCRYPT 2020], devised to achieve compact ciphertexts in the context of indistinguishability-based payload-hiding security, into the setting of simulation-based adaptive attribute-hiding security.

⋆ This is the full version of an extended abstract that will appear in ASIACRYPT 2021.

Contents

1 Introduction

Functional Encryption: *Functional encryption* (FE), formally introduced by Boneh et al. [BSW11] and O'Neill [O'N10], redefines the classical encryption procedure with the motivation to overcome the limitation of the "all-or-nothing" paradigm of decryption. In a traditional encryption system, there is a single secret key such that a user given a ciphertext can either recover the whole message or learns nothing about it, depending on the availability of the secret key. FE in contrast provides fine grained access control over encrypted data by generating artistic secret keys according to the desired functions of the encrypted data to be disclosed. More specifically, in a public-key FE scheme for a function class $\mathcal{F}$, there is a setup authority which produces a master secret key and publishes a master public key. Using the master secret key, the setup authority can derive secret keys or functional decryption keys SK_f associated to functions $f \in \mathcal{F}$. Anyone can encrypt messages msg belonging to a specified message space $\mathsf{msg} \in \mathbb{M}$ using the master public key to produce a ciphertext CT. The ciphertext CT along with a secret key SK_f recovers the function of the message $f(\mathsf{msg})$ at the time of decryption, while unable to extract any other information about msg. More specifically, the security of FE requires *collusion resistance* meaning that any polynomial number of secret keys together cannot gather more information about an encrypted message except the union of what each of the secret keys can learn individually.

FE for Attribute-Weighted Sum: Recently, Abdalla, Gong and Wee [AGW20] proposed an FE scheme for a new class of functionalities which they termed as "attribute-weighted sums". This is a generalization of the inner product functional encryption (IPFE) [ABDCP15, ALS16]. In such a scheme, a database of N attribute-value pairs $(x_i, z_i)_{i=1,\ldots,N}$ are encrypted using the master public key of the scheme, where x_i is a public attribute (e.g., demographic data) and z_i is a private attribute containing sensitive information (e.g., salary, medical condition, loans, college admission outcomes). A recipient having a secret key corresponding to a weight function f can learn the attribute-weighted sum of the database, i.e., $\sum_{i=1}^{N} f(x_i)z_i$. The attribute-weighted sum functionality appears naturally in several real life applications. For instance, as discussed by Abdalla et al. [AGW20] if we consider the weight function f as a boolean predicate, then the attribute-weighted sum functionality $\sum_{i=1}^{N} f(x_i)z_i$ would correspond to the average z_i over all users whose attribute $\boldsymbol{x}_i$ satisfies the predicate f. Important practical scenarios include average salaries of minority groups holding a particular job (z_i = salary) and approval ratings of an election candidate amongst specific demographic groups in a particular state (z_i = rating). Similarly, if z_i is boolean, then the attribute-weighted sum becomes $\sum_{i:z_i=1} f(x_i)$. This could capture for instance the number of and average age of smokers with lung cancer (z_i = lung cancer, f = numbers/age).

The work of [AGW20] considered a more general case of the notion where the domain and range of the weight functions are vectors over some finite field $\mathbb{Z}_p$. In particular, the database consists of N pairs of public/private attribute vectors $(\boldsymbol{x}_i, \boldsymbol{z}_i)_{i=1,\ldots,N}$ which is encrypted to a ciphertext CT. A secret key SK_f generated for a weight function f allows a recipient to learn $\sum_{i=1}^{N} f(\boldsymbol{x}_i)^\top \boldsymbol{z}_i$ from CT without revealing any information about the private attribute vectors $(\boldsymbol{z}_i)_{i=1,\ldots,N}$. To handle a large database where the number of users are not a-priori bounded, Abdalla et al. considered the notion of *unbounded-slot* FE scheme for attribute-weighted sum. Thus, in their scheme, the number of *slots* N is not fixed while generating the system parameters and any secret key SK_f can decrypt an encrypted database having an arbitrary number of slots. Another advantage of unbounded-slot FE is that the same system parameters and secret keys can be reused for different databases with variable lengths, which saves storage space and reduces communication cost significantly.

The unbounded-slot FE of [AGW20] supports expressive function class of *arithmetic branching programs* (ABPs) which is capable of capturing boolean formulas, boolean span programs, combinatorial computations, and arithmetic span programs. The FE scheme of [AGW20] is built in asymmetric bilinear groups of prime order and is proven secure in the simulation-based security model, which is known to be the desirable security model for FE [O'N10, BSW11], under the k-Linear (k-Lin)/*Matrix Diffie-Hellman* (MDDH) assumption. Moreover, their scheme enjoys ciphertext size that grows with the number of slots and the size of the private attribute vectors but is independent of the size of the public attribute vectors. Towards constructing their unbounded-slot scheme, Abdalla et al. first constructed a one-slot scheme and then bootstrap to the unbounded-slot scheme via a semi-generic transformation.

However, one significant limitation of the FE scheme of [AGW20] is that the scheme only achieves semi-adaptive security. While semi-adaptive security, where the adversary is restricted to making secret key queries only after making the ciphertext queries, may be sufficient for certain applications, it is much weaker compared to the strongest and most natural notion of adaptive security which lets the adversary request secret keys both before and after making the ciphertext queries. Thus it is desirable to have an adaptively secure scheme for this important functionality that supports unbounded number of slots.

One artifact of the standard techniques for proving adaptive security of FE schemes based on the so called dual system encryption methodology [Wat09, LW10, LOS+10] is the use of a core information theoretic transition limiting the appearance of an attribute in the description of the associated functions at most once (or an a-priori bounded number of times at the expense of ciphertext and key sizes scaling with that upper bound [OT10, LW11, Wat11]). Recently Kowalczyk and Wee [KW19] and Lin and Luo [LL20] presented advanced techniques to overcome the one-use restriction. However, their techniques were designed in the context of attribute-based encryption (ABE) where attributes are totally public. Currently, it is not known how to remove the one-use restriction in the context of adaptively secure FE schemes where attributes are not fully public as is the case for the attribute-weighted sum functionality. This leads us to the following open problem explicitly posed by Abdalla et al. [AGW20]:

Open Problem *Can we construct adaptively simulation-secure one-slot/unbounded-slot* FE *scheme for the attribute-weighted sum functionality with the weight functions expressed as arithmetic branching programs featuring compact ciphertexts, that is, having ciphertexts that do not grow with the number of appearances of the attributes within the weight functions, from the k-*Lin *assumption?*

Our Contributions: In this work, we resolve the above open problem. More precisely, we make the following contributions.

(a) We start by presenting the *first* one-slot FE scheme for the attribute-weighted sum functionality with the weight functions represented as ABPs that achieves adaptive simulation-based security and compact ciphertexts, that is, the ciphertext size is independent of the number of appearances of the attributes within the weight functions. The scheme is secure against an adversary who is allowed to make an a-priori bounded number of ciphertext queries and an unbounded (polynomial) number of secret key queries both before and after the ciphertext queries, which is the best possible level of security one could hope to achieve in adaptive simulation-based framework [BSW11]. Since simulation-based security also implies indistinguishability-based security and indistinguishability-based security against single and multiple ciphertexts are equivalent [BSW11, O'N10], the proposed FE scheme is also adaptively secure in the indistinguishability-based model against adversaries making unbounded number of ciphertext and secret key queries in any arbitrary order.

(b) We next bootstrap our one-slot scheme to an unbounded-slot scheme that also achieves simulation-based adaptive security against a bounded number of ciphertext queries and an unbounded polynomial number of secret key queries. Just like our one-slot scheme, the ciphertexts of our unbounded-slot scheme also do not depend on the number of appearances of the attributes within the weight functions. However, the caveat here is that the number of pre-ciphertext secret key queries is a priori bounded and all parameters of the scheme, namely, the master public key, ciphertexts, and secret keys scale linearly with that upper bound.

Like Abdalla et al. [AGW20], our FE schemes are build upon asymmetric bilinear groups of prime order. We prove the security of our FE schemes based on the standard (bilateral) k-Lin/ (bilateral) MDDH assumption(s) [EHK$^+$17]. Thus our results can be summarized as follows.

Theorem 1 (Informal) *Under the (bilateral) k-Lin/MDDH assumption(s), there exist adaptively simulation secure one-slot/unbounded-slot FE scheme for attribute-weighted sums against a bounded number of ciphertext and an unbounded number of secret-key queries, and having compact ciphertexts, that is, without the one-use restriction, in bilinear groups of prime order.*

The bilateral MDDH assumption is the plain MDDH assumption except that the elements are available in the exponents of both source groups of a bilinear group simultaneously. This assumption has recently been utilized in the context of achieving FE for quadratic functions in the standard model [AGT20, Wee20] and broadcast encryption scheme with $O(N^{1/3})$ parameter sizes from bilinear maps, where N is the total number of users in the system [?]. Unlike [AGW20], our construction is semi-generic and is built upon two cryptographic building blocks, namely a slotted inner product functional encryption (IPFE) [LV16, LL20], which is a hybrid of a public-key IPFE and a private-key function-hiding IPFE, and an information theoretic primitive called arithmetic key garbling scheme (AKGS) [IW14, LL20]. For bootstrapping from one-slot to unbounded-slot construction, we make use of the same semi-generic transformation proposed in [AGW20], but analyze its security in the adaptive simulation-based setting as opposed to the semi-adaptive setting. Table 1 shows the current state of the art in the development of efficient attribute-hiding[3] FE schemes under standard computational assumptions.

On the technical side, our contributions lie in extending the recent framework of Lin and Luo [LL20]. The techniques of [LL20] are developed to achieve compact ciphertexts, that is, without the one-use restriction in the context of indistinguishability-based adaptively secure ABE (that is, for payload-hiding security and not attribute-hiding). In this work, we extend their techniques to overcome the one-use restriction into the context of adaptive simulation-based attribute-hiding security for the first time. The high level approach of [LL20] to mitigate the one-use restriction is to replace the core information theoretic step of the dual system technique with a computational step. However the application of this strategy in their framework crucially

[3] In this paper, by attribute-hiding, we mean the so-called "strong" attribute-hiding, as stipulated by the security definitions of FE, meaning that private attributes must remain hidden even to decryptors who are able to perform a successful decryption.

rely on the payload hiding security requirement, that is, the adversaries are not allowed to query secret keys that enable a successful decryption. In contrast, in the setting of attribute-hiding, adversaries are allowed to request secret keys enabling successful decryption and extending the technique of [LL20] into this context appears to be non-trivial. We resolve this by developing a three-slot variant of their framework, integrating the pre-image sampleability of the inner product functionality [O'N10, DOT18], and carefully exploiting the structures of the underlying building blocks, namely AKGS and slotted IPFE.

Related Works: Even before it was formally introduced by [BSW11], FE has been studied for various simplistic functionalities such as equality testing [Sha84, BF01, Coc01], subset membership [BGW05, SW05, BW07], inner product predicates [KSW08], and NC^1 access policies [GPSW06]. Sahai and Seyalioglu [SS10] and Gorbunov, Vaikuntanathan, and Wee [GVW12] considered the problem of constructing FE for general functions under standard computational assumptions. Main drawbacks of these constructions are that the schemes support a-priori bounded number of functional keys and ciphertext size grows linearly with the number of secret keys of the system. Moreover, the ciphertext size is *non-succinct* meaning that the ciphertext size scales with the worst-case circuit size of the functions in the function class. Goldwasser et al. [GKP+13] built a *succinct* FE scheme for general circuits, which enables the authority to release only one secret decryption key under the LWE assumption. Here, succinctness means that the ciphertext size depends on the maximum depth of function class supported by the scheme rather than the size of it. Another line of works [GGH+16, PST14, LV16, LV17, AS17, LT17] based on multilinear maps [GGH13, CLT13], constructs collusion resistant FE scheme for general circuits with succinct ciphertexts. Since multilinear maps are highly inefficient and suffers from many non-trivial attacks [CHL+15, CGH+15, MSZ16], consequently these FE schemes are not assumed to be secure any more. As it seems hard to achieve efficient FE schemes for general circuits from standard assumptions since such an FE scheme would directly imply $i\mathcal{O}$ for general circuits [AJ15, BV15, AJS15], building efficient FE schemes for specific practically useful classes of function has drawn special attention in the community, e.g. attribute-based encryption [BGG+14, GVW15a, Agr17, AY20a, AY20b, DKW20], predicate encryption (PE) [LOS+10, KSW08, GVW15b], partially-hiding PE [Wee17, DOT18], IPFE [ABDCP15, ALS16, DDM16, LV16, LT17, JLS19] attribute-based IPFE [ACGU20] and FE for quadratic functions [BCFG17, Lin17, AGT20, Gay20, Wee20, Wee21].

Paper Organization: We discuss detailed technical overview of our results in Section 2. The preliminaries, definitions and tools are provided in Section 3. We present our private-key (public-key) one-slot FE scheme for the attribute-weighted sum functionality that is proven simulation secure against a single ciphertext query and a single (arbitrary polynomial number of) secret key queries either (both) before or (and) after the ciphertext query in Sections 4 (Section 5). We develop a private-key one-slot FE scheme for an extended attribute-weighted sum functionality (abbreviated as extFE) that is proven simulation secure against a single ciphertext query and a single secret key query either before or after the ciphertext query in Section 6. This scheme is designed for the security reduction of our public-key one-slot extFE scheme that is proven secure against one ciphertext query, an a priori-bounded number of pre-ciphertext secret key queries, and an arbitrary polynomial number of post-ciphertext secret key queries described in Section 7. Next, in Section 8, we apply the bootstrapping transformation from [AGW20] on our public-key one-slot extFE scheme from Section 7 leading to an unbounded-slot FE scheme that is

Table 1: Current State of the Art in Attribute-Hiding FE

Scheme	Functionality	Number of Slots	IND Security	SIM Security	$\lvert\mathsf{CT}\rvert$	Assumption
[KSW08]	$\phi_{\boldsymbol{y}\in\mathbb{Z}_p^n} : \mathbb{Z}_p^n \to \{0,1\}, \phi_{\boldsymbol{y}}(\boldsymbol{z}) = (\boldsymbol{z}^\top\boldsymbol{y} \overset{?}{=} 0)$	1	$(-, \mathsf{poly}, \mathsf{poly})$-AD	$\times$	$O(\lvert\boldsymbol{z}\rvert)$	2 non-standard assumptions
[OT12a]	$\phi_{\boldsymbol{y}\in\mathbb{Z}_p^n} : \mathbb{Z}_p^n \to \{0,1\}, \phi_{\boldsymbol{y}}(\boldsymbol{z}) = (\boldsymbol{z}^\top\boldsymbol{y} \overset{?}{=} 0)$	1	$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-AD	$\times$	$O(\lvert\boldsymbol{z}\rvert)$	DLIN
[ABDCP15]	$\phi_{\boldsymbol{y}\in\mathbb{Z}_p^n} : \mathbb{Z}_p^n \to \mathbb{Z}_p, \phi_{\boldsymbol{y}}(\boldsymbol{z}) = \boldsymbol{z}^\top\boldsymbol{y}$	1	$(-, \mathsf{poly}, \mathsf{poly})$-Sel	$\times$	$O(\lvert\boldsymbol{z}\rvert)$	DDH, LWE
[ALS16, ALMT20]	$\phi_{\boldsymbol{y}\in\mathbb{Z}_p^n} : \mathbb{Z}_p^n \to \mathbb{Z}_p, \phi_{\boldsymbol{y}}(\boldsymbol{z}) = \boldsymbol{z}^\top\boldsymbol{y}$	1	$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-AD	$(\mathsf{poly}, \mathsf{bdd}, \mathsf{poly})$-Sel	$O(\lvert\boldsymbol{z}\rvert)$	DDH, DCR, LWE
[Agr17]	$\phi_{f\in\mathsf{GC}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \to \{0,1\}, \phi_f(\boldsymbol{x},\boldsymbol{z}) = (f(\boldsymbol{x})^\top\boldsymbol{z} \overset{?}{=} 0)$	1	$(-, \mathsf{poly}, \mathsf{bdd})$-S-AD	$(-, 1, \mathsf{bdd})$-S-AD	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert)$	LWE
[Wee17]	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \to \{0,1\}, \phi_f(\boldsymbol{x},\boldsymbol{z}) = (f(\boldsymbol{x})^\top\boldsymbol{z} \overset{?}{=} 0)$	1	$(-, \mathsf{poly}, \mathsf{poly})$-S-AD	$(-, 1, \mathsf{poly})$-S-AD	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert)$	k-Lin
[DOT18]	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \to \{0,1\}, \phi_f(\boldsymbol{x},\boldsymbol{z}) = (f(\boldsymbol{x})^\top\boldsymbol{z} \overset{?}{=} 0)$	1	$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-AD	$(\mathsf{poly}, \mathsf{bdd}, \mathsf{poly})$-AD	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert)$	SXDLIN
[ACGU20]	$\phi_{(f\in(\mathsf{NC}^1)^{(n)},\boldsymbol{y}\in\mathbb{Z}_p^{n'})} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \to \mathbb{Z}_p, \phi_{(f,\boldsymbol{y})}(\boldsymbol{x},\boldsymbol{z}) = (f(\boldsymbol{x}) \overset{?}{=} 0)\cdot\boldsymbol{z}^\top\boldsymbol{y}$	1	$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-AD	$\times$	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert)$	SXDH
[AGW20]	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \to \mathbb{Z}_p, \phi_f(\boldsymbol{x},\boldsymbol{z}) = f(\boldsymbol{x})^\top\boldsymbol{z}$	unbounded	$(-, \mathsf{poly}, \mathsf{poly})$-AD	$(-, \mathsf{bdd}, \mathsf{poly})$-S-AD	$O(\lvert\boldsymbol{z}\rvert)$	k-Lin
[Wee20]	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n_1n_2)}} : \mathbb{Z}_p^n \times (\mathbb{Z}_p^{n_1} \times \mathbb{Z}_p^{n_2}) \to \mathbb{Z}_p, \phi_f(\boldsymbol{x},(\boldsymbol{z}_1,\boldsymbol{z}_2)) = f(\boldsymbol{x})^\top(\boldsymbol{z}_1 \otimes \boldsymbol{z}_2)$	1	$(-, \mathsf{poly}, \mathsf{poly})$-S-AD	$(-, \mathsf{bdd}, \mathsf{poly})$-S-AD	$O(\lvert\boldsymbol{z}_1\rvert + \lvert\boldsymbol{z}_2\rvert)$	bilateral k-Lin and k-Lin
This Work	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \to \mathbb{Z}_p, \phi_f(\boldsymbol{x},\boldsymbol{z}) = f(\boldsymbol{x})^\top\boldsymbol{z}$	1	$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-AD	$(\mathsf{poly}, \mathsf{bdd}, \mathsf{poly})$-AD	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert)$	k-Lin
This Work	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \to \mathbb{Z}_p, \phi_f(\boldsymbol{x},\boldsymbol{z}) = f(\boldsymbol{x})^\top\boldsymbol{z}$	unbounded	$(\mathsf{bdd}, \mathsf{poly}, \mathsf{poly})$-AD	$(\mathsf{bdd}, \mathsf{bdd}, \mathsf{poly})$-AD	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert + B)$	bilateral k-Lin and k-Lin

The notations used in this table have the following meanings:
- GC: General polynomial-size circuits, ABP: Arithmetic branching programs
- IND: Indistinguishability-based security, SIM: Simulation-based security
- AD: Adaptive security, S-AD: Semi-adaptive security, Sel: Selective security
- poly: Arbitrary polynomial in the security parameter, bdd: A-priori bounded by the public parameters
- $\lvert x\rvert$: Size of x, B: A bound on the number of pre-ciphertext decryption key queries

In this table, (U, V, W) signifies that the adversary is allowed to make V number of ciphertext queries in the relevant security experiment, while U and W number of decryption key queries in the pre- and post-ciphertext phases respectively.

proven adaptively simulation secure against one ciphertext query, an a priori-bounded number of pre-ciphertext secret key queries, and an arbitrary polynomial number of post-ciphertext secret key queries. We emphasize that our idea of achieving one-slot FE scheme can be employed to build a public-key one-slot extFE scheme that is proven secure against a single ciphertext query and an arbitrary polynomial number of secret key queries both before and after the ciphertext query, however it would not be compatible for the transformation of [AGW20] in the context of adaptive simulation security. For the sake of completeness, in Appendix A, we provide a private-key one-slot extFE scheme secure against a single ciphertext query and a single secret key query which enables in constructing a public-key one-slot extFE scheme secure against an arbitrary polynomial number of secret key queries both before and after the ciphertext query presented in Appendix B.

2 Technical Overview

In this section, we present our main technical ideas. Let $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a bilinear group of prime order p and $[\![a]\!]_i$ denotes g_i^a for any $a \in \mathbb{Z}_p$ and $i \in \{1, 2, T\}$, which notation can also be extended in case of vectors and matrices. At the top most level of strategy, we follow [AGW20] to first design an adaptively simulation-secure one-slot FE scheme and then apply a compiler to bootstrap to an unbounded-slot scheme. For the later part, we use the same compiler as the one presented in [AGW20]. However, [AGW20] only showed that the compiler works in the context of semi-adaptive security, that is, they show that their compiler can bootstrap a semi-adaptively secure one-slot FE scheme to a semi-adaptively secure unbounded-slot scheme. In contrast, we analyze the security of the same transformation in the context of the simulation-based adaptive security framework. We observe that in order to prove the adaptive security for the compiler, the (bilateral) k-Lin/(bilateral) MDDH assumption is needed whereas for semi-adaptive security, the plain k-Lin/MDDH was sufficient [AGW20]. Moreover, we are only able to establish the simulation-based adaptive security for the transformation for settings where only a bounded number of secret-key queries are allowed prior to making the ciphertext queries.

The majority of our technical ideas in this paper lies in the design and analysis of our one-slot scheme which we describe first in this technical overview. Next, we would briefly explain the modifications to our one-slot scheme leading to our extended one-slot scheme, followed by explaining our analysis of the one-slot to unbounded-slot bootstrapping compiler from [AGW20] applied on our one-slot extended FE (extFE) scheme.

Recall that the adaptive simulation security of an FE scheme is proven by showing the indistinguishability between a real game with all the real algorithms and an ideal game where a simulator simulates all the ciphertexts and secret keys queried by the adversary. When an adversary makes a pre-ciphertext query for some function f, the simulator provides the secret key to the adversary. When the adversary makes a challenge ciphertext query for an attribute vector pair $(\boldsymbol{x}, \boldsymbol{z})$, the simulator receives the information of $\boldsymbol{x}$ but not $\boldsymbol{z}$. Instead it receives the functional values $f(\boldsymbol{x})^\top \boldsymbol{z}$ for all the pre-ciphertext secret keys. Based on this information, the simulator must simulate the challenge ciphertext. Finally, when an adversary makes a secret-key query for some function f after making a ciphertext query, the simulator receives f along with the functional value $f(\boldsymbol{x})^\top \boldsymbol{z}$ for that key and simulates the key based on this information.

2.1 Designing Adaptively Simulation Secure One-slot extFE

Abdalla et al. [AGW20] built their one-slot FE scheme for attribute-weighted sums by extending the techniques devised by Wee [Wee17] in the context of partially hiding predicate encryptions for predicates expressed as ABPs over public attributes followed by inner product evaluations over private attributes. The proof strategy of [AGW20, Wee17] is designed to achieve selective type security where during the security reduction, the challenge ciphertext is made completely random and then the secret keys are simulated using the functional value and the randomness used in the challenge ciphertext. In particular, its simulated secret key is divided into two parts — the first part is computed similar to the original key generation algorithm and is used for decrypting the honestly computed ciphertext whereas the second part contains the functional value and is used for decrypting the simulated ciphertext correctly. However, in the adaptive setting, we must embed the correct functional values for the functions associated with the pre-ciphertext secret keys into the challenge ciphertext and therefore the proof technique of [AGW20, Wee17] does not seem to extend to the adaptive setting. Datta et al. [DOT20] designed an adaptively simulation secure predicate encryption scheme for the same class of predicates as [Wee17], but their ciphertexts do not preserve compactness as they had to impose a read-once restriction on the attributes due to the usual information theoretic argument required in dual system encryption.

Overcoming the one-use restriction of the dual system proof techniques for adaptive security, Lin and Luo [LL20] developed new techniques to obtain adaptive indistinguishability secure ABE with compact ciphertexts for the class of predicates expressed as ABPs. [LL20] takes a semi-generic approach to design their ABE schemes. Their main idea is to replace the core information theoretic step of the dual system methodology with a computational step and thereby avoid the one-use restriction. Two main ingredients of [LL20] are arithmetic key garbling scheme (AKGS) which is the information theoretic component and function-hiding *slotted* inner product functional encryption (IPFE) which is the computational component. We try to adopt the techniques of [LL20] into our setting of simulation-based security for FE without the one-use restriction. However, a straight-forward adaptation of the [LL20] framework into our setting presents several challenges which we overcome with new ideas. Before describing those challenges and our ideas, we first give a high-level overview of the two primitives, namely, AKGS and function-hiding slotted IPFE.

Arithmetic Key Garbling Schemes: The notion of partial garbling scheme was proposed in [IW14] and recently it was further refined by [LL20] in the context of arithmetic computations. The refined notion is called arithmetic key garbling scheme (AKGS) which garbles a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ along with two secrets $\alpha, \beta \in \mathbb{Z}_p$ so that the evaluation with an input $\boldsymbol{x} \in \mathbb{Z}_p^n$ gives the value $\alpha f(\boldsymbol{x}) + \beta$. Note that the evaluation does not reveal any information about α and β. In particular, the AKGS has the following algorithms:

- $(\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}) \leftarrow \mathsf{Garble}(\alpha f(\boldsymbol{x}) + \beta; \boldsymbol{r})$: The garbling algorithm outputs $(m + 1)$ affine label functions $L_1, \ldots, L_{m+1}$, described by their coefficient vectors $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}$ over $\mathbb{Z}_p$, using the randomness $\boldsymbol{r} \in \mathbb{Z}_p^m$ where $(m + 1)$ denotes the size of the function f.
- $\gamma \leftarrow \mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1})$: The linear evaluation procedure recovers $\gamma = \alpha f(\boldsymbol{x}) + \beta$ using the input $\boldsymbol{x}$ and the label function values $\ell_j = L_j(\boldsymbol{x}) = \boldsymbol{\ell}_j \cdot (1, \boldsymbol{x}) \in \mathbb{Z}_p$.

AKGS is a partial garbling process as it only hides α, β which is captured by the usual simulation security given by [IW14]. The simulator produces simulated labels $(\widehat{\ell}_1, \ldots, \widehat{\ell}_{m+1}) \leftarrow$

$\mathsf{SimGarble}(f, \boldsymbol{x}, \alpha f(\boldsymbol{x}) + \beta)$ which is the same distribution as the actual label function values evaluated at input $\boldsymbol{x}$. Additionally, [LL20] defines *piecewise* security of AKGS that consists of two structural properties, namely *reverse sampleability* and *marginal randomness*. The partial garbling scheme for ABPs of Ishai and Wee [IW14] directly implies a piecewise secure AKGS for ABPs. (See Section 3.6 for further details.)

Function-Hiding Slotted IPFE: A private-key function-hiding inner product functional encryption (IPFE) scheme based on a bilinear group $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ generates secret keys IPFE.SK for vectors $[\![\boldsymbol{v}]\!]_2 \in \mathbb{G}_2^n$ and produces ciphertexts IPFE.CT for vectors $[\![\boldsymbol{u}]\!]_1 \in \mathbb{G}_1^n$ using the master secret key of the system. Both the key generation and encryption algorithm perform linear operations in the exponent of the source groups $\mathbb{G}_2, \mathbb{G}_1$ respectively. The decryption recovers the inner product $[\![\boldsymbol{v} \cdot \boldsymbol{u}]\!]_T \in \mathbb{G}_T$ in the exponent of the target group. The sizes of the secret keys, IPFE.SK, and ciphertexts, IPFE.CT, in such a system grow linearly with the sizes of the vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ respectively. Roughly, the function-hiding security of an IPFE ensures that no information about the vectors $\boldsymbol{v}, \boldsymbol{u}$ is revealed from IPFE.SK and IPFE.CT except the inner product value $\boldsymbol{v} \cdot \boldsymbol{u}$ which is trivially extracted using the decryption algorithm. A slotted version of IPFE introduced in [LV16, LL20] is a hybrid between a secret-key function-hiding IPFE and a public-key IPFE. The index set of the vectors $\boldsymbol{u}$ is divided into two subsets: public slots S_{pub} and private slot S_{priv} so that the vector $\boldsymbol{u}$ is written as $\boldsymbol{u} = (\boldsymbol{u}_{\mathsf{pub}} \parallel \boldsymbol{u}_{\mathsf{priv}})$. With addition to the usual (secret-key) encryption algorithm, the slotted IPFE has another encryption algorithm that uses the master public key of the system to encrypt the public slots of $\boldsymbol{u}$, i.e. vectors with $\boldsymbol{u}_{\mathsf{priv}} = \boldsymbol{0}$. The slotted IPFE preserves the function-hiding security with respect to the private slots only as anyone can encrypt arbitrary vectors into the public slots.

Challenges with Adapting the Framework of [LL20] and Our Ideas. We now briefly explain at a high level, the main challenges in adapting the [LL20] technique into our setting and our ideas to overcome those challenges.

1. To handle the pre-challenge secret-key queries, [LL20] formulates new properties of AKGS such as *reverse sampling* and *marginal randomness*. Using such structural properties of AKGS, their main motivation was to reversely sample the first garbling label using the challenge attribute so that it can be shifted into the ciphertext component and make the remaining labels uniformly random. This procedure works fine for arguing zero advantage for the adversary at the end of the hybrid sequence in case of ABE as functions in the queried secret keys do not vanish on the challenge attribute and hence the challenge ciphertext can never be decrypted using such secret keys available to the adversary such that the value $\alpha f(\boldsymbol{x}) + \beta$ becomes completely random. But, FE permits the adversary to have secret keys that decrypts the challenge ciphertext, that is, we cannot afford to have $f_t(\boldsymbol{x})\boldsymbol{z}[t] + \beta_t$ completely random. In order to handle this, we carefully integrate the techniques of *pre-image sampleability* [O'N10, DOT20] with the reverse sampling and marginal randomness properties of AKGS to handle the pre-challenge queries.
2. The security proof of [LL20] implements a version of the dual system encryption methodology [Wat09, LW10, LOS$^+$10] via the function-hiding slotted IPFE. Since the ABE is only payload hiding, the usual dual system encryption technique is sufficient for achieving adaptive security where only one hidden subspace is required. More precisely, the secret keys are made of two slots, out of which the first public slot contains the honestly computed components which may

be used to decrypt any honestly computed ciphertext and the other hidden slot is used to embed its interaction with the challenge ciphertext. This dual system encryption technique has been used in several prior works [Wat09, LW10, LOS+10, OT10, OT12b, OT13, DOT20, LL20]. Here, a single hidden slot is enough to handle the interaction between all ciphertext and secret-key queries since by the game restrictions, no secret key queried by the adversary can decrypt the challenge ciphertext and thus their interactions with the challenge ciphertext always result in random outputs. For our application, a portion of the attribute must be kept hidden from an adversary in the context of FE, who is allowed to have polynomially many secret keys that successfully decrypts the challenge ciphertext. The usual dual system encryption is not sufficient for our purpose. We need three hidden subspaces for our security reduction. The first hidden subspace of the challenge ciphertext is kept for handling the interactions with the post-ciphertext secret keys. The second hidden subspace is required to place the dummy vector (obtained from pre-image sampleability) which helps in simulating the interactions between the challenge ciphertext and the pre-ciphertext secret keys. The last hidden subspace is used as a temporary way station to switch each pre-ciphertext secret key from interacting with the original hidden attribute of the challenge ciphertext to interacting with the dummy attribute sampled using the pre-image sampleability. We extend the framework of [LL20] to implement a three-slot dual system encryption procedure for building our one-slot FE scheme.

Our One-Slot FE. We aim to design our decryption algorithm such that given a secret key for a weight function ABP $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ with coordinate functions $f_1, \ldots, f_{n'} : \mathbb{Z}_p^n \to \mathbb{Z}_p$ and an encryption of an attribute vector pair $(\boldsymbol{x}, \boldsymbol{z}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$, the decryption algorithm would first recover the value for each coordinate $\boldsymbol{z}[t]f_t(\boldsymbol{x})$ masked with a random scalar β_t, that is, $\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \beta_t$ and then sum over all these values to obtain the desired functional value (we take the scalars $\{\beta_t\}_{t\in[n']}$ such that $\sum_{t=1}^{n'} \beta_t = 0 \mod p$). Thus we want our key generation algorithm to use AKGS to garble the functions $\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \beta_t$. Note that here, β_t is a constant but $\boldsymbol{z}[t]$ is a variable. While doing this garbling, we also want the label functions to involve either only the variables $\boldsymbol{x}$ or the variable $\boldsymbol{z}[t]$. This is because, in the construction we need to handle $\boldsymbol{x}$ and $\boldsymbol{z}[t]$ separately since $\boldsymbol{x}$ is public whereas $\boldsymbol{z}[t]$ is private. This is unlike [LL20] which garbles $\alpha f(\boldsymbol{x}) + \beta$ where both α, β are known constants and only $\boldsymbol{x}$ is a variable. To solve this issue, we garble an extended ABP where we extend the original ABP f_t by adding a new sink node and connecting the original sink node of f_t to this new sink node with a directed edge labeled with the variable $\boldsymbol{z}[t]$.

We also make use of a particular instantiation of AKGS given by [IW14] where we observe that the first m coefficient vectors $\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}$ are independent of $\boldsymbol{z}[t]$ and the last coefficient vector $\boldsymbol{\ell}_{m+1,t}$ involves only the variable $\boldsymbol{z}[t]$. In the setup phase, two pairs of IPFE keys $(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK})$ and $(\widehat{\mathsf{IPFE.MSK}}, \widehat{\mathsf{IPFE.MPK}})$ for a slotted IPFE are generated for appropriate public and private index sets. The first instance of IPFE is used to handle the public attributes $\boldsymbol{x}$, whereas the second instance for the private attributes $\boldsymbol{z}$. Let $f = (f_1, \ldots, f_{n'}) : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ be a given weight function ABP such that $f_t : \mathbb{Z}_p^n \to \mathbb{Z}_p$ is the t-th coordinate ABP of f. To produce a secret-key SK_f, we proceed as follows:

– Sample vectors $\boldsymbol{\alpha}, \boldsymbol{\beta}_t \leftarrow \mathbb{Z}_p^k$ such that $\sum_{t\in[n']} \boldsymbol{\beta}_t[\iota] = 0 \mod p \;\forall \iota \in [k]$
– Suppose we want to base the security of the proposed scheme under the MDDH_k assumption. Generate k instances of the garblings $(\boldsymbol{\ell}_{1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{m+1,t}^{(\iota)}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}[\iota]\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \boldsymbol{\beta}_t[\iota]; \boldsymbol{r}_t^{(\iota)})$ for

$\iota \in [k]$ where $\boldsymbol{r}_t^{(\iota)} \leftarrow \mathbb{Z}_p^m$. Using the instantiation of AKGS given by [IW14], we have that the $(m+1)$-th label functions $L_{m+1,t}^{(\iota)}$ take the form $L_{m+1,t}^{(\iota)}(\boldsymbol{z}[t]) = \boldsymbol{\alpha}[\iota]\boldsymbol{z}[t] - r_t^{(\iota)}[m]$ with $\boldsymbol{\alpha}[\iota]$ a constant.

– Compute the IPFE secret keys

$$\begin{aligned}
\mathsf{IPFE.SK} &= \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\alpha}, \mathbf{0}_{kn} \parallel 0, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2)\\
\mathsf{IPFE.SK}_{j,t} &= \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\ell}_{j,t}^{(1)}, \dots, \boldsymbol{\ell}_{j,t}^{(k)} \parallel 0, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2) \text{ for } j \in [m]\\
\widehat{\mathsf{IPFE.SK}}_{m+1,t} &= \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{r}_t^{(1)}[m], \dots, \boldsymbol{r}_t^{(k)}[m], \boldsymbol{\alpha} \parallel 0,0,0,0,0,0,0]\!]_2)
\end{aligned}$$

– Return $\mathsf{SK}_f = (\mathsf{IPFE.SK}, \{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$

Here, we separate public and private slots by " $\parallel$ " and $\mathbf{0}$ denotes a vector of all zero elements. Now, to produce a ciphertext CT for some attribute vectors $(\boldsymbol{x}, \boldsymbol{z})$, we use the following steps:

– Sample $\boldsymbol{s} \leftarrow \mathbb{Z}_p^k$ and use the slotted encryption of IPFE to compute the ciphertexts

$$\begin{aligned}
\mathsf{IPFE.CT} &= \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{s}, \boldsymbol{s} \otimes \boldsymbol{x}]\!]_1)\\
\widehat{\mathsf{IPFE.CT}}_t &= \mathsf{IPFE.SlotEnc}(\widehat{\mathsf{IPFE.MSK}}, [\![-\boldsymbol{s}, \boldsymbol{s} \cdot \boldsymbol{z}[t]]\!]_1) \quad \text{for all } t \in [n']
\end{aligned}$$

where $\otimes$ denotes the tensor product.

– return $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$

Decryption first uses $\mathsf{IPFE.Dec}$ to compute

$$\boldsymbol{v} \cdot \boldsymbol{u} = [\![\boldsymbol{\alpha} \cdot \boldsymbol{s}]\!]_T \tag{1}$$

$$\boldsymbol{v}_{j,t} \cdot \boldsymbol{u} = [\![\sum_\iota \boldsymbol{s}[\iota](\boldsymbol{\ell}_{j,t}^{(\iota)} \cdot (1, \boldsymbol{x}))]\!]_T = [\![\ell_{j,t}]\!]_T \quad \text{for } j \in [m], t \in [n'] \tag{2}$$

$$\boldsymbol{v}_{m+1,t} \cdot \boldsymbol{h}_t = [\![\sum_\iota \boldsymbol{s}[\iota](\boldsymbol{\alpha}[\iota]\boldsymbol{z}[t] - \boldsymbol{r}_t^{(\iota)}[m])]\!]_T = [\![\ell_{m+1,t}]\!]_T \quad \text{for } t \in [n'] \tag{3}$$

and then apply the evaluation procedure of AKGS to get

$$\mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \dots, [\![\ell_{m+1,t}]\!]_T) = [\![(\boldsymbol{\alpha} \cdot \boldsymbol{s}) \cdot \boldsymbol{z}[t] f_t(\boldsymbol{x}) + \boldsymbol{\beta}_t \cdot \boldsymbol{s}]\!]_T. \tag{4}$$

Finally, multiplying all these evaluated values and utilizing the fact $\sum_{t\in[n']} \boldsymbol{\beta}_t \cdot \boldsymbol{s} = 0$, we recover $f(\boldsymbol{x})^\top \boldsymbol{z} = \sum_{t\in[n']} \boldsymbol{z}[t] f_t(\boldsymbol{x})$.

The Simulator for Our One-Slot FE Scheme: We now describe our simulator of the adaptive game for our one-slot FE scheme. Note that the private slots on the right side of " $\parallel$ " will be used by the simulator and we program them during the security analysis. For the q-th secret-key query corresponding to a function $f_q = (f_{q,1}, \dots, f_{q,n'})$, the simulator sets public slots of all the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ for $j \in \{1, \dots, m_q + 1\}$ as in the original key generation algorithm. Instead of using the linear combination of the label vectors, the simulator uses freshly sampled garblings to set the private slots. The *pre-challenge* secret key SK_{f_q} takes the form

$$\begin{aligned}
\mathsf{IPFE.SK}_q &= \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\alpha}[\iota], \mathbf{0}_{kn} \parallel \widetilde{\alpha}_q, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2)\\
\mathsf{IPFE.SK}_{q,j,t} &= \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\ell}_{q,j,t}^{(1)}, \dots, \boldsymbol{\ell}_{q,j,t}^{(k)} \parallel \widetilde{\boldsymbol{\ell}}_{q,j,t}, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2) \quad \text{for } j \in [m_q]\\
\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} &= \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{r}_t^{(1)}[m_q], \dots, \boldsymbol{r}_t^{(k)}[m_q], \boldsymbol{\alpha} \parallel 0, 0, \widetilde{\boldsymbol{r}}_{q,t}[m_q], \widetilde{\alpha}_q, 0, 0, 0]\!]_2)
\end{aligned}$$

where $(\widetilde{\boldsymbol{\ell}}_{q,1,t}, \ldots, \widetilde{\boldsymbol{\ell}}_{q,m_q,t}) \leftarrow \mathsf{Garble}(\widetilde{\alpha}_q \boldsymbol{z}[t] f_{q,t}(\boldsymbol{x}) + \widetilde{\beta}_{q,t}; \widetilde{\boldsymbol{r}}_{q,t}), \widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ such that $\sum_{t \in [n']} \widetilde{\beta}_{q,t} = 0 \mod p$. We write $\mathbf{0}_\xi$ as a vector of length ξ with all zero elements. To simulate the *ciphertext* for the challenge attribute $\boldsymbol{x}^*$, the simulator uses the set of all functional values $\mathcal{V} = \{(f_q, f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^*) : q \in [Q_{\mathsf{pre}}]\}$ to compute a dummy vector $\boldsymbol{d}$ satisfying $f_q(\boldsymbol{x}^*)^\top \boldsymbol{d} = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$ for all $q \in [Q_{\mathsf{pre}}]$. Since the inner product functionality is *pre-image sampleable* and both $f_q, \boldsymbol{x}^*$ are known to the simulator, a dummy vector $\boldsymbol{d}$ can be efficiently computed via a polynomial time algorithm given by O'Niell [O'N10]. The simulated ciphertext becomes

$$\begin{aligned} \mathsf{IPFE.CT} &= \mathsf{IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\mathbf{0}_k, \mathbf{0}_{kn} \parallel 1, \boldsymbol{x}^*, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_1) \\ \widehat{\mathsf{IPFE.CT}}_t &= \mathsf{IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\mathbf{0}_k, \mathbf{0}_k \parallel 1, 0, -1, \boldsymbol{d}[t], 0, 0, 0]\!]_1) \end{aligned}$$

The *post-challenge* secret-key query for the q-th function $f_q = (f_{q,1}, \ldots, f_{q,n'})$ with $q > Q_{\mathsf{pre}}$ is answered using the simulator of AKGS. In particular, we choose $\beta_{q,t} \leftarrow \mathbb{Z}_p$ satisfying $\sum_{t \in [n']} \beta_{q,t} = 0 \mod p$ and compute the simulated labels as follows:

$$(\widehat{\ell}_{q,1,1}, \ldots, \widehat{\ell}_{q,m_q+1,1}) \leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, \widetilde{\alpha}_q \cdot f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \beta_{q,1}) \quad (5)$$
$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \beta_{q,t}) \quad \text{for } 1 < t \leq n' \quad (6)$$

Note that, for post-challenge secret keys the functional value $f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$ is known and hence the simulator can directly embed the value into the secret keys. The post-challenge secret key SK_{f_q} takes the form

$$\begin{aligned} \mathsf{IPFE.SK}_q &= \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\alpha}, \mathbf{0}_{kn} \parallel \widetilde{\alpha}_q, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2) \\ \mathsf{IPFE.SK}_{q,j,t} &= \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\ell}_{j,t}^{(1)}, \ldots, \boldsymbol{\ell}_{j,t}^{(k)} \parallel \ell_{q,j,t}, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2) \quad \text{for } j \in [m_q] \\ \widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} &= \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{r}_t^{(1)}[m_q], \ldots, \boldsymbol{r}_t^{(k)}[m_q], \boldsymbol{\alpha} \parallel \ell_{q,m_q+1,t}, 0, 0, 0, 0, 0, 0, 0]\!]_2) \end{aligned}$$

Security Analysis of Our One-Slot FE Scheme. To show the adaptive simulation-based security of our FE scheme, we follow a sequence of hybrid experiments to move from the real game to the ideal game with the simulated algorithms described above. The security analysis has three steps where in the first step we apply function-hiding IPFE and MDDH assumption to use freshly sampled garblings instead of linearly combined coefficient vectors. In the second step, the dummy vector $\boldsymbol{d}$ is utilized in the challenge ciphertext to handle pre-challenge secret-key queries. Here, we need to extend the framework of [LL20] to implement a three slot encryption technique using function-hiding IPFE. Finally, in the third step, we use the simulator of AKGS for simulating the post-challenge secret-key queries.

Step 1

Hybrid $\mathbf{H_0}$: This is the real adaptive simulation security game with all the real algorithms described above.

Hybrid $\mathbf{H_1}$: Indistinguishable from H_0 by the slot-mode correctness of the IPFE where we replace the $\mathsf{SlotEnc}$ algorithm with the Enc algorithm of slotted IPFE.

$$\begin{aligned} \boldsymbol{u} &= (\boldsymbol{s}, \boldsymbol{s} \otimes \boldsymbol{x}^* \parallel \boxed{0}, \boxed{\mathbf{0}_n}, \boxed{\mathbf{0}_{n'}}, \boxed{\mathbf{0}_{n'}}), \\ \boldsymbol{h}_t &= (-\boldsymbol{s}, \boldsymbol{s} \cdot \boldsymbol{z}^*[t] \parallel \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}). \end{aligned}$$

Hybrid $\mathbf{H_2}$: Indistinguishable from H_1 by function-hiding IPFE

$$\begin{aligned}
\boldsymbol{v}_q &= (\ \boldsymbol{\alpha}, \quad \mathbf{0}_{kn} \quad \| \ \boxed{\overline{\alpha}_q}, \ \mathbf{0}_n, \ \mathbf{0}_{n'}, \mathbf{0}_{n'} \ ) \\
\boldsymbol{v}_{q,j,t} &= (\ \boldsymbol{\ell}^{(1)}_{q,j,t}, \ldots, \boldsymbol{\ell}^{(k)}_{q,j,t} \ \| \ \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}}, \ \mathbf{0}_{n'}, \mathbf{0}_{n'} \ ) \quad \text{for } j \in [m_q] \\
\boldsymbol{u} &= (\ \boxed{\mathbf{0}_k}, \ \boxed{\mathbf{0}_{kn}} \ \| \ \boxed{1}, \ \boxed{\boldsymbol{x}^*}, \mathbf{0}_{n'}, \mathbf{0}_{n'} \ ) \\
\boldsymbol{v}_{q,m_q+1,t} &= (\ \boldsymbol{r}^{(1)}_t[m_q], \ldots, \boldsymbol{r}^{(k)}_t[m_q], \ \boldsymbol{\alpha} \ \| \ \boxed{\overline{\boldsymbol{r}}_{q,t}[m_q]}, \ \boxed{\overline{\alpha}_q}, \ 0, 0, 0, 0, 0 \ ) \\
\boldsymbol{h}_t &= (\ \boxed{\mathbf{0}_k}, \ \boxed{\mathbf{0}_k} \ \| \ \boxed{-1}, \ \boxed{\boldsymbol{z}^*[t]}, 0, 0, 0, 0, 0 \ )
\end{aligned}$$

where $\overline{\alpha}_q = \boldsymbol{\alpha}_q \cdot \boldsymbol{s}, \overline{\boldsymbol{\ell}}_{q,j,t} = \sum_\iota \boldsymbol{s}[\iota]\boldsymbol{\ell}^{(\iota)}_{q,j,t}$ and $\overline{\boldsymbol{r}}_{q,t}[m_q] = \sum_\iota \boldsymbol{s}[\iota]\boldsymbol{r}^{(\iota)}_{q,t}[m_q]$. Since the inner product values between the vectors remain the same, the indistinguishability follows from the function-hiding property of IPFE.

Hybrid $\mathbf{H_3}$: Indistinguishable from H_2 by MDDH assumption

$$\begin{aligned}
\boldsymbol{v}_q &= (\ \boldsymbol{\alpha}, \quad \mathbf{0}_{kn} \quad \| \ \boxed{\widetilde{\alpha}_q}, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'} \ ) \\
\boldsymbol{v}_{q,j,t} &= (\ \boldsymbol{\ell}^{(1)}_{j,t}, \ldots, \boldsymbol{\ell}^{(k)}_{j,t} \ \| \ \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}} \ \mathbf{0}_{n'}, \mathbf{0}_{n'} \ ) \quad \text{for } j \in [m_q] \\
\boldsymbol{v}_{q,m_q+1,t} &= (\boldsymbol{r}^{(1)}_t[m_q], \ldots, \boldsymbol{r}^{(k)}_t[m_q], \boldsymbol{\alpha} \ \| \ \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, \boxed{\widetilde{\alpha}_q}, 0, 0, 0, 0, 0)
\end{aligned}$$

where $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0 \mod p$ and $(\widetilde{\boldsymbol{\ell}}_{q,1,t}, \ldots, \widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\widetilde{\alpha}_q \boldsymbol{z}[t] f_{q,t}(\boldsymbol{x}) + \widetilde{\beta}_{q,t}; \widetilde{\boldsymbol{r}}_{q,t})$. The indistinguishability follows from the MDDH assumption in the source group $\mathbb{G}_2$. This completes the first step of the security analysis. In the next step, we use the dummy vector $\boldsymbol{d}$ obtained via the pre-image sampling algorithm [O'N10] and execute our *three slot* dual system encryption variant devised by extending the framework of [LL20].

Step 2

Hybrid $\mathbf{H_4}$: Indistinguishable from H_3 by function-hiding security of IPFE

$$\begin{aligned}
\boldsymbol{v}_{q,m_q+1,t} &= (\cdots \ \| \ \widetilde{\boldsymbol{r}}_{q,t}[m_q], \ \widetilde{\alpha}_q, \quad 0, \quad 0, \quad 0, \quad 0, \quad 0 \ ) \\
\boldsymbol{h}_t &= (\cdots \ \| \quad -1, \quad \boldsymbol{z}^*[t], \boxed{-1}, \boxed{\boldsymbol{d}[t]}, \boxed{-1}, \boxed{\boldsymbol{z}^*[t]}, 0 \ )
\end{aligned}$$

Hybrid $\mathbf{H_{5,q}}(\boldsymbol{q} \in [\boldsymbol{Q}_{\mathsf{pre}}])$: Indistinguishable from $\mathsf{H}_{5,(q-1)}$ via a sequence of sub-hybrids $\{\mathsf{H}_{5,q,1}, \mathsf{H}_{5,q,2}, \mathsf{H}_{5,q,3}\}$. Hybrid $\mathsf{H}_{5,0}$ coincides with H_4.

$$\begin{aligned}
\boldsymbol{v}_{q',m_q+1,t} &= (\cdots \ \| \quad 0, \quad 0, \ \boxed{\widetilde{\boldsymbol{r}}_{q',t}[m_q]}, \boxed{\widetilde{\alpha}_{q'}}, 0, 0, 0 \ ) \text{ for } q' \le q \\
\boldsymbol{v}_{q',m_q+1,t} &= (\cdots \ \| \ \widetilde{\boldsymbol{r}}_{q',t}[m_q], \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \ 0, 0, 0 \ ) \text{ for } q < q' < Q_{\mathsf{pre}}
\end{aligned}$$

Hybrid $\mathbf{H_{5,q,1}}(\boldsymbol{q} \in [\boldsymbol{Q}_{\mathsf{pre}}])$: Indistinguishable from $\mathsf{H}_{5,(q-1)}$ by function-hiding security of IPFE.

$$\begin{aligned}
\boldsymbol{v}_{q',m_q+1,t} &= (\cdots \ \| \quad 0, \quad 0, \ \widetilde{\boldsymbol{r}}_{q',t}[m_q], \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \ 0 \ ) \text{ for } q' < q \\
\boldsymbol{v}_{q,m_q+1,t} &= (\cdots \ \| \ \boxed{0}, \ \boxed{0}, \quad 0, \quad 0, \ \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, \boxed{\widetilde{\alpha}_q}, 0 \ ) \\
\boldsymbol{v}_{q',m_q+1,t} &= (\cdots \ \| \ \widetilde{\boldsymbol{r}}_{q',t}[m_q], \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \quad 0, \quad 0, \ 0 \ ) \text{ for } q < q' < Q_{\mathsf{pre}}
\end{aligned}$$

Hybrid $\mathsf{H}_{5,q,2}(q \in [Q_{\mathsf{pre}}])$: Indistinguishable from $\mathsf{H}_{5,q,1}$ by piecewise security of AKGS and function-hiding security of IPFE.

$$\boldsymbol{h}_t = (\cdots \ \| \ -1, \boldsymbol{z}^*[t], -1, \boldsymbol{d}[t], -1, \boxed{\boldsymbol{d}[t]}, 0)$$

In order to establish the indistinguishability between $\mathsf{H}_{5,q,1}$ and $\mathsf{H}_{5,q,2}$, we actually rely on a computational problem, namely the 1-key 1-ciphertext simulation security of a secret-key FE scheme for attribute-weighted sums where the single key query is made before making the challenge ciphertext query. This scheme is presented in Section 4. The security of (secret-key) one FE scheme follows from the piecewise security of AKGS and the function-hiding security of IPFE. This is the core indistinguishability step that have been information theoretic in all prior applications of the extended dual system encryption methodology for adaptive attribute-hiding security [OT12a, DOT18]. Built on the techniques of [LL20], we are able to make this core indistinguishability step computational and thus remove the one-use restriction in the context of adaptive attribute-hiding security for the first time.

Hybrid $\mathsf{H}_{5,q,3}(q \in [Q_{\mathsf{pre}}])$: Indistinguishable from $\mathsf{H}_{5,q,2}$ by function-hiding security of IPFE.

$$\begin{array}{rl}
\boldsymbol{v}_{q',m_q+1,t} = (\cdots \ \| & 0, \quad 0, \quad \widetilde{\boldsymbol{r}}_{q',t}[m_q], \quad \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \quad 0\,) \text{ for } q' < q \\
\boldsymbol{v}_{q,m_q+1,t} = (\cdots \ \| & 0, \quad 0, \quad \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, \boxed{\widetilde{\alpha}_q}, \boxed{0}, \quad \boxed{0}, \quad 0\,) \\
\boldsymbol{v}_{q',m_q+1,t} = (\cdots \ \| & \widetilde{\boldsymbol{r}}_{q',t}[m_q], \quad \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \quad 0, \quad 0, \quad 0\,) \text{ for } q < q' < Q_{\mathsf{pre}} \\
\boldsymbol{h}_t = (\cdots \ \| & -1, \quad \boldsymbol{z}^*[t], \quad -1, \quad \boldsymbol{d}[t], -1, \boxed{\boldsymbol{z}^*[t]}, 0\,)
\end{array}$$

Observe that $\mathsf{H}_{5,q,3}$ coincides with $\mathsf{H}_{5,q}$.

Hybrid H_6: Indistinguishable from $\mathsf{H}_{5,Q_{\mathsf{pre}}}$ by function-hiding security of IPFE

$$\boldsymbol{h}_t = (\cdots \ \| \ -1, \boldsymbol{z}^*[t], -1, \boldsymbol{d}[t], \boxed{0}, \boxed{0}, 0)$$

The second step of the security analysis is now over as all the pre-challenge secret keys decrypt the challenge ciphertext using dummy vector $\boldsymbol{d}$, instead of using the private attribute $\boldsymbol{z}^*$. However, we still require $\boldsymbol{z}^*$ to be present in the vector $\boldsymbol{h}_t$ for the successful decryption of the challenge ciphertext by post-challenge secret keys since we have not yet altered the forms of the post-ciphertext secret keys. The last step of the security analysis is similar to the selective game of [AGW20] where the simulator of AKGS is employed to remove $\boldsymbol{z}^*$ from the challenge ciphertext and functional values are directly plugged into the post-challenge secret keys.

Step 3

Hybrid H_7: Indistinguishable from H_6 by function-hiding security IPFE.

$$\boldsymbol{v}_{q,j,t} = (\cdots \ \| \ \boxed{\widetilde{\ell}_{q,j,t}}, \boxed{\boldsymbol{0}_n}, \boldsymbol{0}_{n'}, \boldsymbol{0}_{n'}) \quad \text{for } j \in [m_q], q > Q_{\mathsf{pre}}$$

$$\begin{array}{rl}
\boldsymbol{v}_{q,m_q+1,t} = (\cdots \ \| & \boxed{\widetilde{\ell}_{q,m_q+1,t}}, \boxed{0}, \quad 0, \quad 0, \quad 0, 0, 0\,) \quad \text{for } q > Q_{\mathsf{pre}} \\
\boldsymbol{h}_t = (\cdots \ \| & \boxed{1}, \quad \boxed{0}, -1, \boldsymbol{d}[t], 0, 0, 0\,)
\end{array}$$

Hybrid $\mathbf{H_8}$: Indistinguishable from H_7 by simulation security of AKGS.

$$\boldsymbol{v}_{q,j,t} = (\cdots \parallel \boxed{\widehat{\ell}_{q,j,t}}, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}) \quad \text{for } j \in [m_q], q > Q_{\mathsf{pre}}$$
$$\boldsymbol{v}_{q,m_q+1,t} = (\cdots \parallel \boxed{\widehat{\ell}_{q,m_q+1,t}}, 0, 0, 0, 0, 0, 0) \quad \text{for } q > Q_{\mathsf{pre}}$$

In hybrid H_7, we use the honestly computed value $\widetilde{\ell}_{q,j,t} = \widetilde{L}_{q,j,t}(\boldsymbol{x}^*)$ for $j \in [m_q]$ and $\widetilde{\ell}_{q,m_q+1,t} = \widetilde{\alpha}_q \boldsymbol{z}^*[t] - \widetilde{\boldsymbol{r}}_{q,t}[m_q]$. After that, in H_8, we utilize simulator of AKGS to simulate $\widetilde{\alpha}_q \cdot \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\beta}_{q,t}$ using $\widetilde{\ell}_{q,j,t}$.

Hybrid $\mathbf{H_9}$: Statistically close to H_8.

$$\boldsymbol{v}_{q,j,t} = (\cdots \parallel \boxed{\widehat{\ell}_{q,j,t}}, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}) \quad \text{for } j \in [m_q], q > Q_{\mathsf{pre}}$$
$$\boldsymbol{v}_{q,m_q+1,t} = (\cdots \parallel \boxed{\widehat{\ell}_{q,m_q+1,t}}, 0, 0, 0, 0, 0, 0) \quad \text{for } q > Q_{\mathsf{pre}}$$

Finally, we change the distribution of $\{\widetilde{\beta}_{q,t}\}$ to embed the value $\widetilde{\alpha}_q \cdot f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \widetilde{\beta}_{q,1}$ into $\widehat{\ell}_{q,j,1}$ and the value $\widetilde{\beta}_{q,t}$ into $\widehat{\ell}_{q,j,1}$ for $1 < t \le n'$, as in equations 5 and 6. We observe that hybrid H_9 is exactly the same as the simulator of our FE scheme.

From One-Slot FE to One-Slot extFE. We extend our one-slot FE to an extended FE scheme which is required for applying the compiler of [AGW20] to bootstrap to the unbounded-slot FE scheme. In an extFE scheme, as opposed to just taking a weight function f as input, the key generation procedure additionally takes a vector $\boldsymbol{y}$ as input. Similarly, the encryption algorithm takes an additional vector $\boldsymbol{w}$ in addition to a usual public/private vector pair $(\boldsymbol{x}, \boldsymbol{z})$ such that

$$\mathsf{SK}_{f,\boldsymbol{y}} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, (f, \boldsymbol{y})), \quad \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, (\boldsymbol{x}, \boldsymbol{z} \parallel \boldsymbol{w}))$$

The decryption procedure recovers $f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w}$ instead of $f(\boldsymbol{x})^\top \boldsymbol{z}$ like a regular one-slot scheme. The main idea is to use the linearity of the Eval algorithm of AKGS. We add an extra term $\psi_t = \nu_t \cdot (\boldsymbol{\alpha} \cdot \boldsymbol{s}) \boldsymbol{y}^\top \boldsymbol{w}$ to the first garbling value $\ell_{1,t}$ so that equation 4 becomes

$$\begin{aligned}
&\mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) \\
&= \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) \cdot [\![\psi_t]\!]_T \\
&= [\![(\boldsymbol{\alpha} \cdot \boldsymbol{s}) \cdot (f_t(\boldsymbol{x}) \boldsymbol{z}[t] + \nu_t \boldsymbol{y}^\top \boldsymbol{w}) + \boldsymbol{\beta}_t \cdot \boldsymbol{s}]\!]_T
\end{aligned}$$

where $\nu_t \leftarrow \mathbb{Z}_p$ for $t \in [n']$ be such that $\sum_{t \in [n']} \nu_t = 1 \mod p$. Therefore, multiplying all the evaluated terms and using the inner product $\boldsymbol{v} \cdot \boldsymbol{u} = \boldsymbol{\alpha} \cdot \boldsymbol{s}$, as in our one-slot FE scheme, we get $[\![f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w}]\!]_T$ using the fact that $\sum_{t \in [n']} \boldsymbol{\beta}_t \cdot \boldsymbol{s} = 0$. The security analysis is similar to our one-slot scheme.

2.2 Bootstrapping from One-Slot FE to Unbounded-Slot FE

Abdalla et al. [AGW20] devised a compiler that upgrades the one-slot FE into an unbounded-slot FE scheme where the number of slots N can be arbitrarily chosen at the time of encryption. The transformation also preserves the compactness of ciphertexts of the underlying one-slot scheme. However, their transformation actually needs a one-slot extFE scheme as defined above.

The extFE scheme of [AGW20] is built in a bilinear group $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ where ciphertexts are encoded in the group $\mathbb{G}_1$ and secret keys in the group $\mathbb{G}_2$. Interestingly, the structure of the extFE scheme of [AGW20] is such that the key generation procedure can still be run if the vector $\boldsymbol{y}$ is given in the exponent of $\mathbb{G}_2$, that is, $[\![\boldsymbol{y}]\!]_2$. The decryption, given $(\mathsf{SK}_{f,\boldsymbol{y}}, (f, [\![\boldsymbol{y}]\!]_2)), (\mathsf{CT}, \boldsymbol{x})$, recovers $[\![f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w}]\!]_T$ without leaking any additional information about the vectors $\boldsymbol{z}, \boldsymbol{w}$. Now, the unbounded-slot FE (ubdFE) scheme follows a natural masking procedure over the original one-slot scheme. More specifically, we use N extFE encryptions to obtain ciphertexts $\{\mathsf{CT}_i\}_{i\in[N]}$ where CT_i encrypts $(\boldsymbol{x}_i, \boldsymbol{z}_i \parallel \boldsymbol{w}_i)$ with $\sum_{i\in[N]} \boldsymbol{w}_i = \boldsymbol{0} \bmod p$. The decryption procedure first computes individual sum $[\![f(\boldsymbol{x}_i)^\top \boldsymbol{z}_i + \boldsymbol{y}^\top \boldsymbol{w}_i]\!]_T$ and then multiply all the sums to learn $\sum_{i\in[N]} f(\boldsymbol{x}_i)^\top \boldsymbol{z}_i$ via solving a discrete logarithm problem (using brute force). Abdalla et al. [AGW20] proved the semi-adaptive simulation-based security of the scheme assuming MDDH assumption in the source group $\mathbb{G}_2$. The main idea was to gradually shift the sum $\sum_{i\in[2,N]} f(\boldsymbol{x}_i)^\top \boldsymbol{z}_i$ from the last $(N-1)$ ciphertexts $\{\mathsf{CT}_i\}_{i\in[2,N]}$ to the first component of the ciphertext CT_1.

We apply the same high level strategy for proving the adaptive simulation security of the transformation. However, in order to do so, we face two main obstacles. First, the reduction must incorporate the decryption results of all the pre-ciphertext secret keys into the challenge ciphertext. Therefore, for all the pre-ciphertext secret key queries $(f, \boldsymbol{y})$, the reduction needs to know $[\![\boldsymbol{y}]\!]_1$ in order to simulate the challenge ciphertext and $[\![\boldsymbol{y}]\!]_2$ to simulate the key. The reason why $\boldsymbol{y}$ cannot be made available to the reduction in the clear at a high level, is that the shifting of the sums into the first ciphertext component CT_1 from a subsequent ciphertext component, say CT_η, once both CT_1 and CT_η are in the simulated form is to be done via a computational transition based on some MDDH-like assumption. In case of [AGW20], there was no pre-ciphertext key queries and hence the MDDH assumption in $\mathbb{G}_2$ was sufficient. However, in our case, the MDDH assumption only in the source group $\mathbb{G}_2$ is not sufficient to shift the sum $\sum_{i\in[2,N]} f(\boldsymbol{x}_i)^\top \boldsymbol{z}_i$ to the first ciphertext component without changing the adversary's view. Thus, we consider the bilateral MDDH (bMDDH) assumption [EHK+17, AGT20, Wee20] which allows the vector components to be available in the exponent of both the source groups $\mathbb{G}_1, \mathbb{G}_2$:

$$\{[\![\boldsymbol{y}]\!]_1, [\![\boldsymbol{y}]\!]_2, [\![\boldsymbol{y}^\top \boldsymbol{w}_i]\!]_1, [\![\boldsymbol{y}^\top \boldsymbol{w}_i]\!]_2\} \overset{c}{\approx} \{[\![\boldsymbol{y}]\!]_1, [\![\boldsymbol{y}]\!]_2, [\![\boldsymbol{u}]\!]_1, [\![\boldsymbol{u}]\!]_2\}$$

where $\boldsymbol{u}$ is uniform.

The second and more subtle obstacle arises in handling the pre-ciphertext secret key queries in the simulated game. The simulator algorithm of [AGW20] uses the simulator of the underlying one-slot scheme to simulate the ciphertext and secret key components for the first slot while it generates all other ciphertexts and secret key components normally. Now recall that in the simulated adaptive security game, the simulator embed the outputs of all the functions $\{f_q\}_{q\in[Q_{\mathsf{pre}}]}$, for which the pre-ciphertext secret key queries are made, on the challenge message $\{(\boldsymbol{x}_i, \boldsymbol{z}_i)\}_{i\in[N]}$, that is, the values $\{\sum_{i\in[N]} f_q(\boldsymbol{x}_i)^\top \boldsymbol{z}_i\}_{q\in[Q_{\mathsf{pre}}]}$ into the challenge ciphertext. Since the simulator is only generating the ciphertext and secret key components for the first slot in simulated format, we must embed the functional values $\{\sum_{i\in[N]} f_q(\boldsymbol{x}_i)^\top \boldsymbol{z}_i\}_{q\in[Q_{\mathsf{pre}}]}$ into the ciphertext component corresponding to the first slot. As for the one-slot scheme, we aim to make use of the pre-image sampling procedure for this embedding. However, this means we need to solve the system of equations $\{f_q(\boldsymbol{x}_1)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2 = \sum_{i\in[N]} f_q(\boldsymbol{x}_i)^\top \boldsymbol{z}_i\}_{q\in[Q_{\mathsf{pre}}]}$ for $(\boldsymbol{d}_1, \boldsymbol{d}_2)$. Clearly, this system of equations may not possess a solution since the right-hand side contains the sum of the functional values for all the slots while the left-hand side only involves entries corresponding to the first slot. Further, even if solution exists information theoretically, finding it out in polynomial time may not be possible given the fact that the simulator does not receive the vectors $\{\boldsymbol{y}_q\}_{q\in[Q_{\mathsf{pre}}]}$ in the clear, rather in the exponent of group elements.

In order to overcome the above problem, rather than solving the above system of equations, we instead solve the system of equations $\{f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2 + \boldsymbol{e}_q^\top \boldsymbol{d}_3 = \sum_{i\in[N]} f_q(\boldsymbol{x}_i)^\top \boldsymbol{z}_i\}_{q\in[Q_{\mathsf{pre}}]}$ for $(\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3)$, where $\boldsymbol{e}_q$ is the q-th unit vector. Note that this system of equations can be easily solved by sampling the vectors $\boldsymbol{d}_1, \boldsymbol{d}_2$ randomly and then setting the q-th entry of the vector $\boldsymbol{d}_3$ to be $\sum_{i\in[N]} f_q(\boldsymbol{x}_i)^\top \boldsymbol{z}_i - f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 - \boldsymbol{y}_q^\top \boldsymbol{d}_2$ for all $q \in [Q_{\mathsf{pre}}]$. However, this strategy would necessitate the introduction of Q_{pre} many additional subspaces into the ciphertext and secret key components for the underlying one-slot extFE scheme to accommodate for $\boldsymbol{d}_3$. (Those subspaces will contain 0s in the real scheme and only become active in the security proof). This, in turn, requires setting a bound on Q_{pre}, that is, the number of pre-ciphertext secret key queries, for both the underlying extFE scheme and the resulting ubdFE scheme.

Based on the bMDDH assumption and the above pre-image sampling strategy, we are able to show that the ubdFE scheme provides adaptive simulation-based security against a bounded number of pre-ciphertext secret key queries and an arbitrary polynomial number of post-ciphertext secret key queries if the underlying extFE scheme is adaptive simulation secure against such many secret key queries.

3 Preliminaries

In this section, we provide the necessary definitions and backgrounds that will be used in the sequence.

3.1 Notations

We denote by λ the security parameter that belongs to the set of natural number $\mathbb{N}$ and 1^λ denotes its unary representation. We use the notation $s \leftarrow S$ to indicate the fact that s is sampled uniformly at random from the finite set S. For a distribution $\mathcal{X}$, we write $x \leftarrow \mathcal{X}$ to denote that x is sampled at random according to distribution $\mathcal{X}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is said to be a negligible function of λ, if for every $c \in \mathbb{N}$ there exists a $\lambda_c \in \mathbb{N}$ such that for all $\lambda > \lambda_c$, $|\mathsf{negl}(\lambda)| < \lambda^{-c}$.

Let Expt be an interactive security experiment played between a challenger and an adversary, which always outputs a single bit. We assume that $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C}}$ is a function of λ and it is parametrized by an adversary $\mathcal{A}$ and a cryptographic protocol C. Let $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0}$ and $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}$ be two such experiment. The experiments are computationally/statistically indistinguishable if for any PPT/computationally unbounded adversary $\mathcal{A}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{C}}(\lambda) = |\Pr[1 \leftarrow \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0}(1^\lambda)] - \Pr[1 \leftarrow \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}(1^\lambda)]| < \mathsf{negl}(\lambda)$$

We write $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0} \overset{c}{\approx} \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}$ if they are *computationally indistinguishable* (or simply *indistinguishable*). Similarly, $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0} \overset{s}{\approx} \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}$ means *statistically indistinguishable* and $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},0} \equiv \mathsf{Expt}_{\mathcal{A}}^{\mathsf{C},1}$ means they are *identically* distributed.

For $n \in \mathbb{N}$, we denote $[n]$ the set $\{1, 2, \ldots, n\}$ and for $n, m \in \mathbb{N}$ with $n < m$, we denote $[n, m]$ be the set $\{n, n+1, \ldots, m\}$. We use lowercase boldface, e.g., $\boldsymbol{v}$, to denote column vectors in $\mathbb{Z}_p^n$

and uppercase boldface, e.g., $\mathbf{M}$, to denote matrices in $\mathbb{Z}_p^{n\times m}$ for $p, n, m \in \mathbb{N}$. The i-th component of a vector $\boldsymbol{v} \in \mathbb{Z}_p^n$ is written as $\boldsymbol{v}[i]$ and the (i, j)-th element of a matrix $\mathbf{M} \in \mathbb{Z}_p^{n\times m}$ is denoted by $\mathbf{M}[i, j]$. The transpose of a matrix $\mathbf{M}$ is denoted by $\mathbf{M}^\top$ such that $\mathbf{M}^\top[i, j] = \mathbf{M}[j, i]$.To write a vector of length n with all zero elements, we write $\mathbf{0}_n$ or simply $\mathbf{0}$ when the length is clear from the context. Let $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_p^n$, then the inner product between the vectors is denoted as $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}^\top \boldsymbol{v} = \sum_{i\in[n]} \boldsymbol{u}[i]\boldsymbol{v}[i] \in \mathbb{Z}_p$.

Let $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ be an affine function with coefficient vector $\mathbf{f} = (\mathbf{f}[\mathsf{const}], \mathbf{f}[\mathsf{coef}_1], \ldots, \mathbf{f}[\mathsf{coef}_n])$. Then for any $\boldsymbol{x} \in \mathbb{Z}_p^n$, we have $f(\boldsymbol{x}) = \mathbf{f}[\mathsf{const}] + \sum_{i\in[n]} \mathbf{f}[\mathsf{coef}_i]\boldsymbol{x}[i] \in \mathbb{Z}_p$.

3.2 Bilinear Groups and Hardness Assumptions

We use a pairing group generator $\mathcal{G}$ that takes as input 1^λ and outputs a tuple $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order $p = p(\lambda)$ and g_i is a generator of the group $\mathbb{G}_i$ for $i \in \{1, 2\}$. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfies the following properties:

– *bilinear*: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
– *non-degenerate*: $e(g_1, g_2)$ generates $\mathbb{G}_T$.

The group operations in $\mathbb{G}_i$ for $i \in \{1, 2, T\}$ and the map e are efficiently computable in deterministic polynomial time in the security parameter λ. For a matrix $\mathbf{A}$ and each $i \in \{1, 2, T\}$, we use the notation $[\![\mathbf{A}]\!]_i$ to denote $g_i^{\mathbf{A}}$ where the exponentiation is element-wise. The group operation is written additively while using the bracket notation, i.e. $[\![\mathbf{A}_i + \mathbf{B}]\!]_i = [\![\mathbf{A}]\!] + [\![\mathbf{B}]\!]_i$ for matrices $\mathbf{A}$ and $\mathbf{B}$. Observe that, given $\mathbf{A}$ and $[\![\mathbf{B}]\!]_i$, we can efficiently compute $[\![\mathbf{AB}]\!]_i = \mathbf{A} \cdot [\![\mathbf{B}]\!]_i$. We write the pairing operation multiplicatively, i.e. $e([\![\mathbf{A}]\!]_1, [\![\mathbf{B}]\!]_2) = [\![\mathbf{A}]\!]_1[\![\mathbf{B}]\!]_2 = [\![\mathbf{AB}]\!]_T$.

Assumption 1 (Matrix Diffie-Hellman Assumption) *Let $k = k(\lambda), \ell = \ell(\lambda), q = q(\lambda)$ be positive integers. We say that the $\mathsf{MDDH}^q_{k,\ell}$ assumption holds in $\mathbb{G}_i$ $(i \in \{1, 2, T\})$ if for all* PPT *adversary $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MDDH}^q_{k,\ell}}(\lambda) = |\Pr[1 \leftarrow \mathcal{A}(\mathsf{G}, [\![\mathbf{A}]\!]_i, [\![\mathbf{S}^\top\mathbf{A}]\!]_i)] - \Pr[1 \leftarrow \mathcal{A}(\mathsf{G}, [\![\mathbf{A}]\!]_i, [\![\mathbf{U}]\!]_i)]| < \mathsf{negl}(\lambda)$$

where $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda), \mathbf{A} \leftarrow \mathbb{Z}_p^{k\times\ell}, \mathbf{S} \leftarrow \mathbb{Z}_p^{k\times q}$ *and* $\mathbf{U} \leftarrow \mathbb{Z}_p^{q\times\ell}$.

Escala et al. [EHK+17] showed that the k-Linear (k-Lin) assumption [BBS04] implies $\mathsf{MDDH}^1_{k,k+1}$ and $\mathsf{MDDH}^1_{k,k+1}$ implies $\mathsf{MDDH}^q_{k,\ell}$ for all $k, q \in \mathbb{N}$ and $\ell > k$ with a tight security reduction. Henceforth, we will use MDDH_k to denote $\mathsf{MDDH}^1_{k,k+1}$.
We consider bilateral $\mathsf{MDDH}^q_{k,\ell}$ assumption which is a strengthening of the $\mathsf{MDDH}^q_{k,\ell}$ assumption. The bilateral $\mathsf{MDDH}^q_{k,\ell}$ assumption is defined as follows.

Assumption 2 (Bilateral Matrix Diffie-Hellman Assumption) *Let $k = k(\lambda), \ell = \ell(\lambda), q = q(\lambda)$ be positive integers. We say that the bilateral $\mathsf{MDDH}^q_{k,\ell}$ ($\mathsf{bMDDH}^q_{k,\ell}$) assumption holds if for all* PPT *adversary $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{bMDDH}^q_{k,\ell}}(\lambda) = |\Pr[1 \leftarrow \mathcal{A}(\mathsf{G}, \{[\![\mathbf{A}]\!]_i, [\![\mathbf{S}^\top\mathbf{A}]\!]_i\}_{i\in\{1,2\}})] - \Pr[1 \leftarrow \mathcal{A}(\mathsf{G}, \{[\![\mathbf{A}]\!]_i, [\![\mathbf{U}]\!]_i\}_{i\in\{1,2\}})]| < \mathsf{negl}(\lambda)$$

where $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda), \mathbf{A} \leftarrow \mathbb{Z}_p^{k\times\ell}, \mathbf{S} \leftarrow \mathbb{Z}_p^{k\times q}$ *and* $\mathbf{U} \leftarrow \mathbb{Z}_p^{q\times\ell}$.

We consider the following lemma which will be useful in our security proof. This lemma is a direct adaptation of Lemma 1 of [AGW20] in context of bMDDH.

Lemma 1 *For any $Q \in \mathbb{N}$ and $\{\mu_q\}_{q\in[Q]} \in \mathbb{Z}_p$, we have*

$$\begin{aligned}
&\{[\![-\boldsymbol{w}^\top \boldsymbol{y}_q]\!]_1, [\![-\boldsymbol{w}^\top \boldsymbol{y}_q]\!]_2, && [\![\mu_q + \boldsymbol{w}^\top \boldsymbol{y}_q]\!]_1, [\![\mu_q + \boldsymbol{w}^\top \boldsymbol{y}_q]\!]_2, && [\![\boldsymbol{y}_q]\!]_1, [\![\boldsymbol{y}_q]\!]_2\}_{q\in[Q]},\\
\overset{c}{\approx}&\{[\![\mu_q - \boldsymbol{w}^\top \boldsymbol{y}_q]\!]_1, [\![\mu_q - \boldsymbol{w}^\top \boldsymbol{y}_q]\!]_2, && [\![\boldsymbol{w}^\top \boldsymbol{y}_q]\!]_1, [\![\boldsymbol{w}^\top \boldsymbol{y}_q]\!]_2, && [\![\boldsymbol{y}_q]\!]_1, [\![\boldsymbol{y}_q]\!]_2\}_{q\in[Q]}
\end{aligned}$$

where $\boldsymbol{w}, \{\boldsymbol{y}_q\}_{q\in[Q]} \leftarrow \mathbb{Z}_p^k$, under the $\mathsf{bMDDH}^1_{k,Q}$ assumption. More specifically, for any adversary $\mathcal{A}$ distinguishing the two distributions, there exists an adversary $\mathcal{B}$ against the $\mathsf{bMDDH}^1_{k,Q}$ problem such that the distinguishing advantage of $\mathcal{A}$ is bounded by $2 \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{bMDDH}^1_{k,Q}}(\lambda)$.

Proof. The lemma can be proved by three simple hybrids as follows:

$$\begin{aligned}
&\{[\![-\boldsymbol{w}^\top \boldsymbol{y}_q]\!]_1, [\![-\boldsymbol{w}^\top \boldsymbol{y}_q]\!]_2, && [\![\mu_q + \boldsymbol{w}^\top \boldsymbol{y}_q]\!]_1, [\![\mu_q + \boldsymbol{w}^\top \boldsymbol{y}_q]\!]_2, && [\![\boldsymbol{y}_q]\!]_1, [\![\boldsymbol{y}_q]\!]_2\}_{q\in[Q]}\\
\overset{c}{\approx}&\{[\![-\boxed{u_q}]\!]_1, [\![-\boxed{u_q}]\!]_2, && [\![\mu_q + \boxed{u_q}]\!]_1, [\![\mu_q + \boxed{u_q}]\!]_2, && [\![\boldsymbol{y}_q]\!]_1, [\![\boldsymbol{y}_q]\!]_2\}_{q\in[Q]}\\
\overset{s}{\approx}&\{\boxed{[\![\mu_q - u_q]\!]_1, [\![\mu_q - u_q]\!]_2}, && \boxed{[\![u_q]\!]_1, [\![u_q]\!]_2}, && [\![\boldsymbol{y}_q]\!]_1, [\![\boldsymbol{y}_q]\!]_2\}_{q\in[Q]}\\
\overset{c}{\approx}&\{[\![\mu_q - \boxed{\boldsymbol{w}^\top \boldsymbol{y}_q}]\!]_1, [\![\mu_q - \boxed{\boldsymbol{w}^\top \boldsymbol{y}_q}]\!]_2, && \boxed{[\![\boldsymbol{w}^\top \boldsymbol{y}_q]\!]_1, [\![\boldsymbol{w}^\top \boldsymbol{y}_q]\!]_2}, && [\![\boldsymbol{y}_q]\!]_1, [\![\boldsymbol{y}_q]\!]_2\}_{q\in[Q]}
\end{aligned}$$

where u_q is uniform over $\mathbb{Z}_p$. The first computational indistinguishability holds due to $\mathsf{bMDDH}^1_{k,Q}$ assumption. The second indistinguishability is statistical as we have changed the variable u_q by $u_q - \mu_q$ where both μ_q, u_q are uniform over $\mathbb{Z}_p$. Finally, the last computational indistinguishability holds again due to $\mathsf{bMDDH}^1_{k,Q}$ assumption. $\square$

3.3 Arithmetic Branching Program

Arithmetic Branching Program (ABP) is a computational model [Nis91] that can be used to model boolean formula, boolean branching program or arithmetic formula through a linear time reduction with a constant blow-up in their respective sizes. In this work, we consider ABP over $\mathbb{Z}_p$.

Definition 1 (Arithmetic Branching Program) An arithmetic branching program (ABP) over $\mathbb{Z}_p^n$ is a weighted directed acyclic graph (V, E, ϕ, v_0, v_1), where V is the set of all vertices, E is the set of all edges, $\phi : E \to (\mathbb{Z}_p^n \to \mathbb{Z}_p)$ specifies an affine weight function for each edge, and $v_0, v_1 \in V$ are two distinguished vertices (called the source and the sink respectively). The in-degree of v_0 and the out-degree of v_1 are 0. It computes a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ given by

$$f(\boldsymbol{x}) = \sum_{P\in\mathfrak{P}} \prod_{e\in P} \phi(e)(\boldsymbol{x})$$

where $\mathfrak{P}$ is the set of all v_0-v_1 path and $e \in P$ denotes an edge in the path $P \in \mathfrak{P}$. The size of the ABP is $|V|$, the number of vertices.

We denote by $\mathcal{F}^{(n)}_{\mathsf{ABP}}$ the class of ABPs over $\mathbb{Z}_p^n$:

$$\mathcal{F}^{(n)}_{\mathsf{ABP}} = \{f | f \text{ is an ABP over } \mathbb{Z}_p^n \text{ for some prime } p \text{ and positive integer } n\}$$

The class of ABP can be extended in a coordinate-wise manner to a ABPs $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$. More precisely, an ABP $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ has all its weight functions $\phi = (\phi_1, \ldots, \phi_{n'}) : E \to (\mathbb{Z}_p^n \to \mathbb{Z}_p^{n'})$

with each coordinate function ϕ_t for $t \in [n']$ of ϕ being an affine function in $\boldsymbol{x}$ having scalar constants and coefficients. Therefore, such a function f can be viewed as $f = (f_1, \ldots, f_{n'})$ with each coordinate function $f_t : \mathbb{Z}_p^n \to \mathbb{Z}_p$ being an ABP that has the same underlying graph structure as that of f and having $\phi_t : E \to (\mathbb{Z}_p^n \to \mathbb{Z}_p)$ as the weight functions. The class of all such functions is given by

$$\mathcal{F}_{\mathsf{ABP}}^{(n,n')} = \{f = (f_1, \ldots, f_{n'}) : \mathbb{Z}_p^n \to \mathbb{Z}_p^{(n')} | f_t \in \mathcal{F}_{\mathsf{ABP}}^{(n)} \text{ for } t \in [n']\}$$

Thus $\mathcal{F}_{\mathsf{ABP}}^{(n)}$ can alternatively be viewed as $\mathcal{F}_{\mathsf{ABP}}^{(n,1)}$.

Lemma 2 *[IK20] Let $f = (V, E, \phi, v_0, v_1) \in \mathcal{F}_{\mathsf{ABP}}^{(n,1)}$ be an ABP of size m and $v_0, v_2, \ldots, v_{m-1}, v_1$ be stored topologically. Let $\mathbf{M}$ be a square matrix of order $(m-1)$ defined by*

$$\mathbf{M}[i+1, j] = \begin{cases} 0, & i > j; \\ -1, & i = j; \\ 0, & i < j, e_{i,j} = (v_i, v_j) \notin E; \\ \phi(e_{i,j}), & i < j, e_{i,j} = (v_i, v_j) \in E. \end{cases}$$

Then the entries of $\mathbf{M}$ are affine in $\boldsymbol{x}$ and $f(\boldsymbol{x}) = \det(\mathbf{M})$.

3.4 Functional Encryption for Attribute-Weighted Sum

We formally present the syntax of FE for attribute-weighted sum and define adaptive simulation security of the primitive. We consider the function class $\mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ and message space $\mathcal{M} = (\mathbb{Z}_p^n \times \mathbb{Z}_p^{n'})^*$.

Definition 2 (The Attribute-Weighted Sum Functionality) For any $n, n' \in \mathbb{N}$, the class of attribute-weighted sum functionalities is defined as

$$\left\{(\boldsymbol{x} \in \mathbb{Z}_p^n, \boldsymbol{z} \in \mathbb{Z}_p^{n'}) \mapsto f(\boldsymbol{x})^\top \boldsymbol{z} = \sum_{t \in [n']} f_t(\boldsymbol{x})\boldsymbol{z}[t] \mid f = (f_1, \ldots, f_{n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}\right\}$$

Definition 3 (Functional Encryption for Attribute-Weighted Sum) An unbounded-slot FE for attribute-weighted sum associated to the function class $\mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ and the message space $\mathcal{M}$ consists of four PPT algorithms defined as follows:

Setup$(1^\lambda, 1^n, 1^{n'})$: The setup algorithm takes as input a security parameter λ along with two positive integers n, n' representing the lengths of message vectors. It outputs the master secret-key MSK and the master public-key MPK.

KeyGen(MSK, f): The key generation algorithm takes as input MSK and a function $f \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$. It outputs a secret-key SK_f and make f available publicly.

Enc(MPK, $(\boldsymbol{x}_i, \boldsymbol{z}_i)_{i\in[N]}$): The encryption algorithm takes as input MPK and a message $(\boldsymbol{x}_i, \boldsymbol{z}_i)_{i\in[N]} \in (\mathbb{Z}_p^n \times \mathbb{Z}_p^{n'})^*$. It outputs a ciphertext CT and make $(\boldsymbol{x}_i)_{i\in[N]}$ available publicly.

Dec((SK_f, f), (CT, $(\boldsymbol{x}_i)_{i\in[N]}$)): The decryption algorithm takes as input SK_f and CT along with f and $(\boldsymbol{x}_i)_{i\in[N]}$. It outputs a value in $\mathbb{Z}_p$.

Correctness: The unbounded-slot FE for attribute-weighted sum is said to be correct if for all $(\boldsymbol{x}_i, \boldsymbol{z}_i)_{i\in[N]} \in (\mathbb{Z}_p^n \times \mathbb{Z}_p^{n'})^*$ and $f \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, we get

$$\Pr\left[\mathsf{Dec}((\mathsf{SK}_f, f), (\mathsf{CT}, (\boldsymbol{x}_i)_{i\in[N]})) = \sum_{i\in[N]} f(\boldsymbol{x}_i)^\top \boldsymbol{z}_i : \begin{array}{l} (\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^{n'}), \\ \mathsf{SK}_f \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, (\boldsymbol{x}_i, \boldsymbol{z}_i)_{i\in[N]}) \end{array}\right] = 1$$

We consider adaptively simulation-based security of FE for attribute-weighted sum.

Definition 4 Let (Setup, KeyGen, Enc, Dec) be an unbounded-slot FE for attribute-weighted sum for function class $\mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ and message space $\mathcal{M}$. The scheme is said to be $(Q_{\mathsf{pre}}, Q_{\mathsf{CT}}, Q_{\mathsf{post}})$-adaptively simulation secure if for any PPT adversary $\mathcal{A}$ making at most Q_{CT} ciphertext queries and $Q_{\mathsf{pre}}, Q_{\mathsf{post}}$ secret key queries before and after the ciphertext queries respectively, we have $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,ubdFE}}(1^\lambda) \overset{c}{\approx} \mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,ubdFE}}(1^\lambda)$, where the experiments are defined as follows. Also, an unbounded-slot FE for attribute-weighted sums is said to be $(\mathsf{poly}, Q_{\mathsf{CT}}, \mathsf{poly})$-adaptively simulation secure if it is $(Q_{\mathsf{pre}}, Q_{\mathsf{CT}}, Q_{\mathsf{post}})$-adaptively simulation secure as well as Q_{pre} and Q_{post} are unbounded polynomials in the security parameter λ.

```
$\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,ubdFE}}(1^\lambda)$
1. $1^N \leftarrow \mathcal{A}(1^\lambda)$;
2. $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$;
3. $((\boldsymbol{x}_i^*, \boldsymbol{z}_i^*)_{i\in[N]}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}}(\mathsf{MPK})$;
4. $\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{MPK}, (\boldsymbol{x}_i^*, \boldsymbol{z}_i^*)_{i\in[N]})$;
5. return $\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}}(\mathsf{MPK}, \mathsf{CT}^*)$

$\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,ubdFE}}(1^\lambda)$
1. $1^N \leftarrow \mathcal{A}(1^\lambda)$;
2. $(\mathsf{MSK}^*, \mathsf{MPK}) \leftarrow \mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'}, 1^N)$;
3. $((\boldsymbol{x}_i^*, \boldsymbol{z}_i^*)_{i\in[N]}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}_0^*(\mathsf{MSK}^*,\cdot)}}(\mathsf{MPK})$
4. $\mathsf{CT}^* \leftarrow \mathsf{Enc}^*(\mathsf{MPK}, \mathsf{MSK}^*, (\boldsymbol{x}_i^*)_{i\in[N]}, \mathcal{V})$;
5. return $\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}_1^*(\mathsf{MSK}^*,(\boldsymbol{x}_i^*)_{i\in[N]},\cdot,\cdot)}}(\mathsf{MPK}, \mathsf{CT}^*)$

$\mathcal{O}_{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}$
1. input: $f$
2. output: $\mathsf{SK}_f$

$\mathcal{O}_{\mathsf{KeyGen}_0^*(\mathsf{MSK}^*,\cdot)}$
1. input: $f_q$ for $q \in [Q_{\mathsf{pre}}]$
2. output: $\mathsf{SK}_{f_q}^*$

$\mathsf{Enc}^*(\mathsf{MPK}, \mathsf{MSK}^*, (\boldsymbol{x}_i^*)_{i\in[N]}, \cdot)$
1. input:
   $\mathcal{V} = \{((f_q, \mathsf{SK}_{f_q}), \sum_{i\in[N]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^*) : q \in [Q_{\mathsf{pre}}]\}$
2. output: $\mathsf{CT}^*$

$\mathcal{O}_{\mathsf{KeyGen}_1^*(\mathsf{MSK}^*,(\boldsymbol{x}_i^*)_{i\in[N]},\cdot,\cdot)}$
1. input: $f_q, \sum_{i\in[N]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^*$ for $q > Q_{\mathsf{pre}}$
2. output: $\mathsf{SK}_{f_q}^*$
```

3.5 Function-Hiding Slotted Inner Product Functional Encryption

A slotted inner product functional encryption (*slotted* IPFE), as defined by Lin and Luo [LL20], is a hybrid variant of secret-key and public-key IPFE. More specifically, the index set S of the vectors is partitioned into two sets S_{pub} containing public slots and S_{priv} containing the private slots. While computing secret-keys, the slotted IPFE encodes elements of the vector in public/private slots using the master secret-key, similar to the case of secret-key IPFE. However, the encryption procedure is only allowed to encode vector elements in the public slots using master public-key as is the case for public-key IPFE. Lin and Luo [LL20] demonstrated that slotted IPFE lets us use the dual system encryption techniques [Wat09, LW10, LOS+10] during the security analysis of the cryptographic constructions built from it.Following Lin and Luo [LL20] we consider the definition of slotted IPFE with respect to some pairing group , that is, all the vectors and inner products in the scheme are encoded in the exponent of the underlying pairing group.

We present the formal notion of slotted IPFE almost verbatim from [LL20].

Definition 5 (Slotted Inner Product Functional Encryption, [LL20]) Let $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a tuple of pairing groups of prime order p. A slotted inner product functional encryption (IPFE) scheme based on G consists of 5 efficient algorithms:

$\mathsf{IPFE.Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}})$: The setup algorithm takes as input a security parameter λ and two disjoint index sets, the public slot S_{pub} and the private slot S_{priv}. It outputs the master secret-key $\mathsf{IPFE.MSK}$ and the master public-key $\mathsf{IPFE.MPK}$. Let $S = S_{\mathsf{pub}} \cup S_{\mathsf{priv}}$ be the whole index set and $|S|, |S_{\mathsf{pub}}|, |S_{\mathsf{priv}}|$ denote the number of indices in S, S_{pub} and S_{priv} respectively.

$\mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}]\!]_2)$: The key generation algorithm takes as input $\mathsf{IPFE.MSK}$ and a vector $[\![\boldsymbol{v}]\!]_2 \in \mathbb{G}_2^{|S|}$. It outputs a secret-key $\mathsf{IPFE.SK}$ for $\boldsymbol{v} \in \mathbb{Z}_p^{|S|}$.

$\mathsf{IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)$: The encryption algorithm takes as input $\mathsf{IPFE.MSK}$ and a vector $[\![\boldsymbol{u}]\!]_1 \in \mathbb{G}_1^{|S|}$. It outputs a ciphertext $\mathsf{IPFE.CT}$ for $\boldsymbol{u} \in \mathbb{Z}_p^{|S|}$.

$\mathsf{IPFE.Dec}(\mathsf{IPFE.SK}, \mathsf{IPFE.CT})$: The decryption algorithm takes as input a secret-key $\mathsf{IPFE.SK}$ and a ciphertext $\mathsf{IPFE.CT}$. It outputs an element from $\mathbb{G}_T$.

$\mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)$: The slot encryption algorithm takes as input $\mathsf{IPFE.MPK}$ and a vector $[\![\boldsymbol{u}]\!]_1 \in \mathbb{G}_1^{|S_{\mathsf{pub}}|}$. It outputs a ciphertext $\mathsf{IPFE.CT}$ for $(\boldsymbol{u}||\mathbf{0}_{|S_{\mathsf{priv}}|}) \in \mathbb{Z}_p^{|S|}$.

Correctness: The correctness of a slotted IPFE scheme requires the following two properties.

- Decryption Correctness: The slotted IPFE is said to satisfy decryption correctness if for all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_p^{|S|}$, we have

$$\Pr\left[\mathsf{Dec}(\mathsf{IPFE.SK}, \mathsf{IPFE.CT}) = [\![\boldsymbol{v}\cdot\boldsymbol{u}]\!]_T : \begin{array}{l}(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}}),\\ \mathsf{IPFE.SK} \leftarrow \mathsf{KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}]\!]_2),\\ \mathsf{IPFE.CT} \leftarrow \mathsf{Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)\end{array}\right] = 1$$

- Slot-Mode Correctness: The slotted IPFE is said to satisfy the slot-mode correctness if for all vectors $\boldsymbol{u} \in \mathbb{Z}_p^{|S_{\mathsf{pub}}|}$, we have

$$\left\{(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}, \mathsf{IPFE.CT}) : \begin{array}{l}(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}}),\\ \mathsf{IPFE.CT} \leftarrow \mathsf{Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}||\mathbf{0}_{|S_{\mathsf{priv}}|}]\!]_1)\end{array}\right\},$$
$$\equiv \left\{(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}, \mathsf{IPFE.CT}) : \begin{array}{l}(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}}),\\ \mathsf{IPFE.CT} \leftarrow \mathsf{SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)\end{array}\right\}$$

Security: Let $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec}, \mathsf{IPFE.SlotEnc})$ be a slotted IPFE. The scheme is said to be adaptively function-hiding secure if for all PPT adversary $\mathcal{A}$, we have $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FH\text{-}IPFE}}(1^\lambda, 0) \overset{c}{\approx} \mathsf{Expt}_{\mathcal{A}}^{\mathsf{FH\text{-}IPFE}}(1^\lambda, 1)$, where the experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FH\text{-}IPFE}}(1^\lambda, b)$ for $b \in \{0, 1\}$ is defined as follows:

$\underline{\mathsf{Expt}_{\mathcal{A}}^{\mathsf{FH\text{-}IPFE}}(1^\lambda, b)}$

1. $(S_{\mathsf{pub}}, S_{\mathsf{priv}}) \leftarrow \mathcal{A}(1^\lambda)$;
2. $(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{Setup}(1^\lambda, S_{\mathsf{pub}}, S_{\mathsf{priv}})$;
3. return $\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}_b}(\cdot,\cdot), \mathcal{O}_{\mathsf{Enc}_b}(\cdot,\cdot)}(\mathsf{IPFE.MPK})$ if $\boldsymbol{v}_j^0|_{S_{\mathsf{pub}}} = \boldsymbol{v}_j^1|_{S_{\mathsf{pub}}}$ and $\boldsymbol{v}_j^0 \cdot \boldsymbol{u}_i^0 = \boldsymbol{v}_j^1 \cdot \boldsymbol{u}_i^1$ for all $\{[\![\boldsymbol{v}_j^0]\!]_2, [\![\boldsymbol{v}_j^1]\!]_2\}_j, \{[\![\boldsymbol{u}_i^0]\!]_1, [\![\boldsymbol{u}_i^1]\!]_1\}_i$ queried by $\mathcal{A}$ to $\mathcal{O}_{\mathsf{KeyGen}_b}(\cdot,\cdot)$ and $\mathcal{O}_{\mathsf{Enc}_b}(\cdot,\cdot)$ respectively.

$\underline{\mathcal{O}_{\mathsf{KeyGen}_b}(\cdot,\cdot)}$:

1. input: $[\![\boldsymbol{v}_j^0]\!]_2, [\![\boldsymbol{v}_j^1]\!]_2 \in \mathbb{G}_2^{|S|}$
2. output
   $\mathsf{IPFE.SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_j^b]\!]_2)$

$\underline{\mathcal{O}_{\mathsf{Enc}_b}(\cdot,\cdot)}$:

1. input: $[\![\boldsymbol{u}_i^0]\!]_1, [\![\boldsymbol{u}_i^1]\!]_1 \in \mathbb{G}_1^{|S|}$
2. output
   $\mathsf{IPFE.CT}_i \leftarrow \mathsf{Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}_i^b]\!]_1)$

where $\boldsymbol{v}_j|_{S_{\mathsf{pub}}}$ represents the elements of $\boldsymbol{v}_j$ sitting at the indices in S_{pub}.

Lemma 3 (**[Lin17, LL20]**) *Let* $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ *be a tuple of pairing groups of prime order* p *and* $k \geq 1$ *an integer constant. If* MDDH_k *holds in both groups* $\mathbb{G}_1, \mathbb{G}_2$*, then there is an adaptively function-hiding secure* IPFE *scheme based on* G.

3.6 Arithmetic Key Garbling Scheme

Lin and Luo [LL20] introduced arithmetic key garbling scheme (AKGS). The notion of AKGS is an information theoretic primitive, inspired by randomized encodings [AIK11] and partial garbling schemes [IW14]. It garbles a function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ (possibly of size $(m+1)$) along with two secrets $z, \beta \in \mathbb{Z}_p$ and produces affine label functions $L_1, \ldots, L_{m+1} : \mathbb{Z}_p^n \to \mathbb{Z}_p$. Given f, an input $\boldsymbol{x} \in \mathbb{Z}_p^n$ and the values $L_1(\boldsymbol{x}), \ldots, L_{m+1}(\boldsymbol{x})$, there is an efficient algorithm which computes $zf(\boldsymbol{x}) + \beta$ without revealing any information about z and β.

Definition 6 (Arithmetic Key Garbling Scheme (AKGS), [IW14, LL20]) An arithmetic garbling scheme (AKGS) for a function class $\mathcal{F} = \{f\}$, where $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, consists of two efficient algorithms:

$\mathsf{Garble}(zf(\boldsymbol{x}) + \beta)$: The garbling is a randomized algorithm that takes as input a description of the function $zf(\boldsymbol{x}) + \beta$ with $f \in \mathcal{F}$ and scalars $z, \beta \in \mathbb{Z}_p$ where $z, \boldsymbol{x}$ are treated as variables. It outputs $(m+1)$ affine functions $L_1, \ldots, L_{m+1} : \mathbb{Z}_p^{n+1} \to \mathbb{Z}_p$ which are called label functions that specifies how input is encoded as labels. Pragmatically, it outputs the coefficient vectors $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}$.

$\mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1})$: The evaluation is a deterministic algorithm that takes as input a function $f \in \mathcal{F}$, an input vector $\boldsymbol{x} \in \mathbb{Z}_p^n$ and integers $\ell_1, \ldots, \ell_{m+1} \in \mathbb{Z}_p$ which are supposed to be the values of the label functions at $(\boldsymbol{x}, z)$. It outputs a value in $\mathbb{Z}_p$.

Correctness: The AKGS is said to be correct if for all $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}, z, \beta \in \mathbb{Z}_p$ and $\boldsymbol{x} \in \mathbb{Z}_p^n$, we have

$$\Pr\left[\mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1}) = zf(\boldsymbol{x}) + \beta : \begin{array}{l} (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}) \leftarrow \mathsf{Garble}(zf(\boldsymbol{x}) + \beta), \\ \ell_j \leftarrow L_j(\boldsymbol{x}, z) \text{ for } j \in [m+1] \end{array}\right] = 1$$

The scheme have *deterministic shape*, meaning that m is determined solely by f, independent of z, β and the randomness in Garble. The number of label functions, $(m+1)$, is called the *garbling size* of f under this scheme.

Linearity: The AKGS is said to be *linear* if the following conditions hold:

– $\mathsf{Garble}(zf(\boldsymbol{x}) + \beta)$ uses a uniformly random vector $\boldsymbol{r} \leftarrow \mathbb{Z}_p^{m'}$ as its randomness, where m' is determined solely by f, independent of z, β.
– The coefficient vectors $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}$ produced by $\mathsf{Garble}(zf(\boldsymbol{x}) + \beta)$ are linear in $(z, \beta, \boldsymbol{r})$.
– $\mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1})$ is linear in $\ell_1, \ldots, \ell_{m+1}$.

Simulation-Based Security: In this work, we consider linear AKGS for our application. Now, we state the usual simulation-based security of AKGS, which is similar to the security of partial garbling scheme [IW14].

An $\mathsf{AKGS} = (\mathsf{Garble}, \mathsf{Eval})$ for a function class $\mathcal{F}$ is secure if there exists an efficient algorithm $\mathsf{SimGarble}$ such that for all $f : \mathbb{Z}_p^n \to \mathbb{Z}_p, z, \beta \in \mathbb{Z}_p$ and $\boldsymbol{x} \in \mathbb{Z}_p^n$, the following distributions are identically distributed:

$$\left\{ (\ell_1, \ldots, \ell_{m+1}) : \begin{array}{l} (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}) \leftarrow \mathsf{Garble}(zf(\boldsymbol{x}) + \beta), \\ \ell_j \leftarrow L_j(\boldsymbol{x}, z) \text{ for } j \in [m+1] \end{array} \right\},$$

$$\left\{ (\widehat{\ell}_1, \ldots, \widehat{\ell}_{m+1}) : (\widehat{\ell}_1, \ldots, \widehat{\ell}_{m+1}) \leftarrow \mathsf{SimGarble}(f, \boldsymbol{x}, zf(\boldsymbol{x}) + \beta) \right\}$$

The simulation security of AKGS is used to obtain semi-adaptive or selective security of FE for attribute-weighted sum [AGW20], however it is not sufficient for achieving adaptive security. We consider the *piecewise* security of AKGS proposed by Lin and Luo [LL20] where they used it to get adaptive security for ABE.

Definition 7 (Piecewise Security of AKGS, [LL20]) An $\mathsf{AKGS} = (\mathsf{Garble}, \mathsf{Eval})$ for a function class $\mathcal{F}$ is *piecewise* secure if the following conditions hold:

– The first label value is *reversely sampleable* from the other labels together with f and $\boldsymbol{x}$. This reconstruction is perfect even given all the other label functions. Formally, there exists an efficient algorithm $\mathsf{RevSamp}$ such that for all $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}, z, \beta \in \mathbb{Z}_p$ and $\boldsymbol{x} \in \mathbb{Z}_p^n$, the following distributions are identical:

$$\left\{ (\ell_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_{m+1}) : \begin{array}{l} (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}) \leftarrow \mathsf{Garble}(zf(\boldsymbol{x}) + \beta), \\ \ell_1 \leftarrow L_1(\boldsymbol{x}, z) \end{array} \right\},$$

$$\left\{ (\ell_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_{m+1}) : \begin{array}{l} (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}) \leftarrow \mathsf{Garble}(zf(\boldsymbol{x}) + \beta), \\ \ell_j \leftarrow L_j(\boldsymbol{x}, z) \text{ for } j \in [2, m+1], \\ \ell_1 \leftarrow \mathsf{RevSamp}(f, \boldsymbol{x}, zf(\boldsymbol{x}) + \beta, \ell_2, \ldots, \ell_{m+1}) \end{array} \right\}$$

– For the other labels, each is *marginally random* even given all the label functions after it. Formally, this means for all $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}, z, \beta \in \mathbb{Z}_p, \boldsymbol{x} \in \mathbb{Z}_p^n$ and all $j \in [2, m+1]$, the following distributions are identical:

$$\left\{ (\ell_j, \boldsymbol{\ell}_{j+1}, \ldots, \boldsymbol{\ell}_{m+1}) : \begin{array}{l} (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}) \leftarrow \mathsf{Garble}(zf(\boldsymbol{x}) + \beta), \\ \ell_j \leftarrow L_j(\boldsymbol{x}, z) \end{array} \right\},$$

$$\left\{ (\ell_j, \boldsymbol{\ell}_{j+1}, \ldots, \boldsymbol{\ell}_{m+1}) : \begin{array}{l} (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}) \leftarrow \mathsf{Garble}(zf(\boldsymbol{x}) + \beta), \\ \ell_j \leftarrow \mathbb{Z}_p \end{array} \right\}$$

Lemma 4 ([LL20]) *A piecewise secure* AKGS = (Garble, Eval) *for a function class* $\mathcal{F}$ *is also simulation secure.*

We now define special structural properties of AKGS as given in [LL20], related to the piecewise security of it.

Definition 8 (Special Piecewise Security of AKGS, [LL20]) An AKGS = (Garble, Eval) for a function class $\mathcal{F}$ is *special* piecewise secure if for any $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}, z, \beta \in \mathbb{Z}_p$ and $\boldsymbol{x} \in \mathbb{Z}_p^n$, it has the following special form:

- The first label value ℓ_1 is always non-zero, i.e., $\mathsf{Eval}(f, \boldsymbol{x}, 1, 0, \ldots, 0) \neq 0$ where we take $\ell_1 = 1$ and $\ell_j = 0$ for $1 < j \leq (m + 1)$.
- Let $\boldsymbol{r} \leftarrow \mathbb{Z}_p^{m'}$ be the randomness used in $\mathsf{Garble}(zf(\boldsymbol{x}) + \beta)$. For all $j \in [2, m + 1]$. the label function L_j produced by $\mathsf{Garble}(zf(\boldsymbol{x}) + \beta; \boldsymbol{r})$ can be written as

$$L_j(\boldsymbol{x}) = k_j\boldsymbol{r}[j - 1] + L'_j(\boldsymbol{x}; z, \beta, \boldsymbol{r}[j], \boldsymbol{r}[j + 1], \ldots, \boldsymbol{r}[m'])$$

  where $k_j \in \mathbb{Z}_p$ is a non-zero constant (not depending on $\boldsymbol{x}, z, \beta, \boldsymbol{r}$) and L'_j is an affine function of $\boldsymbol{x}$ whose coefficient vector is linear in $(z, \beta, \boldsymbol{r}[j], \boldsymbol{r}[j+1], \ldots, \boldsymbol{r}[m'])$. The component $\boldsymbol{r}[j-1]$ is called the randomizer of L_j and ℓ_j.

Lemma 5 ([LL20]) *A special piecewise secure* AKGS = (Garble, Eval) *for a function class* $\mathcal{F}$ *is also piecewise secure. The* RevSamp *algorithm (required in piecewise security) obtained for a special piecewise secure* AKGS *is linear in* $\gamma, \ell_2, \ldots, \ell_{m+1}$ *and perfectly recovers* ℓ_1 *even if the randomness of* Garble *is not uniformly sampled. More specifically, we have the following:*

$$\mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1}) = \ell_1 \mathsf{Eval}(f, \boldsymbol{x}, 1, 0, \ldots, 0) + \mathsf{Eval}(f, \boldsymbol{x}, 0, \ell_2, \ldots, \ell_{m+1}) \quad (7)$$
$$\mathsf{RevSamp}(f, \boldsymbol{x}, \gamma, \ell_2, \ldots, \ell_{m+1}) = (\mathsf{Eval}(f, \boldsymbol{x}, 1, 0, \ldots, 0))^{-1}(\gamma - \mathsf{Eval}(f, \boldsymbol{x}, 0, \ell_2, \ldots, \ell_{m+1})) \quad (8)$$

Note that, equation 7 follows from the linearity of Eval and equation 8 ensures that RevSamp perfectly computes ℓ_1 (which can be verified by equation 7 with $\gamma = zf(\boldsymbol{x}) + \beta$).

Lemma 6 ([LL20]) *A piecewise secure* AKGS = (Garble, Eval) *is also special piecewise secure after an appropriate change of variable for the randomness used by* Garble*.*

Instantiation of AKGS: ([IW14, LL20]). We now discuss an instantiation of AKGS = (Garble, Eval) for the function class $\mathcal{F} = \mathcal{F}_{\mathsf{ABP}}^{(n,1)}$ following [IW14, LL20].

Garble($zf(x) + \beta$): It takes input an ABP $f : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p \in \mathcal{F}_{\mathsf{ABP}}^{(n,1)}$ of size $(m + 1)$ and two secrets $z, \beta \in \mathbb{Z}_p$. The algorithm works as follows:

1. Using Lemma 2, it computes a matrix $\mathbf{M} \in \mathbb{Z}_p^{m \times m}$ such that $\det(\mathbf{M})$ is the output of the function f.

2. Next, it augments $\mathbf{M}$ into an $(m+1)\times(m+1)$ matrix $\mathbf{M}'$:

$$\mathbf{M}' = \left(\begin{array}{ccccc|c} * & * & \cdots & * & * & \beta \\ -1 & * & \cdots & * & * & 0 \\ & -1 & \cdots & * & * & 0 \\ & & \ddots & \vdots & \vdots & \vdots \\ 0 & & & -1 & * & 0 \\ \hline 0 & 0 & \cdots & 0 & -1 & z \end{array}\right) = \left(\begin{array}{c|c} \mathbf{M} & \boldsymbol{m}_1 \\ \hline \boldsymbol{m}_2^\top & z \end{array}\right)$$

3. It samples its randomness $\boldsymbol{r} \leftarrow \mathbb{Z}_p^m$ and sets $\mathbf{N} = \begin{pmatrix} \mathbf{I}_m & \boldsymbol{r} \\ \mathbf{0} & 1 \end{pmatrix}$.
4. Finally, it defines the label functions by computing

$$\widehat{\mathbf{M}} = \mathbf{M}'\mathbf{N} = \begin{pmatrix} \mathbf{M} & \mathbf{M}\boldsymbol{r} + \boldsymbol{m}_1 \\ \boldsymbol{m}_2^\top & \boldsymbol{m}_2^\top \boldsymbol{r} + z \end{pmatrix} = \left(\begin{array}{cccc|c} & & & & L_1(\boldsymbol{x}) \\ & & & & L_2(\boldsymbol{x}) \\ & & \mathbf{M} & & \vdots \\ & & & & L_m(\boldsymbol{x}) \\ \hline 0 & 0 & \cdots & 0 \quad -1 & L_{m+1}(z) \end{array}\right)$$

and outputs the coefficient vectors $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}$ of $L_1, \ldots, L_{m+1}$.

Remark 1 We note down some structural properties of Garble as follows:

– The label function L_j for every $j \in [m]$ is an *affine* function of the input $\boldsymbol{x}$ and L_{m+1} is an *affine* function of z. It follows from the fact that $\mathbf{M}'$ is affine in $\boldsymbol{x}, z$ and $\mathbf{N}$ is independent of $\boldsymbol{x}, z$. Hence, the last column of the product $\mathbf{M}'\mathbf{N}$, which is the label functions $L_1, \ldots, L_{m+1}$, are affine in $\boldsymbol{x}, z$.
– The output size of Garble is determined solely by the size of f (as an ABP), hence Garble has *deterministic shape*.
– Note that Garble is *linear* in $(z, \beta, \boldsymbol{r})$, i.e., the coefficient vectors $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}$ are linear in $(z, \beta, \boldsymbol{r})$. It follows from the fact that $\mathbf{M}, \boldsymbol{m}_2$ are independent of $(z, \beta, \boldsymbol{r})$, and $\boldsymbol{r}, \boldsymbol{m}_1, z$ are linear in $(z, \beta, \boldsymbol{r})$. Hence, $\mathbf{M}\boldsymbol{r}+\boldsymbol{m}_1$, which defines the label functions $L_1, \ldots, L_m$, and $\boldsymbol{m}_2^\top\boldsymbol{r}+z$, which is the label function L_{m+1}, are linear in $(z, \beta, \boldsymbol{r})$.
– The last label function L_{m+1} is in a *special form*, meaning that it is independent of $\boldsymbol{x}, \beta$ and $\boldsymbol{r}[j < m]$. In particular, it takes the form $L_m = \boldsymbol{m}_2^\top \boldsymbol{r} + z = z - \boldsymbol{r}[m]$. Thus, the elements of the coefficient vector $\boldsymbol{\ell}_{m+1}$ are all zero except the constant term, i.e., $\boldsymbol{\ell}_m[\mathsf{const}] = z - \boldsymbol{r}[m]$ and $\boldsymbol{\ell}_m[\mathsf{coef}_i] = 0$ for all $i \in [n]$.

$\mathsf{Eval}(\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_m)$: It takes input an ABP $f : \mathbb{Z}_p^n \to \mathbb{Z}_p \in \mathcal{F}_{\mathsf{ABP}}^{(n,1)}$ of size $(m+1)$, an input $\boldsymbol{x} \in \mathbb{Z}_p^n$ and $(m+1)$ labels $\ell_1, \ldots, \ell_{m+1}$. It proceeds as follows:

1. It computes the matrix $\mathbf{M}$ using Lemma 2 after substituting $\boldsymbol{x}$.

2. Next, it augments $\mathbf{M}$ to get the matrix

$$\widehat{\mathbf{M}} = \left(\begin{array}{ccccc|c} & & & & & \ell_1 \\ & & & & & \ell_2 \\ & & \mathbf{M} & & & \vdots \\ & & & & & \ell_m \\ \hline 0 & 0 & \cdots & 0 & -1 & \ell_{m+1} \end{array}\right)$$

3. It returns $\det(\widehat{\mathbf{M}})$.

For correctness of the evaluation procedure, we see that when $\ell_j = L_j(\boldsymbol{x})$ for all $j \in [m]$ and $\ell_{m+1} = L_{m+1}(z)$, Eval computes

$$\det(\widehat{\mathbf{M}}) = \det(\mathbf{M}'\mathbf{N}) = \det(\mathbf{M}')\det(\mathbf{N}) = \det(\mathbf{M}') = z\det(\mathbf{M}) + \beta = zf(\boldsymbol{x}) + \beta.$$

The determinant of $\mathbf{M}'$ is calculated via Laplace expansion in the last column.

Remark 2 Here, we observe some structural properties of Eval which we require for our application.

- If we consider the Laplace expansion of $\det(\widehat{\mathbf{M}})$ in the last column then Eval can be written as
$$\mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1}) = A_1\ell_1 + A_2\ell_2 + \cdots + A_{m+1}\ell_{m+1} \tag{9}$$
where A_j is the $(j, (m+1))$-cofactor of $\widehat{\mathbf{M}}$. This shows that Eval is *linear* in $\ell_1, \ldots, \ell_{m+1}$. Due to this linearity feature, Eval can be computed in the exponent of any bilinear group. More precisely, suppose $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be a bilinear group then for any $i \in \{1, 2, T\}$, we have $\mathsf{Eval}(f, \boldsymbol{x}, [\![\ell_1]\!]_i, \ldots, [\![\ell_{m+1}]\!]_i) = [\![\mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1})]\!]_i$.
- Now, in particular, the coefficient of ℓ_1 is $A_1 = (-1)^{2+m}(-1)^m = 1$. Therefore, for any non-zero $\delta \in \mathbb{Z}_p$, we can write
$$\begin{aligned} \delta + \mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1}) &= \mathsf{Eval}(f, \boldsymbol{x}, \delta, 0, \ldots, 0) + \mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1}) \quad (10) \\ &= \mathsf{Eval}(f, \boldsymbol{x}, \ell_1 + \delta, \ell_2, \ldots, \ell_{m+1}) \quad (11) \end{aligned}$$
where equation 10 holds due to equation 9 and $A_1 = 1$; and equation 11 holds by the linearity of Eval. We will utilize equation 11 in our extended one slot FE construction.

Now, we describe the simulator of AKGS which simulates the values of label functions by using $f, \boldsymbol{x}$ and $zf(\boldsymbol{x}) + \beta$.

SimGarble$(\boldsymbol{f}, \boldsymbol{x}, \boldsymbol{zf(x) + \beta})$: The simulator works as follows:

1. It defines a set $H = \left\{ \begin{pmatrix} \mathbf{I}_m & \boldsymbol{r} \\ \mathbf{0} & 1 \end{pmatrix} \middle| \boldsymbol{r} \in \mathbb{Z}_p^m \right\}$ which forms a matrix subgroup.

2. Following Lemma 2, it computes the matrix $\mathbf{M}$ using $f, \boldsymbol{x}$ and sets the matrix

$$\mathbf{M}'' = \left(\begin{array}{ccccc|c} & & & & & zf(\boldsymbol{x}) + \beta \\ & & & & & 0 \\ & & \mathbf{M} & & & \vdots \\ & & & & & 0 \\ \hline 0 & 0 & \cdots & 0 & -1 & 0 \end{array}\right)$$

   which defines a left coset $\mathbf{M}''H = \{\mathbf{M}''\mathbf{N} | \mathbf{N} \in H\}$.
3. It uniformly samples a random matrix from the coset $\mathbf{M}''H$ and returns the last column of the matrix as simulated values of the label functions.

The simulation security follows from [IW14]. They observed that $\mathbf{M}''$ belongs to the coset $\mathbf{M}'H$ and hence by the property of cosets $\mathbf{M}''H = \mathbf{M}'H$ which proves the security. We omit the details here and state the security of AKGS in the following lemma.

Lemma 7 ([LL20]) *The above construction of* AKGS = (Garble, Eval) *is secure. Moreover, it is special piecewise secure as per Definition 8.*

4 1-Key 1-Ciphertext Secure One-Slot FE for Attribute-Weighted Sums

In this section, we first describe a private-key one-slot FE scheme for the attribute-weighted sum functionality that is proven simulation secure against a single ciphertext query and a single secret key query either before or after the ciphertext query. This scheme would be crucially embedded into the hidden slots for our full-fledged public-key one-slot FE scheme for attribute-weighted sums presented in the next section. We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ for simplicity of notations. Let (Garble, Eval) be a special piecewise secure AKGS for a function class $\mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ a tuple of pairing groups of prime order p, and (SK-IPFE.Setup.SK-IPFE.KeyGen, SK-IPFE.Enc, SK-IPFE.Dec) a secret-key function-hiding SK-IPFE based on G.

Setup($1^n, 1^{n'}$): Define the index sets as follows

$$S_{\text{1-FE}} = \left\{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \{\mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}_{\tau\in[n']}\right\},\ \widehat{S}_{\text{1-FE}} = \{\widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}^*}\}$$

It generates IPFE.MSK ← SK-IPFE.Setup($S_{\text{1-FE}}$) and $\widehat{\mathsf{IPFE.MSK}}$ ← SK-IPFE.Setup($\widehat{S}_{\text{1-FE}}$). Finally, it returns MSK = (IPFE.MSK, $\widehat{\mathsf{IPFE.MSK}}$).

KeyGen(MSK, $\boldsymbol{f}$): Let $f \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ be a function such that $f = (f_1, \ldots, f_{n'}) : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \to \mathbb{Z}_p$ where $f_1, \ldots, f_{n'} : \mathbb{Z}_p^n \to \mathbb{Z}_p$ are ABPs of size $(m+1)$. Sample $\beta_t \leftarrow \mathbb{Z}_p$ for $t \in [n']$ such that $\sum_{t\in[n']} \beta_t = 0 \mod p$. Next, sample independent random vectors $\boldsymbol{r}_t \leftarrow \mathbb{Z}_p^m$ for garbling and compute the coefficient vectors

$$(\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}, \boldsymbol{\ell}_{m+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \beta_t; \boldsymbol{r}_t)$$

for all $t \in [n']$. Here we make use of the instantiation of the AKGS described in Section 3.6. From the description of that AKGS instantiation, we note that the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}$ would be of the form $\boldsymbol{\ell}_{m+1,t} = \boldsymbol{z}[t] - \boldsymbol{r}_t[m]$. Also all the label functions $\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}$ involve only the variables $\boldsymbol{x}$ and not the variable $\boldsymbol{z}[t]$. Next, for all $j \in [m]$ and $t \in [n']$, it defines the vectors $\boldsymbol{v}_{j,t}$ corresponding to the label functions $\boldsymbol{\ell}_{j,t}$ obtained from the partial garbling:

vector	const	coef_i	sim_τ	sim^*_τ
$\boldsymbol{v}_{j,t}$	$\boldsymbol{\ell}_{j,t}[\mathsf{const}]$	$\boldsymbol{\ell}_{j,t}[\mathsf{coef}_i]$	0	0

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{m+1,t}$	$\boldsymbol{r}_t[m]$	1	0

It generates the secret-keys as

$$\mathsf{IPFE.SK}_{j,t} \leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2) \qquad \text{for } j \in [m], t \in [n']$$
$$\widehat{\mathsf{IPFE.SK}}_{m+1,t} \leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2) \qquad \text{for } t \in [n']$$

It returns the secret-key as $\mathsf{SK}_f = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$.

$\mathbf{Enc}(\mathbf{MSK}, \boldsymbol{x} \in \mathbb{Z}_p^n, \boldsymbol{z} \in \mathbb{Z}_p^{n'})$: It sets the vectors

vector	const	coef_i	sim_τ	sim^*_τ
$\boldsymbol{u}$	1	$\boldsymbol{x}[i]$	0	0

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{h}_t$	-1	$\boldsymbol{z}[t]$	0

for all $t \in [n']$. It encrypts the vectors as

$$\mathsf{IPFE.CT} \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)$$
$$\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{h}_t]\!]_1) \quad \text{for } t \in [n']$$

and returns the ciphertext as $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

$\mathbf{Dec}((\mathbf{SK}_f, f), (\mathbf{CT}, \boldsymbol{x}))$: It parses the secret-key $\mathsf{SK}_f = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ and the ciphertext $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$. It uses the decryption algorithm of $\mathsf{SK\text{-}IPFE}$ to compute

$$[\![\ell_{j,t}]\!]_T = \mathsf{SK\text{-}IPFE.Dec}(\mathsf{IPFE.SK}_{j,t}, \mathsf{IPFE.CT}) \text{ for } j \in [m], t \in [n']$$
$$[\![\ell_{m+1,t}]\!]_T = \mathsf{SK\text{-}IPFE.Dec}(\widehat{\mathsf{IPFE.SK}}_{m+1,t}, \widehat{\mathsf{IPFE.CT}}_t) \text{ for } t \in [n']$$

Next, it utilizes the evaluation procedure of AKGS and obtain a combined value

$$[\![\rho]\!]_T = \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T).$$

Finally, it returns a value ρ by solving a discrete logarithm problem. Similar to [AGW20], we assume that the desired attribute-weighted sum lies within a specified polynomial-sized domain so that discrete logarithm can be solved via brute force.

Correctness: By the correctness of IPFE, we have $\mathsf{SK\text{-}IPFE.Dec}(\mathsf{IPFE.SK}_{j,t}, \mathsf{IPFE.CT}) = [\![\ell_{j,t}]\!]_T = [\![L_{j,t}(\boldsymbol{x})]\!]_T$ for all $j \in [m], t \in [n']$ and $\mathsf{SK\text{-}IPFE.Dec}(\widehat{\mathsf{IPFE.SK}}_{m+1,t}, \widehat{\mathsf{IPFE.CT}}_t) = [\![\ell_{m+1,t}]\!]_T = [\![\boldsymbol{z}[t] - \boldsymbol{r}_t[m]]\!]_T$ for all $t \in [n']$. Next, using the correctness of AKGS and the linearity of the Eval function, we have

$$\mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) = [\![f_t(\boldsymbol{x})\boldsymbol{z}[t] + \beta_t]\!]_T$$

Therefore, we get by multiplying

$$\begin{aligned}
[\![\rho]\!]_T &= \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) \\
&= [\![\sum_{t=1}^{n'} \mathsf{Eval}(f_t, \boldsymbol{x}, \ell_{1,t}, \ldots, \ell_{m+1,t})]\!]_T \\
&= [\![\sum_{t=1}^{n'} f_t(\boldsymbol{x})\boldsymbol{z}[t] + \beta_t]\!]_T \\
&= [\![f(\boldsymbol{x})^\top \boldsymbol{z}]\!]_T
\end{aligned}$$

where the last equality holds since $\sum_{t\in[n']} \beta_t = 0 \mod p$.

4.1 Security Analysis

Theorem 2 *The 1-*FE *scheme for attribute-weighted sum is 1-key, 1-ciphertext adaptive simulation secure as per Definition 4 assuming the* AKGS *is piecewise secure as per Definition 7 and the* IPFE *is function hiding as per Definition 5.*

We proceeds with the description of the simulator and then security reduction of our 1-key 1-ciphertext secure one-slot FE. Recall that, we have designed the 1-key 1-ciphertext secure one-slot FE for the purpose of showing the indistinguishability in a particular hybrid required in the security reduction of the one-slot FE of Section 5. In that particular hybrid, we deal with a single pre-ciphertext secret key query of the one-slot FE scheme. Thus, while proving the security of our 1-key 1-ciphertext secure one-slot FE, we assume that the adversary queries a single secret key before the challenge ciphertext is sent. However, we emphasize that if we consider the single secret key query after the challenge phase then the security can also be proved using the techniques involved in the security reduction (in Section 5.1) of our one-slot FE.

The Simulator We describe the simulator for the 1-FE scheme. Let us assume that f is the only secret-key query made by the adversary before it sends the challenge ciphertext vectors.

Setup$^*(1^\lambda, 1^n, 1^{n'})$: To generate the master secret-key, it executes as follows:

1. Define the index sets as follows
$$S_{\text{1-FE}} = \left\{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \{\mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}_{\tau\in[n']}\right\}, \ \widehat{S}_{\text{1-FE}} = \{\widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}^*}\}$$
2. It then generates $\mathsf{IPFE.MSK} \leftarrow \mathsf{SK\text{-}IPFE.Setup}(S_{\text{1-FE}})$ and $\widehat{\mathsf{IPFE.MSK}} \leftarrow \mathsf{SK\text{-}IPFE.Setup}(\widehat{S}_{\text{1-FE}})$.
3. It outputs $\mathsf{MSK}^* = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$.

$\mathsf{KeyGen}_0^*(\mathsf{MSK}^*, \boldsymbol{f})$: On input MSK^* and a function $f = (f_1, \ldots f_{n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, the simulator proceeds as in the original scheme:

1. It first samples $\{\beta_t \leftarrow \mathbb{Z}_p\}_{t\in[n']}$ and $\{\boldsymbol{r}_t = (\boldsymbol{r}_t[1], \ldots, \boldsymbol{r}_t[m]) \leftarrow \mathbb{Z}_p^m\}_{t\in[n']}$ where it holds that $\sum_{t\in[n']} \beta_t = 0 \mod p$.
2. Next, it computes the coefficient vectors for the label functions as

$$(\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}, \boldsymbol{\ell}_{m+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{z}^*[t] f_t(\boldsymbol{x}^*) + \beta_t; \boldsymbol{r}_t)$$

   for each $t \in [n']$. From the description of AKGS, we note that the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}$ would be of the form $\boldsymbol{\ell}_{m+1,t} = \boldsymbol{z}[t] - \boldsymbol{r}_t[m]$.
3. It sets the following vectors

vector	const	coef_i	sim_τ	sim_τ^*
$\boldsymbol{v}_{j,t}$	$\boldsymbol{\ell}_{j,t}[\mathsf{const}]$	$\boldsymbol{\ell}_{j,t}[\mathsf{coef}_i]$	0	0

   for all $j \in [m]$ and $t \in [n']$. It also sets the following vectors

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}^*}$
$\boldsymbol{v}_{m+1,t}$	$\boldsymbol{r}_t[m]$	1	0

   for all $t \in [n']$.
4. It generates the IPFE secret-keys

$$\mathsf{IPFE.SK}_{j,t} \leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2) \qquad \text{for } j \in [m], t \in [n']$$
$$\widehat{\mathsf{IPFE.SK}}_{m+1,t} \leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2) \qquad \text{for } t \in [n']$$

5. Finally, it returns the secret-key $\mathsf{SK}_f = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$.

$\mathsf{Enc}^*(\mathsf{MSK}^*, \boldsymbol{x}^*, (\boldsymbol{f}, \boldsymbol{f}(\boldsymbol{x}^*)^\top \boldsymbol{z}^*))$: On input MSK^*, a vector $\boldsymbol{x}^* \in \mathbb{Z}_p^n$ and the tuple $(f, f(\boldsymbol{x}^*)^\top \boldsymbol{z}^*)$ for some $f \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ and $\boldsymbol{z}^* \in \mathbb{Z}_p^{n'}$ the simulator executes the following steps:

1. It samples a dummy vector $\boldsymbol{d} \leftarrow D$ from the set

$$D = \{\boldsymbol{d} \in \mathbb{Z}_p^{n'} : f(\boldsymbol{x}^*)^\top \boldsymbol{d} = f(\boldsymbol{x}^*)^\top \boldsymbol{z}^*\}.$$

   The simulator does this by finding a random vector $\boldsymbol{d} \in \mathbb{Z}_p^{n'}$ such that $\sum_{t\in[n']} f_t(\boldsymbol{x}^*)\boldsymbol{z}^*[t] = \sum_{t\in[n']} f_t(\boldsymbol{x}^*)\boldsymbol{d}[t]$. Hence, D is identical to the set $D_{\mathsf{IP}} = \{\boldsymbol{d} \in \mathbb{Z}_p^{n'} : (f_1(\boldsymbol{x}^*), \ldots, f_{n'}(\boldsymbol{x}^*)) \cdot (\boldsymbol{d}[1], \ldots, \boldsymbol{d}[n']) = f(\boldsymbol{x}^*)^\top \boldsymbol{z}^*\}$. A vector $\boldsymbol{d}$ from a set of the form D_{IP} can be efficiently sampled via a polynomial time algorithm given by O'Neill [O'N10] as the inner product functionality is *pre-image-sampleable*. Therefore, given $\boldsymbol{x}^*$ and $(f, f(\boldsymbol{x}^*)^\top \boldsymbol{z}^*)$, the simulator can find a dummy vector $\boldsymbol{d}$ such that $f(\boldsymbol{x}^*)^\top \boldsymbol{d} = f(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$.
2. Next, it sets the following vectors

vector	const	coef_i	sim_τ	sim_τ^*
$\boldsymbol{u}$	1	$\boldsymbol{x}^*[i]$	0	0

vector	$\widehat{\text{const}}$	$\widehat{\text{coef}}$	$\widehat{\text{sim}^*}$
$\boldsymbol{h}_t$	-1	$\boldsymbol{d}[t]$	0

for all $t \in [n']$.

3. It encrypts the vectors as

$$\mathsf{IPFE.CT} \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)$$
$$\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\widehat{\mathsf{IPFE.MPK}}, [\![\boldsymbol{h}_t]\!]_1) \text{ for } t \in [n']$$

4. It returns the ciphertext as $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

Hybrids and Reductions

Proof. We employ a sequence of hybrid experiments to demonstrate the indistinguishability between the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,1\text{-}FE}}(1^\lambda)$ and the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,1\text{-}FE}}(1^\lambda)$ with the simulator described above where $\mathcal{A}$ is any PPT adversary. We assume that in each experiment, $\mathcal{A}$ queries the single secret-key query for a function $f \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ before submitting the challenge message $(\boldsymbol{x}^*, \boldsymbol{z}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$.

Hybrid $\mathbf{H_0}$: : This is the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,1\text{-}FE}}(1^\lambda)$ defined in Section 3.4. The secret-key $\mathsf{SK}_f = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ is associated with the vectors $\boldsymbol{v}_{j,t}$ given by

vector	const	coef_i	sim_τ	sim^*_τ
$\boldsymbol{v}_{j,t}$	$\boldsymbol{\ell}_{j,t}[\text{const}]$	$\boldsymbol{\ell}_{j,t}[\text{coef}_i]$	0	0

for $j \in [m]$ and $t \in [n']$ and

vector	$\widehat{\text{const}}$	$\widehat{\text{coef}}$	$\widehat{\text{sim}^*}$
$\boldsymbol{v}_{m+1,t}$	$\boldsymbol{r}_t[m]$	1	0

for $t \in [n']$ where

$$(\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}, \boldsymbol{\ell}_{m+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{z}^*[t]f_t(\boldsymbol{x}^*) + \beta_t; \boldsymbol{r}_t)$$

such that $f = (f_1, \ldots, f_{n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, $\boldsymbol{r}_t \leftarrow \mathbb{Z}_p^m$ and $\beta_t \leftarrow \mathbb{Z}_p$ with $\sum_{t\in[n']}\beta_t = 0 \mod p$. The challenge ciphertext $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$ corresponds to $(\boldsymbol{x}^*, \boldsymbol{z}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$ is associated with the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ given by

vector	const	coef_i	sim_τ	sim^*_τ
$\boldsymbol{u}$	1	$\boldsymbol{x}^*[i]$	0	0

vector	$\widehat{\text{const}}$	$\widehat{\text{coef}}$	$\widehat{\text{sim}^*}$
$\boldsymbol{h}_t$	-1	$\boldsymbol{z}^*[t]$	0

for $t \in [n']$. In the subsequent hybrids, we'll omit the names of the indices of the vectors $\{\boldsymbol{v}_{j,t}\}_{j\in[m+1],t\in[n']}, \boldsymbol{u}, \{\boldsymbol{h}_t\}_{t\in[n']}$ and we'll assume that the entries of those vectors lie in those indices as in the order mentioned in H_0.

Hybrid H_1: : This hybrid is exactly the same as H_0 except that we change the vectors $\boldsymbol{v}_{1,t}$ in the secret-key and $\boldsymbol{u}$ in the challenge ciphertext as follows.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\quad \boxed{0}, \quad \boxed{0}, \quad \boxed{\delta_{t\tau}}, \ 0\ ),\\
\boldsymbol{v}_{j,t} &= (\ \boldsymbol{\ell}_{j,t}[\mathsf{const}], \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i], \ 0, \ 0\ ) & \forall 1 < j \le m,\\
\boldsymbol{v}_{m+1,t} &= (\quad \boldsymbol{r}_t[m], \quad 1, \quad 0, \ 0\ ),\\
\boldsymbol{u} &= (\quad 1, \quad \boldsymbol{x}^*[i], \ \boxed{\ell_{1,\tau}}, 0\ ),\\
\boldsymbol{h}_t &= (\quad -1, \quad \boldsymbol{z}^*[t], \quad 0 \quad).
\end{aligned}$$

Here $\delta_{t\tau}$ is the Kronecker Delta where $\delta_{t\tau} = 1$ if $t = \tau$, and 0 otherwise. Thus the difference between H_0 and H_1 is that instead of embedding the coefficient vectors $\boldsymbol{\ell}_{1,t}$ of the label functions $L_{1,t}$ obtained from $\mathsf{Garble}(\boldsymbol{z}^*[t]f_t(\boldsymbol{x}^*)+\beta_t; \boldsymbol{r}_t)$, we embed the value of the label functions $L_{1,t}(\boldsymbol{x}^*) = \ell_{1,t}$ within the ciphertext vector $\boldsymbol{u}$. Note that the inner products $\boldsymbol{v}_{1,t} \cdot \boldsymbol{u} = \ell_{1,t}$, for all $t \in [n']$, remain the same as in H_0. Therefore, the function hiding security of $\mathsf{SK\text{-}IPFE}$ ensures the indistinguishability between the hybrids H_0 and H_1.

Hybrid H_2: : This hybrid is identical to H_1 except that we replace the actual garbling values $\ell_{1,t}$ with the reverse sampling $\widetilde{\ell}_{1,t}$ of AKGS computed as

$$\widetilde{\ell}_{1,t} \leftarrow \mathsf{RevSamp}(f_t, \boldsymbol{x}^*, f_t(\boldsymbol{x}^*)\boldsymbol{z}^*[t] + \beta_t, \ell_{2,t}, \ldots, \ell_{m,t}, \ell_{m+1,t})$$

where $\ell_{j,t} = L_{j,t}(\boldsymbol{x}^*)$ for all $j \in [2, m]$ and $\ell_{m+1,t} = \boldsymbol{z}^*[t] - \boldsymbol{r}_t[m]$ obtained by running $\mathsf{Garble}(\boldsymbol{z}^*[t]$ $f_t(\boldsymbol{x}^*) + \beta_t; \boldsymbol{r}_t)$ honestly. Therefore, the challenge ciphertext is now associated with the vectors

$$\begin{aligned}
\boldsymbol{u} &= (\ 1, \ \boldsymbol{x}^*[i], \boxed{\widetilde{\ell}_{1,\tau}}, 0\ ),\\
\boldsymbol{h}_t &= (\ -1, \boldsymbol{z}^*[t], \quad 0 \quad).
\end{aligned}$$

For each $t \in [n']$, the piecewise security of AKGS guarantees that given $(\boldsymbol{\ell}_{2,t}, \ldots, \boldsymbol{\ell}_{m,t}, \boldsymbol{\ell}_{m+1,t})$, the actual garbling $\ell_{1,t}$ and the reversely sampled value $\widetilde{\ell}_{1,t}$ are identically distributed. Hence, the hybrids H_1 and H_2 are indistinguishable by the reverse sampleability of AKGS.

Hybrid $\mathsf{H}_{3,j}$ ($j \in [m]$): : This is analogous to H_2 except that we change the secret-key as follows. For all j' such that $1 < j' < j$, the coefficient vector $\boldsymbol{\ell}_{j',t}$ is taken away from $\boldsymbol{v}_{j',t}$ and a random value $\ell'_{j',t} \leftarrow \mathbb{Z}_p$ is put into $\boldsymbol{v}_{j',t}[\mathsf{const}]$. The modified secret-key is now associated with the vectors

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\quad 0, \quad 0, \quad \delta_{t\tau}, 0\ ),\\
\boldsymbol{v}_{j',t} &= (\quad \boxed{\ell'_{j',t}}, \quad \boxed{0}, \quad 0, \ 0\ ) & \forall 1 < j' < j,\\
\boldsymbol{v}_{j,t} &= (\ \boldsymbol{\ell}_{j,t}[\mathsf{const}], \ \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i], \ 0, \ 0\ ),\\
\boldsymbol{v}_{j',t} &= (\ \boldsymbol{\ell}_{j',t}[\mathsf{const}], \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], \ 0, \ 0\ ) & \forall j < j' \le m,\\
\boldsymbol{v}_{m+1,t} &= (\quad \boldsymbol{r}_t[m], \quad 1, \quad 0 \quad)
\end{aligned}$$

Note that, in this hybrid $\widetilde{\ell}_{1,t}$ is reversely sampled using the random values $\ell'_{2,t}, \ldots, \ell'_{j-1,t}$ and the actual values $\ell_{j,t}, \ldots, \ell_{m+1,t}$ for each $t \in [n']$. Observe that $\mathsf{H}_{3,1}$ coincides with H_2. We will show that for all $j \in [2, m]$, the hybrids $\mathsf{H}_{3,(j-1)}$ and $\mathsf{H}_{3,j}$ are indistinguishable via the following sequence of sub-hybrids, namely, $\{\mathsf{H}_{3,j,1}, \mathsf{H}_{3,j,2}, \mathsf{H}_{3,j,3}\}_{j\in[2,m]}$.

Hybrid $\mathsf{H}_{3,j,1}$ ($j \in [2,m]$): : This is exactly same as $\mathsf{H}_{3,(j-1)}$ except that the coefficient vector $\boldsymbol{\ell}_{j,t}$ is removed from $\boldsymbol{v}_{j,t}$ and $\boldsymbol{v}_{j,t}[\mathsf{sim}^*_\tau]$ is set to $\delta_{t\tau}$. We hardwire the actual garbling value $\ell_{j,\tau} = L_{j,\tau}(\boldsymbol{x}^*)$ into $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ to ensure the inner product $\boldsymbol{v}_{j,\tau} \cdot \boldsymbol{u}$ remains the same as in $\mathsf{H}_{3,(j-1)}$. The changes in the vectors associated with the secret-key and the challenge ciphertext are given below.

$$\begin{array}{rlllll}
\boldsymbol{v}_{1,t} = (& 0, & 0, & \delta_{t\tau}, & 0 &), \\
\boldsymbol{v}_{j',t} = (& \ell'_{j',t}, & 0, & 0 & 0 &) \quad \forall 1 < j' < j, \\
\boldsymbol{v}_{j,t} = (& \boxed{0}, & \boxed{0}, & 0, & \boxed{\delta_{t\tau}} &), \\
\boldsymbol{v}_{j',t} = (& \boldsymbol{\ell}_{j',t}[\mathsf{const}], & \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], & 0, & 0 &) \quad \forall j < j' \leq m, \\
\boldsymbol{v}_{m+1,t} = (& \boldsymbol{r}_t[m], & 1, & 0 & &), \\
\boldsymbol{u} = (& 1, & \boldsymbol{x}^*[i], & \widetilde{\ell}_{1,\tau}, & \boxed{\ell_{j,\tau}} &), \\
\boldsymbol{h}_t = (& -1, & \boldsymbol{z}^*[t], & 0 & &).
\end{array}$$

Therefore, the hybrids $\mathsf{H}_{3,(j-1)}$ and $\mathsf{H}_{3,j,1}$ are indistinguishable by the function hiding security of IPFE.

Hybrid $\mathsf{H}_{3,j,2}$ ($j \in [2,m]$): : It proceeds exactly the same as $\mathsf{H}_{3,j,1}$ except that the label $\ell_{j,\tau}$ (sitting at $\boldsymbol{u}[\mathsf{sim}^*_\tau]$) is replaced with a random value $\ell'_{j,\tau} \leftarrow \mathbb{Z}_p$. The vectors associated to the challenge ciphertext are given by

$$\begin{array}{rllll}
\boldsymbol{u} = (& 1, & \boldsymbol{x}^*[i], \widetilde{\ell}_{1,\tau}, & \boxed{\ell'_{j,\tau}} &), \\
\boldsymbol{h}_t = (& -1, & \boldsymbol{z}^*[t], \ \ 0 & &)
\end{array}$$

where $\ell'_{j,\tau}$ are randomly sampled from $\mathbb{Z}_p$. Now the first label $\widetilde{\ell}_{1,t}$ is reversely sampled using the random values $\ell'_{2,t}, \ldots, \ell'_{j,t}$ and the actual labels $\ell_{j+1,t} = L_{j+1,t}(\boldsymbol{x}^*), \ldots, \ell_{m,t} = L_{m,t}(\boldsymbol{x}^*), \ell_{m+1,t} = -\boldsymbol{r}_t[m] + \boldsymbol{z}^*[t]$. Hence, the marginal randomness property of AKGS ensures that the hybrids $\mathsf{H}_{3,j,1}$ and $\mathsf{H}_{3,j,2}$ are identically distributed.

Hybrid $\mathsf{H}_{3,j,3}$ ($j \in [2,m]$): : This hybrid is exactly the same as $\mathsf{H}_{3,j,2}$ except that the random value $\ell'_{j,\tau}$ is sifted from $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ to $\boldsymbol{v}_{j,t}[\mathsf{const}]$. Also, the positions $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ and $\boldsymbol{v}_{j,t}[\mathsf{sim}^*_\tau]$ are set to zero. The vectors associated to the secret-key and the challenge ciphertext become

$$\begin{array}{rlllll}
\boldsymbol{v}_{1,t} = (& 0, & 0, & \delta_{t\tau}, & 0 &), \\
\boldsymbol{v}_{j',t} = (& \ell'_{j',t}, & 0, & 0, & 0 &) \quad \forall 1 < j' < j, \\
\boldsymbol{v}_{j,t} = (& \boxed{\ell'_{j,t}}, & 0, & 0, & \boxed{0} &), \\
\boldsymbol{v}_{j',t} = (& \boldsymbol{\ell}_{j',t}[\mathsf{const}], & \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], & 0, & 0 &) \quad \forall j < j' \leq m, \\
\boldsymbol{v}_{m+1,t} = (& \boldsymbol{r}_t[m], & 1, & 0 & &), \\
\boldsymbol{u} = (& 1, & \boldsymbol{x}^*[i], & \widetilde{\ell}_{1,\tau}, & \boxed{0} &), \\
\boldsymbol{h}_t = (& -1, & \boldsymbol{z}^*[t], & 0 & &).
\end{array}$$

Since the inner products $\boldsymbol{v}_{j,t} \cdot \boldsymbol{u}$ for all $j \in [m], t \in [n']$ remain the same as in $\mathsf{H}_{3,j,2}$, the indistinguishability between the hybrids $\mathsf{H}_{3,j,2}$ and $\mathsf{H}_{3,j,3}$ follows from the function hiding security of IPFE. We observe that the hybrids $\mathsf{H}_{3,j,3}$ is identical to $\mathsf{H}_{3,j}$ for all $j \in [2,m]$.

Hybrid $\mathsf{H_4}$: : It proceeds exactly the same as hybrid $\mathsf{H}_{3,m}$ except that the the actual garbling value $\ell_{m+1,t} = \boldsymbol{z}^*[t] - \boldsymbol{r}_t[m]$ for the label function $L_{m+1,t}$ obtained from the Garble algorithm is used in $\boldsymbol{h}_t[\widehat{\mathsf{sim}^*}]$. The changes are given by

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\ \ 0,\ \ \delta_{t\tau},\ \ 0\ ), \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t},\ \ 0,\ \ 0,\ \ 0\ ) \qquad \forall 1 < j \leq m, \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{0},\ \boxed{0},\ \ \boxed{1}\ \ ), \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i],\ \ \widetilde{\ell}_{1,\tau},\ \ 0\ ), \\
\boldsymbol{h}_t &= (\ \boxed{1},\ \boxed{0},\ \boxed{\ell_{m+1,t}}\ ).
\end{aligned}$$

Since the inner products $\boldsymbol{v}_{m+1,t} \cdot \boldsymbol{h}_t$ for all $t \in [n']$ remain the same as in $\mathsf{H}_{3,m}$, the indistinguishability between the hybrids $\mathsf{H}_{3,m}$ and H_4 follows from the function hiding security of IPFE.

Hybrid $\mathsf{H_5}$: : It is exactly the same as H_4 except that the actual label $\ell_{m+1,t}$ is now replaced with a random value $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$. The vectors used in the challenge ciphertext are as follows.

$$\begin{aligned}
\boldsymbol{u} &= (\ 1, \boldsymbol{x}^*[i],\ \ \widetilde{\ell}_{1,\tau},\ \ 0\ ), \\
\boldsymbol{h}_t &= (\ 1,\ \ 0,\ \ \boxed{\ell'_{m+1,t}}\ )
\end{aligned}$$

Note that, in this hybrid the labels $\widetilde{\ell}_{1,t}$ for $t \in [n']$ are now reversely sampled using all random values $\ell'_{2,t}, \ldots, \ell'_{m+1,t}$ which are randomly picked from $\mathbb{Z}_p$. By the marginal randomness property of AKGS, the hybrids H_4 and H_5 are identically distributed.

Hybrid $\mathsf{H_6}$: : This hybrid proceeds exactly the same as H_5 except that the random values $\ell'_{m+1,t}$ are shifted from $\boldsymbol{h}_t[\widehat{\mathsf{sim}^*}]$ to $\boldsymbol{v}_{m+1,t}[\widehat{\mathsf{const}}]$. The changes are indicated as follows.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\quad 0,\qquad 0,\ \ \delta_{t\tau},\ 0\ ), \\
\boldsymbol{v}_{j,t} &= (\quad \ell'_{j,t},\qquad 0,\quad 0,\ 0\ ) \qquad \forall 1 < j \leq m, \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{\ell'_{m+1,t}},\quad 0,\quad \boxed{0}\ \ ), \\
\boldsymbol{u} &= (\quad 1,\quad \boldsymbol{x}^*[i], \widetilde{\ell}_{1,\tau}, 0\ ), \\
\boldsymbol{h}_t &= (\quad 1,\qquad 0,\quad \boxed{0}\ \ ).
\end{aligned}$$

Observe that the inner products $\boldsymbol{v}_{m+1,t} \cdot \boldsymbol{h}_t$ for $t \in [n']$ are unchanged as in H_5. Hence, the function hiding security of IPFE ensures the indistinguishability between the hybrids H_5 and H_6.

Hybrid $\mathsf{H_7}$: : It is analogous to H_6 except that the values $f_t(\boldsymbol{x}^*)\boldsymbol{z}^*[t]$ is removed from $\widetilde{\ell}_{1,t}$ for all $1 < t \leq n'$ and the value $f(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$ is directly encoded into the label $\widetilde{\ell}_{1,1}$. For this, we replace the random elements β_t by $\beta'_t = \beta_t - f_t(\boldsymbol{x}^*)\boldsymbol{z}^*[t]$ for all $1 < t \leq n'$ and change the element β_1 with $\beta'_1 = \beta_1 - f_1(\boldsymbol{x}^*)\boldsymbol{z}^*[1] + f(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$. Note that, the distributions

$$\{\beta_t \leftarrow \mathbb{Z}_p : \sum_{t \in [n']} \beta_t = 0\} \text{ and } \{\beta'_t : \sum_{t \in [n']} \beta_t = 0\}$$

are statistically close since $\{\beta'_t\}_{t \in [n']}$ are also uniform over $\mathbb{Z}_p$ and $\sum_{t \in [n']} \beta'_t = 0$. Thus the vectors of the challenge ciphertext become

$$\begin{aligned}
\boldsymbol{u} &= (\ 1, \boldsymbol{x}^*[i], \boxed{\widetilde{\ell}_{1,\tau}}, 0\ ), \\
\boldsymbol{h}_t &= (\ 1,\quad 0,\qquad 0\quad).
\end{aligned}$$

where the labels $\widetilde{\ell}_{1,\tau}$ are given by

$$\begin{aligned}\widetilde{\ell}_{1,1} &\leftarrow \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f_1(\boldsymbol{x}^*)\boldsymbol{z}^*[1] + \beta_1', \ell_{2,1}, \ldots, \ell_{m+1,1}) \\ &= \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \beta_1, \ell_{2,1}, \ldots, \ell_{m+1,1}) \\ \widetilde{\ell}_{1,\tau} &\leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, f_\tau(\boldsymbol{x}^*)\boldsymbol{z}[\tau] + \beta_\tau, \ell_{2,\tau}, \ldots, \ell_{m+1,\tau}) \;\; \forall 1 < \tau \le n' \\ &= \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, \beta_\tau, \ell_{2,\tau}, \ldots, \ell_{m+1,\tau})\end{aligned}$$

Thus, H_6 and H_7 are indistinguishable as they are statistically close.

Hybrid H_8: : This hybrid is exactly the same as H_7 except that we use a dummy vector $\boldsymbol{d}$ such that $f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* = f(\boldsymbol{x}^*)^\top \boldsymbol{d}$ while generating $\widetilde{\ell}_{1,1}$. After the secret-key query made by $\mathcal{A}$, the dummy vector $\boldsymbol{d}$ can be sampled via an efficient algorithm which only need $f_1(\boldsymbol{x}^*), \ldots, f_{n'}(\boldsymbol{x}^*)$ and $f(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$. This is due to the pre-image-sampleability property of inner product functionality demonstrated by [O'N10]. Thus, the vector $\boldsymbol{u}$ associated with the challenge ciphertext is now defined as

$$\boldsymbol{u} = (1, \overbrace{\boldsymbol{x}^*[1], \ldots, \boldsymbol{x}^*[n]}^{\mathsf{coef}_i}, \overbrace{\boxed{\widetilde{\ell}_{1,1}}, \widetilde{\ell}_{1,2} \ldots, \widetilde{\ell}_{1,n'}}^{\mathsf{sim}_\tau}, \overbrace{0, \ldots, 0}^{\mathsf{sim}^*_\tau})$$

where the labels $\{\widetilde{\ell}_{1,\tau}\}_{\tau \in [n']}$ are computed as

$$\begin{aligned}\widetilde{\ell}_{1,1} &\leftarrow \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f(\boldsymbol{x}^*)^\top \boldsymbol{d} + \beta_1, \ell_{2,1}, \ldots, \ell_{m+1,1}) \\ \widetilde{\ell}_{1,\tau} &\leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, \beta_\tau, \ell_{2,\tau}, \ldots, \ell_{m+1,\tau}) \;\; \forall 1 < \tau \le n'.\end{aligned}$$

Above, we write the full expression of the vector $\boldsymbol{u}$ as opposed to its compressed expression used so far in order to highlight the change. Since the inner products $\boldsymbol{v}_{j,t} \cdot \boldsymbol{u}$ for $j \in [m], t \in [n']$ are unaltered between the two hybrids, the function hiding security of IPFE preserved the indistinguishability of the hybrids H_7 and H_8.

Hybrid H_9: : The following sequence of hybrids is basically the reverse of the previous hybrids with $\boldsymbol{z}^*$ replaced with $\boldsymbol{d}$. Therefore, in this hybrid the vectors of the challenge ciphertext are distributed as

$$\begin{aligned}\boldsymbol{u} &= (\,1, \boldsymbol{x}^*[i], \boxed{\widetilde{\ell}_{1,\tau}}, 0\,), \\ \boldsymbol{h}_t &= (\,1, \;\; 0, \;\;\; 0 \;\;).\end{aligned}$$

where $\widetilde{\ell}_{1,\tau} \leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, f_\tau(\boldsymbol{x}^*)\boldsymbol{d}[\tau] + \beta_\tau, \ell_{2,\tau}, \ldots, \ell_{m+1,\tau})$. This can be done by replacing β_1 by $\beta_1 - f(\boldsymbol{x}^*)^\top \boldsymbol{d} + f_1(\boldsymbol{x}^*)\boldsymbol{d}[1]$ and for $\tau > 1$, β_τ is replaced by $\beta_\tau + f_\tau(\boldsymbol{x}^*)\boldsymbol{d}[\tau]$. Note that, H_8 and H_9 are statistically close.

Hybrid H_{10}: : In this hybrid we change the vectors $\boldsymbol{v}_{m+1,t}$ and $\boldsymbol{h}_t$ as follows

$$\begin{aligned}\boldsymbol{v}_{1,t} &= (\;0, \;\; 0, \;\; \delta_{t\tau}, \;\; 0\;), \\ \boldsymbol{v}_{j,t} &= (\;\ell'_{j,t}, \;\; 0, \;\; 0, \;\; 0\;) \qquad \forall 1 < j \le m, \\ \boldsymbol{v}_{m+1,t} &= (\;\boxed{0}, \;\; 0, \;\; \boxed{1} \;\;), \\ \boldsymbol{u} &= (\;1, \;\; \boldsymbol{x}^*[i], \;\; \widetilde{\ell}_{1,\tau}, \;\; 0\;), \\ \boldsymbol{h}_t &= (\;1, \;\; 0, \;\; \boxed{\ell'_{m+1,t}}\;).\end{aligned}$$

where $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$. The indistinguishability between the hybrids H_9 and H_{10} follows from the function hiding security of IPFE.

Hybrid H_{11}: : It is exactly the same as H_{10} except that the random value $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$ is changed to actual $\ell_{m+1,t} = \boldsymbol{d}[t] - \boldsymbol{r}_t[m]$. Then the vectors in the challenge ciphertext become

$$\begin{aligned} \boldsymbol{u} &= (\ 1, \boldsymbol{x}^*[i],\ \ \widetilde{\ell}_{1,\tau},\ \ 0\ ), \\ \boldsymbol{h}_t &= (\ 1,\ \ 0,\ \ \boxed{\ell_{m+1,t}}\ ). \end{aligned}$$

The hybrids H_{10} and H_{11} are identical due to the marginal randomness property of AKGS.

Hybrid H_{12}: : In this hybrid we change the vectors $\boldsymbol{v}_{m+1,t}$ and $\boldsymbol{h}_t$ as follows

$$\begin{aligned} \boldsymbol{v}_{1,t} &= (\ \ 0,\ \ 0,\ \ \delta_{t\tau},\ 0\ ), \\ \boldsymbol{v}_{j,t} &= (\ \ \ell'_{j,t},\ \ 0,\ \ 0,\ 0\ ) \qquad \forall 1 < j \le m, \\ \boldsymbol{v}_{m+1,t} &= (\ \boxed{\boldsymbol{r}_t[m]},\ \boxed{1},\ \boxed{0}\ ), \\ \boldsymbol{u} &= (\ \ 1,\ \ \boldsymbol{x}^*[i],\ \widetilde{\ell}_{1,\tau},\ 0\ ), \\ \boldsymbol{h}_t &= (\ \boxed{-1},\ \boxed{\boldsymbol{d}[t]},\ \boxed{0}\ ). \end{aligned}$$

The indistinguishability between the hybrids H_{11} and H_{12} follows from the function hiding security of IPFE.

Hybrid $\mathsf{H}_{13,m+1-j}$ ($j \in [m-1]$): : It is analogous to H_{12} except the secret-key is modified as follows. For all j' such that $m+1-j \le j' < m+1$, the random value $\ell'_{j',t} \leftarrow \mathbb{Z}_p$ is discarded from $\boldsymbol{v}_{j',t}[\mathsf{const}]$ and the coefficient vector $\boldsymbol{\ell}_{j',t}$ is used in $\boldsymbol{v}_{j',t}$.

$$\begin{aligned} \boldsymbol{v}_{1,t} &= (\ \ 0,\ \ 0,\ \ \delta_{t\tau},\ 0\ ), \\ \boldsymbol{v}_{j',t} &= (\ \ \ell'_{j',t},\ \ 0,\ \ 0,\ 0\ ) \ \ \forall 1 < j' < m+1-j, \\ \boldsymbol{v}_{m+1-j,t} &= (\ \ \ell'_{m+1-j,t},\ \ 0,\ \ 0,\ 0\ ), \\ \boldsymbol{v}_{j',t} &= (\ \boxed{\boldsymbol{\ell}_{j',t}[\mathsf{const}]},\ \boxed{\boldsymbol{\ell}_{j',t}[\mathsf{coef}_i]},\ 0,\ 0\ ) \ \ \forall m+1-j \le j' < m+1, \\ \boldsymbol{v}_{m+1,t} &= (\ \ \boldsymbol{r}_t[m],\ \ 1,\ \ 0\ ). \end{aligned}$$

In this hybrid, the label $\widetilde{\ell}_{1,t}$ is reversely sampled using the random values $\ell'_{2,t}, \ldots, \ell'_{m+1-j,t}$ and the actual values $\ell_{m-j+2,t}, \ldots, \ell_{m+1,t}$ for each $t \in [n']$. The hybrids $\mathsf{H}_{13,m+1-(j-1)}$ and $\mathsf{H}_{13,m+1-j}$ can be shown to be indistinguishable via the following sequence of sub-hybrids, namely, $\{\mathsf{H}_{13,m+1-j,1}, \mathsf{H}_{13,m+1-j,2}, \mathsf{H}_{13,m+1-j,3}\}$.

Hybrid $\mathsf{H}_{13,m+1-j,1}$ ($j \in [m-1]$): : It proceeds exactly the same as $\mathsf{H}_{13,m+1-(j-1)}$ except that the random values $\ell'_{m+1-j,t}$ are sifted from $\boldsymbol{v}_{m+1-j,t}[\mathsf{const}]$ to $\boldsymbol{u}[\mathsf{sim}^*_\tau]$. We modify vectors associated with the secret-key and the challenge ciphertext as follows

$$\begin{aligned} \boldsymbol{v}_{1,t} &= (\ \ 0,\ \ 0,\ \ \delta_{t\tau},\ \ 0\ ), \\ \boldsymbol{v}_{j',t} &= (\ \ \ell'_{j',t},\ \ 0,\ \ 0,\ \ 0\ ) \ \ \forall 1 < j' < m+1-j, \\ \boldsymbol{v}_{m+1-j,t} &= (\ \boxed{0},\ \ 0,\ \ 0,\ \ \boxed{\delta_{t\tau}}\ ), \\ \boldsymbol{v}_{j',t} &= (\ \boldsymbol{\ell}_{j',t}[\mathsf{const}], \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i],\ 0,\ \ 0\ ) \ \ \forall m+1-j \le j' < m+1, \\ \boldsymbol{v}_{m+1,t} &= (\ \ \boldsymbol{r}_t[m],\ \ 1,\ \ 0\ ), \\ \boldsymbol{u} &= (\ \ 1,\ \ \boldsymbol{x}^*[i],\ \ \widetilde{\ell}_{1,\tau},\ \boxed{\ell'_{m+1-j,\tau}}\ ), \\ \boldsymbol{h}_t &= (\ \ -1,\ \ \boldsymbol{d}[t],\ \ 0\ ). \end{aligned}$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-)j-1)}$ and $\mathsf{H}_{13,m+1-j,1}$ follows from the function hiding security of IPFE.

Hybrid $\mathsf{H}_{13,m+1-j,2}$ ($j \in [m-1]$): : It is exactly same as $\mathsf{H}_{13,m+1-j,1}$ except that the random values $\ell'_{m+1-j,\tau}$ at $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ are now replaced with the actual labels $\ell_{m+1-j,\tau} = L_{m+1-j,\tau}(\boldsymbol{x}^*)$. The change in the vector $\boldsymbol{u}$ associated to the challenge ciphertext is indicated as below.

$$\begin{aligned}\boldsymbol{u} &= (\ \ 1,\ \boldsymbol{x}^*[i], \widetilde{\ell}_{1,\tau}, \boxed{\ell_{m+1-j,\tau}}\ ),\\ \boldsymbol{h}_t &= (\ -1,\ \boldsymbol{d}[t],\ \ 0,\ \ \ \ 0\ \ \ ).\end{aligned}$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-j,1}$ and $\mathsf{H}_{13,m+1-j,2}$ follows from the marginal randomness property of AKGS.

Hybrid $\mathsf{H}_{13,m+1-j,3}$ ($j \in [m-1]$): : It proceeds analogously to $\mathsf{H}_{13,m+1-j,2}$ except that instead of the actual labels $\ell_{m+1-j,t} = L_{m+1-j,t}(\boldsymbol{x}^*)$ we use the coefficient vectors $\boldsymbol{\ell}_{m+1-j,t}$ to set $\boldsymbol{v}_{m+1-j,t}$. Also, the positions $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ are set to zero to keep the inner products $\boldsymbol{v}_{m+1-j,t} \cdot \boldsymbol{u}$ unaltered as in $\mathsf{H}_{13,m+1-j,2}$. The changes in vectors associated with the secret-key and the challenge ciphertext are shown below.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\quad 0, \quad 0, \quad \delta_{t\tau},\ 0\ ),\\
\boldsymbol{v}_{j',t} &= (\quad \ell'_{j',t}, \quad 0, \quad 0,\ 0\ ) \quad \forall 1 < j' < m+1-j,\\
\boldsymbol{v}_{m+1-j,t} &= (\ \boxed{\boldsymbol{\ell}_{m+1-j,t}[\mathsf{const}]}, \boxed{\boldsymbol{\ell}_{m+1-j,t}[\mathsf{coef}_i]},\ 0,\ \boxed{0}\ ),\\
\boldsymbol{v}_{j',t} &= (\quad \boldsymbol{\ell}_{j',t}[\mathsf{const}], \quad \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], \quad 0,\ 0\ ) \quad \forall m+1-j \le j' < m+1,\\
\boldsymbol{v}_{m+1,t} &= (\quad \boldsymbol{r}_t[m], \quad 1, \quad 0,\ 0\ ),\\
\boldsymbol{u} &= (\quad 1, \quad \boldsymbol{x}^*[i], \quad \widetilde{\ell}_{1,\tau},\ \boxed{0}\ ),\\
\boldsymbol{h}_t &= (\quad -1, \quad \boldsymbol{d}[t], \quad 0,\ 0\ ).
\end{aligned}$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-j,2}$ and $\mathsf{H}_{13,m+1-j,3}$ follows from the function hiding security of IPFE. We observe that $\mathsf{H}_{13,m+1-j,3}$ is identical to $\mathsf{H}_{13,m+1-j}$ for all $j \in [m-1]$.

Hybrid H_{14}: : This hybrid proceeds exactly the same as $\mathsf{H}_{13,2}$ except that the reversely sampled labels $\widetilde{\ell}_{1,\tau}$ are replaced with the actual labels $\ell_{1,\tau} = L_{1,\tau}(\boldsymbol{x}^*)$ when setting $\boldsymbol{u}[\mathsf{sim}_\tau]$. The vectors associated with the challenge ciphertext are given by

$$\begin{aligned}\boldsymbol{u} &= (\ \ 1,\ \boldsymbol{x}^*[i], \boxed{\ell_{1,\tau}}, 0\ ),\\ \boldsymbol{h}_t &= (\ -1,\ \boldsymbol{d}[t],\ \ 0\ \ \ ).\end{aligned}$$

The indistinguishability between the hybrids $\mathsf{H}_{13,2}$ and H_{14} follows from the reversely sampleability guaranteed by the piecewise security of AKGS.

Hybrid H_{15}: : It is analogous to H_{14} except that the actual labels $\ell_{1,\tau} = L_{1,\tau}(\boldsymbol{x}^*)$ are removed from $\boldsymbol{u}[\mathsf{sim}_\tau]$ and the coefficient vectors $\boldsymbol{\ell}_{1,t}$ are utilized while setting the vectors $\boldsymbol{v}_{1,t}$ for all $t \in [n']$. The vectors associated with the secret-key and the challenge ciphertext are shown below.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ \boxed{\boldsymbol{\ell}_{1,t}[\mathsf{const}]}, \boxed{\boldsymbol{\ell}_{1,t}[\mathsf{coef}_i]}, \boxed{0}, 0\ ),\\
\boldsymbol{v}_{j,t} &= (\ \boldsymbol{\ell}_{j,t}[\mathsf{const}],\ \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i],\ 0,\ 0\ ) \qquad \forall 1 < j \le m,\\
\boldsymbol{v}_{m+1,t} &= (\quad \boldsymbol{r}_t[m], \quad 1, \quad 0\ ),\\
\boldsymbol{u} &= (\quad 1, \quad \boldsymbol{x}^*[i], \quad \boxed{0}, 0\ ),\\
\boldsymbol{h}_t &= (\quad -1, \quad \boldsymbol{d}[t], \quad 0\ ).
\end{aligned}$$

Since the inner products $\boldsymbol{v}_{1,t} \cdot \boldsymbol{u} = \ell_{1,t}$, for all $t \in [n']$, remain the same as in H_{14}, the function hiding security of IPFE ensures the indistinguishability between the hybrids H_{14} and H_{15}. Observe that the hybrid H_{15} coincides with the ideal experiment $\mathsf{Expt}^{\mathsf{Ideal,1\text{-}FE}}_{\mathcal{A}}(1^\lambda)$. $\square$

5 One-Slot FE for Attribute-Weighted Sums

In this section we present our public-key one-slot FE scheme Π_{one} for the attribute-weighted sum functionality that is proven adaptively simulation secure against a single ciphertext query and an arbitrary polynomial number of secret key queries both before and after the ciphertext query. As outlined in Remark 3 below, this scheme can naturally be extended to one supporting a bounded number of ciphertext queries. We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ for simplicity of notations. Let $(\mathsf{Garble}, \mathsf{Eval})$ be a special piecewise secure AKGS for a function class $\mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ a tuple of pairing groups of prime order p such that MDDH_k holds in $\mathbb{G}_2$, and $(\mathsf{IPFE.Setup}.\mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ a slotted IPFE based on G. We construct an FE scheme for attribute-weighted sums with the message space $\mathbb{M} = \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$.

Setup$(1^n, 1^{n'})$: Define the following index sets as follows

$$\begin{aligned} S_{\mathsf{pub}} &= \left\{\{\mathsf{const}^{(\iota)}\}_{\iota\in[k]}, \{\mathsf{coef}_i^{(\iota)}\}_{\iota\in[k],i\in[n]}\right\}, \widehat{S}_{\mathsf{pub}} = \left\{\widehat{\mathsf{const}}^{(\iota)}, \widehat{\mathsf{coef}}^{(\iota)}\right\}_{\iota\in[k]} \\ S_{\mathsf{priv}} &= \left\{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \{\mathsf{sim}_\tau, \mathsf{sim}_\tau^*\}_{\tau\in[n']}\right\}, \\ \widehat{S}_{\mathsf{priv}} &= \{\widehat{\mathsf{const}}_1, \widehat{\mathsf{coef}}_1, \widehat{\mathsf{const}}_2, \widehat{\mathsf{coef}}_2, \widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}. \end{aligned}$$

It generates $(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{IPFE.Setup}(S_{\mathsf{pub}}, S_{\mathsf{priv}})$ and $(\widehat{\mathsf{IPFE.MSK}}, \widehat{\mathsf{IPFE.MPK}}) \leftarrow \mathsf{IPFE.Setup}(\widehat{S}_{\mathsf{pub}}, \widehat{S}_{\mathsf{priv}})$. Finally, it returns $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$ and $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widehat{\mathsf{IPFE.MPK}})$.

KeyGen(MSK, f): Let $f = (f_1, \ldots, f_{n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$. Sample $\boldsymbol{\alpha}, \boldsymbol{\beta}_t \leftarrow \mathbb{Z}_p^k$ for $t \in [n']$ such that

$$\sum_{t\in[n']} \boldsymbol{\beta}_t[\iota] = 0 \mod p \text{ for all } \iota \in [k]$$

Next, sample independent random vectors $\boldsymbol{r}_t^{(\iota)} \leftarrow \mathbb{Z}_p^m$ and computes

$$(\boldsymbol{\ell}_{1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{m,t}^{(\iota)}, \boldsymbol{\ell}_{m+1,t}^{(\iota)}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}[\iota]\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \boldsymbol{\beta}_t[\iota]; \boldsymbol{r}_t^{(\iota)})$$

for all $\iota \in [k], t \in [n']$. Here we make use of the instantiation of the AKGS described in Section 3.6. From the description of that AKGS instantiation, we note that the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}^{(\iota)}$ would be of the form $\boldsymbol{\ell}_{m+1,t}^{(\iota)} = \boldsymbol{\alpha}[\iota]\boldsymbol{z}[t] - \boldsymbol{r}_t^{(\iota)}[m]$ where $\boldsymbol{\alpha}[\iota]$ is a constant. Also all the label functions $\boldsymbol{\ell}_{1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{m,t}^{(\iota)}$ involve only the variables $\boldsymbol{x}$ and not the variable $\boldsymbol{z}[t]$. Next, for all $j \in [m]$ and $t \in [n']$, it defines the vectors $\boldsymbol{v}_{j,t}$ corresponding to the label functions $\boldsymbol{\ell}_{j,t}^{(\iota)}$ obtained from the partial garbling above as

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	S_{priv}
$\boldsymbol{v}$	$\boldsymbol{\alpha}[\iota]$	0	0
$\boldsymbol{v}_{j,t}$	$\boldsymbol{\ell}_{j,t}^{(\iota)}[\mathsf{const}]$	$\boldsymbol{\ell}_{j,t}^{(\iota)}[\mathsf{coef}_i]$	0

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$	$\widehat{S}_{\mathsf{priv}}$
$\boldsymbol{v}_{m+1,t}$	$\boldsymbol{r}_t^{(\iota)}[m]$	$\boldsymbol{\alpha}[\iota]$	0

It generates the secret-keys as

$$\mathsf{IPFE.SK} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}]\!]_2)$$
$$\mathsf{IPFE.SK}_{j,t} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2) \text{ for } j \in [m], t \in [n']$$
$$\widehat{\mathsf{IPFE.SK}}_{m+1,t} \leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2) \text{ for } t \in [n']$$

It returns $\mathsf{SK}_f = (\mathsf{IPFE.SK}, \{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$.

$\mathsf{Enc}(\mathsf{MPK}, \boldsymbol{x} \in \mathbb{Z}_p^n, \boldsymbol{z} \in \mathbb{Z}_p^{n'})$: It samples $\boldsymbol{s} \leftarrow \mathbb{Z}_p^k$ and set the vectors

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$
$\boldsymbol{u}$	$\boldsymbol{s}[\iota]$	$\boldsymbol{s}[\iota]\boldsymbol{x}[i]$

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$
$\boldsymbol{h}_t$	$-\boldsymbol{s}[\iota]$	$\boldsymbol{s}[\iota]\boldsymbol{z}[t]$

for all $t \in [n']$. It encrypts the vectors as

$$\mathsf{IPFE.CT} \leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)$$
$$\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{IPFE.SlotEnc}(\widehat{\mathsf{IPFE.MPK}}, [\![\boldsymbol{h}_t]\!]_1) \text{ for } t \in [n']$$

and returns the ciphertext as $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

$\mathsf{Dec}((\mathsf{SK}_f, f), (\mathsf{CT}, \boldsymbol{x}))$: It parses the secret-key $\mathsf{SK}_f = (\mathsf{IPFE.MSK}, \{\mathsf{IPFE.MSK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.MSK}}_{m+1,t}\}_{t\in[n']})$ and the ciphertext $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$. It uses the decryption algorithm of IPFE to compute

$$[\![\mu]\!]_T = \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}, \mathsf{IPFE.CT})$$
$$[\![\ell_{j,t}]\!]_T = \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{j,t}, \mathsf{IPFE.CT}) \text{ for } j \in [m], t \in [n']$$
$$[\![\ell_{m+1,t}]\!]_T = \mathsf{IPFE.Dec}(\widehat{\mathsf{IPFE.SK}}_{m+1,t}, \widehat{\mathsf{IPFE.CT}}_t) \text{ for } t \in [n']$$

Next, it utilizes the evaluation procedure of AKGS and obtain a combined value

$$[\![\rho]\!]_T = \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T).$$

Finally, it returns a value ζ from a polynomially bounded set $\mathcal{P}$ such that $[\![\rho]\!]_T = [\![\mu]\!]_T \cdot [\![\zeta]\!]_T$; otherwise $\perp$.

Correctness: By the correctness of IPFE, AKGS and the linearity of the Eval function we have

$$\begin{aligned}&\mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T)\\ &= [\![\sum_{\iota=1}^{k} \boldsymbol{\alpha}[\iota]\boldsymbol{s}[\iota] \cdot f_t(\boldsymbol{x})\boldsymbol{z}[t] + \boldsymbol{\beta}_t[\iota]\boldsymbol{s}[\iota]]\!]_T\\ &= [\![\boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot f_t(\boldsymbol{x})\boldsymbol{z}[t] + \boldsymbol{\beta}_t \cdot \boldsymbol{s}]\!]_T\end{aligned}$$

Therefore, $[\![\rho]\!]_T = [\![\sum_{t=1}^{n'} \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot f_t(\boldsymbol{x})\boldsymbol{z}[t] + \boldsymbol{\beta}_t \cdot \boldsymbol{s}]\!]_T = [\![\boldsymbol{\alpha} \cdot \boldsymbol{s} f(\boldsymbol{x})^\top \boldsymbol{z}]\!]_T$ since $\sum_{t\in[n']} \boldsymbol{\beta}_t[\iota] = 0 \bmod p$ for all $\iota \in [k]$. Also, by the correctness of IPFE we see that $[\![\mu]\!]_T = [\![\boldsymbol{\alpha} \cdot \boldsymbol{s}]\!]_T$ and hence $[\![\zeta]\!]_T = [\![f(\boldsymbol{x})^\top \boldsymbol{z}]\!]_T \in \mathcal{P}$.

Remark 3 (Multi-Ciphertext Scheme) The one-slot FE scheme Π_{one} described above is secure against adversaries that are restricted to query a single ciphertext. However, we can easily modify the FE scheme to another FE that is secure for any a-priori bounded number of ciphertext queries from the adversary's end. For the extension, we introduce additional $(2n'+2)q_{\mathsf{CT}}$ private slots on each ciphertext and decryption key sides, where q_{CT} denotes the number of ciphertext queries. More specifically, we add $2n'q_{\mathsf{CT}}$ and $2q_{\mathsf{CT}}$ dimensional hidden slots to $\mathcal{S}_{\mathsf{priv}}$ and $\widehat{\mathcal{S}}_{\mathsf{priv}}$ respectively to handle the q_{CT} ciphertext queries during the security reduction. Consequently, the sizes of system parameters, secret-keys and ciphertext would grow linearly with q_{CT}. A similar strategy can be followed to convert our extended one-slot FE scheme (of Section B) that only supports a single ciphertext query to one that is secure for any a-priori bounded number of ciphertext queries.

5.1 Security Analysis

Theorem 3 *The one slot* FE *scheme* Π_{one} *for attribute-weighted sum is adaptively simulation-secure assuming the* AKGS *is piecewise secure as per Definition 7, the* MDDH_k *assumption holds in group* $\mathbb{G}_2$ *as per Assumption 1, and the slotted* IPFE *is function hiding as per Definition 5.*

The Simulator We describe the simulator for the one slot FE scheme Π_{one}.

$\mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'})$: To generate the master public/secret keys, it executes as follows:

1. Define the following index sets as follows

$$\begin{aligned}
S_{\mathsf{pub}} &= \left\{\{\mathsf{const}^{(\iota)}\}_{\iota\in[k]}, \{\mathsf{coef}_i^{(\iota)}\}_{\iota\in[k],i\in[n]}\right\},\\
\widehat{S}_{\mathsf{pub}} &= \left\{\widehat{\mathsf{const}}^{(\iota)}, \widehat{\mathsf{coef}}^{(\iota)}\right\}_{\iota\in[k]}\\
S_{\mathsf{priv}} &= \left\{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \{\mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}_{\tau\in[n']}\right\},\\
\widehat{S}_{\mathsf{priv}} &= \{\widehat{\mathsf{const}}_1, \widehat{\mathsf{coef}}_1, \widehat{\mathsf{const}}_2, \widehat{\mathsf{coef}}_2, \widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}.
\end{aligned}$$

2. It generates $(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{IPFE.Setup}(S_{\mathsf{pub}}, S_{\mathsf{priv}})$ and $(\widehat{\mathsf{IPFE.MSK}}, \widehat{\mathsf{IPFE.MPK}}) \leftarrow \mathsf{IPFE.Setup}(\widehat{S}_{\mathsf{pub}}, \widehat{S}_{\mathsf{priv}})$.
3. It outputs $\mathsf{MSK}^* = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$ and $\mathsf{MPK}^* = (\mathsf{IPFE.MPK}, \widehat{\mathsf{IPFE.MPK}})$.

$\mathsf{KeyGen}_0^*(\mathsf{MSK}^*, \boldsymbol{f_q})$: On input MSK^*, a function $f_q = (f_{q,1}, \ldots f_{q,n'})$ for $q \in [Q_{\mathsf{pre}}]$ the simulator proceeds as follows:
Setting Public Positions: The public positions are set as in the original scheme.

1. It first samples $\boldsymbol{\beta}_{q,t} = (\boldsymbol{\beta}_{q,t}[1], \ldots, \boldsymbol{\beta}_{q,t}[k]) \leftarrow \mathbb{Z}_p^k$ and $\boldsymbol{r}_{q,t}^{(\iota)} = (\boldsymbol{r}_{q,t}^{(\iota)}[1], \ldots, \boldsymbol{r}_{q,t}^{(\iota)}[m_q]) \leftarrow \mathbb{Z}_p^{m_q}$ where it holds that
$$\sum_{t\in[n']} \boldsymbol{\beta}_{q,t}[\iota] = 0 \mod p \text{ for all } \iota \in [k].$$
2. Next, it computes the coefficient vectors for the label functions as
$$(\boldsymbol{\ell}_{q,1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{q,m_q,t}^{(\iota)}, \boldsymbol{\ell}_{q,m_q+1,t}^{(\iota)}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t]f_{q,t}(\boldsymbol{x}^*) + \boldsymbol{\beta}_{q,t}[\iota]; \boldsymbol{r}_{q,t}^{(\iota)})$$
for all $\iota \in [k], t \in [n']$. From the description of AKGS, we note that the (m_q+1)-th label function $\boldsymbol{\ell}_{q,m_q+1,t}^{(\iota)}$ would be of the form $\boldsymbol{\ell}_{q,m_q+1,t}^{(\iota)} = \boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t] - \boldsymbol{r}_{q,t}^{(\iota)}[m_q]$.

3. It picks $\boldsymbol{\alpha}_q \leftarrow \mathbb{Z}_p^k$ and sets the public positions at the indexes in $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ of following vectors

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$
$\boldsymbol{v}_q$	$\boldsymbol{\alpha}_q[\iota]$	0
$\boldsymbol{v}_{q,j,t}$	$\boldsymbol{\ell}_{q,j,t}^{(\iota)}[\mathsf{const}]$	$\boldsymbol{\ell}_{q,j,t}^{(\iota)}[\mathsf{coef}_i]$

for all $j \in [m_q]$ and $t \in [n']$. It also sets the following vectors

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$
$\boldsymbol{v}_{q,m_q+1,t}$	$\boldsymbol{r}_{q,t}^{(\iota)}[m_q]$	$\boldsymbol{\alpha}_q[\iota]$

for all $t \in [n']$.

Setting Private Positions:

4. It samples $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ for $t \in [n']$ satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0$.
5. Next, it picks $\widetilde{\boldsymbol{r}}_{q,t} \leftarrow \mathbb{Z}_p^{m_q}$ and computes the coefficient vectors for the label functions as

$$(\widetilde{\boldsymbol{\ell}}_{q,1,t}, \ldots, \widetilde{\boldsymbol{\ell}}_{q,m_q,t}, \widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\widetilde{\alpha}_q \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\beta}_{q,t}; \widetilde{\boldsymbol{r}}_{q,t}).$$

for all $t \in [n']$. From the description of AKGS, we note that the (m_q+1)-th label function $\widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}$ would be of the form $\widetilde{\boldsymbol{\ell}}_{q,m_q+1,t} = \widetilde{\alpha}_q \boldsymbol{z}^*[t] - \widetilde{\boldsymbol{r}}_{q,t}[m_q]$.
6. Now, it fills the private positions at the indexes in $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ as follows

vector	const	coef_i	sim_τ	sim_τ^*
$\boldsymbol{v}_q$	$\widetilde{\alpha}_q$	0	0	0
$\boldsymbol{v}_{q,j,t}$	$\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]$	$\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]$	0	0

for all $j \in [m_q]$ and $t \in [n']$; and

vector	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{q,m_q+1,t}$	0	0	$\widetilde{\boldsymbol{r}}_{q,t}[m_q]$	$\widetilde{\alpha}_q$	0	0	0

for all $t \in [n']$.
7. It generates the IPFE secret-keys as

$$\mathsf{IPFE.SK}_q \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_q]\!]_2)$$
$$\mathsf{IPFE.SK}_{q,j,t} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{q,j,t}]\!]_2) \text{ for } j \in [m_q], t \in [n']$$
$$\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} \leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{q,m_q+1,t}]\!]_2) \text{ for } t \in [n']$$

8. Finally, it returns $\mathsf{SK}_{f_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$.

$\mathbf{Enc^*(MPK^*, MSK^*, \boldsymbol{x}^*, \mathcal{V})}$: On input MSK^*, a vector $\boldsymbol{x}^* \in \mathbb{Z}_p^n$ and a set $\mathcal{V} = \{(f_q, f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^*) : q \in [Q_{\mathsf{pre}}]\}$ the simulator executes the following steps:

1. It samples a dummy vector $\boldsymbol{d}$ from the set
$$D = \{\boldsymbol{d} \in \mathbb{Z}_p^{n'} : f_q(\boldsymbol{x}^*)^\top \boldsymbol{d} = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* \text{ for all } q \in [Q_{\mathsf{pre}}]\}.$$
The simulator does this by finding a random vector $\boldsymbol{d} \in \mathbb{Z}_p^{n'}$ such that $\sum_{t\in[n']} f_{q,t}(\boldsymbol{x}^*)\boldsymbol{d}[t] = \sum_{t\in[n']} f_{q,t}(\boldsymbol{x}^*)\boldsymbol{z}^*[t]$ for all $q \in [Q_{\mathsf{pre}}]$. Hence, D is identical to the set $D_{\mathsf{IP}} = \{\boldsymbol{d} \in \mathbb{Z}_p^{n'} : (f_{q,1}(\boldsymbol{x}^*), \ldots, f_{q,n'}(\boldsymbol{x}^*)) \cdot (\boldsymbol{d}[1], \ldots, \boldsymbol{d}[n']) = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* \text{ for all } q \in [Q_{\mathsf{pre}}]\}$. A vector $\boldsymbol{d}$ from a set of the form D_{IP} can be efficiently sampled via a polynomial time algorithm given by O'Neill [O'N10] as noted earlier. Therefore, given $\boldsymbol{x}^*$ and $\mathcal{V}$, the simulator can find a dummy vector $\boldsymbol{d}$ such that $f_q(\boldsymbol{x}^*)^\top \boldsymbol{d} = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$ holds for every $q \in [Q_{\mathsf{pre}}]$.
2. Next, it sets the following vectors

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	const	coef_i	sim_τ	sim^*_τ
$\boldsymbol{u}$	0	0	1	$\boldsymbol{x}^*[i]$	0	0

and

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$
$\boldsymbol{h}_t$	0	0

vector	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{h}_t$	1	0	-1	$\boldsymbol{d}[t]$	0	0	0

for all $t \in [n']$.
3. It encrypts the vectors as
$$\mathsf{IPFE.CT} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)$$
$$\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{IPFE.Enc}(\widehat{\mathsf{IPFE.MPK}}, [\![\boldsymbol{h}_t]\!]_1) \text{ for } t \in [n']$$
4. It returns the ciphertext as $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

$\mathbf{KeyGen_1^*(MSK^*, \boldsymbol{x}^*, \boldsymbol{f_q}, \boldsymbol{f_q}(\boldsymbol{x}^*)^\top \boldsymbol{z}^*)}$: On input MSK^*, $\boldsymbol{x}^* \in \mathbb{Z}_p^n$, a function $f_q = (f_{q,1}, \ldots f_{q,n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ for $q \in [Q_{\mathsf{pre}} + 1, Q]$ and $f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* \in \mathbb{Z}_p$ the simulator proceeds as follows:

Setting Public Positions:

1. The simulator sets the public positions at the indexes in $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ analogous to $\mathsf{KeyGen}_0^*(\mathsf{MSK}^*, f_q)$.

Setting Private Positions:

2. First, it samples a random elements $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$, for $t \in [n']$, satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0$ and then runs the simulator of the AKGS to obtain
$$(\widehat{\ell}_{q,1,1}, \ldots, \widehat{\ell}_{q,m_q,1}, \widehat{\ell}_{q,m_q+1,1}) \leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, \widetilde{\alpha}_q \cdot f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \widetilde{\beta}_{q,1})$$
$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \widetilde{\beta}_{q,t}) \quad \text{for } 1 < t \leq n'.$$

3. Next, it fills the private positions at the indices in $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ as follows

vector	const	coef_i	sim_τ	sim^*_τ
$\boldsymbol{v}_q$	$\widetilde{\alpha}_q$	0	0	0
$\boldsymbol{v}_{q,j,t}$	$\widehat{\ell}_{q,j,t}$	0	0	0

for all $j \in [m_q]$ and $t \in [n']$; and

vector	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{q,m_q+1,t}$	$\widehat{\ell}_{q,m_q+1,t}$	0	0	0	0	0	0

for all $t \in [n']$.

4. It generates the IPFE secret-keys as

$$\mathsf{IPFE.SK}_q \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_q]\!]_2)$$
$$\mathsf{IPFE.SK}_{q,j,t} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{q,j,t}]\!]_2) \text{ for } j \in [m_q], t \in [n']$$
$$\widehat{\mathsf{IPFE.MSK}}_{q,m_q+1,t} \leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{q,m_q+1,t}]\!]_2) \text{ for } t \in [n']$$

5. It outputs $\mathsf{SK}_{f_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$.

Hybrids and Reductions

Proof. We employ a sequence of hybrid experiments to demonstrate the indistinguishability between the real experiment $\mathsf{Expt}^{\mathsf{Real,\ FE}}_{\mathcal{A}}(1^\lambda)$ and the ideal experiment $\mathsf{Expt}^{\mathsf{Ideal,\ FE}}_{\mathcal{A}}(1^\lambda)$ with the simulator described above where $\mathcal{A}$ is any PPT adversary. In each experiment, $\mathcal{A}$ can query a polynomial number of secret-key queries for functions $f_q \in \mathcal{F}^{(n,n')}_{\mathsf{ABP}}$, both before and after submitting the challenge message $(\boldsymbol{x}^*, \boldsymbol{z}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$. Let Q be the total number of secret-key queries and Q_{pre} $(< Q)$ be the number of secret-keys queried before making the challenge message. We denote the q-th secret-key by SK_{f_q} corresponding to a function f_q. For the ease of presentation, we write the vector elements sitting in the public slots $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ in *blue* color and the vector elements sitting in the private slots $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ in *red* color. More precisely, we do this so that while describing the hybrid games, we sometimes omit the public parts of the vectors and write down only the private parts when the changes occur only in the private parts. Now, we describe the hybrids as follows:

Hybrid H_0: This is the real experiment $\mathsf{Expt}^{\mathsf{Real,\ FE}}_{\mathcal{A}}(1^\lambda)$ defined in Definition 4 (with single slot, i.e., $N = 1$). For any $q \in [Q]$, the q-th secret-key $\mathsf{SK}_{f_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}$, $\{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$ is associated with the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ given by

$$\boldsymbol{v}_q = (\quad \boldsymbol{\alpha}_q[\iota], \qquad 0, \quad 0, 0, 0, 0\ ),$$
$$\boldsymbol{v}_{q,j,t} = (\ \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], 0, 0, 0, 0\ ),$$
$$\boldsymbol{v}_{q,m_q+1,t} = (\ \boldsymbol{r}^{(\iota)}_{q,t}[m_q], \boldsymbol{\alpha}_q[\iota], 0, 0, 0, 0, 0, 0, 0\ ).$$

for $j \in [m_q]$ and $t \in [n']$. Note that $\boldsymbol{\alpha}_q$ and $\boldsymbol{r}_{q,t}^{(\iota)}$ are random vectors sampled from $\mathbb{Z}_p^k$ and $\mathbb{Z}_p^{m_q}$ respectively. For all $t \in [n']$, the garblings are computed as

$$(\boldsymbol{\ell}_{q,1,t}, \ldots, \boldsymbol{\ell}_{q,m_q,t}, \boldsymbol{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t]f_{q,t}(\boldsymbol{x}^*) + \boldsymbol{\beta}_{q,t}[\iota]; \boldsymbol{r}_{q,t}^{(\iota)})$$

where $f_q = (f_{q,1}, \ldots, f_{q,n'})$ and $\boldsymbol{\beta}_{q,t} \leftarrow \mathbb{Z}_p^k$ with $\sum_{t\in[n']} \boldsymbol{\beta}_{q,t}[\iota] = 0 \mod p\ \forall \iota \in [k]$. The challenge ciphertext $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$ corresponds to $(\boldsymbol{x}^*, \boldsymbol{z}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$ is associated with the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ given by

$$\begin{aligned}\boldsymbol{u} &= (\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{x}^*[i], \perp, \perp, \perp, \perp),\\ \boldsymbol{h}_t &= (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t], \perp, \perp, \perp, \perp, \perp, \perp, \perp),\end{aligned}$$

for $t \in [n']$ and $\boldsymbol{s} \leftarrow \mathbb{Z}_p^k$. Note that, in the real experiment, CT^* is computed using $\mathsf{IPFE.SlotEnc}$ and therefore the elements sitting at the indexes in S_{priv} are set as $\perp$ for the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$.

Hybrid H_1: It is exactly the same as hybrid H_0 except the fact that here the challenge ciphertext CT^* is generated using $\mathsf{IPFE.Enc}$ using $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$. As a result the private positions of $\boldsymbol{u}$ and $\boldsymbol{h}_t$ (in CT^*) are changed from $\perp$ to 0. Thus the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ become

$$\begin{aligned}\boldsymbol{u} &= (\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{x}^*[i], \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}),\\ \boldsymbol{h}_t &= (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t], \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}).\end{aligned}$$

The slot-mode correctness of IPFE guarantees that the two hybrids H_0 and H_1 are identically distributed.

Hybrid H_2: This hybrid is similar to H_1 except that in the private slots of the vectors used to compute SK_{f_q}, we put one single garbling that linearly combines k garblings with weight vector $\boldsymbol{s} \in \mathbb{Z}_p^k$ instead of using k independent garblings associated to each index $j \in [m_q]$ of the vectors $\boldsymbol{v}_{q,j,t}$ and a single random element combining the weight vector $\boldsymbol{s}$ in $\boldsymbol{v}_q$ instead of using a random vector $\boldsymbol{\alpha}_q$. Accordingly, we modify the challenge ciphertext CT^* by omitting the weight vector $\boldsymbol{s}$ and setting the public slots to zero of the vectors $\boldsymbol{u}, \boldsymbol{h}_t$ to ensure the inner products computed at the time of decryption remains the same in both the hybrids.

In H_1, the public slots of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ are occupied by a vector $\boldsymbol{\alpha}_q \in \mathbb{Z}_p^k$ and the garblings $\boldsymbol{\ell}_{q,j,t}^{(\iota)}$ computed using randomness $\boldsymbol{r}_{q,t}^{(\iota)} \in \mathbb{Z}_p^{m_q}$. In the public slots of the vectors $\boldsymbol{u}, \boldsymbol{h}_t$, we use $(\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{x}^*[i]), (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t])$ respectively. Therefore, the IPFE decryption lets us recover $[\![\mu_q]\!]_T, [\![\ell_{q,j,t}]\!]_T$ such that

$$\mu_q = \boldsymbol{\alpha}_q \cdot \boldsymbol{s} = \overline{\alpha}_q \text{ (say)},$$

$$\begin{aligned}\ell_{q,j,t} &= (\boldsymbol{\ell}_{q,j,t}^{(1)}, \ldots, \boldsymbol{\ell}_{q,j,t}^{(k)}) \cdot (\boldsymbol{s}[1](1, \boldsymbol{x}^*), \ldots, \boldsymbol{s}[k](1, \boldsymbol{x}^*))\\ &= (\boldsymbol{s}[1]\boldsymbol{\ell}_{q,j,t}^{(1)}, \ldots, \boldsymbol{s}[k]\boldsymbol{\ell}_{q,j,t}^{(k)}) \cdot ((1, \boldsymbol{x}^*), \ldots, (1, \boldsymbol{x}^*))\\ &= \overline{\boldsymbol{\ell}}_{q,j,t} \cdot (1, \boldsymbol{x}^*)\end{aligned}$$

where $\overline{\boldsymbol{\ell}}_{q,j,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{\ell}_{q,j,t}^{(\iota)}$ for all $j \in [m_q]$ and $t \in [n']$. Similarly, the $(m_q + 1)$-th garbling returns

$$\begin{aligned}\ell_{q,m_q+1,t} &= ((\boldsymbol{r}_{q,t}^{(1)}[m_q], \boldsymbol{\alpha}_q[1]), \ldots, (\boldsymbol{r}_{q,t}^{(k)}[m_q], \boldsymbol{\alpha}_q[k])) \cdot (\boldsymbol{s}[1](-1, \boldsymbol{z}^*[t]), \ldots, \boldsymbol{s}[k](-1, \boldsymbol{z}^*[t]))\\ &= (\boldsymbol{s}[1](\boldsymbol{r}_{q,t}^{(1)}[m_q], \boldsymbol{\alpha}_q[1]), \ldots, \boldsymbol{s}[k](\boldsymbol{r}_{q,t}^{(k)}[m_q], \boldsymbol{\alpha}_q[k])) \cdot ((-1, \boldsymbol{z}^*[t]), \ldots, (-1, \boldsymbol{z}^*[t]))\\ &= (\overline{\boldsymbol{r}}_{q,t}[m_q], \overline{\alpha}_q) \cdot (-1, \boldsymbol{z}^*[t])\end{aligned}$$

where $\overline{\boldsymbol{r}}_{q,t}[m_q] = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{r}^{(\iota)}_{q,t}[m_q]$. In H_2, we use $\overline{\alpha}_q, \overline{\boldsymbol{\ell}}_{q,j,t}$ and $\overline{\boldsymbol{r}}_{q,t}[m_q]$ in the private slots of the vectors associated to SK_{f_q} as described below

$$
\begin{aligned}
\boldsymbol{v}_q &= (\quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad \boxed{\overline{\alpha}_q}, \quad 0, \quad 0, 0\ ),\\
\boldsymbol{v}_{q,j,t} &= (\ \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]}, \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]}, 0, 0\ ),\\
\boldsymbol{v}_{q,m_q+1,t} &= (\boldsymbol{r}^{(\iota)}_{q,t}[m_q], \boldsymbol{\alpha}_q[\iota], \boxed{\overline{\boldsymbol{r}}_{q,t}[m_q]}, \boxed{\overline{\alpha}_q}, 0, 0, 0, 0, 0)
\end{aligned}
$$

Since the weight vector $\boldsymbol{s}$ is not required to generate the challenge ciphertext CT^*, we omit it in the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$. Moreover, the public slots of $\boldsymbol{u}$ and $\boldsymbol{h}_t$ are set to zero as the inner product is computed through the private slots only. We describe the changes below.

$$
\begin{aligned}
\boldsymbol{u} &= (\boxed{0}, \boxed{0}, \boxed{1}, \boxed{\boldsymbol{x}^*[i]}, 0, 0),\\
\boldsymbol{h}_t &= (\boxed{0}, \boxed{0}, \boxed{-1}, \boxed{\boldsymbol{z}^*[t]}, 0, 0, 0, 0, 0),
\end{aligned}
$$

Finally, we observe that the inner products $\boldsymbol{v}_q \cdot \boldsymbol{u}, \boldsymbol{v}_{q,j,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ for all $j \in [m_q], t \in [n']$ remain the same as in H_1. Thus, the function hiding property of IPFE preserves the indistinguishability between the hybrids H_1 and H_2.

Note that, in this hybrid we pick $\boldsymbol{\alpha}_q, \boldsymbol{\beta}_{q,t}, \boldsymbol{s} \leftarrow \mathbb{Z}_p^k$ and $\boldsymbol{r}^{(\iota)}_{q,t} \leftarrow \mathbb{Z}_p^{m_q}$ for all $t \in [n'], \iota \in [k]$ satisfying $\sum_{t\in[n']} \boldsymbol{\beta}_{q,t}[\iota] = 0 \mod p$ for each $\iota \in [k]$. Then, the linearity of the Garble algorithm allows us to write

$$
(\overline{\boldsymbol{\ell}}_{q,1,t}, \ldots, \overline{\boldsymbol{\ell}}_{q,m_q,t}, \overline{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\overline{\alpha}_q \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \overline{\beta}_{q,t}; \overline{\boldsymbol{r}}_{q,t})
$$

where $\overline{\boldsymbol{\ell}}_{q,j,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{\ell}^{(\iota)}_{q,j,t}, \overline{\boldsymbol{r}}_{q,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{r}^{(\iota)}_{q,t}$ and $\overline{\beta}_{q,t} = \boldsymbol{\beta}_{q,t} \cdot \boldsymbol{s}$.

Hybrid H_3: It is analogous to H_2 except the liner combinations $\overline{\alpha}_q, \overline{\boldsymbol{\ell}}_{q,j,t}, \overline{\boldsymbol{r}}_{q,t}$ in the private slots of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}, \boldsymbol{v}_{q,m_q+1,t}$ are replaced with freshly and independently generated random values and garblings $\widetilde{\alpha}_q, \widetilde{\boldsymbol{\ell}}_{q,j,t}, \widetilde{\boldsymbol{r}}_{q,t}$. More specifically, we sample random elements $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ for all $t \in [n']$ such that $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0 \mod p$ and a vector $\widetilde{\boldsymbol{r}}_{q,t} \leftarrow \mathbb{Z}_p^{m_q}$. Then, the garblings are computed as

$$
(\widetilde{\boldsymbol{\ell}}_{q,1,t}, \ldots, \widetilde{\boldsymbol{\ell}}_{q,m_q,t}, \widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\widetilde{\alpha}_q \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\beta}_{q,t}; \widetilde{\boldsymbol{r}}_{q,t})
$$

for all $t \in [n']$. The vectors involved in SK_{f_q} are modified as follows:

$$
\begin{aligned}
\boldsymbol{v}_q &= (\quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad \boxed{\widetilde{\alpha}_q}, \quad 0, \quad 0, 0\ ),\\
\boldsymbol{v}_{q,j,t} &= (\ \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]}, \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]}, 0, 0\ ),\\
\boldsymbol{v}_{q,m_q+1,t} &= (\boldsymbol{r}^{(\iota)}_{q,t}[m_q], \boldsymbol{\alpha}_q[\iota], \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, \boxed{\widetilde{\alpha}_q}, 0, 0, 0, 0, 0)
\end{aligned}
$$

Recall that in H_2, the following linear combinations

$$
\overline{\alpha}_q = \boldsymbol{\alpha}_q \cdot \boldsymbol{s}, \quad \overline{\beta}_{q,t} = \boldsymbol{\beta}_{q,t} \cdot \boldsymbol{s}, \quad \overline{\boldsymbol{r}}_{q,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{r}^{(\iota)}_{q,t}
$$

where a common weight vector $\boldsymbol{s}$ has been used to set $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$. On the other hand, in H_3, fresh and independent random elements $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t}, \widetilde{\boldsymbol{r}}_{q,t}$ are used to compute SK_{f_q}. Note that the elements of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ are only used in the exponent of the source group $\mathbb{G}_2$ while generating the IPFE secret-keys. Let us consider the matrix $\mathbf{A}_{q,t} = (\boldsymbol{\alpha}_q \parallel \boldsymbol{\beta}_{q,t} \parallel (\mathbf{R}_{q,t})^\top) \in \mathbb{Z}_p^{k\times(m_q+2)}$ where $\mathbf{R}_{q,t} = (\boldsymbol{r}_{q,t}^{(1)} \parallel \ldots \parallel \boldsymbol{r}_{q,t}^{(k)}) \in \mathbb{Z}_p^{m_q \times k}$. Since the matrix $\mathbf{A}_{q,t}$ is uniformly chosen from $\mathbb{Z}_p^{k\times(m_q+2)}$ and $\boldsymbol{s}$ is uniform over $\mathbb{Z}_p^k$, by the MDDH_k assumption in group $\mathbb{G}_2$ we have

$$\underbrace{(\llbracket \mathbf{A}_{q,t} \rrbracket_2, \llbracket \boldsymbol{s}^\top \mathbf{A}_{q,t} \rrbracket_2)}_{\text{in } \mathsf{H}_2} \overset{c}{\approx} \underbrace{(\llbracket \mathbf{A}_{q,t} \rrbracket_2, \llbracket (\widetilde{\alpha}_q, \widetilde{\beta}_{q,t}, \widetilde{\boldsymbol{r}}_{q,t}) \rrbracket_2)}_{\text{in } \mathsf{H}_3}$$

holds for all $q \in [Q]$ and $t \in [n']$. Hence, the two hybrids H_2 and H_3 are indistinguishable under the MDDH_k assumption with $k < m_q + 2$.

Hybrid H_4: It is exactly the same as hybrid H_3 except we change the way the vectors $\boldsymbol{h}_t$ for all $t \in [n']$ are computed while producing the challenge ciphertext. After all the pre-challenge secret-key queries made by $\mathcal{A}$, a dummy vector $\boldsymbol{d}$ is picked from the set

$$D = \{\boldsymbol{d} \in \mathbb{Z}_p^{n'} : f_q(\boldsymbol{x}^*)^\top \boldsymbol{d} = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* \text{ for all } q \in [Q_{\mathsf{pre}}]\}$$

via an efficient algorithm proposed in [O'N10], and then the vectors $\boldsymbol{u}, \boldsymbol{h}_t$ associated with the ciphertext are defined as below.

$$\begin{aligned}
\boldsymbol{u} &= (0, 0, 1, \boldsymbol{x}^*[i], 0, 0),\\
\boldsymbol{h}_t &= (0, 0, -1, \boldsymbol{z}^*[t], \boxed{-1}, \boxed{\boldsymbol{d}[t]}, \boxed{-1}, \boxed{\boldsymbol{z}^*[t]}, 0),
\end{aligned}$$

Note that, these changes in $\boldsymbol{h}_t$ have no effect in the final inner product between $\boldsymbol{v}_{q,m_q+1,t}$ and $\boldsymbol{h}_t$ since the slots $(\widehat{\mathsf{const}}_2, \widehat{\mathsf{coef}}_2, \widehat{\mathsf{const}}, \widehat{\mathsf{coef}})$ where the changes take place in $\boldsymbol{h}_t$ correspond to zero entries in $\boldsymbol{v}_{q,m_q+1,t}$. Therefore, by the function hiding property of IPFE, the hybrids H_3 and H_4 remain indistinguishable to the adversary.

From the next hybrid we will modify the pre-challenge secret-key queries and the challenge ciphertext so that the decryption results become $f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}$ for all $q \in [Q_{\mathsf{pre}}]$ for some vector $\boldsymbol{d} \in \mathbb{Z}_p^{n'}$. Note that, $\boldsymbol{d}$ is a dummy vector which is sampled from $\mathbb{Z}_p^{n'}$ such that $f_q(\boldsymbol{x}^*)^\top \boldsymbol{d} = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$ for all $q \in [Q_{\mathsf{pre}}]$. This is done through a loop of hybrids described below.

Hybrid $\mathsf{H}_{5,q}$ ($q \in [Q_{\mathsf{pre}}]$): It proceeds similar to H_4 except that for each $1 \le q' \le q$, we modify the vector $\boldsymbol{v}_{q,m_q+1,t}$ as described below.

$$\begin{aligned}
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\quad 0, \quad 0, \boxed{\widetilde{\boldsymbol{r}}_{q',t}[m_{q'}]}, \boxed{\widetilde{\alpha}_{q'}}, 0, 0, 0\,) && \text{for } q' \le q\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\,\widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \; 0, 0, 0\,) && \text{for } q < q' \le Q_{\mathsf{pre}}
\end{aligned}$$

Note that, the post-challenge secret-key queries are still answered according to H_4. Observe that $\mathsf{H}_{5,0}$ coincides with H_4. We will prove that $\mathsf{H}_{5,(q-1)}$ and $\mathsf{H}_{5,q}$ are indistinguishable via the following sequence of sub-hybrids, namely $\{\mathsf{H}_{5,q,1}, \mathsf{H}_{5,q,2}, \mathsf{H}_{5,q,3}\}$.

Hybrid $\mathsf{H}_{5,q,1}$ ($q \in [Q_{\mathsf{pre}}]$): It is analogous to $\mathsf{H}_{5,(q-1)}$ except that in the qth secret-key query the vector $\boldsymbol{v}_{q,m_q+1,t}$ is modified as follows.

$$\begin{aligned}
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\quad 0, \quad 0, \ \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \ 0\ ) \quad \text{for } q' < q\\
\boldsymbol{v}_{q,m_q+1,t} &= (\quad \boxed{0}, \quad \boxed{0}, \quad 0, \quad 0, \ \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, \boxed{\widetilde{\alpha}_q}, 0\ ),\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\ \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \quad 0, \quad 0, \ 0\ ) \quad \text{for } q < q' \leq Q_{\mathsf{pre}}
\end{aligned}$$

We observe that this change in $\boldsymbol{v}_{q,m_q+1,t}$ has no effect in the inner product $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ for all $t \in [n']$. Therefore, the function hiding security of IPFE ensures that the hybrids $\mathsf{H}_{5,(q-1)}$ and $\mathsf{H}_{5,q,1}$ are indistinguishable.

In this hybrid, the positions of $\boldsymbol{v}_{q,j,t}|_{S_{\mathsf{priv}}}$ and $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{sim}}]$, $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{sim}}^*]$ are exactly the same as in the secret-key of our 1-FE scheme. Similarly, in the case of the challenge ciphertext, the positions of $\boldsymbol{u}|_{S_{\mathsf{priv}}}$ and $\boldsymbol{h}_t[\widehat{\mathsf{const}}], \boldsymbol{h}_t[\widehat{\mathsf{coef}}], \boldsymbol{h}_t[\widehat{\mathsf{sim}}], \boldsymbol{h}_t[\widehat{\mathsf{sim}}^*]$ are also identical to the ciphertext of our 1-FE scheme.

Hybrid $\mathsf{H}_{5,q,2}$ ($q \in [Q_{\mathsf{pre}}]$): It is exactly the same as $\mathsf{H}_{5,q,1}$ except that the position $\boldsymbol{h}_t[\widehat{\mathsf{coef}}]$ is changed from $\boldsymbol{z}^*[t]$ to $\boldsymbol{d}[t]$ as shown below.

$$\begin{aligned}
\boldsymbol{u} &= (0, 0, 1, \boldsymbol{x}^*[i], 0, 0),\\
\boldsymbol{h}_t &= (0, 0, -1, \boldsymbol{z}^*[t], -1, \boldsymbol{d}[t], -1, \boxed{\boldsymbol{d}[t]}, 0),
\end{aligned}$$

All the secret-keys are answered as in the previous hybrid. The indistinguishability follows from the security of our 1-FE scheme. We note that the security of our 1-FE relies on the function hiding security of IPFE and the security of AKGS. In particular, we use the security of IPFE and AKGS to reversely sample the first label and make all the other labels random as shown below

$$\begin{aligned}
&\widetilde{\ell}_{q,1,1} \leftarrow \mathsf{RevSamp}(f_{q,1}, \boldsymbol{x}^*, \widetilde{\alpha}_q f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \widetilde{\beta}_{q,1}, \ell_{2,1}, \ldots, \ell_{m_q+1,1})\\
&\widetilde{\ell}_{q,1,\tau} \leftarrow \mathsf{RevSamp}(f_{q,\tau}, \boldsymbol{x}^*, \widetilde{\beta}_{q,\tau}, \ell_{2,\tau}, \ldots, \ell_{m_q+1,\tau}) \quad \text{for } 1 < \tau \leq n',
\end{aligned}$$

where $\sum_{\tau \in [n']} \widetilde{\beta}_{q,\tau} = 0$. Then, the dummy vector $\boldsymbol{d}$ replaces $\boldsymbol{z}^*$ while computing $\widetilde{\ell}_{q,1,1}$ and $\boldsymbol{d}[t]$ is placed at $\boldsymbol{h}_t[\widehat{\mathsf{coef}}]$. Finally, we move in the reverse direction so that the vectors $\boldsymbol{v}_{q,j,t}$ for all $j \in [m_q]$ and $t \in [n']$ are back in the form as they were in $\mathsf{H}_{5,q,1}$. Note that, the hybrids involved in our 1-FE scheme uses the positions $\mathsf{sim}_\tau, \mathsf{sim}^*_\tau, \widehat{\mathsf{sim}}, \widehat{\mathsf{sim}}^*$ of the vectors $\boldsymbol{v}_{q,j,t}, \boldsymbol{u}$ and $\boldsymbol{h}_t$, which does not effect the decryption using any post-challenge secret-key.

Hybrid $\mathsf{H}_{5,q,3}$ ($q \in [Q_{\mathsf{pre}}]$): It proceeds analogously to $\mathsf{H}_{5,q,2}$ except that we change $\boldsymbol{v}_{q,m_q+1,t}$ and $\boldsymbol{h}_t$ as below.

$$\begin{aligned}
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\quad 0, \quad 0, \ \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \ \widetilde{\alpha}_{q'}, \ 0, \ 0, \ 0\ ) \quad \text{for } q' < q\\
\boldsymbol{v}_{q,m_q+1,t} &= (\quad 0, \quad 0, \ \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, \boxed{\widetilde{\alpha}_q}, \boxed{0}, \boxed{0}, 0\ ),\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\ \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \ 0, \ 0, \ 0\ ) \quad \text{for } q < q' \leq Q_{\mathsf{pre}}\\
\boldsymbol{u} &= (1, \boldsymbol{x}^*[i], 0, 0)\\
\boldsymbol{h}_t &= (-1, \boldsymbol{z}^*[t], -1, \boldsymbol{d}[t], -1, \boxed{\boldsymbol{z}^*[t]}, 0)
\end{aligned}$$

Note that the inner product $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ remains the same as in $\mathsf{H}_{5,q,2}$. Therefore, the hybrids $\mathsf{H}_{5,q,2}$ and $\mathsf{H}_{5,q,3}$ are indistinguishable due to the function hiding security of IPFE. We observe that $\mathsf{H}_{5,q,3}$ is identical to $\mathsf{H}_{5,q}$ for all $q \in [Q_{\mathsf{pre}}]$.

Hybrid H_6: It is exactly the same as $\mathsf{H}_{5,Q_{\mathsf{pre}}}$ except that the positions $\boldsymbol{h}_t[\widehat{\mathsf{const}}]$ and $\boldsymbol{h}_t[\widehat{\mathsf{coef}}]$ are set to zero. We describe the vectors associated with the pre-ciphertext secret-key queries and the challenge ciphertext below. Note that the post-challenge secret-key queries are answered in the same way as in H_4 (or in $\mathsf{H}_{5,Q_{\mathsf{pre}}}$).

$$1 \le q < Q_{\mathsf{pre}} \begin{cases} \boldsymbol{v}_q = (\quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad \widetilde{\alpha}_q, \quad 0, \quad 0, 0\ ), \\ \boldsymbol{v}_{q,j,t} = (\ \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i], 0, 0\ ), \\ \boldsymbol{v}_{q,m_q+1,t} = (\boldsymbol{r}^{(\iota)}_{q,t}[m_q], \boldsymbol{\alpha}_q[\iota], 0, 0, \widetilde{\boldsymbol{r}}_{q,t}[m_q], \widetilde{\alpha}_q, 0, 0, 0) \end{cases}$$

$$\boldsymbol{u} = (0, 0, 1, \boldsymbol{x}^*[i], 0, 0),$$

$$\boldsymbol{h}_t = (0, 0, -1, \boldsymbol{z}^*[t], -1, \boldsymbol{d}[t], \boxed{0}, \boxed{0}, 0)$$

$$Q_{\mathsf{pre}} < q \le Q \begin{cases} \boldsymbol{v}_q = (\quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad \widetilde{\alpha}_q, \quad 0, \quad 0, 0\ ), \\ \boldsymbol{v}_{q,j,t} = (\ \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i], 0, 0\ ), \\ \boldsymbol{v}_{q,m_q+1,t} = (\boldsymbol{r}^{(\iota)}_{q,t}[m_q], \boldsymbol{\alpha}_q[\iota], \widetilde{\boldsymbol{r}}_{q,t}[m_q], \widetilde{\alpha}_q, 0, 0, 0, 0, 0) \end{cases}$$

Since the inner product $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ for all $q \in [Q], t \in [n']$ is unaltered with this change, the function hiding security of IPFE ensures indistinguishability between the hybrids $\mathsf{H}_{5,Q_{\mathsf{pre}}}$ and H_6.

Hybrid H_7: This hybrid proceeds exactly similar to H_6 except that we use the honest levels $\widetilde{\ell}_{q,j,t} = \widetilde{\boldsymbol{\ell}}_{q,j,t}(\boldsymbol{x}^*)$ for $j \in [m_q]$ and $\widetilde{\ell}_{q,m_q+1,t} = \widetilde{\alpha}_q \boldsymbol{z}^*[t] - \widetilde{\boldsymbol{r}}_{q,t}[m_q]$ at the index const of the vectors $\boldsymbol{v}_{q,j,t}$ in all the post-challenge secret-key queries. Moreover, all the other private positions of $\boldsymbol{v}_{q,j,t}$ are set to zero for all $j \in [m_q]$. We also modify $\boldsymbol{h}_t$ of the challenge ciphertext as shown below.

$$\boldsymbol{u} = (0, 0, 1, \boldsymbol{x}^*[i], 0, 0),$$

$$\boldsymbol{h}_t = (0, 0, \boxed{1}, \boxed{0}, -1, \boldsymbol{d}[t], 0, 0, 0)$$

$$Q_{\mathsf{pre}} < q \le Q \begin{cases} \boldsymbol{v}_q = (\quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad \widetilde{\alpha}_q, \quad 0, \ 0, 0\ ), \\ \boldsymbol{v}_{q,j,t} = (\ \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], \boxed{\widetilde{\ell}_{q,j,t}}, \boxed{0}, 0, 0\ ), \\ \boldsymbol{v}_{q,m_q+1,t} = (\boldsymbol{r}^{(\iota)}_{q,t}[m_q], \boldsymbol{\alpha}_q[\iota], \boxed{\widetilde{\ell}_{q,m_q+1,t}}, \boxed{0}, 0, 0, 0, 0, 0) \end{cases}$$

Since the inner products $\boldsymbol{v}_{q,j,t} \cdot \boldsymbol{u}, \boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ gives the same result as in the previous hybrid, the function hiding property of IPFE ensures that the hybrids H_6 and H_7 are indistinguishable.

Hybrid H_8: This hybrid proceeds analogous to H_7 except that in the post-challenge secret-key queries we use the simulated garblings instead of the honest garblings. More specifically, we sample $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ satisfying $\sum_{t \in [n']} \widetilde{\beta}_{q,t} = 0$ and compute the simulated garblings

$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \widetilde{\alpha}_q \cdot \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\beta}_{q,t})$$

for all $q \in [Q_{\mathsf{pre}} + 1, Q]$ and $t \in [n']$. Then, the post-challenge secret-keys are generated using the vectors given below.

$$\boldsymbol{v}_q = (\quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad \widetilde{\alpha}_q, \ 0, 0, 0\ ),$$

$$\boldsymbol{v}_{q,j,t} = (\ \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], \boxed{\widehat{\ell}_{q,j,t}}, 0, 0, 0\ ),$$

$$\boldsymbol{v}_{q,m_q+1,t} = (\boldsymbol{r}^{(\iota)}_{q,t}[m_q], \boldsymbol{\alpha}_q[\iota], \boxed{\widehat{\ell}_{q,m_q+1,t}}, 0, 0, 0, 0, 0, 0)$$

The simulated levels of AKGS is used in place of actual garblings. The simulation security of AKGS implies that the hybrids H_7 and H_8 are indistinguishable.

Hybrid H_9: This is exactly the same as H_8 except that the distribution of $\{\widetilde{\beta}_{q,t}\}_{t\in[n']}$ is changed. We replace $\widetilde{\beta}_{q,t}$ by $\widetilde{\beta}'_{q,t} = \widetilde{\beta}_{q,t} - \widetilde{\alpha}_q \cdot \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*)$ for all $1 < t \leq n'$ and replace the element $\widetilde{\beta}_{q,1}$ by $\widetilde{\beta}'_{q,1} = \widetilde{\beta}_{q,1} - \widetilde{\alpha}_q \cdot \boldsymbol{z}^*[t] f_{q,1}(\boldsymbol{x}^*) + \widetilde{\alpha}_q \cdot f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^*$. Note that, the distributions

$$\{\widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p : \sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0\} \text{ and } \{\widetilde{\beta}'_{q,t} : \sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0\}$$

are statistically close since $\{\widetilde{\beta}'_{q,t}\}_{t\in[n']}$ are also uniform over $\mathbb{Z}_p$ and $\sum_{t\in[n']} \widetilde{\beta}'_{q,t} = 0$. Finally, the vectors associated to the post-challenge secret-keys are given by

$$\begin{aligned}
\boldsymbol{v}_q &= (\quad \boldsymbol{\alpha}_q[\iota], \qquad 0, \qquad \widetilde{\alpha}_q, \quad 0, 0, 0\ ),\\
\boldsymbol{v}_{q,j,t} &= (\ \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], \boxed{\widehat{\ell}_{q,j,t}}, 0, 0, 0\ ),\\
\boldsymbol{v}_{q,m_q+1,t} &= (\boldsymbol{r}^{(\iota)}_{q,t}[m_q], \boldsymbol{\alpha}_q[\iota], \boxed{\widehat{\ell}_{q,m_q+1,t}}, 0, 0, 0, 0, 0, 0)
\end{aligned}$$

where the simulated garblings take the form

$$\begin{aligned}
(\widehat{\ell}_{q,1,1}, \ldots, \widehat{\ell}_{q,m_q,1}, \widehat{\ell}_{q,m_q+1,1}) &\leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, \boxed{\widetilde{\alpha}_q \cdot f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \widetilde{\beta}_{q,1}})\\
(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) &\leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \boxed{\widetilde{\beta}_{q,t}}) \quad \text{for } 1 < t \leq n'.
\end{aligned}$$

Observe that H_9 is the same as the ideal experiment $\mathsf{Expt}^{\mathsf{FE},\ \mathsf{Ideal}}_{\mathcal{A}}(1^\lambda)$. This completes the security analysis. $\square$

6 1-Key 1-Ciphertext Secure One-Slot Extended FE Designed for Bounded-Key One-Slot Extended FE for Attribute-Weighted Sums

In this section, we present a private-key one-slot FE scheme for an extended attribute-weighted sum functionality that is proven simulation secure against a single ciphertext query and a single secret key query either before or after the ciphertext query. This scheme will be embedded into the hidden subspaces of the public-key multi-key FE scheme for the same functionality presented in the next section in its security proof. We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ for simplicity of notations. Let $(\mathsf{Garble}, \mathsf{Eval})$ be a special piecewise secure AKGS for a function class $\mathcal{F}^{(n,n')}_{\mathsf{ABP}}$, $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ a tuple of pairing groups of prime order p, and $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ a secret-key function-hiding SK-IPFE based on G.

Setup$(1^\lambda, 1^n, 1^{n'})$: Define the following index sets as follows

$$S_{\text{1-extFE}} = \left\{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \{\mathsf{extnd}_\kappa\}_{\kappa\in[k]}, \mathsf{query}, \{\mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}_{\tau\in[n']}\right\},$$
$$\widehat{S}_{\text{1-extFE}} = \{\widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}^*}\}$$

It generates two IPFE master secret-keys $\mathsf{IPFE.MSK} \leftarrow \mathsf{SK\text{-}IPFE.Setup}(S_{\text{1-extFE}})$ and $\widehat{\mathsf{IPFE.MSK}} \leftarrow \mathsf{SK\text{-}IPFE.Setup}(\widehat{S}_{\text{1-extFE}})$. Finally, it returns $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$.

KeyGen(MSK, $(\boldsymbol{f}, \boldsymbol{y})$): Let $f = (f_1, \ldots, f_{n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ and $\boldsymbol{y} \in \mathbb{Z}_p^k$. Samples integers $\nu_t, \beta_t \leftarrow \mathbb{Z}_p$ for $t \in [n']$ such that

$$\sum_{t\in[n']} \nu_t = 1 \text{ and } \sum_{t\in[n']} \beta_t = 0 \quad \text{modulo } p.$$

Next, samples independent random vectors $\boldsymbol{r}_t \leftarrow \mathbb{Z}_p^m$ for garbling and computes the coefficient vectors

$$(\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}, \boldsymbol{\ell}_{m+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \beta_t; \boldsymbol{r}_t)$$

for each $t \in [n']$. Here we make use of the instantiation of the AKGS described in Section 3.6. From the description of that AKGS instantiation, we note that the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}$ would be of the form $\boldsymbol{\ell}_{m+1,t} = \boldsymbol{z}[t] - \boldsymbol{r}_t[m]$. Also all the label functions $\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}$ involve only the variables $\boldsymbol{x}$ and not the variable $\boldsymbol{z}[t]$. Next, for all $j \in [m]$ and $t \in [n']$, it defines the vectors $\boldsymbol{v}_{j,t}$ corresponding to the label functions $\boldsymbol{\ell}_{j,t}$ obtained from the partial garbling above and the vector $\boldsymbol{y}$ as

vector	const	coef_i	extnd_κ	query	sim_τ	sim^*_τ
$\boldsymbol{v}_{1,t}$	$\boldsymbol{\ell}_{1,t}[\mathsf{const}]$	$\boldsymbol{\ell}_{1,t}[\mathsf{coef}_i]$	$\boldsymbol{y}[\kappa]\nu_t$	0	0	0
$\boldsymbol{v}_{j,t}$	$\boldsymbol{\ell}_{j,t}[\mathsf{const}]$	$\boldsymbol{\ell}_{j,t}[\mathsf{coef}_i]$	0	0	0	0

It also sets the vectors $\boldsymbol{v}_{m+1,t}$ for $t \in [n']$ corresponding to the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}$ as

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{m+1,t}$	$\boldsymbol{r}_t[m]$	1	0

Now, it uses the key generation algorithm of IPFE to generate the secret-keys

$$\begin{aligned} \mathsf{IPFE.SK}_{j,t} &\leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2) && \text{for } j \in [m], t \in [n'] \\ \widehat{\mathsf{IPFE.SK}}_{m+1,t} &\leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2) && \text{for } t \in [n'] \end{aligned}$$

It returns the secret-key as $\mathsf{SK}_{f,\boldsymbol{y}} = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$.

Remark. We note that the key-generation process can be performed if the vector $\boldsymbol{y}$ is not given in the clear, but $[\![\boldsymbol{y}]\!]_2 \in \mathbb{G}_2^k$ is known. This is because while running the IPFE.KeyGen *algorithm above, the vectors $\boldsymbol{v}_{j,t}$ are not inputted in the clear but in the exponent of the group $\mathbb{G}_2$. This fact will be used in the security analysis of our unbounded* FE *scheme.*

Enc(MSK, $(\boldsymbol{x}, \boldsymbol{z}||\boldsymbol{w}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$): It sets the following vectors:

vector	const	coef_i	extnd_κ	query	sim_τ	sim^*_τ
$\boldsymbol{u}$	1	$\boldsymbol{x}[i]$	$\boldsymbol{w}[\kappa]$	0	0	0

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{h}_t$	-1	$\boldsymbol{z}[t]$	0

for all $t \in [n']$. Then, it encrypts the vectors using IPFE and obtain the ciphertexts

$$\begin{aligned} \mathsf{IPFE.CT} &\leftarrow \mathsf{SK\text{-}IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1) \\ \widehat{\mathsf{IPFE.CT}}_t &\leftarrow \mathsf{SK\text{-}IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{h}_t]\!]_1) \qquad \text{for } t \in [n'] \end{aligned}$$

Finally, it returns the ciphertext as $\mathsf{CT}_{\boldsymbol{x},\boldsymbol{z}||\boldsymbol{w}} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

$\mathsf{Dec}((\mathsf{SK}_{\boldsymbol{f},\boldsymbol{y}}, \boldsymbol{f}), (\mathsf{CT}_{\boldsymbol{x},\boldsymbol{z}||\boldsymbol{w}}, \boldsymbol{x}))$: It parses $\mathsf{SK}_{f,\boldsymbol{y}} = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ and $\mathsf{CT}_{\boldsymbol{x},\boldsymbol{z}||\boldsymbol{w}} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$. It uses the decryption algorithm of $\mathsf{SK\text{-}IPFE}$ to compute

$$\begin{aligned} [\![\ell_{1,t} + \psi_t]\!]_T &\leftarrow \mathsf{SK\text{-}IPFE.Dec}(\mathsf{IPFE.SK}_{1,t}, \mathsf{IPFE.CT}) && \text{for } t \in [n'] \\ [\![\ell_{j,t}]\!]_T &\leftarrow \mathsf{SK\text{-}IPFE.Dec}(\mathsf{IPFE.SK}_{j,t}, \mathsf{IPFE.CT}) && \text{for } j \in [2,m], t \in [n'] \\ [\![\ell_{m+1,t}]\!]_T &\leftarrow \mathsf{SK\text{-}IPFE.Dec}(\widehat{\mathsf{IPFE.SK}}_{m+1,t}, \widehat{\mathsf{IPFE.CT}}_t) && \text{for } t \in [n'] \end{aligned}$$

where $\psi_t = \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}$. Next, it utilizes the evaluation procedure of AKGS and returns the combined value

$$[\![\rho]\!]_T = \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T).$$

Correctness: From the correctness of IPFE, we have $\mathsf{SK\text{-}IPFE.Dec}(\mathsf{IPFE.SK}_{1,t}, \mathsf{IPFE.CT}) = [\![\ell_{1,t} + \psi_t]\!]_T$ where $\psi_t = \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}$. Next, using the correctness of IPFE and AKGS evaluation, we get

$$\begin{aligned} &\mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) \\ &= \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) + \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\psi_t]\!]_T, [\![0]\!]_T, \ldots, [\![0]\!]_T) \\ &= [\![\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \beta_t + \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}]\!]_T \end{aligned}$$

The first equality follows from the linearity of Eval function. Now, multiplying all the evaluated values we have

$$\begin{aligned} [\![\rho]\!]_T &= \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) \\ &= [\![\sum_{t=1}^{n'} (\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w} + \beta_t)]\!]_T \\ &= [\![f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w}]\!]_T \end{aligned}$$

The last equality is obtained from the fact that $\sum_{t\in[n']} \nu_t = 1$ and $\sum_{t\in[n']} \beta_t = 0$.

6.1 Security Analysis

Theorem 4 *The 1-*extFE *scheme for attribute-weighted sum is 1-key, 1-ciphertext simulation-secure as per Definition 4 assuming the* AKGS *is piecewise secure as per Definition 7 and the* IPFE *is function hiding as per Definition 5.*

As in the case of our 1-key 1-ciphertext secure one-slot FE, here also we assume that the adversary queries the single secret key before the challenge ciphertext is sent. This is because we will use the security of the 1-key 1-ciphertext secure one-slot extFE in a particular hybrid of the security reduction of our one-slot extFE scheme (presented in Section B) where we deal with a single pre-ciphertext secret key of the one-slot extFE. However, we emphasize that if we consider the single secret key query after the challenge phase then the security can also be proved using the security reduction of our one-slot extFE, provided in Supplementary Material **??**.

The Simulator We describe the simulator for the 1-extFE scheme. Let us assume that $(f, \boldsymbol{y}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')} \times \mathbb{Z}_p^k$ is the only secret-key query made by the adversary before it sends challenge vectors $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$. The algorithm $\mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'})$ is exactly the same as $\mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$ which outputs a master secret-key $\mathsf{MSK}^* = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$. The key generation procedure $\mathsf{KeyGen}_0^*(\mathsf{MSK}^*, (f, \boldsymbol{y}))$ of the simulator is similar to the original algorithm $\mathsf{KeyGen}(\mathsf{MSK}^*, (f, \boldsymbol{y}))$ except the fact that $\boldsymbol{v}_{1,t}[\mathsf{query}] = \nu_t$. We describe the encryption process of the simulator which uses the information $\mu = f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^*$.

$\mathsf{Enc}^*(\mathsf{MSK}^*, \boldsymbol{x}^*, ((f, \boldsymbol{y}), \mu))$: On input MSK^*, a vector $\boldsymbol{x}^* \in \mathbb{Z}_p^n$, the tuple $(f, \boldsymbol{y}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')} \times \mathbb{Z}_p^k$ and an integer $\mu \in \mathbb{Z}_p$ the simulator executes the following steps:

1. First, it picks two random vectors $\boldsymbol{d}_1 \leftarrow \mathbb{Z}_p^{n'}, \boldsymbol{d}_2 \leftarrow \mathbb{Z}_p^k$ and sets $\sigma = \mu - f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 - \boldsymbol{y}^\top \boldsymbol{d}_2$.
2. Next, it sets the following vectors

vector	const	coef_i	extnd_κ	query	sim_τ	sim^*_τ
$\boldsymbol{u}$	1	$\boldsymbol{x}^*[i]$	$\boldsymbol{d}_2[\kappa]$	σ	0	0

and

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}^*}$
$\boldsymbol{h}_t$	-1	$\boldsymbol{d}_1[t]$	0

for all $t \in [n']$.

3. Finally, it encrypts the vectors as

$$\mathsf{IPFE.CT} \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)$$
$$\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{h}_t]\!]_1) \qquad \text{for } t \in [n']$$

4. It returns the simulated ciphertext as $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

Remark. Observe that Enc^* is designed in such a way that the simulator is also able to generate the ciphertext CT^* even when it gets $[\![\boldsymbol{y}]\!]_1, [\![\mu]\!]_1$ instead of $\boldsymbol{y}, \mu$ in the clear. In such a scenario, the simulator will obtain $[\![\sigma]\!]_1 = [\![\mu]\!]_1 \cdot [\![\boldsymbol{y}^\top \boldsymbol{d}_2]\!]_1^{-1} \cdot [\![f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1]\!]^{-1}$ by sampling $\boldsymbol{d}_1 \leftarrow \mathbb{Z}_p^{n'}, \boldsymbol{d}_2 \leftarrow \mathbb{Z}_p^k$. Hence, it can define $[\![\boldsymbol{u}]\!]_1$ and $[\![\boldsymbol{h}_t]\!]_1$ as above before applying the encryption process of IPFE. This procedure is indeed required for the security analysis of our unbounded FE construction.

Hybrids and Reductions

Proof. We employ a sequence of hybrid experiments to demonstrate the indistinguishability between the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,1\text{-}extFE}}(1^\lambda)$ and the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,1\text{-}extFE}}(1^\lambda)$ where $\mathcal{A}$ is any PPT adversary. We assume that in each experiment, $\mathcal{A}$ queries the single secret-key query for a pair $(f, \boldsymbol{y}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')} \times \mathbb{Z}_p^k$ before submitting the challenge message $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$.

Hybrid H_0: This is the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,1\text{-}extFE}}(1^\lambda)$ where the secret-key $\mathsf{SK}_{f,\boldsymbol{y}} = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ such that $\mathsf{IPFE.SK}_{j,t} \leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2)$ for $j \in [m], t \in [n']$ and $\widehat{\mathsf{IPFE.SK}}_{m+1,t} \leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2)$ for $t \in [n']$ where the vectors $\boldsymbol{v}_{j,t}, \boldsymbol{v}_{m+1,t}$ are given as follows:

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ \boldsymbol{\ell}_{1,t}[\mathsf{const}], \boldsymbol{\ell}_{1,t}[\mathsf{coef}_i], \boldsymbol{y}[\kappa]\nu_t, 0, 0, 0\ )\\
\boldsymbol{v}_{j,t} &= (\ \boldsymbol{\ell}_{j,t}[\mathsf{const}], \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i], \quad 0, \quad 0, 0, 0\ ) \qquad \text{for } 1 < j \leq m,\\
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m], 1, 0\ )
\end{aligned}$$

for $j \in [m]$, $t \in [n']$ and $\boldsymbol{r}_t \leftarrow \mathbb{Z}_p^m$. Note that $\{\nu_t\}_{t\in[n']} \leftarrow \mathbb{Z}_p$ is such that $\sum_{t\in[n']} \nu_t = 1$ modulo p. Then, the garblings are computed as

$$(\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}, \boldsymbol{\ell}_{m+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{z}^*[t]f_t(\boldsymbol{x}^*) + \beta_t; \boldsymbol{r}_t)$$

where $\beta_t \leftarrow \mathbb{Z}_p$ for all $t \in [n']$ with $\sum_{t\in[n']} \beta_t = 0$ modulo p. The challenge ciphertext $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$ corresponding to the challenge message $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$ is given by $\mathsf{IPFE.CT} \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)$ and $\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{h}_t]\!]_1)$ for $t \in [n']$ where

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], \boldsymbol{w}[\kappa], 0, 0, 0\ ), \quad \boldsymbol{h}_t = (\ -1, \boldsymbol{z}^*[t], 0\ )$$

for $t \in [n']$. Note that the components of the vectors $\boldsymbol{u}$ and $\boldsymbol{v}_{j,t}$ are associated with the indices in $S_{\mathsf{1\text{-}extFE}}$, and the components of the vectors $\boldsymbol{h}_t$ and $\boldsymbol{v}_{m+1,t}$ are associated with the indices in $\widehat{S}_{\mathsf{1\text{-}extFE}}$.

Hybrid H_1: This hybrid is exactly the same as H_0 except that we directly hardwire the value $\ell_{1,\tau} + \psi_\tau = \boldsymbol{\ell}_{1,\tau}(\boldsymbol{x}^*) + \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{w}$ into $\boldsymbol{u}[\mathsf{sim}_\tau]$ for all $\tau \in [n']$ and remove the coefficient vector $\boldsymbol{\ell}_{1,t}$ from $\boldsymbol{v}_{1,t}$ for all $t \in [n']$. We change the vectors $\boldsymbol{v}_{1,t}$ in the secret-key and $\boldsymbol{u}$ in the challenge ciphertext as follows:

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\quad \boxed{0}, \qquad \boxed{0}, \quad \boxed{0}, 0, \quad \boxed{\delta_{t\tau}}, \quad 0\ )\\
\boldsymbol{v}_{j,t} &= (\ \boldsymbol{\ell}_{j,t}[\mathsf{const}], \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i], \ 0,\ 0, \quad 0, \quad 0\ ) \qquad \text{for } 1 < j < m,\\
\boldsymbol{u} &= (\quad 1, \qquad \boldsymbol{x}^*[i], \quad \boxed{0}, 0, \boxed{\ell_{1,\tau} + \psi_\tau}, 0\ )\\
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m], \quad 1, \quad 0\ )\\
\boldsymbol{h}_t &= (\quad -1, \quad \boldsymbol{z}^*[t], 0\ )
\end{aligned}$$

We denote by $\delta_{t\tau}$ the usual *Kronecker delta* function such that $\delta_{t\tau} = 1$ if $t = \tau$, 0 otherwise. Note that the inner product $\boldsymbol{v}_{1,t} \cdot \boldsymbol{u} = \ell_{1,t} + \psi_t$, for all $t \in [n']$, remain the same as in H_0. Therefore, the function hiding security of IPFE ensures the indistinguishability between the hybrids H_0 and H_1.

Hybrid H_2: This is analogous to H_1 except that instead of using the actual garbling value $\ell_{1,\tau}$ at $\boldsymbol{u}[\mathsf{sim}_\tau]$, we now use $\widetilde{\ell}_{1,\tau}$ which is computed via reverse sampling algorithm of AKGS:

$$\widetilde{\ell}_{1,\tau} \leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau] + \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{w} + \beta_\tau, \ell_{2,\tau}, \ldots, \ell_{m+1,\tau})$$

where $\ell_{j,\tau} = \boldsymbol{\ell}_{j,\tau}(\boldsymbol{x}^*)$ for all $j \in [2, m]$ and $\ell_{m+1,\tau} = -\boldsymbol{r}_\tau[m] + \boldsymbol{z}^*[\tau]$ for all $\tau \in [n']$. Therefore, the vectors in the challenge ciphertext becomes

$$\boldsymbol{u} = (\, 1, \boldsymbol{x}^*[i], 0, 0, \boxed{\widetilde{\ell}_{1,\tau}}, 0\,), \quad \boldsymbol{h}_t = (\, -1, \boldsymbol{z}^*[t], 0\,).$$

For each $\tau \in [n']$, the piecewise security of AKGS guarantees that given the label functions $(\boldsymbol{\ell}_{2,\tau}, \ldots, \boldsymbol{\ell}_{m,\tau}, \boldsymbol{\ell}_{m+1,\tau})$, the actual garbled label $\ell_{1,\tau}$ and the reversely sampled value $\widetilde{\ell}_{1,\tau}$ are identically distributed. Hence, the hybrids H_1 and H_2 are indistinguishable by the reverse sampleability of AKGS.

Remark. Suppose in this hybrid instead of the vector $\boldsymbol{y}$, the challenger only receives $[\![\boldsymbol{y}]\!]_1$ from the adversary as part of its secret-key query. Then, it can also simulate the game by computing the vector $[\![\boldsymbol{u}]\!]_1$ using the fact

$$\begin{aligned}&\mathit{RevSamp}(f_\tau, \boldsymbol{x}^*, [\![\gamma_\tau]\!]_1, [\![\ell_{2,\tau}]\!]_1, \ldots, [\![\ell_{m+1,\tau}]\!]_1)\\ &= [\![\gamma_\tau]\!]_1 \cdot ([\![\mathit{Eval}(f_\tau, \boldsymbol{x}^*, 0, \ell_{2,\tau}, \ldots, \ell_{m+1,\tau})]\!]_1)^{-1}\end{aligned}$$

with $\gamma_\tau = f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau] + \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{w} + \beta_\tau$. Although, it is not necessary for this proof, we will need this formulation of $\mathsf{RevSamp}$ during the security analysis of our unbounded FE scheme.

Hybrid $\mathsf{H}_{3,j}$ ($j \in [2, m]$): The hybrid proceeds similar to H_2 except that we change the secret-key as follows. For all j' such that $1 < j' < j$, the coefficient vector $\boldsymbol{\ell}_{j,t}$ is taken away from $\boldsymbol{v}_{j',t}$ and a random value $\ell'_{j',t} \leftarrow \mathbb{Z}_p$ is put into $\boldsymbol{v}_{j',t}[\mathsf{const}]$. We describe the vectors associated with the secret-key and the ciphertext below.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\quad 0, \quad\quad 0, \quad 0, 0, \delta_{t\tau}, 0\,) \\
\boldsymbol{v}_{j',t} &= (\quad \boxed{\ell'_{j',t}}, \quad \boxed{0}, \quad 0, 0, 0, 0\,) && \text{for } 1 < j' \le j, \\
\boldsymbol{v}_{j',t} &= (\, \boldsymbol{\ell}_{j',t}[\mathsf{const}], \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], 0, 0, 0, 0\,) && \text{for } j < j' \le m, \\
\boldsymbol{u} &= (\quad 1, \quad \boldsymbol{x}^*[i], \quad 0, 0, \widetilde{\ell}_{1,\tau}, 0\,) \\
\boldsymbol{v}_{m+1,t} &= (\, \boldsymbol{r}_t[m], \quad 1, \quad 0\,) \\
\boldsymbol{h}_t &= (\quad -1, \quad \boldsymbol{z}^*[t], 0\,)
\end{aligned}$$

Note that, in this hybrid $\widetilde{\ell}_{1,\tau}$ is reversely sampled using the random values $\ell_{2,\tau}, \ldots, \ell_{j-1,\tau}$ (which are randomly chosen from $\mathbb{Z}_p$) and the actual values $\ell_{j,\tau}, \ldots, \ell_{m+1,\tau}$ for each $\tau \in [n']$. Observe that $\mathsf{H}_{3,1}$ coincides with H_2. We will show that for all $j \in [2, m]$, the hybrids $\mathsf{H}_{3,(j-1)}$ and $\mathsf{H}_{3,j}$ are indistinguishable via the following sequence of sub-hybrids, namely, $\{\mathsf{H}_{3,j,1}, \mathsf{H}_{3,j,2}, \mathsf{H}_{3,j,3}\}_{j \in [2,m]}$.

Hybrid $\mathsf{H}_{3,j,1}$ ($j \in [2, m]$): This is exactly the same as $\mathsf{H}_{3,(j-1)}$ except that the coefficient vector $\boldsymbol{\ell}_{j,t}$ is removed from $\boldsymbol{v}_{j,t}$ and $\boldsymbol{v}_{j,t}[\mathsf{sim}^*_\tau]$ is set to $\delta_{t\tau}$. The actual garbling value $\ell_{j,\tau} = \boldsymbol{\ell}_{j,\tau}(\boldsymbol{x}^*)$ is hardwired into $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ to ensure the inner product $\boldsymbol{v}_{j,\tau} \cdot \boldsymbol{u}$ remains the same as in $\mathsf{H}_{3,(j-1)}$.

The changes in the vectors involved while computing secret-key and the challenge ciphertext as given below.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\quad 0, \quad 0, \quad 0, 0,\ \delta_{t\tau},\ \ 0\ ) \\
\boldsymbol{v}_{j',t} &= (\quad \ell'_{j',t}, \quad 0, \quad 0, 0,\ 0,\ \ 0\ ) && \text{for } 1 < j' < j, \\
\boldsymbol{v}_{j,t} &= (\quad \boxed{0}, \quad \boxed{0}, \quad 0, 0,\ 0,\ \boxed{\delta_{t\tau}}\ ) \\
\boldsymbol{v}_{j',t} &= (\ \boldsymbol{\ell}_{j',t}[\mathsf{const}], \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], 0, 0,\ 0,\ \ 0\ ) && \text{for } j < j' \leq m, \\
\boldsymbol{u} &= (\quad 1, \quad \boldsymbol{x}^*[i], \quad 0, 0, \widetilde{\ell}_{1,\tau}, \boxed{\ell_{j,\tau}}\ )
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m],\ \ 1,\ \ 0\ ) \\
\boldsymbol{h}_t &= (\ \ -1,\ \ \boldsymbol{z}^*[t], 0\ )
\end{aligned}$$

The hybrids $\mathsf{H}_{3,(j-1)}$ and $\mathsf{H}_{3,j,1}$ are indistinguishable by the function hiding security of IPFE since the inner product $\boldsymbol{v}_{j,\tau} \cdot \boldsymbol{u}$ for all $\tau \in [n']$ remains the same as in $\mathsf{H}_{3,(j-1)}$.

Hybrid $\mathbf{H}_{3,j,2}$ ($\boldsymbol{j} \in [\mathbf{2}, \boldsymbol{m}]$): It proceeds exactly the same as $\mathsf{H}_{3,j,1}$ except that the actual label $\ell_{j,\tau}$ (sitting at $\boldsymbol{u}[\mathsf{sim}^*_\tau]$) is replaced with a random value $\ell'_{j,\tau} \leftarrow \mathbb{Z}_p$. The vectors associated to the challenge ciphertext are given by

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], 0, 0, \widetilde{\ell}_{1,\tau}, \boxed{\ell'_{j,\tau}}\ ), \quad \boldsymbol{h}_t = (\ -1, \boldsymbol{z}^*[t], 0\ )$$

where $\ell'_{j,\tau}$ is randomly sampled from $\mathbb{Z}_p$. Now, the first label $\widetilde{\ell}_{1,\tau}$ is reversely sampled using the random values $\ell'_{2,\tau}, \ldots, \ell'_{j,\tau}$ and the actual labels $\ell_{j+1,\tau} = \boldsymbol{\ell}_{j+1,\tau}(\boldsymbol{x}^*), \ldots, \ell_{m,\tau} = \boldsymbol{\ell}_{m,\tau}(\boldsymbol{x}^*), \ell_{m+1,\tau} = -\boldsymbol{r}_\tau[m] + \boldsymbol{z}^*[\tau]$. The marginal randomness property of AKGS implies that the hybrids $\mathsf{H}_{3,j,1}$ and $\mathsf{H}_{3,j,2}$ are identically distributed.

Hybrid $\mathbf{H}_{3,j,3}$ ($\boldsymbol{j} \in [\mathbf{2}, \boldsymbol{m}]$): The hybrid is analogous to $\mathsf{H}_{3,j,2}$ except that the random value $\ell'_{j,\tau}$ is sifted from the ciphertext component $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ to the secret-key component $\boldsymbol{v}_{j,t}[\mathsf{const}]$. Also, the positions $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ and $\boldsymbol{v}_{j,t}[\mathsf{sim}^*_\tau]$ are set to zero. Thus, the vectors in the secret-key and the challenge ciphertext become

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\quad 0, \quad 0, \quad 0, 0,\ \delta_{t\tau},\ \ 0\ ) \\
\boldsymbol{v}_{j',t} &= (\quad \ell'_{j',t}, \quad 0, \quad 0, 0,\ 0,\ \ 0\ ) && \text{for } 1 < j' < j, \\
\boldsymbol{v}_{j,t} &= (\quad \boxed{\ell'_{j,t}}, \quad 0, \quad 0, 0,\ 0,\ \boxed{0}\ ) \\
\boldsymbol{v}_{j',t} &= (\ \boldsymbol{\ell}_{j',t}[\mathsf{const}], \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], 0, 0,\ 0,\ \ 0\ ) && \text{for } j < j' \leq m, \\
\boldsymbol{u} &= (\quad 1, \quad \boldsymbol{x}^*[i], \quad 0, 0, \widetilde{\ell}_{1,\tau}, \boxed{0}\ )
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m],\ \ 1,\ \ 0\ ) \\
\boldsymbol{h}_t &= (\ \ -1,\ \ \boldsymbol{z}^*[t], 0\ )
\end{aligned}$$

Since the inner products $\boldsymbol{v}_{j,t} \cdot \boldsymbol{u}$ for all j, t remain the same as in $\mathsf{H}_{3,j,2}$, the indistinguishability between the hybrids $\mathsf{H}_{3,j,2}$ and $\mathsf{H}_{3,j,3}$ follows from the function hiding security of IPFE. We observe that the hybrids $\mathsf{H}_{3,j,3}$ is identical to $\mathsf{H}_{3,j}$ for all $j \in [2, m]$.

Hybrid $\mathbf{H}_4$: It proceeds exactly the same as hybrid $\mathsf{H}_{3,m}$ except that the actual garbling value $\ell_{m+1,t} = -\boldsymbol{r}_t[m] + \boldsymbol{z}^*[t]$ is used in $\boldsymbol{h}_t[\widehat{\mathsf{sim}^*}]$. Also, $\boldsymbol{h}_t[\widehat{\mathsf{coef}}]$,

$\boldsymbol{v}_{m+1,t}[\widehat{\mathsf{const}}], \boldsymbol{v}_{m+1,t}[\widehat{\mathsf{coef}}]$ are set to zero. The changes are indicated below.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0, \quad 0, \quad 0, 0, \ \delta_{t\tau}, \ 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t}, \quad 0, \quad 0, 0, \ \ 0, \ \ 0\ ) \qquad \text{for } 1 < j \leq m, \\
\boldsymbol{u} &= (\ 1, \ \boldsymbol{x}^*[i], 0, 0, \widetilde{\ell}_{1,\tau}, 0\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{0}, \boxed{0}, \quad \boxed{1} \quad) \\
\boldsymbol{h}_t &= (\ \boxed{1}, \boxed{0}, \boxed{\ell_{m+1,t}}\ )
\end{aligned}$$

Since the inner products $\boldsymbol{v}_{m+1,t} \cdot \boldsymbol{h}_t$ for all $t \in [n']$ are unaltered as in H_4, the indistinguishability between the hybrids $\mathsf{H}_{3,m}$ and H_4 follows from the function hiding security of IPFE.

Hybrid H_5: It is analogous to H_4 except that the actual label $\ell_{m+1,t}$ is now replaced with a random value $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$. The vectors associated with the challenge ciphertext are modified as follows.

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], 0, 0, \widetilde{\ell}_{1,\tau}, 0\ ), \quad \boldsymbol{h}_t = (\ 1, 0, \boxed{\ell'_{m+1,t}}\ )$$

Note that, in this hybrid the labels $\widetilde{\ell}_{1,t}$ for $t \in [n']$ are now reversely sampled using all random values $\ell'_{2,t}, \ldots, \ell'_{m+1,t}$ which are randomly picked from $\mathbb{Z}_p$. By the marginal randomness property of AKGS, the hybrids H_4 and H_5 are identically distributed.

Hybrid H_6: This hybrid proceeds exactly the same as H_5 except that the simulated labels $\ell'_{m+1,t}$ are shifted from $\boldsymbol{h}_t[\widehat{\mathsf{sim}^*}]$ to $\boldsymbol{v}_{m+1,t}[\widehat{\mathsf{rand}}]$. The positions $\boldsymbol{v}_{m+1,t}[\widehat{\mathsf{sim}}^*]$ and $\boldsymbol{h}_t[\widehat{\mathsf{sim}}^*]$ are set to zero. The changes are indicated as follows.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0, \quad 0, \quad 0, 0, \ \delta_{t\tau}, \ 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t}, \quad 0, \quad 0, 0, \ \ 0, \ \ 0\ ) \qquad \text{for } 1 < j \leq m, \\
\boldsymbol{u} &= (\ 1, \ \boldsymbol{x}^*[i], 0, 0, \widetilde{\ell}_{1,\tau}, 0\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{\ell'_{m+1,t}}, 0, \boxed{0}\ ) \\
\boldsymbol{h}_t &= (\quad 1, \quad 0, \boxed{0}\ )
\end{aligned}$$

Observe that the inner products $\boldsymbol{v}_{m+1,t} \cdot \boldsymbol{h}_t$ for all $t \in [n']$ are unchanged as in H_5. Hence, the function-hiding security of IPFE ensures the indistinguishability between the hybrids H_5 and H_6.

Hybrid H_7: It is analogous to H_6 except that the value $f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau]$ is removed from $\widetilde{\ell}_{1,\tau}$ for all $1 < \tau \leq n'$ and the value $f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^*$ is directly encoded into the label $\widetilde{\ell}_{1,1}$. To make this change, we replace the random elements β_τ by $\beta'_\tau = \beta_\tau - f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau] - \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{w}^*$ for all $1 < \tau \leq n'$ and change the element β_1 with $\beta'_1 = \beta_1 - (f_1(\boldsymbol{x}^*)\boldsymbol{z}^*[1] + \nu_1 \cdot \boldsymbol{y}^\top \boldsymbol{w}^*) + f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^*$. Note that, the distributions

$$\{\beta_\tau \leftarrow \mathbb{Z}_p : \sum_{\tau \in [n']} \beta_\tau = 0 \mod p\} \text{ and } \{\beta'_\tau : \sum_{\tau \in [n']} \beta_\tau = 0 \mod p\}$$

are statistically close since β'_τ is also uniform over $\mathbb{Z}_p$ and $\sum_{\tau \in [n']} \beta'_\tau = 0 \mod p$. Thus the vectors associated to the challenge ciphertext become

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], 0, 0, \boxed{\widetilde{\ell}_{1,\tau}}, 0\ ), \quad \boldsymbol{h}_t = (\ 1, 0, 0\ )$$

where the labels $\widetilde{\ell}_{1,\tau}$ are given by

$$\begin{aligned}\widetilde{\ell}_{1,1} &\leftarrow \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f_1(\boldsymbol{x}^*)\boldsymbol{z}^*[1] + \nu_1 \cdot \boldsymbol{y}^\top \boldsymbol{w}^* + \beta_1', \ell_{2,1}', \ldots, \ell_{m+1,1}') \\ &= \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^* + \beta_1, \ell_{2,1}', \ldots, \ell_{m+1,1}') \\ \widetilde{\ell}_{1,\tau} &\leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau] + \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{w}^* + \beta_\tau', \ell_{2,\tau}', \ldots, \ell_{m+1,\tau}') \\ &= \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, \beta_\tau, \ell_{2,\tau}', \ldots, \ell_{m+1,\tau}') \qquad \text{for } 1 < \tau \leq n'\end{aligned}$$

Thus, H_6 and H_7 are indistinguishable from the adversary's view as they are statistically close. As discussed in the remark of H_2, the challenger can also simulate this hybrid when $[\![\boldsymbol{y}]\!]_1$ is known instead of $\boldsymbol{y}$.

Hybrid H_8: This hybrid is exactly the same as H_7 except that we use a dummy vector $(\boldsymbol{d}_1 \parallel \boldsymbol{d}_2) \in \mathbb{Z}_p^{n'+k}$ in place of $(\boldsymbol{z}^* \parallel \boldsymbol{w}^*)$ while computing $\widetilde{\ell}_{1,1}$ where it holds that $\mu = f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^* = f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma$. In particular, we choose $\boldsymbol{d}_1 \leftarrow \mathbb{Z}_p^{n'}, \boldsymbol{d}_2 \leftarrow \mathbb{Z}_p^k$ and set $\sigma = \mu - f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 - \boldsymbol{y}^\top \boldsymbol{d}_2 \in \mathbb{Z}_p$. It can be seen that $f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma = \mu$ as required. The vector $\boldsymbol{u}$ is now defined as

$$\boldsymbol{u} = (\, 1, \overbrace{\boldsymbol{x}^*[1], \ldots, \boldsymbol{x}^*[n]}^{\mathsf{coef}_i}, \overbrace{0, \ldots, 0}^{\mathsf{extnd}_\kappa}, 0, \overbrace{\boxed{\widetilde{\ell}_{1,1}}, \widetilde{\ell}_{1,2} \ldots, \widetilde{\ell}_{1,n'}}^{\mathsf{sim}_\tau}, \overbrace{0, \ldots, 0}^{\mathsf{sim}_\tau^*}\,)$$

where the labels are computed as

$$\begin{aligned}\widetilde{\ell}_{1,1} &\leftarrow \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma + \beta_1, \ell_{2,1}', \ldots, \ell_{m+1,1}') \\ \widetilde{\ell}_{1,\tau} &\leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, \beta_\tau, \ell_{2,\tau}', \ldots, \ell_{m+1,\tau}') \qquad \text{for } 1 < \tau \leq n'\end{aligned}$$

Above, we write the full expression of the vector $\boldsymbol{u}$ as opposed to its compressed expression used so far in order to highlight the change. Since β_1 is uniformly distributed and $f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^* = f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma$, hybrids H_7 and H_8 are statistically close.

Remark. Suppose, the vector $[\![\boldsymbol{y}]\!]_1$ is known to the challenger instead of $\boldsymbol{y}$, then it can directly computes $[\![\sigma]\!]_1 = [\![\mu]\!]_1 \cdot [\![f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1]\!]_1^{-1} \cdot [\![\boldsymbol{y}^\top \boldsymbol{d}_2]\!]_1^{-1}$. To simulate this hybrid the challenger uses $[\![f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma + \beta_1]\!]_1$ to obtain $[\![\widetilde{\ell}_{1,1}]\!]_1$ as it has $\boldsymbol{d}_1 \in \mathbb{Z}_p^{n'}, \boldsymbol{d}_2 \in \mathbb{Z}_p^k, [\![\sigma]\!]_1 \in \mathbb{G}_1$ and $\beta_1 \in \mathbb{Z}_p$.

Hybrid H_9: The following sequence of hybrids is basically the reverse of the previous hybrids with $(\boldsymbol{z}^* \parallel \boldsymbol{w}^*)$ replaced with $(\boldsymbol{d}_1 \parallel \boldsymbol{d}_2)$. In this hybrid, we change the distribution of β_τ similar to what we did in H_7. In particular, β_τ is replaced with $\beta_\tau' = \beta_\tau + f_\tau(\boldsymbol{x}^*)\boldsymbol{d}_1[\tau] + \nu_\tau \cdot (\boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma)$ and β_1 is replaced with $\beta_1' = \beta_1 + f_1(\boldsymbol{x}^*)\boldsymbol{d}_1[1] + \nu_1 \cdot (\boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma) - (f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma)$. So, the vectors associated with challenge ciphertext are distributed as

$$\boldsymbol{u} = (\, 1, \boldsymbol{x}^*[i], 0, 0, \boxed{\widetilde{\ell}_{1,\tau}}, 0\,), \quad \boldsymbol{h}_t = (\, 1, 0, 0\,)$$

where $\widetilde{\ell}_{1,\tau} \leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, f_\tau(\boldsymbol{x}^*)\boldsymbol{d}_1[\tau] + \nu_\tau \cdot (\boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma) + \beta_\tau, \ell_{2,\tau}', \ldots, \ell_{m+1,\tau}')$ Note that, H_8 and H_9 are statistically close as $\{\beta_\tau : \tau \in [n']\}$ and $\{\beta_\tau' : \tau \in [n']\}$ are both uniform over $\mathbb{Z}_p$ with $\sum_{\tau \in [n']} \beta_\tau = \sum_{\tau \in [n']} \beta_\tau' = 0 \mod p$. Hence, hybrids H_8 and H_9 are indistinguishable.

Hybrid H_{10}: In this hybrid we change the vectors $\boldsymbol{v}_{m+1,t}$ and $\boldsymbol{h}_t$ as follows

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\quad 0,\quad 0, 0,\ \delta_{t\tau},\ 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t},\quad 0,\quad 0, 0,\ \ 0,\ \ 0\ ) \qquad \text{for } 1 < j \leq m, \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i], 0, 0, \widetilde{\ell}_{1,\tau}, 0\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{0}, 0,\quad \boxed{1}\quad) \\
\boldsymbol{h}_t &= (\ 1,\ 0, \boxed{\ell'_{m+1,t}}\ )
\end{aligned}$$

where $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$. The indistinguishability between the hybrids H_9 and H_{10} follows from the function-hiding security of IPFE.

Hybrid H_{11}: It is exactly the same as H_{10} except that the random values $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$ are changed to the actual label $\ell_{m+1,t} = \boldsymbol{d}_1[t] - \boldsymbol{r}_t[m]$. Then the vectors associated with the challenge ciphertext become

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], 0, 0, \widetilde{\ell}_{1,\tau}, 0\ ), \quad \boldsymbol{h}_t = (\ 1, 0, \boxed{\ell_{m+1,t}}\ )$$

The hybrids H_{11} and H_{12} are identical due to the marginal randomness property of AKGS.

Hybrid H_{12}: In this hybrid we change the vectors $\boldsymbol{v}_{m+1,t}$ and $\boldsymbol{h}_t$ as follows

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\quad 0,\quad 0, 0,\ \delta_{t\tau},\ 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t},\quad 0,\quad 0, 0,\ \ 0,\ \ 0\ ) \qquad \text{for } 1 < j \leq m, \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i], 0, 0, \widetilde{\ell}_{1,\tau}, 0\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{\boldsymbol{r}_t[m]},\ \ \boxed{1},\ \ \boxed{0}\ ) \\
\boldsymbol{h}_t &= (\ \ \boxed{-1},\ \ \boxed{\boldsymbol{d}_1[t]}, \boxed{0}\ )
\end{aligned}$$

The indistinguishability between the hybrids H_{10} and H_{11} follows from the function-hiding security of IPFE.

Hybrid $\mathsf{H}_{13,m+1-j}$ ($j \in [m-1]$): It is analogous to H_{12} except the secret-key is modified as follows. For all j' such that $m+1-j \leq j' < m+1$, the random value $\ell'_{j',t} \leftarrow \mathbb{Z}_p$ is discarded from $\boldsymbol{v}_{j',t}[\mathsf{const}]$ and the coefficient vector $\boldsymbol{\ell}_{j',t}$ is used in $\boldsymbol{v}_{j',t}$.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\qquad 0,\qquad\qquad 0,\qquad 0, 0, \delta_{t\tau}, 0\ ) \\
\boldsymbol{v}_{j',t} &= (\qquad \ell'_{j',t},\qquad\qquad 0,\qquad 0, 0,\ 0,\ 0\ ) \text{ for } 1 < j' < m+1-j, \\
\boldsymbol{v}_{j',t} &= (\ \boxed{\boldsymbol{\ell}_{j',t}[\mathsf{const}]}, \boxed{\boldsymbol{\ell}_{j',t}[\mathsf{coef}_i]}, 0, 0,\ 0,\ 0\ ) \text{ for } m+1-j \leq j' < m+1, \\
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m], 1, 0\ )
\end{aligned}$$

In this hybrid, the label $\widetilde{\ell}_{1,t}$ is reversely sampled using the random values $\ell'_{2,t}, \ldots, \ell'_{m+1-j,t}$ and the actual values $\ell_{m-j+2,t}, \ldots, \ell_{m+1,t}$ for each $t \in [n']$. The hybrids $\mathsf{H}_{13,m+1-(j-1)}$ and $\mathsf{H}_{13,m+1-j}$ can be shown to be indistinguishable via the following sequence of sub-hybrids, namely, $\{\mathsf{H}_{13,m+1-j,1}, \mathsf{H}_{13,m+1-j,2}, \mathsf{H}_{13,m+1-j,3}\}_{j \in [m-1]}$.

Hybrid $\mathsf{H}_{13,m+1-j,1}$ ($j \in [m-1]$): It proceeds exactly the same as $\mathsf{H}_{13,m+1-(j-1)}$ except that the random labels $\ell'_{m+1-j,t}$ are sifted from $\boldsymbol{v}_{m+1-j,t}[\mathsf{const}]$ to $\boldsymbol{u}[\mathsf{sim}^*_\tau]$. We modify the vectors associated with the secret-key and the challenge ciphertext as follows

$$\begin{array}{rll}
\boldsymbol{v}_{1,t} = (& 0, \quad 0, \quad 0, 0, \delta_{t\tau}, \quad 0 &) \\
\boldsymbol{v}_{j',t} = (& \ell'_{j',t}, \quad 0, \quad 0, 0, 0, \quad 0 &) \text{ for } 1 < j' < m+1-j, \\
\boldsymbol{v}_{m+1-j,t} = (& \boxed{0}, \quad 0, \quad 0, 0, 0, \quad \boxed{\delta_{t\tau}} &) \\
\boldsymbol{v}_{j',t} = (& \boldsymbol{\ell}_{j',t}[\mathsf{const}], \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], 0, 0, 0, \quad 0 &) \text{ for } m+1-j < j' < m+1, \\
\boldsymbol{u} = (& 1, \quad \boldsymbol{x}^*[i], \quad 0, 0, \widetilde{\ell}_{1,\tau}, \boxed{\ell'_{m+1-j,\tau}} &)
\end{array}$$

$$\begin{array}{rl}
\boldsymbol{v}_{m+1,t} = & (\boldsymbol{r}_t[m], \quad 1, \quad 0) \\
\boldsymbol{h}_t = & (\ -1, \quad \boldsymbol{d}_1[t], 0)
\end{array}$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-(j-1)}$ and $\mathsf{H}_{13,m+1-j,1}$ follows from the function-hiding security of IPFE.

Hybrid $\mathsf{H}_{13,m+1-j,2}$ ($j \in [m-1]$): It is exactly same as $\mathsf{H}_{13,m+1-j,1}$ except that the random label $\ell'_{m+1-j,\tau} \leftarrow \mathbb{Z}_p$ at $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ are now replaced with the actual labels $\ell_{m+1-j,\tau} = \boldsymbol{\ell}_{m+1-j,\tau}(\boldsymbol{x}^*)$. The change in the vector $\boldsymbol{u}$ associated to the challenge ciphertext is indicated below.

$$\boldsymbol{u} = (1, \boldsymbol{x}^*[i], 0, 0, \widetilde{\ell}_{1,\tau}, \boxed{\ell_{m+1-j,\tau}}), \quad \boldsymbol{h}_t = (-1, \boldsymbol{d}_1[t], 0)$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-j,1}$ and $\mathsf{H}_{13,m+1-j,2}$ follows from the marginal randomness property of AKGS.

Hybrid $\mathsf{H}_{13,m+1-j,3}$ ($j \in [m-1]$): It proceeds analogous to $\mathsf{H}_{13,m+1-j,2}$ except that the actual label $\ell_{m+1-j,\tau} = \boldsymbol{\ell}_{m+1-j,\tau}(\boldsymbol{x}^*)$ is removed from $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ and the coefficient vector $\boldsymbol{\ell}_{m+1-j,t}$ is used to set $\boldsymbol{v}_{m+1-j,t}$. The inner product $\boldsymbol{v}_{m+1-j,t} \cdot \boldsymbol{u}$ is unaltered as in $\mathsf{H}_{13,m+1-j,2}$. The changes in the vectors associated to the secret-key and the challenge ciphertext are shown below.

$$\begin{array}{rll}
\boldsymbol{v}_{1,t} = (& 0, \quad 0, \quad 0, 0, \delta_{t\tau}, \ 0 &) \\
\boldsymbol{v}_{j',t} = (& \ell'_{j',t}, \quad 0, \quad 0, 0, 0, \ 0 &) \text{ for } 1 < j' < m+1-j, \\
\boldsymbol{v}_{m+1-j,t} = (& \boxed{\boldsymbol{\ell}_{m+1-j,t}[\mathsf{const}]}, \boxed{\boldsymbol{\ell}_{m+1-j,t}[\mathsf{coef}_i]}, 0, 0, 0, \ \boxed{0} &) \\
\boldsymbol{v}_{j',t} = (& \boldsymbol{\ell}_{j',t}[\mathsf{const}], \quad \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], \quad 0, 0, 0, \ 0 &) \text{ for } m+1-j < j' < m+1, \\
\boldsymbol{u} = (& 1, \quad \boldsymbol{x}^*[i], \quad 0, 0, \widetilde{\ell}_{1,\tau}, \boxed{0} &)
\end{array}$$

$$\begin{array}{rl}
\boldsymbol{v}_{m+1,t} = & (\boldsymbol{r}_t[m], \quad 1, \quad 0) \\
\boldsymbol{h}_t = & (\ -1, \quad \boldsymbol{d}_1[t], 0)
\end{array}$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-j,2}$ and $\mathsf{H}_{13,m+1-j,3}$ follows from the function-hiding security of IPFE. We observe that $\mathsf{H}_{13,m+1-j,3}$ is identical to $\mathsf{H}_{13,m+1-j}$ for all $j \in [m-1]$.

Hybrid H_{14}: It proceeds exactly the same as $\mathsf{H}_{13,2}$ except that the reversely sampled labels $\widetilde{\ell}_{1,\tau}$ are replaced with the actual labels $\ell_{1,\tau} + \psi_\tau = \boldsymbol{\ell}_{1,\tau}(\boldsymbol{x}^*) + \nu_\tau \cdot (\boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma)$ when setting $\boldsymbol{u}[\mathsf{sim}_\tau]$. The vectors associated with the challenge ciphertext are now written as

$$\boldsymbol{u} = (1, \boldsymbol{x}^*[i], 0, 0, \boxed{\ell_{1,\tau} + \psi_\tau}, 0), \quad \boldsymbol{h}_t = (-1, \boldsymbol{d}_1[t], 0)$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m}$ and H_{14} follows from the piecewise security of AKGS.

Hybrid $\mathbf{H_{15}}$: It is analogous to H_{14} except that the actual label $\ell_{1,\tau} = \boldsymbol{\ell}_{1,\tau}(\boldsymbol{x}^*) + \nu_\tau \cdot (\boldsymbol{y}^\top \boldsymbol{d}_2 + \sigma)$ is removed from $\boldsymbol{u}[\mathsf{sim}_\tau]$ and the coefficient vectors $\boldsymbol{\ell}_{1,t}$ are utilized while setting the vectors $\boldsymbol{v}_{1,t}$ for all $t \in [n']$. Also, the positions $\boldsymbol{v}_{1,t}[\mathsf{extnd}_\kappa], \boldsymbol{v}_{1,t}[\mathsf{query}]$ and $\boldsymbol{u}[\mathsf{extnd}_\kappa], \boldsymbol{u}[\mathsf{query}]$ are set as $\boldsymbol{y}[\kappa]\nu_t, \nu_t$ and $\boldsymbol{d}_2[\kappa], \sigma$ respectively. The vectors associated with the secret-key and the challenge ciphertext are shown below.

$$\begin{array}{rl}
\boldsymbol{v}_{1,t} = (& \boxed{\boldsymbol{\ell}_{1,t}[\mathsf{const}]}, \boxed{\boldsymbol{\ell}_{1,t}[\mathsf{coef}_i]}, \boxed{\boldsymbol{y}[\kappa]\nu_t}, \boxed{\nu_t}, \ 0, \ 0\ ) \\
\boldsymbol{v}_{j,t} = (& \boldsymbol{\ell}_{j,t}[\mathsf{const}], \ \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i], \quad 0, \quad 0, \ 0, \ 0\ ) \qquad \text{for } 1 < j \leq m, \\
\boldsymbol{u} = (& 1, \qquad \boldsymbol{x}^*[i], \quad \boxed{\boldsymbol{d}_2[\kappa]}, \boxed{\sigma}, \boxed{0}, 0\ ) \\
\boldsymbol{v}_{m+1,t} = (& \boldsymbol{r}_t[m], \ 1, \ 0\ ) \\
\boldsymbol{h}_t = (& -1, \ \boldsymbol{d}_1[t], 0\ )
\end{array}$$

Since the inner products $\boldsymbol{v}_{1,t} \cdot \boldsymbol{u} = \ell_{1,t} + \psi_t$, for all $t \in [n']$, remain the same as in H_{14}, the function-hiding security of IPFE ensures the indistinguishability between the hybrids H_{14} and H_{15}. This completes the security analysis as H_{15} is the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal},1-\mathsf{extFE}}(1^\lambda)$. $\square$

7 Bounded-Key One-Slot Extended FE for Attribute-Weighted Sums

In this section, we present a public-key one-slot FE scheme $\Pi^{\mathsf{bdd}}_{\mathsf{extOne}}$ for an extended attribute-weighted sum functionality. This scheme is proven adaptively simulation secure against one ciphertext query, an a priori bounded number of pre-ciphertext secret key queries, and an arbitrary polynomial number of post-ciphertext secret key queries. We will apply the bootstrapping compiler from [AGW20] onto this FE scheme to obtain our unbounded-slot FE scheme for attribute-weighted sums in the next section. We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ for simplicity of notations. Let ($\mathsf{Garble}, \mathsf{Eval}$) be a special piecewise secure AKGS for a function class $\mathcal{F}^{(n,n')}_{\mathsf{ABP}}$, $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ a tuple of pairing groups of prime order p such that MDDH_k holds in $\mathbb{G}_2$, and ($\mathsf{IPFE.Setup.IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec}$) a slotted IPFE based on G. We construct an FE scheme for attribute-weighted sums with the message space $\mathbb{M} = \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$.

$\mathsf{Setup}(1^\lambda, 1^n, 1^{n'}, 1^B)$: Defines the following index sets as follows

$$S_{\mathsf{pub}} = \left\{ \{\mathsf{const}^{(\iota)}\}_{\iota \in [k]}, \{\mathsf{coef}_i^{(\iota)}\}_{\iota \in [k], i \in [n]}, \{\mathsf{extnd}_\kappa^{(\iota)}\}_{\iota, \kappa \in [k]} \right\}, \ \widehat{S}_{\mathsf{pub}} = \{\widehat{\mathsf{const}}^{(\iota)}, \widehat{\mathsf{coef}}^{(\iota)}\}_{\iota \in [k]}$$

$$S_{\mathsf{priv}} = \left\{ \mathsf{const}, \{\mathsf{coef}_i\}_{i \in [n]}, \{\mathsf{extnd}_{\kappa,1}, \mathsf{extnd}_{\kappa,2}, \mathsf{extnd}_\kappa\}_{\kappa \in [k]}, \{\mathsf{query}_\eta\}_{\eta \in [B]}, \{\mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}_{\tau \in [n']} \right\},$$

$$\widehat{S}_{\mathsf{priv}} = \{\widehat{\mathsf{const}}_1, \widehat{\mathsf{coef}}_1, \widehat{\mathsf{const}}_2, \widehat{\mathsf{coef}}_2, \widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}$$

where B denotes a bound on the number of pre-challenge queries. It generates two pair of IPFE keys ($\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}$) $\leftarrow \mathsf{IPFE.Setup}(S_{\mathsf{pub}}, S_{\mathsf{priv}})$ and ($\widehat{\mathsf{IPFE.MSK}}, \widehat{\mathsf{IPFE.MPK}}$) $\leftarrow \mathsf{IPFE.Setup}(\widehat{S}_{\mathsf{pub}}, \widehat{S}_{\mathsf{priv}})$. Finally, it returns the master secret-key of the system as $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$ and master public-key as $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widehat{\mathsf{IPFE.MPK}})$.

KeyGen(MSK, $(\boldsymbol{f}, \boldsymbol{y})$): Let $f = (f_1, \ldots, f_{n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ and $\boldsymbol{y} \in \mathbb{Z}_p^k$. It samples integers $\nu_t \leftarrow \mathbb{Z}_p$ and vectors $\boldsymbol{\alpha}, \boldsymbol{\beta}_t \leftarrow \mathbb{Z}_p^k$ for $t \in [n']$ such that

$$\sum_{t\in[n']} \nu_t = 1 \text{ and } \sum_{t\in[n']} \boldsymbol{\beta}_t[\iota] = 0 \bmod p \text{ for all } \iota \in [k]$$

Next, sample independent random vectors $\boldsymbol{r}_t^{(\iota)} \leftarrow \mathbb{Z}_p^m$ and computes

$$(\boldsymbol{\ell}_{1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{m,t}^{(\iota)}, \boldsymbol{\ell}_{m+1,t}^{(\iota)}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}[\iota]\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \boldsymbol{\beta}_t[\iota]; \boldsymbol{r}_t^{(\iota)})$$

for all $\iota \in [k], t \in [n']$. Here, we make use of the instantiation of the AKGS described in Section 3.6. From the description of that AKGS instantiation, we note that the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}^{(\iota)}$ would be of the form $\boldsymbol{\ell}_{m+1,t}^{(\iota)} = \boldsymbol{\alpha}[\iota]\boldsymbol{z}[t] - \boldsymbol{r}_t^{(\iota)}[m]$ where $\boldsymbol{\alpha}[\iota]$ is a constant. Also all the label functions $\boldsymbol{\ell}_{1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{m,t}^{(\iota)}$ involve only the variables $\boldsymbol{x}$ and not the variable $\boldsymbol{z}[t]$. Next, for all $j \in [2, m]$ and $t \in [n']$, it defines the vectors $\boldsymbol{v}_{j,t}$ corresponding to the label functions $\boldsymbol{\ell}_{j,t}$ obtained from the partial garbling above and the vector $\boldsymbol{y}$ as

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	$\mathsf{extnd}_\kappa^{(\iota)}$	S_{priv}
$\boldsymbol{v}$	$\boldsymbol{\alpha}[\iota]$	0	0	0
$\boldsymbol{v}_{1,t}$	$\boldsymbol{\ell}_{1,t}^{(\iota)}[\mathsf{const}]$	$\boldsymbol{\ell}_{1,t}^{(\iota)}[\mathsf{coef}_i]$	$\boldsymbol{\alpha}[\iota]\boldsymbol{y}[\kappa]\nu_t$	0
$\boldsymbol{v}_{j,t}$	$\boldsymbol{\ell}_{j,t}^{(\iota)}[\mathsf{const}]$	$\boldsymbol{\ell}_{j,t}^{(\iota)}[\mathsf{coef}_i]$	0	0

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$	$\widehat{S}_{\mathsf{priv}}$
$\boldsymbol{v}_{m+1,t}$	$\boldsymbol{r}_t^{(\iota)}[m]$	$\boldsymbol{\alpha}[\iota]$	0

It generates the secret-keys as

$$\begin{aligned}
\mathsf{IPFE.SK} &\leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}]\!]_2)\\
\mathsf{IPFE.SK}_{j,t} &\leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2) \text{ for } j \in [m], t \in [n']\\
\widehat{\mathsf{IPFE.SK}}_{m+1,t} &\leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2) \text{ for } t \in [n']
\end{aligned}$$

Finally, it returns the secret-key as $\mathsf{SK}_{f,\boldsymbol{y}} = (\mathsf{IPFE.SK}, \{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ and $(f, \boldsymbol{y})$.

Remark 4 We note that the vector $\boldsymbol{y}$ is only used to set $\boldsymbol{v}_{1,t}[\mathsf{extnd}_\kappa^{(\iota)}]$ and the IPFE.KeyGen only requires $[\![\boldsymbol{v}_{1,t}]\!]_2 \in \mathbb{G}_2^k$ to compute the secret-key $\mathsf{IPFE.SK}_{1,t}$. Therefore, the key generation process can compute the same secret-key $\mathsf{SK}_{f,\boldsymbol{y}}$ if $(f, [\![\boldsymbol{y}]\!]_2)$ is supplied as input instead of $(f, \boldsymbol{y})$ and we express this by writing $\mathsf{KeyGen}(\mathsf{MSK}, (f, [\![\boldsymbol{y}]\!]_2)) = \mathsf{KeyGen}(\mathsf{MSK}, (f, \boldsymbol{y}))$. This fact will be crucial while describing the unbounded slot FE.

$\mathsf{Enc}(\mathsf{MPK}, (\boldsymbol{x}, \boldsymbol{z}||\boldsymbol{w}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k})$: It samples a random vector $\boldsymbol{s} \leftarrow \mathbb{Z}_p^k$ and sets the vectors

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	$\mathsf{extnd}_\kappa^{(\iota)}$
$\boldsymbol{u}$	$\boldsymbol{s}[\iota]$	$\boldsymbol{s}[\iota]\boldsymbol{x}[i]$	$\boldsymbol{s}[\iota]\boldsymbol{w}[\kappa]$

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$
$\boldsymbol{h}_t$	$-\boldsymbol{s}[\iota]$	$\boldsymbol{s}[\iota]\boldsymbol{z}[t]$

for all $t \in [n']$. It encrypts the vectors as

$$\begin{aligned}\mathsf{IPFE.CT} &\leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)\\ \widehat{\mathsf{IPFE.CT}}_t &\leftarrow \mathsf{IPFE.SlotEnc}(\widehat{\mathsf{IPFE.MPK}}, [\![\boldsymbol{h}_t]\!]_1) \text{ for } t \in [n']\end{aligned}$$

and returns the ciphertext as $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$ and $\boldsymbol{x}$.

$\mathsf{Dec}((\mathsf{SK}_{f,\boldsymbol{y}}, f), (\mathsf{CT}, \boldsymbol{x}))$: It parses the secret-key and ciphertext as $\mathsf{SK}_{f,\boldsymbol{y}} = (\mathsf{IPFE.SK}, \{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ and $\mathsf{CT}_{\boldsymbol{x},\boldsymbol{z}} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$. It uses the decryption algorithm of IPFE to compute

$$\begin{aligned}[\![\rho]\!]_T &\leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}, \mathsf{IPFE.CT})\\ [\![\ell_{1,t} + \psi_t]\!]_T &\leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{1,t}, \mathsf{IPFE.CT})\\ [\![\ell_{j,t}]\!]_T &\leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{j,t}, \mathsf{IPFE.CT}) \text{ for } j \in [2, m], t \in [n']\\ [\![\ell_{m+1,t}]\!]_T &\leftarrow \mathsf{IPFE.Dec}(\widehat{\mathsf{IPFE.SK}}_{m+1,t}, \widehat{\mathsf{IPFE.CT}}_t) \text{ for } t \in [n']\end{aligned}$$

where $\psi_t = \sum_{\iota=1}^k \boldsymbol{\alpha}[\iota]\boldsymbol{s}[\iota] \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w} = \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}$. Next, it utilizes the evaluation procedure of AKGS and obtain a combined value

$$[\![\zeta]\!]_T = \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T).$$

Finally, it returns a value $[\![\mu]\!]_T = [\![\zeta]\!]_T \cdot [\![\rho]\!]_T^{-1} \in \mathbb{G}_T$.

Correctness: First, the IPFE correctness implies $\mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{1,t}, \mathsf{IPFE.CT}) = [\![\ell_{1,t} + \psi_t]\!]$ where $\psi_t = \sum_{\iota=1}^k \boldsymbol{\alpha}[\iota]\boldsymbol{s}[\iota] \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w} = \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}$. Next, by the correctness of IPFE, AKGS we have

$$\begin{aligned}&\mathsf{Eval}(f_t, \boldsymbol{x}, \ell_{1,t} + \psi_t, \ldots, \ell_{m+1,t})\\ &= \mathsf{Eval}(f_t, \boldsymbol{x}, \ell_{1,t}, \ldots, \ell_{m+1,t}) + \mathsf{Eval}(f_t, \boldsymbol{x}, \psi_t, 0, \ldots, 0)\\ &= \mathsf{Eval}(f_t, \boldsymbol{x}, \ell_{1,t}, \ldots, \ell_{m+1,t}) + \psi_t\\ &= \sum_{\iota=1}^k (\boldsymbol{\alpha}[\iota]\boldsymbol{s}[\iota] \cdot \boldsymbol{z}[t] f_t(\boldsymbol{x}) + \boldsymbol{\beta}_t[\iota]\boldsymbol{s}[\iota]) + \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}\\ &= \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot (\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}) + \boldsymbol{\beta}_t \cdot \boldsymbol{s}\end{aligned}$$

The first equality follows from the linearity of Eval algorithm. Therefore, multiplying all the evaluated values we have

$$\begin{aligned}[\![\zeta]\!]_T &= \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) \\ &= [\![\sum_{t=1}^{n'} \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot (\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}) + \boldsymbol{\beta}_t \cdot \boldsymbol{s}]\!]_T = [\![\boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot (f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w})]\!]_T\end{aligned}$$

where the last equality follows from the fact that $\sum_{t\in n'} \nu_t = 1 \bmod p$ and $\sum_{t\in[n']} \boldsymbol{\beta}_t[\iota] = 0 \bmod p$ for all $\iota \in [k]$. Also, by the correctness of IPFE we see that $[\![\rho]\!]_T = [\![\boldsymbol{\alpha} \cdot \boldsymbol{s}]\!]_T$ and hence $[\![\mu]\!]_T = [\![f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w}]\!]_T$.

Theorem 5 *The extended one slot* FE *scheme* $\Pi^{\mathsf{bdd}}_{\mathsf{extOne}}$ *for attribute-weighted sum is adaptively simulation-secure assuming the* AKGS *is piecewise-secure as per Definition 7, the* MDDH_k *assumption holds in group* $\mathbb{G}_2$*, and the slotted* IPFE *is function hiding as per Definition 5.*

7.1 Security Analysis

The Simulator We describe the simulator for the extended one-slot FE scheme $\Pi^{\mathsf{bdd}}_{\mathsf{extOne}}$. The simulated setup algorithm is the same setup of the original scheme. Let $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'}, 1^B) = \mathsf{Setup}(1^\lambda, 1^n, 1^{n'}, 1^B)$ where $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$ and $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widehat{\mathsf{IPFE.MPK}})$.

$\mathbf{KeyGen}_0^*(\mathbf{MSK}, (\boldsymbol{f_q}, \boldsymbol{y_q}))$: On input MSK, a function $f_q = (f_{q,1}, \ldots f_{q,n'}) \in \mathcal{F}^{(n,n')}_{\mathsf{ABP}}$ and a vector $\boldsymbol{y}_q \in \mathbb{Z}_p^k$ the simulator proceeds as follows:

Setting Public Positions: The public positions are set as in the original scheme.

1. It first samples $\boldsymbol{\beta}_{q,t} = (\boldsymbol{\beta}_{q,t}[1], \ldots, \boldsymbol{\beta}_{q,t}[k]) \leftarrow \mathbb{Z}_p^k, \nu_{q,t} \leftarrow \mathbb{Z}_p$ for $t \in [n']$, and $\boldsymbol{r}^{(\iota)}_{q,t} = (\boldsymbol{r}^{(\iota)}_{q,t}[1], \ldots, \boldsymbol{r}^{(\iota)}_{q,t}[m_q]) \leftarrow \mathbb{Z}_p^{m_q}$ where it holds that
$$\sum_{t\in[n']} \boldsymbol{\beta}_{q,t}[\iota] = 0 \bmod p \text{ for all } \iota \in [k] \text{ and } \sum_{t\in[n']} \nu_{q,t} = 1 \bmod p$$
2. Next, it computes the coefficient vectors for the label functions as
$$(\boldsymbol{\ell}^{(\iota)}_{q,1,t}, \ldots, \boldsymbol{\ell}^{(\iota)}_{q,m_q,t}, \boldsymbol{\ell}^{(\iota)}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \boldsymbol{\beta}_{q,t}[\iota]; \boldsymbol{r}^{(\iota)}_{q,t})$$
for all $\iota \in [k], t \in [n']$. From the description of AKGS, we note that the $(m_q + 1)$-th label function $\boldsymbol{\ell}^{(\iota)}_{q,m_q+1,t}$ would be of the form $\boldsymbol{\ell}^{(\iota)}_{q,m_q+1,t} = \boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t] - \boldsymbol{r}^{(\iota)}_{q,t}[m_q]$.
3. It picks $\boldsymbol{\alpha}_q \leftarrow \mathbb{Z}_p^k$ and sets the public positions at the indexes in $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ of following vectors

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	$\mathsf{extnd}_\kappa^{(\iota)}$
$\boldsymbol{v}_q$	$\boldsymbol{\alpha}_q[\iota]$	0	0
$\boldsymbol{v}_{q,1,t}$	$\boldsymbol{\ell}^{(\iota)}_{q,1,t}[\mathsf{const}]$	$\boldsymbol{\ell}^{(\iota)}_{q,1,t}[\mathsf{coef}_i]$	$\boldsymbol{\alpha}_q[\iota]\boldsymbol{y}_q[\kappa]\nu_{q,t}$
$\boldsymbol{v}_{q,j,t}$	$\boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}]$	$\boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i]$	0

for all $j \in [2, m_q]$ and $t \in [n']$. It also sets the following vectors for all $t \in [n']$.

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$
$\boldsymbol{v}_{q,m_q+1,t}$	$\boldsymbol{r}_{q,t}^{(\iota)}[m_q]$	$\boldsymbol{\alpha}_q[\iota]$

Setting Private Positions: It now fills the private indices as follows.

4. It samples $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ for $t \in [n']$ satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0$.
5. Next, it picks $\widetilde{\boldsymbol{r}}_{q,t} \leftarrow \mathbb{Z}_p^{m_q}$ and computes the coefficient vectors for the label functions as

$$(\widetilde{\boldsymbol{\ell}}_{q,1,t}, \ldots, \widetilde{\boldsymbol{\ell}}_{q,m_q,t}, \widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\widetilde{\alpha}_q \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\beta}_{q,t}; \widetilde{\boldsymbol{r}}_{q,t}).$$

for all $t \in [n']$. From the description of AKGS, we note that the $(m_q + 1)$-th label function $\widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}$ would be of the form $\widetilde{\boldsymbol{\ell}}_{q,m_q+1,t} = \widetilde{\alpha}_q \boldsymbol{z}^*[t] - \widetilde{\boldsymbol{r}}_{q,t}[m_q]$.
6. Now, it fills the private positions at the indexes in $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ as follows

vector	const	coef_i	$\mathsf{extnd}_{\kappa,1}$	$\mathsf{extnd}_{\kappa,2}$	extnd_{κ}	query_{η}	sim_{τ}	sim^*_{τ}
$\boldsymbol{v}_q$	$\widetilde{\alpha}_q$	0	0	0	0	0	0	0
$\boldsymbol{v}_{q,1,t}$	$\widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}]$	$\widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i]$	0	$\widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}$	0	$\widetilde{\alpha}_q \boldsymbol{e}_q[\eta]\nu_{q,t}$	0	0
$\boldsymbol{v}_{q,j,t}$	$\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]$	$\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]$	0	0	0	0	0	0

for all $j \in [2, m_q]$ and $t \in [n']$; and for all $t \in [n']$

vector	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{q,m_q+1,t}$	0	0	$\widetilde{\boldsymbol{r}}_{q,t}[m_q]$	$\widetilde{\boldsymbol{\alpha}}_q$	0	0	0

where $\boldsymbol{e}_q \in \{0,1\}^B$ such that $\boldsymbol{e}_q[\eta] = 1$ if $\eta = q$; 0 otherwise.
7. It generates the IPFE secret-keys

$$\mathsf{IPFE.SK}_q \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_q]\!]_2)$$
$$\mathsf{IPFE.SK}_{q,j,t} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{q,j,t}]\!]_2) \text{ for } j \in [m_q], t \in [n']$$
$$\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} \leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{q,m_q+1,t}]\!]_2) \text{ for } t \in [n']$$

8. Finally, it returns the secret-key $\mathsf{SK}_{f_q,\boldsymbol{y}_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$.

Let Q_{pre} be the total number of secret-key queries made before the challenge query and hence without loss of generality we take $B = Q_{\mathsf{pre}}$.

Remark. Suppose the simulator only gets $[\![\boldsymbol{y}_q]\!]_2$ instead of $\boldsymbol{y}_q$. We observe that the components of $\boldsymbol{y}_q$ are used to set $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_{\kappa}^{\iota}]$ and $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_{\kappa,2}]$. Since the elements $\boldsymbol{\alpha}_q[\iota], \widetilde{\alpha}_q$ and $\nu_{q,t}$ are sampled by the simulator, it can compute $[\![\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_{\kappa}^{\iota}]]\!]_2 = \boldsymbol{\alpha}_q[\iota]\nu_{q,t} \cdot [\![\boldsymbol{y}_q[\kappa]]\!]_2$ and $[\![\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_{\kappa,2}]]\!]_2 = \widetilde{\alpha}_q \nu_{q,t} \cdot [\![\boldsymbol{y}_q[\kappa]]\!]_2$. The simulator only requires to know $[\![\boldsymbol{v}_{q,1,t}]\!]_2$ in order to generate $\mathsf{IPFE.SK}_{q,1,t}$. In this context, we write $\mathsf{KeyGen}_0^*(\mathsf{MSK}, (f_q, [\![\boldsymbol{y}_q]\!]_2)) = \mathsf{KeyGen}_0^*(\mathsf{MSK}, (f_q, \boldsymbol{y}_q))$ for all $q \in [Q_{\mathsf{pre}}]$. We emphasize that this fact is crucial for the security analysis of the unbounded slot scheme.

$\mathsf{Enc}^*(\mathsf{MPK}, \mathsf{MSK}, \boldsymbol{x}^*, \mathcal{V})$: On input $\mathsf{MPK}, \mathsf{MSK}$, a vector $\boldsymbol{x}^* \in \mathbb{Z}_p^n$ and a set $\mathcal{V} = \{(f_q, f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) : q \in [Q_{\mathsf{pre}}]\}$ the simulator executes the following steps:

1. It samples a dummy vector $(\boldsymbol{d}_1||\boldsymbol{d}_2||\boldsymbol{d}_3) \in \mathbb{Z}_p^{n'+k+Q_{\mathsf{pre}}}$ from the set

$$\mathcal{D} = \{(\boldsymbol{d}_1||\boldsymbol{d}_2||\boldsymbol{d}_3) \in \mathbb{Z}_p^{n'+k+Q_{\mathsf{pre}}} : f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2 + \boldsymbol{e}_q^\top \boldsymbol{d}_3 = \mu_q \text{ for all } q \in [Q_{\mathsf{pre}}]\}$$

   where $\mu_q = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*$. The sampling procedure works as follows. First, the simulator selects two random vectors $\boldsymbol{d}_1 \in \mathbb{Z}_p^{n'}, \boldsymbol{d}_2 \leftarrow \mathbb{Z}_p^k$ and sets $\sigma_q = \mu_q - f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 - \boldsymbol{y}_q^\top \boldsymbol{d}_2 \in \mathbb{Z}_p$. Then, it sets $\boldsymbol{d}_3[\eta] = \sigma_\eta$[4] for all $\eta \in [Q_{\mathsf{pre}}]$. Therefore, one may observe that $f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2 + \boldsymbol{e}_q^\top \boldsymbol{d}_3 = \mu_q$ for all $q \in [Q_{\mathsf{pre}}]$.
2. Next, it sets the following vectors:

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	$\mathsf{extnd}_\kappa^{(\iota)}$
$\boldsymbol{u}$	0	0	0

vector	const	coef_i	$\mathsf{extnd}_{\kappa,1}$	$\mathsf{extnd}_{\kappa,2}$	extnd_κ	query_η	sim_τ	sim_τ^*
$\boldsymbol{u}$	1	$\boldsymbol{x}^*[i]$	0	$\boldsymbol{d}_2[\kappa]$	0	$\boldsymbol{d}_3[\eta]$	0	0

   and for all $t \in [n']$

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{h}_t$	0	0	1	0	-1	$\boldsymbol{d}_1[t]$	0	0	0

3. It encrypts the vectors as

$$\mathsf{IPFE.CT} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)$$
$$\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{IPFE.Enc}(\widehat{\mathsf{IPFE.MPK}}, [\![\boldsymbol{h}_t]\!]_1) \text{ for } t \in [n']$$

4. It returns the ciphertext as $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

$\mathsf{KeyGen}_1^*(\mathsf{MSK}^*, \boldsymbol{x}^*, (\boldsymbol{f_q}, \boldsymbol{y_q}), \boldsymbol{f_q}(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y_q}^\top \boldsymbol{w}^*)$: On input MSK^*, $\boldsymbol{x}^* \in \mathbb{Z}_p^n$, a function $f_q = (f_{q,1}, \ldots, f_{q,n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, a vector $\boldsymbol{y}_q \in \mathbb{Z}_p^k$ for $q \in [Q_{\mathsf{pre}}+1, Q]$ and $(f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) \in \mathbb{Z}_p$ the simulator proceeds as follows:
Setting Public Positions:

1. The simulator sets the public positions at the indexes in $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ analogous to $\mathsf{KeyGen}_0^*(\mathsf{MSK}^*, (f_q, \boldsymbol{y}_q))$.

Setting Private Positions:

2. First, it samples a random element $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$, for $t \in [n']$, satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0$ and then runs the simulator of the AKGS to obtain

[4] If the number of pre-challenge query Q_{pre} is strictly less than B, then $\boldsymbol{d}_3 \in \mathbb{Z}_p^B$ and we can simply take $\boldsymbol{d}_3[\eta] = 0$ for all $Q_{\mathsf{pre}} < \eta \leq B$.

$$(\widehat{\ell}_{q,1,1}, \ldots, \widehat{\ell}_{q,m_q,1}, \widehat{\ell}_{q,m_q+1,1}) \leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, \widetilde{\alpha}_q \cdot (f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) + \widetilde{\beta}_{q,1})$$
$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \widetilde{\beta}_{q,t}) \quad \text{for } 1 < t \leq n'.$$

3. Next, it fills the private positions at the indices in $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ as follows

vector	const	coef_i	$\mathsf{extnd}_{\kappa,1}$	$\mathsf{extnd}_{\kappa,2}$	extnd_{κ}	query_{η}	sim_{τ}	sim^*_{τ}
$\boldsymbol{v}_q$	$\widetilde{\alpha}_q$	0	0	0	0	0	0	0
$\boldsymbol{v}_{q,j,t}$	$\widehat{\ell}_{q,j,t}$	0	0	0	0	0	0	0

for all $j \in [m_q]$ and $t \in [n']$; and

vector	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{q,m_q+1,t}$	$\widehat{\ell}_{q,m_q+1,t}$	0	0	0	0	0	0

for all $t \in [n']$.

4. It generates the IPFE secret-keys

$$\mathsf{IPFE.SK}_q \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_q]\!]_2)$$
$$\mathsf{IPFE.SK}_{q,j,t} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{q,j,t}]\!]_2) \text{ for } j \in [m_q], t \in [n']$$
$$\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} \leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{q,m_q+1,t}]\!]_2) \text{ for } t \in [n']$$

5. It outputs the secret-key $\mathsf{SK}_{f_q,\boldsymbol{y}_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$.

Remark. *Suppose the simulator is provided with $(f_q, [\![\boldsymbol{y}_q]\!]_2)$ as secret-key query and it only knows $[\![f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*]\!]_2 = [\![\mu_q]\!]_2$. Then, it can simulate the public positions using $[\![\boldsymbol{y}_q]\!]_2$ as described at the end of the description of $\mathsf{KeyGen}_0^*(\cdot)$. Now, for private positions, the simulator samples $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ (as above) and computes $\widetilde{\alpha}_q \cdot [\![\mu_q]\!]_2 = [\![\widetilde{\alpha}_q \mu_q]\!]_2$. Next, it employs the simulator of* AKGS *as follows:*

$$(\widehat{\ell}_{q,1,1}, \ldots, \widehat{\ell}_{q,m_q,1}, \widehat{\ell}_{q,m_q+1,1}) \leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, [\![\widetilde{\alpha}_q \mu_q + \widetilde{\beta}_{q,1}]\!]_2)$$
$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, [\![\widetilde{\beta}_{q,t}]\!]_2) \quad \text{for } 1 < t \leq n'.$$

Thus, the vectors $\boldsymbol{v}_{q,j,t}$ for all $j \in [m_q]$ are available in the exponent of source group $\mathbb{G}_2$ and hence the simulator successfully executes key generation of IPFE *with $[\![\boldsymbol{v}_{q,j,t}]\!]_2$. We express it by writing $\mathsf{KeyGen}_1^*(\mathsf{MSK}^*, \boldsymbol{x}^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\mu_q]\!]_2) = \mathsf{KeyGen}_1^*(\mathsf{MSK}^*, \boldsymbol{x}^*, (f_q, \boldsymbol{y}_q), \mu_q)$ and note that this fact is, in particular, useful for the security analysis of our unbounded slot scheme.*

Hybrids and Reductions

Proof. We use a sequence of hybrid experiments to establish the indistinguishability between the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,extFE}}(1^\lambda)$ and the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,extFE}}(1^\lambda)$ where $\mathcal{A}$ is any PPT adversary. In each experiment, $\mathcal{A}$ can query a polynomial number of secret-key queries for pairs $(f, \boldsymbol{y}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')} \times \mathbb{Z}_p^k$, both before and after submitting the challenge message $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$. Let Q be the total number of secret-key queries and $B = Q_{\mathsf{pre}}\ (\leq Q)$ be the number of secret-keys queried before submitting the challenge message. We denote the q-th secret-key by $\mathsf{SK}_{f_q,\boldsymbol{y}_q}$ corresponding to a function f_q and a vector $\boldsymbol{y}_q$. For the ease of presentation, we write the vector elements sitting in the public slots $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ in *blue* color and the vector elements sitting in the private slots $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ in *red* color. More precisely, we do this so that while describing the hybrid games, we sometimes omit the public parts of the vectors and write down only the private parts when the changes occur only in the private parts. Now, we describe the hybrids as follows:

Hybrid $\mathbf{H_0}$: : This is the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,extFE}}(1^\lambda)$ defined in Definition 4 (with single slot, i.e., $N = 1$). The q-th secret-key $\mathsf{SK}_{f_q,\boldsymbol{y}_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$ is computed using the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ given by

$$\begin{aligned}
\boldsymbol{v}_q &= (\quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad 0, \quad 0, 0, 0, 0, 0, 0, 0, 0),\\
\boldsymbol{v}_{q,1,t} &= (\boldsymbol{\ell}^{(\iota)}_{q,1,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,1,t}[\mathsf{coef}_i], \boldsymbol{\alpha}_q[\iota]\boldsymbol{y}_q[\kappa]\nu_{q,t}, 0, 0, 0, 0, 0, 0, 0, 0),\\
\boldsymbol{v}_{q,j,t} &= (\boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}], \boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i], \quad 0, \quad 0, 0, 0, 0, 0, 0, 0, 0),\\
\boldsymbol{v}_{q,m_q+1,t} &= (\quad \boldsymbol{r}^{(\iota)}_{q,t}[m_q], \quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad 0, 0, 0, 0, 0, 0)
\end{aligned}$$

for $j \in [2, m_q]$ and $t \in [n']$. Note that $\boldsymbol{\alpha}_q$ and $\boldsymbol{r}^{(\iota)}_{q,t}$ are random vectors sampled from $\mathbb{Z}_p^k$ and $\mathbb{Z}_p^{m_q}$ respectively. The integers $\nu_{q,t}$ for $t \in [n']$ is picked randomly from $\mathbb{Z}_p$ such that $\sum_{t\in[n']} \nu_{q,t} = 1$. For all $t \in [n']$, the garblings are computed as

$$(\boldsymbol{\ell}^{(\iota)}_{q,1,t}, \ldots, \boldsymbol{\ell}^{(\iota)}_{q,m_q,t}, \boldsymbol{\ell}^{(\iota)}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t]f_{q,t}(\boldsymbol{x}^*) + \boldsymbol{\beta}_{q,t}[\iota]; \boldsymbol{r}^{(\iota)}_{q,t})$$

where $f_q = (f_{q,1}, \ldots, f_{q,n'})$ and $\boldsymbol{\beta}_{q,t} \leftarrow \mathbb{Z}_p^k$ with $\sum_{t\in[n']} \boldsymbol{\beta}_{q,t}[\iota] = 0\ \forall \iota \in [k]$. The challenge ciphertext $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$ corresponds to the challenge vectors $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$ is computed using the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ given by

$$\begin{aligned}
\boldsymbol{u} &= (\quad \boldsymbol{s}[\iota], \quad \boldsymbol{s}[\iota]\boldsymbol{x}^*[i], \boldsymbol{s}[\iota]\boldsymbol{w}^*[\kappa], \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp),\\
\boldsymbol{h}_t &= (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t], \quad \perp, \quad \perp, \perp, \perp, \perp, \perp, \perp)
\end{aligned}$$

for $t \in [n']$ and $\boldsymbol{s} \leftarrow \mathbb{Z}_p^k$. Note that, in real experiment CT^* is computed using $\mathsf{IPFE.SlotEnc}$ and therefore the elements sitting at the indices in S_{priv} are set as $\perp$ for the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$.

Hybrid $\mathbf{H_1}$: It is exactly the same as hybrid H_0 except the fact that instead of using $\mathsf{IPFE.SlotEnc}$, here the challenge ciphertext CT^* is generated applying $\mathsf{IPFE.Enc}$ which uses $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$ to encrypt the vectors. We indicate this change by changing the private positions of $\boldsymbol{u}$ and $\boldsymbol{h}_t$ from $\perp$ to 0. Thus the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ become

$$\begin{aligned}
\boldsymbol{u} &= (\quad \boldsymbol{s}[\iota], \quad \boldsymbol{s}[\iota]\boldsymbol{x}^*[i], \boldsymbol{s}[\iota]\boldsymbol{w}^*[\kappa], \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}),\\
\boldsymbol{h}_t &= (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t], \quad \boxed{0}, \quad \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}).
\end{aligned}$$

The slot-mode correctness of IPFE guarantees that the two hybrids H_0 and H_1 are identically distributed.

Hybrid $\mathbf{H_2}$: This hybrid is similar to H_1 except that in the private slots of the vectors $\boldsymbol{v}_{q,j,t}$ we put a garbling that linearly combines k garblings (of the public slots) with weight vector $\boldsymbol{s} \in \mathbb{Z}_p^k$ and in the private slots of the vector $\boldsymbol{v}_q$ we use a single random element combining the weight vector $\boldsymbol{s}$. Accordingly, we modify the challenge ciphertext CT^* by omitting the weight vector $\boldsymbol{s}$ and setting the public slots of the vectors $\boldsymbol{u}, \boldsymbol{h}_t$ to zero so that the inner products computed at the time of decryption remains the same in the previous hybrids.

In H_1, the public slots of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ are occupied by vectors $\boldsymbol{\alpha}_q \in \mathbb{Z}_p^k, \nu_{q,t} \in \mathbb{Z}_p$ for $t \in [n']$ and the garblings $\boldsymbol{\ell}_{q,j,t}^{(\iota)}$ computed using randomness $\boldsymbol{r}_{q,t}^{(\iota)} \in \mathbb{Z}_p^{m_q}$. In the public slots of the vectors $\boldsymbol{u}, \boldsymbol{h}_t$, we use $(\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{x}^*[i]), (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t])$ respectively. Therefore, at the time of decryption we recover $[\![\rho_q]\!]_T, [\![\ell_{q,j,t}]\!]_T$ such that

$$\begin{aligned}
\rho_q &= \boldsymbol{\alpha}_q \cdot \boldsymbol{s} = \overline{\alpha}_q \text{ (say)}, \\
\ell_{q,1,t} &= (\boldsymbol{\ell}_{q,1,t}^{(1)}, \ldots, \boldsymbol{\ell}_{q,1,t}^{(k)}) \cdot (\boldsymbol{s}[1](1, \boldsymbol{x}^*), \ldots, \boldsymbol{s}[k](1, \boldsymbol{x}^*)) + \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot \boldsymbol{y}^\top \boldsymbol{w} \cdot \nu_{q,t} \\
&= (\boldsymbol{s}[1]\boldsymbol{\ell}_{q,1,t}^{(1)}, \ldots, \boldsymbol{s}[k]\boldsymbol{\ell}_{q,1,t}^{(k)}) \cdot ((1, \boldsymbol{x}^*), \ldots, (1, \boldsymbol{x}^*)) + \overline{\alpha}_q \cdot \boldsymbol{y}^\top \boldsymbol{w} \cdot \nu_{q,t} \\
&= \overline{\boldsymbol{\ell}}_{q,1,t} \cdot (1, \boldsymbol{x}^*) + \overline{\alpha}_q \cdot \boldsymbol{y}^\top \boldsymbol{w} \cdot \nu_{q,t} \\
\ell_{q,j,t} &= (\boldsymbol{\ell}_{q,j,t}^{(1)}, \ldots, \boldsymbol{\ell}_{q,j,t}^{(k)}) \cdot (\boldsymbol{s}[1](1, \boldsymbol{x}^*), \ldots, \boldsymbol{s}[k](1, \boldsymbol{x}^*)) \\
&= \overline{\boldsymbol{\ell}}_{q,j,t} \cdot (1, \boldsymbol{x}^*)
\end{aligned}$$

where $\overline{\boldsymbol{\ell}}_{q,j,t} = \sum_{\iota \in [k]} \boldsymbol{s}[\iota]\boldsymbol{\ell}_{q,j,t}^{(\iota)}$ for all $j \in [2, m_q]$ and $t \in [n']$. Similarly, the $m_q + 1$-the garbling returns

$$\begin{aligned}
\ell_{q,m_q+1,t} &= ((\boldsymbol{r}_{q,t}^{(1)}[m_q], \boldsymbol{\alpha}_q[1]), \ldots, (\boldsymbol{r}_{q,t}^{(k)}[m_q], \boldsymbol{\alpha}_q[k])) \cdot (\boldsymbol{s}[1](-1, \boldsymbol{z}^*[t]), \ldots, \boldsymbol{s}[k](-1, \boldsymbol{z}^*[t])) \\
&= (\boldsymbol{s}[1](\boldsymbol{r}_{q,t}^{(1)}[m_q], \boldsymbol{\alpha}_q[1]), \ldots, \boldsymbol{s}[k](\boldsymbol{r}_{q,t}^{(k)}[m_q], \boldsymbol{\alpha}_q[k])) \cdot ((-1, \boldsymbol{z}^*[t]), \ldots, (-1, \boldsymbol{z}^*[t])) \\
&= (\overline{\boldsymbol{r}}_{q,t}[m_q], \overline{\alpha}_q) \cdot (-1, \boldsymbol{z}^*[t])
\end{aligned}$$

where $\overline{\boldsymbol{r}}_{q,t}[m_q] = \sum_{\iota \in [k]} \boldsymbol{s}[\iota]\boldsymbol{r}_{q,t}^{(\iota)}[m_q]$. In H_2, we use $\overline{\alpha}_q, \overline{\boldsymbol{\ell}}_{q,j,t}$ and $\overline{\boldsymbol{r}}_{q,t}[m_q]$ in the private slots of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ as described below

$$\begin{aligned}
\boldsymbol{v}_q &= (\ \boxed{\overline{\alpha}_q},\ 0,\ 0,\ 0, 0, 0, 0, 0\ ), \\
\boldsymbol{v}_{q,1,t} &= (\ \boxed{\overline{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}]},\ \boxed{\overline{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i]},\ \boxed{\overline{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}},\ 0, 0, 0, 0, 0\ ), \\
\boldsymbol{v}_{q,j,t} &= (\ \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]},\ \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]},\ 0,\ 0, 0, 0, 0, 0\ ) \quad \text{for } j \in [2, m_q], \\
\boldsymbol{v}_{q,m_q+1,t} &= (\ \boxed{\overline{\boldsymbol{r}}_{q,t}[m_q]},\ \boxed{\overline{\alpha}_q},\ 0,\ 0, 0, 0, 0\ )
\end{aligned}$$

Since the weight vector $\boldsymbol{s}$ is not required to generate the challenge ciphertext CT^*, we omit using it in the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$. Moreover, the public slots of $\boldsymbol{u}$ and $\boldsymbol{h}_t$ are set to zero as the inner product is computed through the private slots only. We describe the changes below.

$$\begin{aligned}
\boldsymbol{u} &= (\ \boxed{0}, \boxed{0}, \boxed{0},\ \boxed{1},\ \boxed{\boldsymbol{x}^*[i]}, \boxed{\boldsymbol{w}^*[\kappa]}, 0, 0, 0, 0, 0\ ), \\
\boldsymbol{h}_t &= (\ \boxed{0}, \boxed{0}, \boxed{-1}, \boxed{\boldsymbol{z}^*[t]},\ 0,\ 0,\ 0, 0, 0\ ).
\end{aligned}$$

Finally, we observe that the inner products $\boldsymbol{v}_q \cdot \boldsymbol{u}, \boldsymbol{v}_{q,j,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ remain the same as in H_1. Thus, the function hiding property of IPFE preserves the indistinguishability between the hybrids H_1 and H_2.

Note that, in this hybrid we pick $\boldsymbol{\alpha}_q, \boldsymbol{\beta}_{q,t}, \boldsymbol{s} \leftarrow \mathbb{Z}_p^k, \nu_{q,t} \leftarrow \mathbb{Z}_p$ and $\boldsymbol{r}_{q,t}^{(\iota)} \leftarrow \mathbb{Z}_p^{m_q}$ for all $t \in [n'], \iota \in [k]$ satisfying $\sum_{t \in [n']} \boldsymbol{\beta}_{q,t}[\iota] = 0$ for each $\iota \in [k]$ and $\sum_{t \in [n']} \nu_{q,t} = 1$. Then, the linearity of the Garble algorithm allows us to write

$$(\overline{\boldsymbol{\ell}}_{q,1,t}, \ldots, \overline{\boldsymbol{\ell}}_{q,m_q,t}, \overline{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\overline{\alpha}_q \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \overline{\beta}_{q,t}; \overline{\boldsymbol{r}}_{q,t})$$

where $\overline{\boldsymbol{\ell}}_{q,j,t} = \sum_{\iota \in [k]} \boldsymbol{s}[\iota]\boldsymbol{\ell}_{q,j,t}^{(\iota)}, \overline{\boldsymbol{r}}_{q,t} = \sum_{\iota \in [k]} \boldsymbol{s}[\iota]\boldsymbol{r}_{q,t}^{(\iota)}$ and $\overline{\beta}_{q,t} = \boldsymbol{\beta}_{q,t} \cdot \boldsymbol{s}$.

From the next hybrid onward the public slots of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ are unaltered for all $q \in [Q], j \in [k]$ and $t \in [n']$. Therefore, we only write the components sitting in the private slots of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ assuming that the components of public slots are the same as in the real experiment. We denote the private slots of the vectors by $\boldsymbol{v}_q|_{S_{\mathsf{priv}}}, \boldsymbol{v}_{q,j,t}|_{S_{\mathsf{priv}}}$ and $\boldsymbol{v}_{q,m_q+1,t}|_{\widehat{S}_{\mathsf{priv}}}$.

Hybrid H_3: It is analogous to H_2 except the liner combinations $\overline{\alpha}_q, \overline{\boldsymbol{\ell}}_{q,j,t}, \overline{\boldsymbol{r}}_{q,t}$ in the private slots of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}, \boldsymbol{v}_{q,m_q+1,t}$ are replaced with freshly and independently generated random values and garblings $\widetilde{\alpha}_q, \widetilde{\boldsymbol{\ell}}_{q,j,t}, \widetilde{\boldsymbol{r}}_{q,t}$. More specifically, we sample random elements $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ for all $t \in [n']$ such that $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0$ and a vector $\overline{\boldsymbol{r}}_{q,t} \leftarrow \mathbb{Z}_p^{m_q}$. Then, the garblings are computed as

$$(\widetilde{\boldsymbol{\ell}}_{q,1,t}, \dots, \widetilde{\boldsymbol{\ell}}_{q,m_q,t}, \widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\widetilde{\alpha}_q \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\beta}_{q,t}; \widetilde{\boldsymbol{r}}_{q,t})$$

for all $t \in [n']$. The vectors involved in the computation of $\mathsf{SK}_{f_q,\boldsymbol{y}_q}$ are as follows:

$$\begin{aligned}
\boldsymbol{v}_q &= (\ \boxed{\widetilde{\alpha}_q},\ 0,\ 0,\ 0, 0, 0,\ 0,\ 0\ ),\\
\boldsymbol{v}_{q,1,t} &= (\ \boxed{\widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}]},\ \boxed{\widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i]},\ \boxed{\widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}},\ 0, 0, 0,\ 0,\ 0\ ),\\
\boldsymbol{v}_{q,j,t} &= (\ \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]},\ \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]},\ 0,\ 0, 0, 0,\ 0,\ 0\ ) \quad \text{for } j \in [2, m_q],\\
\boldsymbol{v}_{q,m_q+1,t} &= (\ \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]},\ \boxed{\widetilde{\alpha}_q},\ 0,\ 0, 0, 0, 0\ )
\end{aligned}$$

Recall that in H_2, the following linear combinations

$$\overline{\alpha}_q = \boldsymbol{\alpha}_q \cdot \boldsymbol{s}, \quad \overline{\beta}_{q,t} = \boldsymbol{\beta}_{q,t} \cdot \boldsymbol{s}, \quad \overline{\boldsymbol{r}}_{q,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{r}_{q,t}^{(\iota)}$$

with a common weight vector $\boldsymbol{s}$ has been used to set $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$. On the other hand, in H_3 fresh and independent random elements $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t}, \widetilde{\boldsymbol{r}}_{q,t}$ are used to compute $\mathsf{SK}_{f_q,\boldsymbol{y}_q}$. Note that the elements of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ are only used in the exponent of the source group $\mathbb{G}_2$ while generating the IPFE secret-keys. Let us consider the matrix $\mathbf{A}_{q,t} = (\boldsymbol{\alpha}_q|\boldsymbol{\beta}_{q,t}|(\mathbf{R}_{q,t})^\top) \in \mathbb{Z}_p^{k\times(m_q+1)}$ where $\mathbf{R}_{q,t} = (\boldsymbol{r}_{q,t}^{(1)}|\dots|\boldsymbol{r}_{q,t}^{(k)}) \in \mathbb{Z}_p^{m\times k}$. Since the matrix $\mathbf{A}_{q,t}$ is uniformly chosen from $\mathbb{Z}_p^{k\times(m_q+1)}$ and $\boldsymbol{s}$ is uniform over $\mathbb{Z}_p^k$, by the MDDH_k assumption in group $\mathbb{G}_2$ we have

$$\underbrace{([\![\mathbf{A}_{q,t}]\!]_2, [\![\mathbf{A}_{q,t}^\top \boldsymbol{s}]\!])}_{\text{in } \mathsf{H}_2} \approx \underbrace{([\![\mathbf{A}_{q,t}]\!]_2, [\![(\widetilde{\alpha}_q, \widetilde{\beta}_{q,t}, \widetilde{\boldsymbol{r}}_{q,t})]\!]_2)}_{\text{in } \mathsf{H}_3}$$

holds for all $q \in [Q]$ and $t \in [n']$. Hence, the two hybrids H_2 and H_3 are indistinguishable under the MDDH_k assumption.

We have completed the first phase of our security analysis as we see that the private slots of the vectors associated to secret-keys and the challenge ciphertext are now computed similar to our extended 1-FE scheme. From the next hybrid, we modify the vectors in such a way that all the pre-challenge secret-key queries decrypt the challenge ciphertext without using the slots of $\boldsymbol{u}$ and $\boldsymbol{h}_t$ where the challenge massage $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*)$ are used.

Hybrid H_4: It proceeds similar to hybrid H_3 except we change the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ for all $t \in [n']$ which are used in the computation of the challenge ciphertext. After all the pre-challenge secret-key queries made by $\mathcal{A}$, a dummy vector $(\boldsymbol{d}_1||\boldsymbol{d}_2||\boldsymbol{d}_3) \in \mathbb{Z}_p^{n'+k+Q_{\mathsf{pre}}}$ is picked from the set

$$\mathcal{D} = \{(\boldsymbol{d}_1||\boldsymbol{d}_2||\boldsymbol{d}_3) \in \mathbb{Z}_p^{n'+k+Q_{\mathsf{pre}}} : f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2 + \boldsymbol{e}_q^\top \boldsymbol{d}_3 = \mu_q \text{ for all } q \in [Q_{\mathsf{pre}}]\}$$

where $\mu_q = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*$. The sampling procedure is as described in the algorithm $\mathsf{Enc}^*(\cdot)$. Then the vectors $\boldsymbol{u}, \boldsymbol{h}_t$ are defined as below.

$$\begin{aligned}
\boldsymbol{u} &= (\ 0, 0,\ 0,\ 1,\ \boldsymbol{x}^*[i],\ \boldsymbol{w}^*[\kappa],\ \boxed{\boldsymbol{d}_2[\kappa]},\ \boxed{\boldsymbol{w}^*[\kappa]},\ 0,\ 0, 0\ ),\\
\boldsymbol{h}_t &= (\ 0, 0, -1,\ \boldsymbol{z}^*[t],\ \boxed{-1},\ \boxed{\boldsymbol{d}_1[t]},\ \boxed{-1},\ \boxed{\boldsymbol{z}^*[t]},\ 0\ ).
\end{aligned}$$

Note that, these changes in $\boldsymbol{u}$ and $\boldsymbol{h}_t$ have no effect in the final inner product values of $\boldsymbol{v}_q \cdot \boldsymbol{u}, \boldsymbol{v}_{q,j,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$. This is because the elements at the slots $(\mathsf{extnd}_{\kappa,2}, \mathsf{extnd}_\kappa)$ of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ $\boldsymbol{h}_t$ and the elements at the slots $(\widehat{\mathsf{const}}_2, \widehat{\mathsf{coef}}_2, \widehat{\mathsf{const}}, \widehat{\mathsf{coef}})$ of the vector $\boldsymbol{v}_{q,m_q+1,t}$ (where the changes take place in $\boldsymbol{u}, \boldsymbol{h}_t$) are all zero. Therefore, by the function hiding property of IPFE the hybrids H_3 and H_4 remain indistinguishable to the adversary.

Hybrid $\mathsf{H}_{5,q}$ ($q \in [Q_{\mathsf{pre}}]$): It proceeds similar to H_4 except that for each $1 \leq q' \leq q$, we modify the vectors $\boldsymbol{v}_{q,1,t}$ and $\boldsymbol{v}_{q,m_q+1,t}$ as described below.

$$\begin{array}{rlllllllll}
\boldsymbol{v}_{q',1,t} = (& \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], & \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], & \boxed{0}, & \boxed{\widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}}, & 0, & \boxed{\widetilde{\alpha}_{q'}\boldsymbol{e}_{q'}[\eta]\nu_{q',t}}, & 0, 0\ ) & \text{for } 1 \leq q' \leq q, \\
\boldsymbol{v}_{q',1,t} = (& \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], & \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], & \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, & 0, & 0, & 0, & 0, 0\ ) & \text{for } q < q' \leq Q_{\mathsf{pre}}, \\
\boldsymbol{v}_{q',m_{q'}+1,t} = (& 0, & 0, & \boxed{\widetilde{\boldsymbol{r}}_{q',t}[m_{q'}]}, & \boxed{\widetilde{\alpha}_{q'}}, & 0, & 0, & 0\ ) & \text{for } 1 \leq q' \leq q, \\
\boldsymbol{v}_{q',m_{q'}+1,t} = (& \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], & \widetilde{\alpha}_{q'}, & 0, & 0, & 0, & 0, & 0\ ) & \text{for } q < q' \leq Q_{\mathsf{pre}}
\end{array}$$

Note that, the post-challenge secret-key queries are still answered according to H_4. Observe that $\mathsf{H}_{5,0}$ coincides with H_4. We will prove that $\mathsf{H}_{5,(q-1)}$ and $\mathsf{H}_{5,q}$ are indistinguishable via the following sequence of sub-hybrids, namely $\{\mathsf{H}_{5,q,1}, \mathsf{H}_{5,q,2}, \mathsf{H}_{5,q,3}\}$.

Hybrid $\mathsf{H}_{5,q,1}$ ($q \in [Q_{\mathsf{pre}}]$): It is analogous to $\mathsf{H}_{5,(q-1)}$ except that in the qth secret-key query the vectors $\boldsymbol{v}_{q,1,t}$ and $\boldsymbol{v}_{q,m_q+1,t}$ are modified as follow. The element $\widetilde{\alpha}_q\boldsymbol{y}_q[\kappa]\nu_{q,t}$ is shifted from $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_{\kappa,1}]$ to $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_\kappa]$ and the elements $\widetilde{\boldsymbol{r}}_{q,t}[m_q], \widetilde{\alpha}_q$ are shifted from $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}_1], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}_1]$ to $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}]$ respectively.

$$\begin{array}{rlllllllll}
\boldsymbol{v}_{q',1,t} = (& \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], & \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], & 0, & \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, & 0, & \widetilde{\alpha}_{q'}\boldsymbol{e}_{q'}[\eta]\nu_{q',t}, & 0, 0\ ) & \text{for } 1 \leq q' < q, \\
\boldsymbol{v}_{q,1,t} = (& \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}], & \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i], & \boxed{0}, & 0, & \boxed{\widetilde{\alpha}_q\boldsymbol{y}_q[\kappa]\nu_{q,t}}, & 0, & 0, 0\ ), & \\
\boldsymbol{v}_{q',1,t} = (& \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], & \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], & \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, & 0, & 0, & 0, & 0, 0\ ) & \text{for } q < q' \leq Q_{\mathsf{pre}}, \\
\boldsymbol{v}_{q',m_{q'}+1,t} = (& 0, & 0, & \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], & \widetilde{\alpha}_{q'}, & 0, & 0, & 0\ ) & \text{for } 1 \leq q' < q, \\
\boldsymbol{v}_{q,m_q+1,t} = (& \boxed{0}, & \boxed{0}, & 0, & 0, & \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, & \boxed{\widetilde{\alpha}_q}, & 0\ ), & \\
\boldsymbol{v}_{q',m_{q'}+1,t} = (& \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], & \widetilde{\alpha}_{q'}, & 0, & 0, & 0, & 0, & 0\ ) & \text{for } q < q' \leq Q_{\mathsf{pre}}
\end{array}$$

We observe that the inner products $\boldsymbol{v}_{q,1,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ are unchanged due to the modification occurred in $\boldsymbol{v}_{q,1,t}$ and $\boldsymbol{v}_{q,m_q+1,t}$. Therefore, the function hiding security of IPFE ensures that the hybrids $\mathsf{H}_{5,(q-1)}$ and $\mathsf{H}_{5,q,1}$ are indistinguishable.

In this hybrid, the components of $\boldsymbol{v}_{q,j,t}$ corresponding to the slots $\{\mathsf{const}, \mathsf{coef}_i, \mathsf{extnd}_\kappa, \mathsf{query}_q, \mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}$ and the components of $\boldsymbol{v}_{q,m_q+1,t}$ corresponding to the slots $\{\widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}$ are exactly the same as in the secret-key of our extended 1-FE scheme. Similarly, in case of the challenge ciphertext, the components of $\boldsymbol{u}$ at the positions $\{\mathsf{const}, \mathsf{coef}_i, \mathsf{extnd}_\kappa, \mathsf{query}_q, \mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}$ and the components of $\boldsymbol{h}_t$ at the positions $\{\widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}$ are also identical to the ciphertext of our extended 1-FE scheme.

Hybrid $\mathsf{H}_{5,q,2}$ ($q \in [Q_{\mathsf{pre}}]$): It is exactly the same as $\mathsf{H}_{5,q,1}$ except that the components $\boldsymbol{u}[\mathsf{extnd}_\kappa], \boldsymbol{u}[\mathsf{query}_q]$ and $\boldsymbol{h}_t[\widehat{\mathsf{coef}}]$ are changed from $\boldsymbol{z}^*[t], 0, \boldsymbol{w}^*[\kappa]$ to $\boldsymbol{d}_1[t], \sigma_q, \boldsymbol{d}_2[\kappa]$ respectively. Thus, the secret key vectors and the vectors $\boldsymbol{u}, \boldsymbol{h}_t$ become

$$\begin{array}{rllllllll}
\boldsymbol{v}_{q',1,t} = (& \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], & 0, & \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, & 0, & \widetilde{\alpha}_{q'}\boldsymbol{e}_{q'}[\eta]\nu_{q',t}, & 0, 0\ ) & \text{for } 1 \le q' < q, \\
\boldsymbol{v}_{q,1,t} = (& \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i], & 0, & 0, & \widetilde{\alpha}_{q}\boldsymbol{y}_{q}[\kappa]\nu_{q,t}, & \boxed{\widetilde{\alpha}_{q}\boldsymbol{e}_{q}[\eta]\nu_{q,t}}, & 0, 0\ ), & \\
\boldsymbol{v}_{q',1,t} = (& \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], & \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, & 0, & 0, & 0, & 0, 0\ ) & \text{for } q < q' \le Q_{\mathsf{pre}},
\end{array}$$

$$\begin{array}{rl}
\boldsymbol{u} = & (\ 0,\ 0,\ \ 0,\ \ \ 1,\ \ \boldsymbol{x}^*[i],\ \boldsymbol{w}^*[\kappa],\ \boldsymbol{d}_2[\kappa],\ \boxed{\boldsymbol{d}_2[\kappa]},\ \boxed{\boldsymbol{d}_3^{\le q}[\eta]}\ 0,\ 0\ ), \\
\boldsymbol{h}_t = & (\ 0,\ 0,\ -1,\ \boldsymbol{z}^*[t],\ \ -1,\ \ \boldsymbol{d}_1[t],\ \ -1,\ \ \boxed{\boldsymbol{d}_1[t]},\ \ \ 0\ )
\end{array}$$

where $\boldsymbol{d}_3^{\le q}[\eta] = \sigma_q$ if $\eta \le q$; 0 otherwise. The indistinguishability follows from the security of 1-extFE scheme. We note that the security of our 1-extFE scheme relies on the function hiding security of IPFE and the security of AKGS. In particular, we use the security of IPFE and AKGS to reversely sample the first label and make all the other labels random as shown below

$$\begin{array}{ll}
\widetilde{\ell}_{q,1,1} \leftarrow \mathsf{RevSamp}(f_{q,1}, \boldsymbol{x}^*, \widetilde{\alpha}_q f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^* + \widetilde{\beta}_{q,1}, \ell_{q,2,1}, \ldots, \ell_{q,m_q,1}) & \\
\widetilde{\ell}_{q,1,\tau} \leftarrow \mathsf{RevSamp}(f_{q,\tau}, \boldsymbol{x}^*, \widetilde{\beta}_{q,\tau}, \ell_{q,2,\tau}, \ldots, \ell_{q,m_q,\tau}) \quad \text{for } 1 < \tau < n',
\end{array}$$

where $\sum_{\tau\in[n']} \widetilde{\beta}_{q,\tau} = 0$ and $\ell_{q,j,\tau}$ is picked randomly for all $j \in [2, m_q]$. Then, the dummy vector $(\boldsymbol{d}_1 || \boldsymbol{d}_2)$ replaces the challenge message $(\boldsymbol{z}^* || \boldsymbol{w}^*)$ and $\boldsymbol{d}_3[q] = \sigma_q$ is added to the term $\widetilde{\alpha}_q f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2$ while computing $\widetilde{\ell}_{q,1,1}$. Finally, we move in the reverse direction so that the vectors $\boldsymbol{v}_{q,j,t}$ for all $j \in [m_q]$ and $t \in [n']$ are back in form as they were in $\mathsf{H}_{5,q,1}$ and $\boldsymbol{d}_1[t], \boldsymbol{d}_2[\kappa]$ are placed at $\boldsymbol{h}_t[\widehat{\mathsf{coef}}], \boldsymbol{u}[\mathsf{extnd}_\kappa]$ respectively. Note that, the hybrids involved in our 1-extFE scheme uses the positions $\mathsf{sim}_\tau, \mathsf{sim}_\tau^*, \widehat{\mathsf{sim}}, \widehat{\mathsf{sim}}^*$ of the vectors $\boldsymbol{v}_{q,j,t}, \boldsymbol{u}$ and $\boldsymbol{h}_t$, which does not affect the decryption using any post-challenge secret-key.

Hybrid $\mathsf{H}_{5,q,3}$ ($q \in [Q_{\mathsf{pre}}]$): It proceeds analogous to $\mathsf{H}_{5,q,2}$ except that we change $\boldsymbol{v}_{q,m_q+1,t}$ and $\boldsymbol{h}_t$ as below. The element $\widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}$ is shifted from $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_\kappa]$ to $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_{\kappa,2}]$ and the elements $\widetilde{\boldsymbol{r}}_{q,t}[m_q], \widetilde{\alpha}_q$ are shifted from $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}]$ to $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}_2], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}_2]$ respectively.

$$\begin{array}{rllllllll}
\boldsymbol{v}_{q',1,t} = (& \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], & 0, & \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, & 0, & \widetilde{\alpha}_{q'}\boldsymbol{e}_{q'}[\eta]\nu_{q',t}, & 0, 0\ ) & \text{for } 1 \le q' < q, \\
\boldsymbol{v}_{q,1,t} = (& \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i], & 0, & \boxed{\widetilde{\alpha}_{q}\boldsymbol{y}_{q}[\kappa]\nu_{q,t}}, & \boxed{0}, & \widetilde{\alpha}_{q}\boldsymbol{e}_{q}[\eta]\nu_{q,t}, & 0, 0\ ), & \\
\boldsymbol{v}_{q',1,t} = (& \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, & 0, & 0, & 0, & 0, 0\ ) & & \text{for } q < q' \le Q_{\mathsf{pre}}, \\
\boldsymbol{v}_{q',m_{q'}+1,t} = (& 0, \quad 0, \quad \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], & \widetilde{\alpha}_{q'}, & 0, & 0, & 0\ ) & & \text{for } 1 \le q' < q, \\
\boldsymbol{v}_{q,m_q+1,t} = (& 0, \quad 0, \quad \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, & \boxed{\widetilde{\alpha}_{q}}, & \boxed{0}, & \boxed{0}, & 0\ ), & & \\
\boldsymbol{v}_{q',m_{q'}+1,t} = (& \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \quad \widetilde{\alpha}_{q'}, \quad 0, & 0, & 0, & 0, & 0\ ) & & \text{for } q < q' \le Q_{\mathsf{pre}},
\end{array}$$

$$\begin{array}{rl}
\boldsymbol{u} = & (\ 0,\ 0,\ \ 0,\ \ \ 1,\ \ \boldsymbol{x}^*[i],\ \boldsymbol{w}^*[\kappa],\ \boldsymbol{d}_2[\kappa],\ \boxed{\boldsymbol{w}^*[\kappa]},\ \boldsymbol{d}_3^{\le q}[\eta],\ 0,\ 0\ ), \\
\boldsymbol{h}_t = & (\ 0,\ 0,\ -1,\ \boldsymbol{z}^*[t],\ \ -1,\ \ \boldsymbol{d}_1[t],\ \ -1,\ \ \boxed{\boldsymbol{z}^*[t]},\ \ \ 0\ )
\end{array}$$

Note that the inner products $\boldsymbol{v}_{q,1,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ remains the same as in $\mathsf{H}_{5,q,2}$. Therefore, the hybrids $\mathsf{H}_{5,q,2}$ and $\mathsf{H}_{5,q,3}$ are indistinguishable due to the function hiding security of IPFE. We observe that $\mathsf{H}_{5,q,3}$ is identical to $\mathsf{H}_{5,q}$ for all $q \in [Q_{\mathsf{pre}}]$.

Hybrid H_6: It is exactly the same as $\mathsf{H}_{5,Q_{\mathsf{pre}}}$ except that the elements $\boldsymbol{u}[\mathsf{extnd}_\kappa]$, $\boldsymbol{h}_t[\widehat{\mathsf{const}}]$ and $\boldsymbol{h}_t[\widehat{\mathsf{coef}}]$ are set to zero. We describe the vectors associated to secret-key queries and the challenge ciphertext below. Note that the post-challenge secret-key queries are released in the same way as in H_4 (or in $\mathsf{H}_{5,Q_{\mathsf{pre}}}$).

$$1 \le q \le Q_{\mathsf{pre}} \begin{cases} \boldsymbol{v}_q = (\ \widetilde{\alpha}_q,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\ ), \\ \boldsymbol{v}_{q,1,t} = (\ \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}],\ \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i],\ 0,\ \widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t},\ 0,\ \widetilde{\alpha}_q \boldsymbol{e}_q[\eta]\nu_{q,t},\ 0,\ 0\ ), \\ \boldsymbol{v}_{q,j,t} = (\ \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}],\ \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i],\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\ ) & \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t} = (\ 0,\ 0,\ \widetilde{\boldsymbol{r}}_{q,t}[m_q],\ \widetilde{\alpha}_q,\ 0,\ 0,\ 0\ ), \end{cases}$$

$$\boldsymbol{u} = (\ 0, 0,\ 0,\ 1,\ \boldsymbol{x}^*[i], \boldsymbol{w}^*[\kappa], \boldsymbol{d}_2[\kappa], \boxed{0}, \boldsymbol{d}_3[\eta], 0, 0\ ),$$
$$\boldsymbol{h}_t = (\ 0, 0, -1, \boldsymbol{z}^*[t],\ -1,\ \boldsymbol{d}_1[t],\ \boxed{0},\ \boxed{0},\ 0\ )$$

$$Q_{\mathsf{pre}} < q \le Q \begin{cases} \boldsymbol{v}_q = (\ \widetilde{\alpha}_q,\ 0,\ 0,\ 0, 0, 0,\ 0,\ 0\ ), \\ \boldsymbol{v}_{q,1,t} = (\ \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}],\ \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i],\ \widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t},\ 0, 0, 0,\ 0,\ 0\ ), \\ \boldsymbol{v}_{q,j,t} = (\ \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}],\ \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i],\ 0,\ 0, 0, 0,\ 0,\ 0\ ) & \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t} = (\ \widetilde{\boldsymbol{r}}_{q,t}[m_q],\ \widetilde{\alpha}_q,\ 0,\ 0, 0, 0, 0\ ) \end{cases}$$

Since the inner products $\boldsymbol{v}_{q,1,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ is unaltered due to the modification in this hybrid, the function hiding security of IPFE ensures indistinguishability between the hybrids $\mathsf{H}_{5,Q_{\mathsf{pre}}}$ and H_6.

The second part of the proof is completed as all the pre-challenge secret-keys are now able to decrypt the challenge ciphertext without the components of $\boldsymbol{u}, \boldsymbol{h}_t$ that make use of $\boldsymbol{z}^*$ and $\boldsymbol{w}^*$. Note that, $\boldsymbol{u}[\mathsf{extnd}_{\kappa,1}] = \boldsymbol{w}^*[\kappa]$ and $\boldsymbol{h}_t[\widehat{\mathsf{coef}_1}] = \boldsymbol{z}^*[t]$ are only needed for the successful decryption of the challenge ciphertext by post-challenge secret-keys. From the next hybrid we change the computation of post-challenge secret-keys so that the challenge ciphertext can be simulated without using $(\boldsymbol{z}^* || \boldsymbol{w}^*)$.

Hybrid H_7: This hybrid proceeds exactly similar to H_6 except that we use the honest levels $\widetilde{\ell}_{q,1,t} = \widetilde{\boldsymbol{\ell}}_{q,1,t}(\boldsymbol{x}^*)$, $\widetilde{\ell}_{q,j,t} = \widetilde{\boldsymbol{\ell}}_{q,j,t}(\boldsymbol{x}^*)$ for $j \in [2, m_q]$ and $\widetilde{\ell}_{q,m_q+1,t} = -\widetilde{\boldsymbol{r}}_{q,t}[m_q] + \widetilde{\alpha}_q \boldsymbol{z}^*[t]$ while defining the vectors $\boldsymbol{v}_{q,j,t}$ in all the *post-challenge* secret-key queries. Moreover, all the other private components $\boldsymbol{v}_{q,j,t}[\mathsf{coef}_i]$ and $\boldsymbol{v}_{q,j,t}[\mathsf{extnd}_{\kappa,1}]$ are zero for all $j \in [m_q]$. We also modify $\boldsymbol{u}$ and $\boldsymbol{h}_t$ of the challenge ciphertext as shown below.

$$\boldsymbol{u} = (\ 0, 0,\ 0,\ 1,\ \boldsymbol{x}^*[i],\ \boxed{0},\ \boldsymbol{d}_2[\kappa], 0, \boldsymbol{d}_3[\eta], 0, 0\ ),$$
$$\boldsymbol{h}_t = (\ 0, 0, \boxed{1}, \boxed{0},\ -1,\ \boldsymbol{d}_1[t],\ 0,\ 0,\ 0\ ),$$

$$Q_{\mathsf{pre}} < q \le Q \begin{cases} \boldsymbol{v}_q = (\ \widetilde{\alpha}_q,\ 0,\ 0,\ 0, 0, 0, 0, 0\ ), \\ \boldsymbol{v}_{q,1,t} = (\ \boxed{\widetilde{\ell}_{q,1,t} + \widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}}, \boxed{0}, \boxed{0}, 0, 0, 0, 0, 0\ ), \\ \boldsymbol{v}_{q,j,t} = (\ \boxed{\widetilde{\ell}_{q,j,t}},\ \boxed{0},\ 0,\ 0, 0, 0, 0, 0\ ) & \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t} = (\ \boxed{\widetilde{\ell}_{q,m_q+1,t}},\ \boxed{0},\ 0,\ 0, 0, 0,\ 0\ ) \end{cases}$$

Since the inner products $\boldsymbol{v}_{q,j,t} \cdot \boldsymbol{u}, \boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ for all $q \in [Q_{\mathsf{pre}} + 1, Q]$ are the same as in the previous hybrid, the function hiding property of IPFE ensures that the hybrids H_6 and H_7 are indistinguishable.

Hybrid H_8: : This hybrid proceeds analogous to H_7 except that the post-challenge secret-key queries use the simulated garblings instead of the honest garblings. More specifically, we sample $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t}, \widetilde{\nu}_{q,t} \leftarrow \mathbb{Z}_p$ satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0, \sum_{t\in[n']} \widetilde{\nu}_{q,t} = 1$ and compute the simulated garblings

$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \widetilde{\alpha}_q \cdot (\boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\nu}_{q,t} \cdot \boldsymbol{y}_q^\top \boldsymbol{w}^*) + \widetilde{\beta}_{q,t})$$

for all $q \in [Q_{\mathsf{pre}} + 1, Q]$ and $t \in [n']$. Then, the post-challenge secret-keys are generated using the vectors described below.

$$Q_{\mathsf{pre}} < q \le Q \begin{cases} \boldsymbol{v}_q|_{S_{\mathsf{priv}}} = (\ \widetilde{\alpha}_q,\ 0, 0, 0, 0, 0,\ 0,\ 0\ ), \\ \boldsymbol{v}_{q,1,t}|_{S_{\mathsf{priv}}} = (\ \boxed{\widehat{\ell}_{q,1,t}},\ 0, 0, 0, 0, 0,\ 0,\ 0\ ) \\ \boldsymbol{v}_{q,j,t}|_{S_{\mathsf{priv}}} = (\ \boxed{\widehat{\ell}_{q,j,t}},\ 0, 0, 0, 0, 0,\ 0,\ 0\ ) & \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t}|_{\widehat{S}_{\mathsf{priv}}} = (\ \boxed{\widehat{\ell}_{q,m_q+1,t}}, 0, 0, 0, 0, 0, 0\ ) \end{cases}$$

The simulated levels of AKGS is used in place of actual garblings. The simulation security of AKGS implies that the hybrids H_7 and H_8 are indistinguishable.

Hybrid H_9: : This proceeds exactly the same as H_8 except that the distribution of $\{\widetilde{\beta}_{q,t}\}_{t\in[n']}$ is changed. We replace $\widetilde{\beta}_{q,t}$ by $\widetilde{\beta}'_{q,t} = \widetilde{\beta}_{q,t} - \widetilde{\alpha}_q \cdot (\boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\nu}_{q,t} \cdot \boldsymbol{y}_q^\top \boldsymbol{w}^*)$ for all $1 < t \leq n'$ and replace the element $\widetilde{\beta}_{q,1}$ by $\widetilde{\beta}'_{q,1} = \widetilde{\beta}_{q,1} - \widetilde{\alpha}_q \cdot (\boldsymbol{z}^*[1] f_{q,1}(\boldsymbol{x}^*) + \widetilde{\nu}_{q,1} \cdot \boldsymbol{y}_q^\top \boldsymbol{w}^*) + \widetilde{\alpha}_q \cdot (f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*)$. Note that, the distributions

$$\{\widetilde{\beta}_{t,q} \leftarrow \mathbb{Z}_p : \sum_{t\in[n']} \widetilde{\beta}_{t,q} = 0\} \text{ and } \{\widetilde{\beta}'_{t,q} : \sum_{t\in[n']} \widetilde{\beta}_{t,q} = 0\}$$

are statistically close since $\{\widetilde{\beta}'_{q,t}\}_{t\in[n']}$ are also uniform over $\mathbb{Z}_p$ and $\sum_{t\in[n']} \widetilde{\beta}'_{q,t} = 0$. Finally, the vectors associated to the post-challenge secret-keys are given by

$$Q_{\mathsf{pre}} < q \leq Q \begin{cases} \boldsymbol{v}_q|_{S_{\mathsf{priv}}} = (\quad \widetilde{\alpha}_q, \quad 0, 0, 0, 0, 0, 0, 0 \,), \\ \boldsymbol{v}_{q,1,t}|_{S_{\mathsf{priv}}} = (\boxed{\widehat{\ell}_{q,1,t}}, \quad 0, 0, 0, 0, 0, 0, 0 \,) \\ \boldsymbol{v}_{q,j,t}|_{S_{\mathsf{priv}}} = (\boxed{\widehat{\ell}_{q,j,t}}, \quad 0, 0, 0, 0, 0, 0, 0 \,) \quad \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t}|_{\widehat{S}_{\mathsf{priv}}} = (\boxed{\widehat{\ell}_{q,m_q+1,t}}, 0, 0, 0, 0, 0, 0 \,) \end{cases}$$

where the simulated garblings take the form

$$(\widehat{\ell}_{q,1,1}, \ldots, \widehat{\ell}_{q,m_q,1}, \widehat{\ell}_{q,m_q+1,1}) \leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, \boxed{\widetilde{\alpha}_q \cdot (f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) + \widetilde{\beta}_{q,1}})$$

$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \boxed{\widetilde{\beta}_{q,t}}) \quad \text{for } 1 < t \leq n'.$$

Observe that H_9 is the same as the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,extFE}}(1^\lambda)$. This completes the security proof.

Note: Recall that the simulation goes through even if the challenger gets $[\![\boldsymbol{y}_q]\!]_2$ (and hence $[\![\widetilde{\alpha}_q f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*]\!]_2$) as we have already mentioned it while describing $\mathsf{KeyGen}_1^*(\cdot)$.

8 Unbounded-Slot FE for Attribute-Weighted Sum

In this section, we describe the transformation from extended one-slot FE to unbounded-slot FE. The conversion is proposed in [AGW20] with semi-adaptive simulation security relying on MDDH_k assumption. We show the same transformation works to achieve adaptive simulation security against an a priori bounded number of pre-ciphertext secret key queries while an arbitrary polynomial number of post-ciphertext secret key queries under the bMDDH_k assumption.

Let $\mathit{\Pi}_{\mathsf{extOne}} = (\mathsf{Setup}_{\mathsf{extFE}}, \mathsf{KeyGen}_{\mathsf{extFE}}, \mathsf{Enc}_{\mathsf{extFE}}, \mathsf{Dec}_{\mathsf{extFE}})$ be the extended one-slot FE scheme described in Section 7. The unbounded-slot FE scheme $\mathit{\Pi}_{\mathsf{ubd}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ works as follows:

$\mathsf{Setup}(1^\lambda, 1^n, 1^{n'}, 1^B)$: On input integers λ, n, n' as unary, the setup algorithm runs

$$(\mathsf{MSK}_1, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B),$$
$$(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

and outputs the master secret-key $\mathsf{MSK} = (\mathsf{MSK}_1, \mathsf{MSK}_2)$ and the master public-key $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$.

$\mathsf{KeyGen}(\mathsf{MSK}, f)$: The key generation algorithm takes input $\mathsf{MSK} = (\mathsf{MSK}_1, \mathsf{MSK}_2)$ and a function $f \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$. It samples $\boldsymbol{y} \leftarrow \mathbb{Z}_p^k$ and computes

$$\mathsf{SK}_{f,1} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_1, (f, [\![\boldsymbol{y}]\!]_2)), \quad \mathsf{SK}_{f,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f, [\![\boldsymbol{y}]\!]_2))$$

Then, It returns the secret-key as $\mathsf{SK}_f = (\mathsf{SK}_{f,1}, \mathsf{SK}_{f,2})$ and f. Here, we use the property of extFE that $\mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_j, (f, \boldsymbol{y})) = \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_j, (f, [\![\boldsymbol{y}]\!]_2))$ for $j \in [2]$.

$\mathsf{Enc}(\mathsf{MPK}, (\boldsymbol{x}_i, \boldsymbol{z}_i)_{i\in[N]})$: The encryption algorithm takes input MPK and message $(\boldsymbol{x}_i, \boldsymbol{z}_i) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$ for $i \in [N]$. It samples random vectors $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and computes

$$\begin{aligned}\mathsf{CT}_1 &\leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_1, (\boldsymbol{x}_1, \boldsymbol{z}_1 || - \sum_{i\in[2,N]} \boldsymbol{w}_i))\\ \mathsf{CT}_i &\leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i, \boldsymbol{z}_i || \boldsymbol{w}_i)), \text{ for } i \in [2, N]\end{aligned}$$

It returns the ciphertext $\mathsf{CT}_{(\boldsymbol{x}_i||\boldsymbol{z}_i)} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N)$.

$\mathsf{Dec}((\mathsf{SK}_f, f), (\mathsf{CT}_{(\boldsymbol{x}_i||\boldsymbol{z}_i)}, (\boldsymbol{x}_i)_{i\in[N]}))$: The decryption algorithm parses the secret-key $\mathsf{SK}_f = (\mathsf{SK}_{f,1}, \mathsf{SK}_{f,2})$ and the ciphertext $\mathsf{CT}_{(\boldsymbol{x}_i||\boldsymbol{z}_i)} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N)$. Then it computes

$$\begin{aligned}&[\![D_1]\!]_T \leftarrow \mathsf{Dec}_{\mathsf{extFE}}((\mathsf{SK}_{f,1}, f), (\mathsf{CT}_1, \boldsymbol{x}_1))\\ &[\![D_i]\!]_T \leftarrow \mathsf{Dec}_{\mathsf{extFE}}((\mathsf{SK}_{f,2}, f), (\mathsf{CT}_i, \boldsymbol{x}_i)) \quad \text{for } i \in [2, N]\end{aligned}$$

and multiply those values to get $[\![D]\!]_T = [\![D_1]\!]_T \cdots [\![D_N]\!]_T$. Finally, it returns D by solving discrete log via brute-force.
Correctness. By the correctness of underlying extFE scheme, we get

$$\begin{aligned}[\![D_1]\!]_T &= [\![f(\boldsymbol{x}_1)^\top \boldsymbol{z}_1 - \sum_{i\in[2,N]} \boldsymbol{y}^\top \boldsymbol{w}_i]\!]_T\\ [\![D_i]\!]_T &= [\![f(\boldsymbol{x}_i)^\top \boldsymbol{z}_i + \boldsymbol{y}^\top \boldsymbol{w}_i]\!]_T \quad \text{for } i \in [2, N]\end{aligned}$$

Therefore, multiplying all $[\![D_i]\!]_T$ for $i \in [N]$, we have $[\![D]\!]_T = [\![\sum_{i\in[N]} f(\boldsymbol{x}_i)^\top \boldsymbol{z}_i]\!]_T$.

8.1 Security Analysis

Theorem 6 *The unbounded-slot* FE *scheme Π_{ubd} for attribute weighted sum is adaptively simulation-secure under bilateral MDDH_k assumption if the underlying extended one-slot* FE *scheme Π_{extOne} is adaptively simulation secure.*

The Simulator In this section, we describe the simulator of our unbounded slot FE scheme Π_{ubd}. First, we recall the syntax of the simulator of our extended one-slot FE scheme presented in Section 7.

Simulator of Π_{extOne}. Let Q be the total number of secret-key queries by the adversary and $B = Q_{\mathsf{pre}}$ be the number of secret-keys asked before the challenge phase. We consider $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*)$ as the challenge message.

- $\mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B) \to (\mathsf{MSK}^*_1, \mathsf{MPK}_1)$
- $\mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2)) \to \mathsf{SK}_{f_q,\boldsymbol{y}}$
- $\mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}^*_1, \boldsymbol{x}^*, \mathcal{V}_1) \to \mathsf{CT}^*$ where $\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$
- $\mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![f_q(\boldsymbol{x}^*)^\top + \boldsymbol{y}_q^\top \boldsymbol{w}^*]\!]_2) \to \mathsf{SK}_{f_q,\boldsymbol{y}}$

Remark 5 Note that, the simulator is given $\boldsymbol{y}_q$ and $f_q(\boldsymbol{x}^*)^\top + \boldsymbol{y}_q^\top \boldsymbol{w}^*$ in the power of the source groups. The simulator still runs efficiently as we are utilizing the following facts from our Π_{extOne}:

1. $\mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2)) = \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_1, (f_q, \boldsymbol{y}_q))$ *in case of our* Π_{extOne} *for all* $q \in [Q_{\mathsf{pre}}]$
2. $\mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}^*_1, \boldsymbol{x}^*, \mathcal{V}_1) = \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}^*_1, \boldsymbol{x}^*, \mathcal{V}'_1)$ *where* $\mathcal{V}'_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) : q \in [Q_{\mathsf{pre}}]\}$
3. $\mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![f_q(\boldsymbol{x}^*)^\top + \boldsymbol{y}_q^\top \boldsymbol{w}^*]\!]_2) = \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}^*, (f_q, \boldsymbol{y}_q), f_q(\boldsymbol{x}^*)^\top + \boldsymbol{y}_q^\top \boldsymbol{w}^*)$ *for all* $q \in [Q_{\mathsf{pre}} + 1, Q]$

Now, we present the simulator of Π_{ubd} as follows:

$\mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'}, 1^B, 1^N)$: On input integers λ, n, n', N and a bound on the pre-challenge query B as unary, the simulated setup algorithm samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and generates the keys

$$(\mathsf{MSK}^*_1, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B),$$
$$(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

It returns $\mathsf{MSK}^* = (\mathsf{MSK}^*_1, \mathsf{MSK}_2, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N)$ and $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$.

$\mathsf{KeyGen}^*_0(\mathsf{MSK}^*, f_q)$: This is the pre-challenge key generation algorithm. On input MSK^* and a function $f_q \in \mathcal{F}^{(n,n')}_{\mathsf{ABP}}$, the algorithm samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and computes

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_1, (f, [\![\boldsymbol{y}_q]\!]_2)),$$
$$\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f, [\![\boldsymbol{y}_q]\!]_2))$$

It outputs the simulated key $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \mathsf{SK}_{f_q,2})$.

Let $B = Q_{\mathsf{pre}}$ be the total number of pre-challenge keys queried by the adversary and $(\boldsymbol{x}^*_i, \boldsymbol{z}^*_i)_{i \in [N]}$ be the challenge message.

$\mathsf{Enc}^*(\mathsf{MPK}, \mathsf{MSK}^*, (\boldsymbol{x}_i)_{i \in [N]}, \mathcal{V})$: On input $\mathsf{MPK}, \mathsf{MSK}^*$, a set of vectors $(\boldsymbol{x}^*_i)_{i \in [N]}$ and a set $\mathcal{V} = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), \mu_q = \sum_{i \in [N]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i) : q \in [Q_{\mathsf{pre}}]\}$, the simulated encryption algorithm defines the set $\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\mu_q - \sum_{i \in [2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$ and computes

$$\mathsf{CT}^*_1 \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}^*_1, \boldsymbol{x}^*_1, \mathcal{V}_1)$$
$$\mathsf{CT}^*_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \boldsymbol{0} || \boldsymbol{w}_i)) \quad \text{for } i \in [2, N]$$

It returns the simulated ciphertext $\mathsf{CT}^* = (\mathsf{CT}^*_1, \mathsf{CT}_2, \ldots, \mathsf{CT}_N)$.

$\mathbf{KeyGen}_1^*(\mathbf{MSK}^*, (\boldsymbol{x}_i^*)_{i\in[2,N]}, \boldsymbol{f_q}, \boldsymbol{\mu_q})$: This is the post-challenge key generation algorithm. On input MSK^*, a set of vectors $(\boldsymbol{x}_i^*)_{i\in[2,N]}$, a function $f_q \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ and an integer $\mu_q = \sum_{i\in[N]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^*$, the algorithm samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and computes

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\mu_q - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2)$$

$$\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

It outputs the simulated secret-key $\mathsf{SK}^*_{f_q} = (\mathsf{SK}^*_{f_q,1}, \mathsf{SK}_{f_q,2})$

Hybrids and Reductions

Proof. We prove the theorem by showing the indistinguishability between the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,ubdFE}}(1^\lambda)$ and the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,ubdFE}}(1^\lambda)$ via a sequence of hybrid games. In each experiment, the adversary $\mathcal{A}$ can query a polynomial number of secret-key queries corresponding to functions $f \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, both before and after submitting the challenge message $(\boldsymbol{x}_i, \boldsymbol{z}_i)_{i\in[N]} \in (\mathbb{Z}_p^n \times \mathbb{Z}_p^{n'})^N$. Let Q be the total number of key queries and without loss of generality let $B = Q_{\mathsf{pre}}$ be the number of keys queried before the challenge phase. We denote the q-th secret-key by SK_{f_q} for a function f_q. The sequence of hybrids are described as follows:

Hybrid $\mathbf{H_0}$: : This is the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,ubdFE}}(1^\lambda)$.

– The master keys are sampled as follows:

$$(\mathsf{MSK}_1, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B),$$
$$(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

The challenger sets $\mathsf{MSK} = (\mathsf{MSK}_1, \mathsf{MSK}_2)$ and $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$.

– The q-th secret-key SK_{f_q}, for all $q \in [Q]$, is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generate the keys

$$\mathsf{SK}_{f_q,1} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_1, (f_q, [\![\boldsymbol{y}_q]\!]_2)),$$
$$\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

The challenger sends $\mathsf{SK}_{f_q} = (\mathsf{SK}_{f_q,1}, \mathsf{SK}_{f_q,2})$.

– The challenge ciphertext is computed as follows: The challenger samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and compute the ciphertexts

$$\mathsf{CT}_1 \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_1, (\boldsymbol{x}_1^*, \boldsymbol{z}_1^* || - \sum_{i\in[2,N]} \boldsymbol{w}_i))$$

$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{z}_i^* || \boldsymbol{w}_i)), \text{ for } i \in [2, N]$$

The challenger returns $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N)$.

Hybrid H_1: : This is exactly the same H_0 except that all the algorithms of the first instant of Π_{extOne} is now replaced with their simulated counterpart. The changes are indicated as follows:

– The master keys as sampled as follows:

$$\boxed{(\mathsf{MSK}_1^*, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)},$$
$$(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

The challenger samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and sets the master keys as $\boxed{\mathsf{MSK} = (\mathsf{MSK}_1^*, \mathsf{MSK}_2, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N)}$ and $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$.

– The q-th secret-key SK_{f_q}, for all $q \in [Q_{\mathsf{pre}}]$, is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generate the keys

$$\boxed{\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2))},$$
$$\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

The challenger sends $\mathsf{SK}_{f_q} = (\boxed{\mathsf{SK}^*_{f_q,1}}, \mathsf{SK}_{f_q,2})$.

– The challenge ciphertext is computed as follows: After all the pre-challenge secret-key queries, the challenger defines a set

$$\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![f_q(\boldsymbol{x}_1^*)^\top \boldsymbol{z}_1^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$

and computes the ciphertexts

$$\boxed{\mathsf{CT}_1^* \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}_1^*, \mathcal{V}_1)}$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{z}_i^* || \boldsymbol{w}_i)),\ \text{for } i \in [2, N]$$

The challenger returns $\mathsf{CT} = (\boxed{\mathsf{CT}_1^*}, \mathsf{CT}_2, \ldots, \mathsf{CT}_N)$.

– The post-challenge secret-key SK_{f_q} for $q \in [Q_{\mathsf{pre}}+1, Q]$ is computed as follows: The challenger $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generates the keys

$$\boxed{\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}_1^*, \boldsymbol{x}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![f_q(\boldsymbol{x}_1^*)^\top \boldsymbol{z}_1^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2)},$$
$$\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

and returns $\mathsf{SK}_{f_q} = (\boxed{\mathsf{SK}^*_{f_q,1}}, \mathsf{SK}_{f_q,2})$

In lemma 8, we show that the hybrids H_0 and H_1 are indistinguishable by the adaptive simulation-security of Π_{extOne} scheme.

Hybrid $\mathsf{H}_{2,\eta}$ ($\eta \in [2, N]$): : It is exactly the same as hybrid H_1 except that the changes indicated below.

– The master keys as sampled as follows:

$$(\mathsf{MSK}_1^*, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B),$$
$$(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

The challenger samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and sets $\mathsf{MSK} = (\mathsf{MSK}_1^*, \mathsf{MSK}_2, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N)$ and $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$.

– The q-th secret-key SK_{f_q}, for all $q \in [Q_{\mathsf{pre}}]$, is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generate the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2)),$$
$$\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

The challenger sends $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \mathsf{SK}_{f_q,2})$.

– The challenge ciphertext is computed as follows: After all the pre-challenge secret-key queries, the challenger defines a set

$$\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\sum_{i\in[\eta-1]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$

and computes the ciphertexts

$$\mathsf{CT}_1^* \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}_1^*, \mathcal{V}_1)$$
$$\boxed{\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{0}||\boldsymbol{w}_i))} \text{ for } i \in [2, \eta],$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{z}_i^*||\boldsymbol{w}_i)) \text{ for } i \in [\eta+1, N]$$

The challenger returns $\mathsf{CT} = (\mathsf{CT}_1^*, \mathsf{CT}_2, \ldots, \mathsf{CT}_{\eta-1}, \mathsf{CT}_\eta, \ldots, \mathsf{CT}_N)$.

– The post-challenge secret-key SK_{f_q} for $q \in [Q_{\mathsf{pre}}+1, Q]$ is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generates the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}_1^*, \boldsymbol{x}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\sum_{i\in[\eta-1]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2),$$
$$\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

and returns $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \mathsf{SK}_{f_q,2})$

Observe that $\mathsf{H}_{2,1}$ coincides with H_1. We will show that for all $\eta \in [2, N]$, the hybrids $\mathsf{H}_{2,(\eta-1)}$ and $\mathsf{H}_{2,\eta}$ are indistinguishable via the following sequence of sub-hybrids, namely, $\{\mathsf{H}_{2,\eta,1}, \mathsf{H}_{2,\eta,2}, \mathsf{H}_{2,\eta,3}\}_{\eta\in[2,N]}$.

Hybrid $\mathsf{H}_{2,\eta,1}$ ($\eta \in [2, N]$): : It is exactly the same as hybrid $\mathsf{H}_{2,(\eta-1)}$ except that the changes indicated below.

– The master keys as sampled as follows:

$$(\mathsf{MSK}_1^*, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B),$$
$$\boxed{(\mathsf{MSK}_2^*, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)}$$

The challenger samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and sets $\mathsf{MSK} = (\mathsf{MSK}_1^*, \boxed{\mathsf{MSK}_2^*}, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N)$ and $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$.

– The q-th secret-key SK_{f_q}, for all $q \in [Q_{\mathsf{pre}}]$, is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generate the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2)),$$
$$\boxed{\mathsf{SK}^*_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))}$$

The challenger sends $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \boxed{\mathsf{SK}^*_{f_q,2}})$.

– The challenge ciphertext is computed as follows: After all the pre-challenge secret-key queries, the challenger defines the sets

$$\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\sum_{i \in [\eta-1]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i - \sum_{i \in [2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$
$$\mathcal{V}_2 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![f_q(\boldsymbol{x}^*_\eta)^\top \boldsymbol{z}^*_\eta + \boldsymbol{y}_q^\top \boldsymbol{w}_\eta]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$

and computes the ciphertexts

$$\mathsf{CT}^*_1 \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}^*_1, \mathcal{V}_1)$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \mathbf{0}||\boldsymbol{w}_i)) \text{ for } i \in [2, \eta-1],$$
$$\boxed{\mathsf{CT}^*_\eta \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_2, \mathsf{MSK}^*_2, \mathcal{V}_2)},$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \boldsymbol{z}^*_i||\boldsymbol{w}_i)) \text{ for } i \in [\eta+1, N]$$

The challenger returns $\mathsf{CT} = (\mathsf{CT}^*_1, \mathsf{CT}_2, \ldots, \mathsf{CT}_{\eta-1}, \boxed{\mathsf{CT}^*_\eta}, \mathsf{CT}_{\eta+1}, \ldots, \mathsf{CT}_N)$.

– The post-challenge secret-key SK_{f_q} for $q \in [Q_{\mathsf{pre}}+1, Q]$ is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generates the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\sum_{i \in [\eta-1]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i - \sum_{i \in [2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2),$$
$$\boxed{\mathsf{SK}^*_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_2, \boldsymbol{x}^*_\eta, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![f_q(\boldsymbol{x}^*_\eta)^\top \boldsymbol{z}^*_\eta + \boldsymbol{y}_q^\top \boldsymbol{w}_\eta]\!]_2)}$$

and returns $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \boxed{\mathsf{SK}^*_{f_q,2}})$

We demonstrate in lemma 9 that $\mathsf{H}_{2,(\eta-1)}$ and $\mathsf{H}_{2,\eta,1}$ are indistinguishable by the adaptive simulation-security of Π_{extOne}.

Hybrid $\mathbf{H}_{2,\eta,2}$ ($\eta \in [2, N]$): : It is exactly the same as hybrid $\mathsf{H}_{2,\eta,1}$ except that the changes indicated below.

– The master keys as sampled as follows:

$$(\mathsf{MSK}^*_1, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B),$$
$$(\mathsf{MSK}^*_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

The challenger samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and sets $\mathsf{MSK} = (\mathsf{MSK}^*_1, \mathsf{MSK}^*_2, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N)$ and $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$.

– The q-th secret-key SK_{f_q}, for all $q \in [Q_{\mathsf{pre}}]$, is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generate the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2)),$$
$$\mathsf{SK}^*_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

The challenger sends $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \mathsf{SK}^*_{f_q,2})$.

– The challenge ciphertext is computed as follows: After all the pre-challenge secret-key queries, the challenger defines the sets

$$\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\boxed{\sum_{i\in[\eta]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i} - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$
$$\mathcal{V}_2 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), \boxed{[\![\boldsymbol{y}_q^\top \boldsymbol{w}_\eta]\!]_1}) : q \in [Q_{\mathsf{pre}}]\}$$

and computes the ciphertexts

$$\mathsf{CT}^*_1 \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}^*_1, \boxed{\mathcal{V}_1})$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \boldsymbol{0}||\boldsymbol{w}_i)) \text{ for } i \in [2, \eta-1],$$
$$\mathsf{CT}^*_\eta \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_2, \mathsf{MSK}^*_2, \boxed{\mathcal{V}_2}),$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \boldsymbol{z}^*_i||\boldsymbol{w}_i)) \text{ for } i \in [\eta+1, N]$$

The challenger returns $\mathsf{CT} = (\boxed{\mathsf{CT}^*_1}, \mathsf{CT}_2, \ldots, \mathsf{CT}_{\eta-1}, \boxed{\mathsf{CT}^*_\eta}, \mathsf{CT}_{\eta+1}, \ldots, \mathsf{CT}_N)$.

– The post-challenge secret-key SK_{f_q} for $q \in [Q_{\mathsf{pre}}+1, Q]$ is computed as follows: The challenger $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generates the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\boxed{\sum_{i\in[\eta]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i} - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2),$$
$$\mathsf{SK}^*_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_2, \boldsymbol{x}^*_\eta, (f_q, [\![\boldsymbol{y}_q]\!]_2), \boxed{[\![\boldsymbol{y}_q^\top \boldsymbol{w}_\eta]\!]_2})$$

and returns $\boxed{\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \mathsf{SK}^*_{f_q,2})}$

Lemma 10 ensures that the hybrids $\mathsf{H}_{2,\eta,1}$ and $\mathsf{H}_{2,\eta,2}$ are indistinguishable due to bilateral MDDH_k assumption.

Hybrid $\mathsf{H}_{2,\eta,3}$ ($\eta \in [2,\eta]$): : It is exactly the same as hybrid $\mathsf{H}_{2,\eta,2}$ except that the changes indicated below.

– The master keys as sampled as follows:

$$(\mathsf{MSK}^*_1, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B),$$
$$\boxed{(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)}$$

The challenger samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and sets $\mathsf{MSK} = (\mathsf{MSK}^*_1, \boxed{\mathsf{MSK}_2}, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N)$ and $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$.

– The q-th secret-key SK_{f_q}, for all $q \in [Q_{\mathsf{pre}}]$, is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generate the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2)),$$
$$\boxed{\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))}$$

The challenger sends $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \boxed{\mathsf{SK}_{f_q,2}})$.

– The challenge ciphertext is computed as follows: After all the pre-challenge secret-key queries, the challenger defines the sets

$$\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\sum_{i\in[\eta]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$

and computes the ciphertexts

$$\mathsf{CT}^*_1 \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}^*_1, \mathcal{V}_1)$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \boldsymbol{0}||\boldsymbol{w}_i)) \text{ for } i \in [2, \eta-1],$$
$$\boxed{\mathsf{CT}_\eta \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_\eta, \boldsymbol{0}||\boldsymbol{w}_\eta))},$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \boldsymbol{z}^*_i||\boldsymbol{w}_i)) \text{ for } i \in [\eta+1, N]$$

The challenger returns $\mathsf{CT} = (\mathsf{CT}^*_1, \mathsf{CT}_2, \dots, \mathsf{CT}_{\eta-1}, \boxed{\mathsf{CT}_\eta}, \mathsf{CT}_{\eta+1}, \dots, \mathsf{CT}_N)$.

– The post-challenge secret-key SK_{f_q} for $q \in [Q_{\mathsf{pre}}+1, Q]$ is computed as follows: The challenger samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generates the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\sum_{i\in[\eta]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2),$$

$$\boxed{\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))}$$

and returns $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \boxed{\mathsf{SK}_{f_q,2}})$

We show in lemma 11 that the hybrids $\mathsf{H}_{2,\eta,2}$ and $\mathsf{H}_{2,\eta,3}$ are indistinguishable by the adaptive simulation security of Π_{extOne}.

Now, we observe that the hybrid $\mathsf{H}_{2,1}$ is identical to H_1 and $\mathsf{H}_{2,\eta,3}$ is identical to $\mathsf{H}_{2,\eta}$ for all $\eta \in [2, N]$. Finally, we note that $\mathsf{H}_{2,N}$ is the ideal experiment $\mathsf{Expt}^{\mathsf{Ideal,ubdFE}}_{\mathcal{A}}(1^\lambda)$.

Lemma 8 *The hybrids H_0 and H_1 are computationally indistinguishable by adaptive simulation-security of Π_{extOne}. More specifically, for any* PPT *adversary $\mathcal{A}$, there exists another* PPT *adversary $\mathcal{B}_1$ such that*

$$|Adv^{H_0}_{\mathcal{A}}(\lambda) - Adv^{H_1}_{\mathcal{A}}(\lambda)| \leq Adv^{extFE}_{\mathcal{B}_1}(\lambda)$$

Proof. We establish the indistinguishability by constructing an adversary $\mathcal{B}_1$ against the adaptive simulation-security of Π_{extOne}. Let $\mathcal{C}_1$ be the challenger of the security experiment of Π_{extOne}. The adversary $\mathcal{B}_1$ works as follows:

– Setup: $\mathcal{B}_1$ gets MPK_1 from $\mathcal{C}_1$ and computes

$$(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

It returns $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$ to $\mathcal{A}$.

– Key Queries: $\mathcal{A}$ asks for a secret-key corresponding to the function f_q at the q-th key query for $q \in [Q]$. First, $\mathcal{B}_1$ samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and generates

$$\mathsf{SK}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

Next, $\mathcal{B}_1$ forwards $(f_q, \boldsymbol{y}_q)$ to $\mathcal{C}_1$ and gets a secret-key $\widetilde{\mathsf{SK}}_{f_q,1}$. Finally, $\mathcal{B}_1$ returns $\mathsf{SK}_{f_q} = (\widetilde{\mathsf{SK}}_{f_q,1}, \mathsf{SK}_{f_q,2})$ to $\mathcal{A}$.

– Ciphertext Query: $\mathcal{A}$ sends the challenge ciphertext $(\boldsymbol{x}_i^*, \boldsymbol{z}_i^*)_{i\in[N]}$. Now, $\mathcal{B}_1$ samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and computes

$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{z}_i^*||\boldsymbol{w}_i)), \text{ for } i \in [2, N]$$

Next, $\mathcal{B}_1$ sends $(\boldsymbol{x}_1^*, \boldsymbol{z}_1^*|| - \sum_{i\in[2,N]} \boldsymbol{w}_i)$ as its challenge ciphertext to $\mathcal{C}_1$ and receives a ciphertext $\widetilde{\mathsf{CT}}_1$. Finally, $\mathcal{B}_1$ sends the challenge ciphertext $\mathsf{CT} = (\widetilde{\mathsf{CT}}_1, \mathsf{CT}_2, \ldots, \mathsf{CT}_N)$ to $\mathcal{A}$.

Observe that, if $\mathcal{C}_1$ chooses the real algorithms of $\mathit{\Pi}_{\mathsf{extOne}}$ then

$$\begin{aligned}
&(\mathsf{MSK}_1, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)\\
&\widetilde{\mathsf{SK}}_{f_q,1} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_1, (f_q, [\![\boldsymbol{y}_q]\!]_2)) \;\; \forall q \in [Q]\\
&\widetilde{\mathsf{CT}}_1 \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_1, (\boldsymbol{x}_1^*, \boldsymbol{z}_1^*|| - \sum_{i\in[2,N]} \boldsymbol{w}_i))
\end{aligned}$$

and hence $\mathcal{B}_1$ simulates H_0. If $\mathcal{C}_1$ chooses the the simulator of $\mathit{\Pi}_{\mathsf{extOne}}$ then

$$\begin{aligned}
&(\mathsf{MSK}_1^*, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)\\
&\widetilde{\mathsf{SK}}_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2)) \;\; \forall q \in [Q_{\mathsf{pre}}]\\
&\widetilde{\mathsf{CT}}_1 \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_1, (\boldsymbol{x}_1^*, \boldsymbol{z}_1^*|| - \sum_{i\in[2,N]} \boldsymbol{w}_i))\\
&\widetilde{\mathsf{SK}}_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}_1^*, \boldsymbol{x}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![f_q(\boldsymbol{x}_1^*)^\top \boldsymbol{z}_1^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall q \in [Q_{\mathsf{pre}} + 1, Q]
\end{aligned}$$

and hence $\mathcal{B}_1$ simulates H_1.

Lemma 9 *The hybrids $\mathsf{H}_{2,(\eta-1)}$ and $\mathsf{H}_{2,\eta,1}$ are computationally indistinguishable by adaptive simulation-security of $\mathit{\Pi}_{\mathsf{extOne}}$. More specifically, for any* PPT *adversary $\mathcal{A}$, there exists another* PPT *adversary $\mathcal{B}_2$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{H}_{2,(\eta-1)}}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{H}_{2,\eta,1}}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{extFE}}(\lambda)$$

Proof. We prove the lemma by constructing an adversary $\mathcal{B}_2$ against the adaptive simulation-security of $\mathit{\Pi}_{\mathsf{extOne}}$. Let $\mathcal{C}_2$ be the challenger of the security experiment of $\mathit{\Pi}_{\mathsf{extOne}}$. The adversary $\mathcal{B}_2$ works as follows:

– Setup: $\mathcal{B}_2$ gets MPK_2 from $\mathcal{C}_2$ and computes

$$(\mathsf{MSK}_1^*, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

It returns $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$ to $\mathcal{A}$.

– Pre-challenge Key Queries: $\mathcal{A}$ asks for a secret-key corresponding to the function f_q at the q-th key query for $q \in [Q_{\mathsf{pre}}]$. First, $\mathcal{B}_2$ samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and computes

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

Next, $\mathcal{B}_2$ forwards $(f_q, \boldsymbol{y}_q)$ to $\mathcal{C}_2$ and gets a secret-key $\widetilde{\mathsf{SK}}_{f_q,2}$. Finally, $\mathcal{B}_2$ returns $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \widetilde{\mathsf{SK}}_{f_q,2})$ to $\mathcal{A}$.

– Ciphertext Query: $\mathcal{A}$ sends the challenge ciphertext $(\boldsymbol{x}_i^*, \boldsymbol{z}_i^*)_{i\in[N]}$. Now, $\mathcal{B}_2$ samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and defines the set

$$\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\sum_{i\in[\eta-1]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$

Now, $\mathcal{B}_2$ computes the ciphertexts

$$\begin{aligned}
&\mathsf{CT}_1^* \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}_1^*, \mathcal{V}_1)\\
&\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{0}||\boldsymbol{w}_i)) \text{ for } i \in [2, \eta-1],\\
&\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{z}_i^*||\boldsymbol{w}_i)) \text{ for } i \in [\eta+1, N]
\end{aligned}$$

Next, $\mathcal{B}_2$ sends $(\boldsymbol{x}_\eta^*, \boldsymbol{z}_\eta^*||\boldsymbol{w}_\eta)$ as its challenge ciphertext to $\mathcal{C}_2$ and receives a ciphertext $\widetilde{\mathsf{CT}}_\eta$. Finally, $\mathcal{B}_2$ sends the challenge ciphertext $\mathsf{CT} = (\mathsf{CT}_1^*, \mathsf{CT}_2, \ldots, \mathsf{CT}_{\eta-1}, \widetilde{\mathsf{CT}}_\eta, \mathsf{CT}_{\eta+1}, \ldots, \mathsf{CT}_N)$ to $\mathcal{A}$.

– Post-challenge Key Queries: $\mathcal{A}$ asks for a secret-key corresponding to the function f_q at the q-th key query for $q \in [Q_{\mathsf{pre}}+1, Q]$. First, $\mathcal{B}_2$ samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and computes

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}_1^*, \boldsymbol{x}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\textstyle\sum_{i\in[\eta-1]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2)$$

Next, $\mathcal{B}_2$ forwards $(f_q, \boldsymbol{y}_q)$ to $\mathcal{C}_2$ and gets a secret-key $\widetilde{\mathsf{SK}}_{f_q,2}$. Finally, $\mathcal{B}_2$ returns $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \widetilde{\mathsf{SK}}_{f_q,2})$ to $\mathcal{A}$.

Observe that, if $\mathcal{C}_2$ chooses the real algorithms of Π_{extOne} then

$$\begin{aligned}
&(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)\\
&\widetilde{\mathsf{SK}}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2)) \ \ \forall q \in [Q]\\
&\widetilde{\mathsf{CT}}_\eta \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_\eta^*, \boldsymbol{z}_\eta^*||\boldsymbol{w}_\eta))
\end{aligned}$$

and hence $\mathcal{B}_2$ simulates $\mathsf{H}_{2,(\eta-1)}$. If $\mathcal{C}_2$ chooses the the simulator of Π_{extOne} then

$$\begin{aligned}
&(\mathsf{MSK}_2^*, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)\\
&\widetilde{\mathsf{SK}}_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}_2^*, (f_q, [\![\boldsymbol{y}_q]\!]_2)) \ \ \forall q \in [Q_{\mathsf{pre}}]\\
&\widetilde{\mathsf{CT}}_\eta \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_2, \mathsf{MSK}_2^*, \mathcal{V}_2)\\
&\widetilde{\mathsf{SK}}_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}_2^*, \boldsymbol{x}_\eta^*, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![f_q(\boldsymbol{x}_\eta^*)^\top \boldsymbol{z}_\eta^* + \boldsymbol{y}_q^\top \boldsymbol{w}_\eta]\!]_2)\\
&\hspace{22em}\forall q \in [Q_{\mathsf{pre}}+1, Q]
\end{aligned}$$

where $\mathcal{V}_2 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![f_q(\boldsymbol{x}_\eta^*)^\top \boldsymbol{z}_\eta^* + \boldsymbol{y}_q^\top \boldsymbol{w}_\eta]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$ and hence $\mathcal{B}_2$ simulates $\mathsf{H}_{2,\eta,1}$.

Lemma 10 *The hybrids $\mathsf{H}_{2,\eta,1}$ and $\mathsf{H}_{2,\eta,2}$ are computationally indistinguishable by bilateral $\mathsf{MDDH}^1_{k,Q}$ assumption. More specifically, for any* PPT *adversary $\mathcal{A}$, there exists another* PPT *adversary $\mathcal{B}_3$ such that*

$$|\mathsf{Adv}^{\mathsf{H}_{2,\eta,1}}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{H}_{2,\eta,2}}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathcal{B}_3}_{\mathsf{bMDDH}^1_{k,Q}}(\lambda)$$

Proof. We prove the indistinguishability using lemma 1 with $\boldsymbol{w} = \boldsymbol{w}_\eta$ and $\mu_q = f_q(\boldsymbol{x}^*_\eta)^\top \boldsymbol{z}^*_\eta$. Let $\mathcal{B}_3$ be an adversary of lemma 1, who gets a challenge instance

$$\{[\![\rho_{q,1}]\!]_1, [\![\rho_{q,1}]\!]_2, \quad [\![\rho_{q,2}]\!]_1, [\![\rho_{q,2}]\!]_2, \quad [\![\boldsymbol{y}_q]\!]_1, [\![\boldsymbol{y}_q]\!]_2\}_{q\in[Q]}$$

from its challenger. Now, $\mathcal{B}_3$ simulates the game as follows:

– Setup: $\mathcal{B}_3$ generates the master keys as follows:

$$(\mathsf{MSK}^*_1, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B),$$
$$(\mathsf{MSK}^*_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

and sends $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$ to $\mathcal{A}$.

– Pre-challenge Key Queries: $\mathcal{A}$ asks for a secret-key corresponding to the function f_q at the q-th key query for $q \in [Q_{\mathsf{pre}}]$. First, $\mathcal{B}_3$ generate the keys

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2)),$$
$$\mathsf{SK}^*_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_2, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

Then it sends $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \mathsf{SK}^*_{f_q,2})$ to $\mathcal{A}$.

– Ciphertext Query: $\mathcal{A}$ sends the challenge ciphertext $(\boldsymbol{x}^*_i, \boldsymbol{z}^*_i)_{i\in[N]}$. Now, $\mathcal{B}_3$ samples $\boldsymbol{w}_i \leftarrow \mathbb{Z}^k_p$ for all $i \in [2, N] \setminus \{\eta\}$ and defines the set

$$\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\sum_{i\in[\eta-1]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i - \sum_{i\in[2,N]\setminus\{\eta\}} \boldsymbol{y}_q^\top \boldsymbol{w}_i + \rho_{q,1}]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$
$$\mathcal{V}_2 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\rho_{q,2}]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$

Next, it computes the ciphertexts

$$\mathsf{CT}^*_1 \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}^*_1, \mathcal{V}_1)$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \boldsymbol{0}||\boldsymbol{w}_i)) \text{ for } i \in [2, \eta-1],$$
$$\mathsf{CT}^*_\eta \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_2, \mathsf{MSK}^*_2, \mathcal{V}_2),$$
$$\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_i, \boldsymbol{z}^*_i||\boldsymbol{w}_i)) \text{ for } i \in [\eta+1, N]$$

and sends the challenge ciphertext as $\mathsf{CT} = (\mathsf{CT}^*_1, \mathsf{CT}_2, \ldots, \mathsf{CT}_{\eta-1}, \mathsf{CT}^*_\eta, \mathsf{CT}_{\eta+1}, \ldots, \mathsf{CT}_N)$.

– Post-challenge Key Queries: $\mathcal{A}$ asks for a secret-key corresponding to the function f_q at the q-th key query for $q \in [Q_{\mathsf{pre}} + 1, Q]$. First, $\mathcal{B}_2$ samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}^k_p$ and computes

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\sum_{i\in[\eta-1]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i - \sum_{i\in[2,N]\setminus\{\eta\}} \boldsymbol{y}_q^\top \boldsymbol{w}_i + \rho_{q,1}]\!]_2),$$
$$\mathsf{SK}^*_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_2, \boldsymbol{x}^*_\eta, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\rho_{q,2}]\!]_2)$$

and sends $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \mathsf{SK}^*_{f_q,2})$ to $\mathcal{A}$.

Observe that, if $\mathcal{B}_3$ gets the challenge instance such that $\rho_{q,1} = \boldsymbol{y}_q^\top \boldsymbol{w}_\eta$ and $\rho_{q,2} = f_q(\boldsymbol{x}_\eta^*)^\top \boldsymbol{z}_\eta^* + \boldsymbol{y}_q^\top \boldsymbol{w}_\eta$ which corresponds to the first distribution in lemma 1, then we have

$$\sum_{i\in[\eta-1]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]\setminus\{\eta\}} \boldsymbol{y}_q^\top \boldsymbol{w}_i + \rho_{q,1} = \sum_{i\in[\eta-1]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i$$

and hence $\mathcal{B}_3$ simulates $\mathsf{H}_{2,\eta,1}$. If $\mathcal{B}_3$ gets the challenge instance such that $\rho_{q,1} = f_q(\boldsymbol{x}_\eta^*)^\top \boldsymbol{z}_\eta^* - \boldsymbol{y}_q^\top \boldsymbol{w}_\eta$ and $\rho_{q,2} = \boldsymbol{y}_q^\top \boldsymbol{w}_\eta$ which corresponds to the second distribution in lemma 1, then we have

$$\sum_{i\in[\eta-1]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]\setminus\{\eta\}} \boldsymbol{y}_q^\top \boldsymbol{w}_i + \rho_{q,1} = \sum_{i\in[\eta]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i$$

and hence $\mathcal{B}_3$ simulates $\mathsf{H}_{2,\eta,2}$.

Lemma 11 *The hybrids $\mathsf{H}_{2,\eta,2}$ and $\mathsf{H}_{2,\eta,3}$ are computationally indistinguishable by adaptive simulation-security of Π_{extOne}. More specifically, for any* PPT *adversary $\mathcal{A}$, there exists another* PPT *adversary $\mathcal{B}_4$ such that*

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{H}_{2,\eta,2}}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{H}_{2,\eta,3}}(\lambda)| \le \mathsf{Adv}_{\mathcal{B}_4}^{\mathsf{extFE}}(\lambda)$$

Proof. The proof is similar to the lemma 9 with a few changes. We construct an adversary $\mathcal{B}_4$ against the adaptive simulation-security of Π_{extOne} depending on the the adversary $\mathcal{A}$. Let $\mathcal{C}_4$ be the challenger of the security experiment of Π_{extOne}. The adversary $\mathcal{B}_4$ works as follows:

- Setup: $\mathcal{B}_4$ gets MPK_2 from $\mathcal{C}_4$ and computes

  $$(\mathsf{MSK}_1^*, \mathsf{MPK}_1) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$

  It returns $\mathsf{MPK} = (\mathsf{MPK}_1, \mathsf{MPK}_2)$ to $\mathcal{A}$.
- Pre-challenge Key Queries: $\mathcal{A}$ asks for a secret-key corresponding to the function f_q at the q-th key query for $q \in [Q_{\mathsf{pre}}]$. First, $\mathcal{B}_4$ samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and computes

  $$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}_1^*, (f_q, [\![\boldsymbol{y}_q]\!]_2))$$

  Next, $\mathcal{B}_4$ forwards $(f_q, \boldsymbol{y}_q)$ to $\mathcal{C}_4$ and gets a secret-key $\widetilde{\mathsf{SK}}_{f_q,2}$. Finally, $\mathcal{B}_4$ returns $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \widetilde{\mathsf{SK}}_{f_q,2})$ to $\mathcal{A}$.
- Ciphertext Query: $\mathcal{A}$ sends the challenge ciphertext $(\boldsymbol{x}_i^*, \boldsymbol{z}_i^*)_{i\in[N]}$. Now, $\mathcal{B}_4$ samples $\boldsymbol{w}_2, \ldots, \boldsymbol{w}_N \leftarrow \mathbb{Z}_p^k$ and defines the set

  $$\mathcal{V}_1 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\sum_{i\in[\eta]} f_q(\boldsymbol{x}_i^*)^\top \boldsymbol{z}_i^* - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$$

  Now, $\mathcal{B}_4$ computes the ciphertexts

  $$\begin{aligned} &\mathsf{CT}_1^* \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_1, \mathsf{MSK}_1^*, \mathcal{V}_1)\\ &\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{0}||\boldsymbol{w}_i)) \text{ for } i \in [2, \eta-1],\\ &\mathsf{CT}_i \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}_i^*, \boldsymbol{z}_i^*||\boldsymbol{w}_i)) \text{ for } i \in [\eta+1, N] \end{aligned}$$

  Next, $\mathcal{B}_4$ sends $(\boldsymbol{x}_\eta^*, \boldsymbol{0}||\boldsymbol{w}_\eta)$ as its challenge ciphertext to $\mathcal{C}_4$ and receives a ciphertext $\widetilde{\mathsf{CT}}_\eta$. Finally, $\mathcal{B}_4$ sends the challenge ciphertext $\mathsf{CT} = (\mathsf{CT}_1^*, \mathsf{CT}_2, \ldots, \mathsf{CT}_{\eta-1}, \widetilde{\mathsf{CT}}_\eta, \mathsf{CT}_{\eta+1}, \ldots, \mathsf{CT}_N)$ to $\mathcal{A}$.

– Post-challenge Key Queries: $\mathcal{A}$ asks for a secret-key corresponding to the function f_q at the q-th key query for $q \in [Q_{\mathsf{pre}}+1, Q]$. First, $\mathcal{B}_4$ samples $\boldsymbol{y}_q \leftarrow \mathbb{Z}_p^k$ and computes

$$\mathsf{SK}^*_{f_q,1} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_1, \boldsymbol{x}^*_1, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\sum_{i\in[\eta]} f_q(\boldsymbol{x}^*_i)^\top \boldsymbol{z}^*_i - \sum_{i\in[2,N]} \boldsymbol{y}_q^\top \boldsymbol{w}_i]\!]_2)$$

Next, $\mathcal{B}_4$ forwards $(f_q, \boldsymbol{y}_q)$ to $\mathcal{C}_4$ and gets a secret-key $\widetilde{\mathsf{SK}}_{f_q,2}$. Finally, $\mathcal{B}_4$ returns $\mathsf{SK}_{f_q} = (\mathsf{SK}^*_{f_q,1}, \widetilde{\mathsf{SK}}_{f_q,2})$ to $\mathcal{A}$.

Observe that, if $\mathcal{C}_4$ chooses the simulator of Π_{extOne} then

$$(\mathsf{MSK}^*_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}^*_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$
$$\widetilde{\mathsf{SK}}_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},0}(\mathsf{MSK}^*_2, (f_q, [\![\boldsymbol{y}_q]\!]_2)) \quad \forall q \in [Q_{\mathsf{pre}}]$$
$$\widetilde{\mathsf{CT}}_\eta \leftarrow \mathsf{Enc}^*_{\mathsf{extFE}}(\mathsf{MPK}_2, \mathsf{MSK}^*_2, \mathcal{V}_2)$$
$$\widetilde{\mathsf{SK}}_{f_q,2} \leftarrow \mathsf{KeyGen}^*_{\mathsf{extFE},1}(\mathsf{MSK}^*_2, \boldsymbol{x}^*_\eta, (f_q, [\![\boldsymbol{y}_q]\!]_2), [\![\boldsymbol{y}_q^\top \boldsymbol{w}_\eta]\!]_2) \quad \forall q \in [Q_{\mathsf{pre}}+1, Q]$$

where $\mathcal{V}_2 = \{((f_q, [\![\boldsymbol{y}_q]\!]_1), [\![\boldsymbol{y}_q^\top \boldsymbol{w}_\eta]\!]_1) : q \in [Q_{\mathsf{pre}}]\}$ and hence $\mathcal{B}_4$ simulates $\mathsf{H}_{2,\eta,2}$. If $\mathcal{C}_4$ chooses the real algorithms of Π_{extOne} then

$$(\mathsf{MSK}_2, \mathsf{MPK}_2) \leftarrow \mathsf{Setup}_{\mathsf{extFE}}(1^\lambda, 1^n, 1^{n'}, 1^B)$$
$$\widetilde{\mathsf{SK}}_{f_q,2} \leftarrow \mathsf{KeyGen}_{\mathsf{extFE}}(\mathsf{MSK}_2, (f_q, [\![\boldsymbol{y}_q]\!]_2)) \quad \forall q \in [Q]$$
$$\widetilde{\mathsf{CT}}_\eta \leftarrow \mathsf{Enc}_{\mathsf{extFE}}(\mathsf{MPK}_2, (\boldsymbol{x}^*_\eta, \boldsymbol{0}||\boldsymbol{w}_\eta))$$

and hence $\mathcal{B}_4$ simulates $\mathsf{H}_{2,\eta,3}$.

References

ABDCP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *PKC 2015*, pages 733–751. Springer, 2015.

ACGU20. Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. *IACR Cryptology ePrint Archive, Report 2020/577*, 2020.

Agr17. Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *CRYPTO 2017*, pages 3–35. Springer, 2017.

AGT20. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. IACR Cryptology ePrint Archive, Report 2020/1285, 2020.

AGW20. Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-Lin. In *CRYPTO 2020*, pages 685–716. Springer, 2020.

AIK11. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In *FOCS 2011*, pages 120–129. IEEE Computer Society, 2011.

AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO 2015*, pages 308–326. Springer, 2015.

AJS15. Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. 2015.

ALMT20. Shweta Agrawal, Benoît Libert, Monosij Maitra, and Radu Titiu. Adaptive simulation security for inner product functional encryption. In *PKC 2020*, pages 34–64. Springer, 2020.

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *CRYPTO 2016*, pages 333–362. Springer, 2016.

AS17. Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT 2017*, pages 152–181. Springer, 2017.

AY20a. Shweta Agrawal and Shota Yamada. CP-ABE for circuits (and more) in the symmetric key setting. In *TCC 2020*, pages 117–148. Springer, 2020.

AY20b. Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In *EUROCRYPT 2020*, pages 13–43. Springer, 2020.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, pages 41–55. Springer, 2004.

BCFG17. Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *CRYPTO 2017*, pages 67–98. Springer, 2017.

BF01. Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *CRYPTO 2001*, pages 213–229. Springer, 2001.

BGG$^+$14. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT 2014*, pages 533–556. Springer, 2014.

BGW05. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO 2005*, pages 258–275. Springer, 2005.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC 2011*, pages 253–273. Springer, 2011.

BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS 2015*, pages 171–190. IEEE Computer Society, 2015.

BW07. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007*, pages 535–554. Springer, 2007.

CGH$^+$15. Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *CRYPTO 2015*, pages 247–266. Springer, 2015.

CHL$^+$15. Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *EUROCRYPT 2015*, pages 3–12. Springer, 2015.

CLT13. Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO 2013*, pages 476–493. Springer, 2013.

Coc01. Clifford C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMACC 2001*, pages 360–363. Springer, 2001.

DDM16. Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC 2016*, pages 164–195. Springer, 2016.

DKW20. Pratish Datta, Ilan Komargodski, and Brent Waters. Decentralized multi-authority ABE for dnfs from LWE. *IACR Cryptology ePrint Archive, Report 2020/1386*, 2020.

DOT18. Pratish Datta, Tatsuaki Okamoto, and Katsuyuki Takashima. Adaptively simulation-secure attribute-hiding predicate encryption. In *ASIACRYPT 2018*, pages 640–672. Springer, 2018.

DOT20. Pratish Datta, Tatsuaki Okamoto, and Katsuyuki Takashima. Adaptively simulation-secure attribute-hiding predicate encryption. *IEICE TRANSACTIONS on Information and Systems*, 103(7):1556–1597, 2020.

EHK$^+$17. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Rafols, and Jorge Villar. An algebraic framework for diffie–hellman assumptions. *Journal of cryptology*, 30(1):242–288, 2017.

Gay20. Romain Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In *PKC 2020*, pages 95–120. Springer, 2020.

GGH13. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT 2013*, pages 1–17. Springer, 2013.

GGH$^+$16. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

GKP$^+$13. Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC 2013*, pages 555–564. ACM, 2013.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS 2006*, pages 89–98. ACM, 2006.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO 2012*, pages 162–179. Springer, 2012.

GVW15a. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. *Journal of the ACM JACM*, 62(6):1–33, 2015.

GVW15b. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO 2015*, pages 503–523. Springer, 2015.

IK20. Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP 2002*, pages 244–256. Springer, 2020.

IW14. Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In *ICALP 2014*, pages 650–662. Springer, 2014.

JLS19. Aayush Jain, Huijia Lin, and Amit Sahai. Simplifying constructions and assumptions for $i\mathcal{O}$. Technical report, IACR Cryptology ePrint Archive, Report 2019/1252, 2019.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, pages 146–162. Springer, 2008.

KW19. Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. *Journal of Cryptology*, pages 1–49, 2019.

Lin17. Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 prgs. In *CRYPTO 2017*, pages 599–629. Springer, 2017.

LL20. Huijia Lin and Ji Luo. Compact adaptively secure abe from k-Lin: Beyond NC^1 and towards NL. In *EUROCRYPT 2020*, pages 247–277. Springer, 2020.

LOS$^+$10. Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT 2010*, pages 62–91. Springer, 2010.

LT17. Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In *CRYPTO 2017*, pages 630–660. Springer, 2017.

LV16. Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In *FOCS 2016*, pages 11–20. IEEE, 2016.

LV17. Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In *TCC 2017*, pages 119–137. Springer, 2017.

LW10. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC 2010*, pages 455–479. Springer, 2010.

LW11. Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT 2011*, pages 568–588. Springer, 2011.

MSZ16. Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In *CRYPTO 2016*, pages 629–658. Springer, 2016.

Nis91. Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *STOC 1991*, pages 410–418. ACM, 1991.

O'N10. Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive, Report 2010/556*, 2010.

OT10. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO 2010*, pages 191–208. Springer, 2010.

OT12a. Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT 2012*, pages 591–608. Springer, 2012.

OT12b. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *ASIACRYPT 2012*, pages 349–366. Springer, 2012.

OT13. Tatsuaki Okamoto and Katsuyuki Takashima. Efficient (hierarchical) inner-product encryption tightly reduced from the decisional linear assumption. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 96(1):42–52, 2013.

PST14. Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *CRYPTO 2014*, pages 500–517. Springer, 2014.

Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 1984*, pages 47–53. Springer, 1984.

SS10. Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *CCS 2010*, pages 463–472. ACM, 2010.

SW05. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, pages 457–473. Springer, 2005.

Wat09. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO 2009*, pages 619–636. Springer, 2009.

Wat11. Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC 2011*, pages 53–70. Springer, 2011.

Wee17. Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In *TCC 2017*, pages 206–233. Springer, 2017.

Wee20. Hoeteck Wee. Functional encryption for quadratic functions from k-Lin, revisited. In *TCC 2020*, pages 210–228. Springer, 2020.

Wee21. Hoeteck Wee. Broadcast encryption with size $\mathrm{n}^{1/3}$ and more from k-lin. In *Annual International Cryptology Conference*, pages 155–178. Springer, 2021.

A 1-Key 1-Ciphertext Secure One-Slot Extended **FE** Designed for Unbounded-Key One-Slot Extended **FE** for Attribute-Weighted Sums

In this section, we present a private-key one-slot FE scheme for an extended attribute-weighted sum functionality that is proven simulation secure against a single ciphertext query and a single secret key query either before or after the ciphertext query. This scheme will be embedded into the hidden subspaces of the public-key multi-key FE scheme for the same functionality presented in the next section in its security proof. We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ for simplicity of notations. Let $(\mathsf{Garble}, \mathsf{Eval})$ be a special piecewise secure AKGS for a function class $\mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ a tuple of pairing groups of prime order p, and $(\mathsf{IPFE.Setup}, \mathsf{IPFE.KeyGen}, \mathsf{IPFE.Enc}, \mathsf{IPFE.Dec})$ a secret-key function-hiding SK-IPFE based on G.

Setup($1^\lambda, 1^n, 1^{n'}$): Define the following index sets as follows

$$S_{\mathsf{1\text{-}extFE}} = \left\{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \{\mathsf{extnd}_\kappa\}_{\kappa\in[k]}, \{\mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}_{\tau\in[n']}\right\},\ \widehat{S}_{\mathsf{1\text{-}extFE}} = \{\widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}$$

It generates two IPFE master secret-keys $\mathsf{IPFE.MSK} \leftarrow \mathsf{SK\text{-}IPFE.Setup}(S_{\mathsf{1\text{-}extFE}})$ and $\widehat{\mathsf{IPFE.MSK}} \leftarrow \mathsf{SK\text{-}IPFE.Setup}(\widehat{S}_{\mathsf{1\text{-}extFE}})$. Finally, it returns $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$.

KeyGen(MSK, $(\boldsymbol{f}, \boldsymbol{y})$): Let $f = (f_1, \ldots, f_{n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$ and $\boldsymbol{y} \in \mathbb{Z}_p^k$. Samples integers $\nu_t, \beta_t \leftarrow \mathbb{Z}_p$ for $t \in [n']$ such that

$$\sum_{t\in[n']} \nu_t = 1 \text{ and } \sum_{t\in[n']} \beta_t = 0 \quad \text{modulo } p.$$

Next, samples independent random vectors $\boldsymbol{r}_t \leftarrow \mathbb{Z}_p^m$ for garbling and computes the coefficient vectors

$$(\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}, \boldsymbol{\ell}_{m+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \beta_t; \boldsymbol{r}_t)$$

for each $t \in [n']$. Here we make use of the instantiation of the AKGS described in Section 3.6. From the description of that AKGS instantiation, we note that the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}$ would be of the form $\boldsymbol{\ell}_{m+1,t} = \boldsymbol{z}[t] - \boldsymbol{r}_t[m]$. Also all the label functions $\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}$ involve only the variables $\boldsymbol{x}$ and not the variable $\boldsymbol{z}[t]$. Next, for all $j \in [m]$ and $t \in [n']$, it defines the vectors $\boldsymbol{v}_{j,t}$ corresponding to the label functions $\boldsymbol{\ell}_{j,t}$ obtained from the partial garbling above and the vector $\boldsymbol{y}$ as

vector	const	coef$_i$	extnd$_\kappa$	sim$_\tau$	sim$^*_\tau$
$\boldsymbol{v}_{1,t}$	$\boldsymbol{\ell}_{1,t}[\mathsf{const}]$	$\boldsymbol{\ell}_{1,t}[\mathsf{coef}_i]$	$\boldsymbol{y}[\kappa]\nu_t$	0	0
$\boldsymbol{v}_{j,t}$	$\boldsymbol{\ell}_{j,t}[\mathsf{const}]$	$\boldsymbol{\ell}_{j,t}[\mathsf{coef}_i]$	0	0	0

It also sets the vectors $\boldsymbol{v}_{m+1,t}$ for $t \in [n']$ corresponding to the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}$ as

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{m+1,t}$	$\boldsymbol{r}_t[m]$	1	0

Now, it uses the key generation algorithm of IPFE to generate the secret-keys

$$\begin{aligned}
\mathsf{IPFE.SK}_{j,t} &\leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2) && \text{for } j \in [m], t \in [n'] \\
\widehat{\mathsf{IPFE.SK}}_{m+1,t} &\leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2) && \text{for } t \in [n']
\end{aligned}$$

It returns the secret-key as $\mathsf{SK}_{f,\boldsymbol{y}} = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$.

$\mathsf{Enc}(\mathsf{MSK}, (\boldsymbol{x}, \boldsymbol{z}||\boldsymbol{w}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k})$: It sets the following vectors:

vector	const	coef_i	extnd_κ	sim_τ	sim^*_τ
$\boldsymbol{u}$	1	$\boldsymbol{x}[i]$	$\boldsymbol{w}[\kappa]$	0	0

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{h}_t$	-1	$\boldsymbol{z}[t]$	0

for all $t \in [n']$. Then, it encrypts the vectors using IPFE and obtain the ciphertexts

$$\begin{aligned}
\mathsf{IPFE.CT} &\leftarrow \mathsf{SK\text{-}IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1) \\
\widehat{\mathsf{IPFE.CT}}_t &\leftarrow \mathsf{SK\text{-}IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{h}_t]\!]_1) \quad \text{for } t \in [n']
\end{aligned}$$

Finally, it returns the ciphertext as $\mathsf{CT}_{\boldsymbol{x},\boldsymbol{z}||\boldsymbol{w}} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

$\mathsf{Dec}((\mathsf{SK}_{f,\boldsymbol{y}}, f), (\mathsf{CT}_{\boldsymbol{x},\boldsymbol{z}||\boldsymbol{w}}, \boldsymbol{x}))$: It parses $\mathsf{SK}_{f,\boldsymbol{y}} = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ and $\mathsf{CT}_{\boldsymbol{x},\boldsymbol{z}||\boldsymbol{w}} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$. It uses the decryption algorithm of $\mathsf{SK\text{-}IPFE}$ to compute

$$\begin{aligned}
[\![\ell_{1,t} + \psi_t]\!]_T &\leftarrow \mathsf{SK\text{-}IPFE.Dec}(\mathsf{IPFE.SK}_{1,t}, \mathsf{IPFE.CT}) && \text{for } t \in [n'] \\
[\![\ell_{j,t}]\!]_T &\leftarrow \mathsf{SK\text{-}IPFE.Dec}(\mathsf{IPFE.SK}_{j,t}, \mathsf{IPFE.CT}) && \text{for } j \in [2, m], t \in [n'] \\
[\![\ell_{m+1,t}]\!]_T &\leftarrow \mathsf{SK\text{-}IPFE.Dec}(\widehat{\mathsf{IPFE.SK}}_{m+1,t}, \widehat{\mathsf{IPFE.CT}}_t) && \text{for } t \in [n']
\end{aligned}$$

where $\psi_t = \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}$. Next, it utilizes the evaluation procedure of AKGS and returns the combined value

$$[\![\rho]\!]_T = \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T).$$

Correctness: From the correctness of IPFE, we have $\mathsf{SK\text{-}IPFE.Dec}(\mathsf{IPFE.SK}_{1,t}, \mathsf{IPFE.CT}) = [\![\ell_{1,t} + \psi_t]\!]_T$ where $\psi_t = \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}$. Next, using the correctness of IPFE and AKGS evaluation, we get

$$\begin{aligned}
&\mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) \\
&= \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t}]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) + \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\psi_t]\!]_T, [\![0]\!]_T, \ldots, [\![0]\!]_T) \\
&= [\![\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \beta_t + \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}]\!]_T
\end{aligned}$$

The first equality follows from the linearity of Eval function. Now, multiplying all the evaluated values we have

$$\begin{aligned}
[\![\rho]\!]_T &= \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T) \\
&= [\![\sum_{t=1}^{n'} (\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w} + \beta_t)]\!]_T \\
&= [\![f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w}]\!]_T
\end{aligned}$$

The last equality is obtained from the fact that $\sum_{t\in[n']} \nu_t = 1$ and $\sum_{t\in[n']} \beta_t = 0$.

A.1 Security Analysis

Theorem 7 *The 1-*extFE *scheme for attribute-weighted sum is 1-key, 1-ciphertext simulation-secure as per Definition 4 assuming the* AKGS *is piecewise secure as per Definition 7 and the* IPFE *is function hiding as per Definition 5.*

As in the case of our 1-key 1-ciphertext secure one-slot FE, here also we assume that the adversary queries the single secret key before the challenge ciphertext is sent. This is because we will use the security of the 1-key 1-ciphertext secure one-slot extFE in a particular hybrid of the security reduction of our one-slot extFE scheme (presented in Section B) where we deal with a single pre-ciphertext secret key of the one-slot extFE. However, we emphasize that if we consider the single secret key query after the challenge phase then the security can also be proved using the security reduction of our one-slot extFE, given in Section B.1.

The Simulator We describe the simulator for the 1-extFE scheme. Let us assume that $(f, \boldsymbol{y}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')} \times \mathbb{Z}_p^k$ is the only secret-key query made by the adversary before it sends challenge vectors $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$. The algorithm $\mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'})$ is exactly the same as $\mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$ which outputs a master secret-key $\mathsf{MSK}^* = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$. The key generation procedure $\mathsf{KeyGen}_0^*(\mathsf{MSK}^*, (f, \boldsymbol{y}))$ of the simulator is also similar to the original algorithm $\mathsf{KeyGen}(\mathsf{MSK}^*, (f, \boldsymbol{y}))$. We describe the encryption process of the simulator which uses the information $\mu = f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^*$.

$\mathsf{\textbf{Enc}}^*(\mathsf{\textbf{MSK}}^*, \boldsymbol{x}^*, ((\boldsymbol{f}, \boldsymbol{y}), \boldsymbol{\mu}))$: On input MSK^*, a vector $\boldsymbol{x}^* \in \mathbb{Z}_p^n$, the tuple $(f, \boldsymbol{y}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')} \times \mathbb{Z}_p^k$ and an integer $\mu \in \mathbb{Z}_p$ the simulator executes the following steps:

1. It finds a dummy vector $(\boldsymbol{d}_1||\boldsymbol{d}_2) \in \mathbb{Z}_p^{n'+k}$ by solving the linear equation $f(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}^\top \boldsymbol{d}_2 = \mu$. Note that by the restriction of the ideal game, there must exist some vector $(\boldsymbol{z}^*, \boldsymbol{w}^*) \in \mathbb{Z}_p^{n'} \times \mathbb{Z}_p^k$ such that $f(\boldsymbol{x}^*)\boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^* = \mu$. Consequently the existence of the vectors $(\boldsymbol{d}_1, \boldsymbol{d}_2) \in \mathbb{Z}_p^{n'} \times \mathbb{Z}_p^k$ is guaranteed.
2. Next, it sets the following vectors

vector	const	coef_i	extnd_κ	sim_τ	sim^*_τ
$\boldsymbol{u}$	1	$\boldsymbol{x}^*[i]$	$\boldsymbol{d}_2[\kappa]$	0	0

and

vector	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}^*}$
$\boldsymbol{h}_t$	-1	$\boldsymbol{d}_1[t]$	0

for all $t \in [n']$.

3. Finally, it encrypts the vectors as

$$\mathsf{IPFE.CT} \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)$$
$$\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{h}_t]\!]_1) \quad \text{for } t \in [n']$$

4. It returns the simulated ciphertext as $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

Hybrids and Reductions

Proof. We employ a sequence of hybrid experiments to demonstrate the indistinguishability between the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,1\text{-}extFE}}(1^\lambda)$ and the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,1\text{-}extFE}}(1^\lambda)$ where $\mathcal{A}$ is any PPT adversary. We assume that in each experiment, $\mathcal{A}$ queries the single secret-key query for a pair $(f, \boldsymbol{y}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')} \times \mathbb{Z}_p^k$ before submitting the challenge message $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$.

Hybrid $\mathbf{H_0}$: This is the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,1\text{-}extFE}}(1^\lambda)$ where the secret-key $\mathsf{SK}_{f,\boldsymbol{y}} = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ such that $\mathsf{IPFE.SK}_{j,t} \leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2)$ for $j \in [m], t \in [n']$ and $\widehat{\mathsf{IPFE.SK}}_{m+1,t} \leftarrow \mathsf{SK\text{-}IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2)$ for $t \in [n']$ where the vectors $\boldsymbol{v}_{j,t}, \boldsymbol{v}_{m+1,t}$ are given as follows:

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ \boldsymbol{\ell}_{1,t}[\mathsf{const}], \boldsymbol{\ell}_{1,t}[\mathsf{coef}_i], \boldsymbol{y}[\kappa]\nu_t, 0, 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \boldsymbol{\ell}_{j,t}[\mathsf{const}], \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i], \quad 0, \quad 0, 0\ ) \qquad \text{for } 1 < j \le m, \\
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m], 1, 0\ )
\end{aligned}$$

for $j \in [m]$, $t \in [n']$ and $\boldsymbol{r}_t \leftarrow \mathbb{Z}_p^m$. Note that $\{\nu_t\}_{t\in[n']} \leftarrow \mathbb{Z}_p$ is such that $\sum_{t\in[n']} \nu_t = 1$ modulo p. Then, the garblings are computed as

$$(\boldsymbol{\ell}_{1,t}, \ldots, \boldsymbol{\ell}_{m,t}, \boldsymbol{\ell}_{m+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{z}^*[t]f_t(\boldsymbol{x}^*) + \beta_t; \boldsymbol{r}_t)$$

where $\beta_t \leftarrow \mathbb{Z}_p$ for all $t \in [n']$ with $\sum_{t\in[n']} \beta_t = 0$ modulo p. The challenge ciphertext $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$ corresponding to the challenge message $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$ is given by $\mathsf{IPFE.CT} \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{u}]\!]_1)$ and $\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{SK\text{-}IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{h}_t]\!]_1)$ for $t \in [n']$ where

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], \boldsymbol{w}[\kappa], 0, 0\ ), \quad \boldsymbol{h}_t = (\ -1, \boldsymbol{z}^*[t], 0\ )$$

for $t \in [n']$. Note that the components of the vectors $\boldsymbol{u}$ and $\boldsymbol{v}_{j,t}$ are associated with the indices in $S_{\mathsf{1\text{-}extFE}}$, and the components of the vectors $\boldsymbol{h}_t$ and $\boldsymbol{v}_{m+1,t}$ are associated with the indices in $\widehat{S}_{\mathsf{1\text{-}extFE}}$.

Hybrid H_1: This hybrid is exactly the same as H_0 except that we directly hardwire the value $\ell_{1,\tau} + \psi_\tau = \boldsymbol{\ell}_{1,\tau}(\boldsymbol{x}^*) + \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{w}$ into $\boldsymbol{u}[\mathsf{sim}_\tau]$ for all $\tau \in [n']$ and remove the coefficient vector $\boldsymbol{\ell}_{1,t}$ from $\boldsymbol{v}_{1,t}$ for all $t \in [n']$. We change the vectors $\boldsymbol{v}_{1,t}$ in the secret-key and $\boldsymbol{u}$ in the challenge ciphertext as follows:

$$\begin{array}{rllllll}
\boldsymbol{v}_{1,t} = (& \boxed{0}, & \boxed{0}, & \boxed{0}, & \boxed{\delta_{t\tau}}, & 0\ ) & \\
\boldsymbol{v}_{j,t} = (& \boldsymbol{\ell}_{j,t}[\mathsf{const}], & \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i], & 0, & 0, & 0\ ) & \text{for } 1 < j < m, \\
\boldsymbol{u} = (& 1, & \boldsymbol{x}^*[i], & \boxed{0}, & \boxed{\ell_{1,\tau} + \psi_\tau}, & 0\ ) &
\end{array}$$

$$\begin{array}{rl}
\boldsymbol{v}_{m+1,t} = & (\ \boldsymbol{r}_t[m],\ \ 1,\ \ 0\ ) \\
\boldsymbol{h}_t = & (\ -1,\ \ \boldsymbol{z}^*[t], 0\ )
\end{array}$$

We denote by $\delta_{t\tau}$ the usual *Kronecker delta* function such that $\delta_{t\tau} = 1$ if $t = \tau$, 0 otherwise. Note that the inner product $\boldsymbol{v}_{1,t} \cdot \boldsymbol{u} = \ell_{1,t} + \psi_t$, for all $t \in [n']$, remain the same as in H_0. Therefore, the function hiding security of IPFE ensures the indistinguishability between the hybrids H_0 and H_1.

Hybrid H_2: This is analogous to H_1 except that instead of using the actual garbling value $\ell_{1,\tau}$ at $\boldsymbol{u}[\mathsf{sim}_\tau]$, we now use $\widetilde{\ell}_{1,\tau}$ which is computed via reverse sampling algorithm of AKGS:

$$\widetilde{\ell}_{1,\tau} \leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau] + \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{w} + \beta_\tau, \ell_{2,\tau}, \ldots, \ell_{m+1,\tau})$$

where $\ell_{j,\tau} = \boldsymbol{\ell}_{j,\tau}(\boldsymbol{x}^*)$ for all $j \in [2, m]$ and $\ell_{m+1,\tau} = -\boldsymbol{r}_\tau[m] + \boldsymbol{z}^*[\tau]$ for all $\tau \in [n']$. Therefore, the vectors in the challenge ciphertext becomes

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], 0, \boxed{\widetilde{\ell}_{1,\tau}}, 0\ ), \quad \boldsymbol{h}_t = (\ -1, \boldsymbol{z}^*[t], 0\ ).$$

For each $\tau \in [n']$, the piecewise security of AKGS guarantees that given the label functions $(\boldsymbol{\ell}_{2,\tau}, \ldots, \boldsymbol{\ell}_{m,\tau}, \boldsymbol{\ell}_{m+1,\tau})$, the actual garbled label $\ell_{1,\tau}$ and the reversely sampled value $\widetilde{\ell}_{1,\tau}$ are identically distributed. Hence, the hybrids H_1 and H_2 are indistinguishable by the reverse sampleability of AKGS.

Hybrid $\mathsf{H}_{3,j}$ ($j \in [2, m]$): The hybrid proceeds similar to H_2 except that we change the secret-key as follows. For all j' such that $1 < j' < j$, the coefficient vector $\boldsymbol{\ell}_{j,t}$ is taken away from $\boldsymbol{v}_{j',t}$ and a random value $\ell'_{j',t} \leftarrow \mathbb{Z}_p$ is put into $\boldsymbol{v}_{j',t}[\mathsf{const}]$. We describe the vectors associated with the secret-key and the ciphertext below.

$$\begin{array}{rllllll}
\boldsymbol{v}_{1,t} = (& 0, & 0, & 0, & \delta_{t\tau}, & 0\ ) & \\
\boldsymbol{v}_{j',t} = (& \boxed{\ell'_{j',t}}, & \boxed{0}, & 0, & 0, & 0\ ) & \text{for } 1 < j' \leq j, \\
\boldsymbol{v}_{j',t} = (& \boldsymbol{\ell}_{j',t}[\mathsf{const}], & \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], & 0, & 0, & 0\ ) & \text{for } j < j' \leq m, \\
\boldsymbol{u} = (& 1, & \boldsymbol{x}^*[i], & 0, & \widetilde{\ell}_{1,\tau}, & 0\ ) &
\end{array}$$

$$\begin{array}{rl}
\boldsymbol{v}_{m+1,t} = & (\ \boldsymbol{r}_t[m],\ \ 1,\ \ 0\ ) \\
\boldsymbol{h}_t = & (\ -1,\ \ \boldsymbol{z}^*[t], 0\ )
\end{array}$$

Note that, in this hybrid $\widetilde{\ell}_{1,\tau}$ is reversely sampled using the random values $\ell_{2,\tau}, \ldots, \ell_{j-1,\tau}$ (which are randomly chosen from $\mathbb{Z}_p$) and the actual values $\ell_{j,\tau}, \ldots, \ell_{m+1,\tau}$ for each $\tau \in [n']$. Observe that $\mathsf{H}_{3,1}$ coincides with H_2. We will show that for all $j \in [2, m]$, the hybrids $\mathsf{H}_{3,(j-1)}$ and $\mathsf{H}_{3,j}$ are indistinguishable via the following sequence of sub-hybrids, namely, $\{\mathsf{H}_{3,j,1}, \mathsf{H}_{3,j,2}, \mathsf{H}_{3,j,3}\}_{j \in [2,m]}$.

Hybrid $\mathsf{H}_{3,j,1}$ ($j \in [2, m]$): This is exactly the same as $\mathsf{H}_{3,(j-1)}$ except that the coefficient vector $\boldsymbol{\ell}_{j,t}$ is removed from $\boldsymbol{v}_{j,t}$ and $\boldsymbol{v}_{j,t}[\mathsf{sim}^*_\tau]$ is set to $\delta_{t\tau}$. The actual garbling value $\ell_{j,\tau} = \boldsymbol{\ell}_{j,\tau}(\boldsymbol{x}^*)$ is hardwired into $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ to ensure the inner product $\boldsymbol{v}_{j,\tau} \cdot \boldsymbol{u}$ remains the same as in $\mathsf{H}_{3,(j-1)}$. The changes in the vectors involved while computing secret-key and the challenge ciphertext as given below.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\ 0,\ 0,\ \delta_{t\tau},\ 0\ ) \\
\boldsymbol{v}_{j',t} &= (\ \ell'_{j',t},\ 0,\ 0,\ 0,\ 0\ ) && \text{for } 1 < j' < j, \\
\boldsymbol{v}_{j,t} &= (\ \boxed{0},\ \boxed{0},\ 0,\ 0,\ \boxed{\delta_{t\tau}}\ ) \\
\boldsymbol{v}_{j',t} &= (\ \boldsymbol{\ell}_{j',t}[\mathsf{const}],\ \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i],\ 0,\ 0,\ 0\ ) && \text{for } j < j' \leq m, \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i],\ 0,\ \widetilde{\ell}_{1,\tau},\ \boxed{\ell_{j,\tau}}\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m],\ 1,\ 0\ ) \\
\boldsymbol{h}_t &= (\ -1,\ \boldsymbol{z}^*[t], 0\ )
\end{aligned}$$

The hybrids $\mathsf{H}_{3,(j-1)}$ and $\mathsf{H}_{3,j,1}$ are indistinguishable by the function hiding security of IPFE since the inner product $\boldsymbol{v}_{j,\tau} \cdot \boldsymbol{u}$ for all $\tau \in [n']$ remains the same as in $\mathsf{H}_{3,(j-1)}$.

Hybrid $\mathsf{H}_{3,j,2}$ ($j \in [2, m]$): It proceeds exactly the same as $\mathsf{H}_{3,j,1}$ except that the actual label $\ell_{j,\tau}$ (sitting at $\boldsymbol{u}[\mathsf{sim}^*_\tau]$) is replaced with a random value $\ell'_{j,\tau} \leftarrow \mathbb{Z}_p$. The vectors associated to the challenge ciphertext are given by

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], 0, \widetilde{\ell}_{1,\tau}, \boxed{\ell'_{j,\tau}}\ ), \quad \boldsymbol{h}_t = (\ -1, \boldsymbol{z}^*[t], 0\ )$$

where $\ell'_{j,\tau}$ is randomly sampled from $\mathbb{Z}_p$. Now, the first label $\widetilde{\ell}_{1,\tau}$ is reversely sampled using the random values $\ell'_{2,\tau}, \ldots, \ell'_{j,\tau}$ and the actual labels $\ell_{j+1,\tau} = \boldsymbol{\ell}_{j+1,\tau}(\boldsymbol{x}^*), \ldots, \ell_{m,\tau} = \boldsymbol{\ell}_{m,\tau}(\boldsymbol{x}^*), \ell_{m+1,\tau} = -\boldsymbol{r}_\tau[m] + \boldsymbol{z}^*[\tau]$. The marginal randomness property of AKGS implies that the hybrids $\mathsf{H}_{3,j,1}$ and $\mathsf{H}_{3,j,2}$ are identically distributed.

Hybrid $\mathsf{H}_{3,j,3}$ ($j \in [2, m]$): The hybrid is analogous to $\mathsf{H}_{3,j,2}$ except that the random value $\ell'_{j,\tau}$ is sifted from the ciphertext component $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ to the secret-key component $\boldsymbol{v}_{j,t}[\mathsf{const}]$. Also, the positions $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ and $\boldsymbol{v}_{j,t}[\mathsf{sim}^*_\tau]$ are set to zero. Thus, the vectors in the secret-key and the challenge ciphertext become

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\ 0,\ 0,\ \delta_{t\tau},\ 0\ ) \\
\boldsymbol{v}_{j',t} &= (\ \ell'_{j',t},\ 0,\ 0,\ 0,\ 0\ ) && \text{for } 1 < j' < j, \\
\boldsymbol{v}_{j,t} &= (\ \boxed{\ell'_{j,t}},\ 0,\ 0,\ 0,\ \boxed{0}\ ) \\
\boldsymbol{v}_{j',t} &= (\ \boldsymbol{\ell}_{j',t}[\mathsf{const}],\ \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i],\ 0,\ 0,\ 0\ ) && \text{for } j < j' \leq m, \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i],\ 0,\ \widetilde{\ell}_{1,\tau},\ \boxed{0}\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m],\ 1,\ 0\ ) \\
\boldsymbol{h}_t &= (\ -1,\ \boldsymbol{z}^*[t], 0\ )
\end{aligned}$$

Since the inner products $\boldsymbol{v}_{j,t} \cdot \boldsymbol{u}$ for all j, t remain the same as in $\mathsf{H}_{3,j,2}$, the indistinguishability between the hybrids $\mathsf{H}_{3,j,2}$ and $\mathsf{H}_{3,j,3}$ follows from the function hiding security of IPFE. We observe that the hybrids $\mathsf{H}_{3,j,3}$ is identical to $\mathsf{H}_{3,j}$ for all $j \in [2, m]$.

Hybrid $\mathsf{H_4}$: It proceeds exactly the same as hybrid $\mathsf{H}_{3,m}$ except that the actual garbling value $\ell_{m+1,t} = -\boldsymbol{r}_t[m] + \boldsymbol{z}^*[t]$ is used in $\boldsymbol{h}_t[\widehat{\mathsf{sim}^*}]$. Also, $\boldsymbol{h}_t[\widehat{\mathsf{coef}}], \boldsymbol{v}_{m+1,t}[\widehat{\mathsf{const}}], \boldsymbol{v}_{m+1,t}[\widehat{\mathsf{coef}}]$ are set to zero. The changes are indicated below.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\quad 0,\quad 0,\ \delta_{t\tau},\ 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t},\quad 0,\quad 0,\ \ 0,\ \ 0\ ) \qquad \text{for } 1 < j \leq m, \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i], 0, \widetilde{\ell}_{1,\tau}, 0\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{0}, \boxed{0}, \quad \boxed{1} \quad) \\
\boldsymbol{h}_t &= (\ \boxed{1}, \boxed{0}, \boxed{\ell_{m+1,t}}\ )
\end{aligned}$$

Since the inner products $\boldsymbol{v}_{m+1,t} \cdot \boldsymbol{h}_t$ for all $t \in [n']$ are unaltered as in H_4, the indistinguishability between the hybrids H_3 and H_4 follows from the function hiding security of IPFE.

Hybrid $\mathsf{H_5}$: It is analogous to H_4 except that the actual label $\ell_{m+1,t}$ is now replaced with a random value $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$. The vectors associated with the challenge ciphertext are modified as follows.

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], 0, \widetilde{\ell}_{1,\tau}, 0\ ), \quad \boldsymbol{h}_t = (\ 1, 0, \boxed{\ell'_{m+1,t}}\ )$$

Note that, in this hybrid the labels $\widetilde{\ell}_{1,t}$ for $t \in [n']$ are now reversely sampled using all random values $\ell'_{2,t}, \ldots, \ell'_{m+1,t}$ which are randomly picked from $\mathbb{Z}_p$. By the marginal randomness property of AKGS, the hybrids H_4 and H_5 are identically distributed.

Hybrid $\mathsf{H_6}$: This hybrid proceeds exactly the same as H_5 except that the simulated labels $\ell'_{m+1,t}$ are shifted from $\boldsymbol{h}_t[\widehat{\mathsf{sim}^*}]$ to $\boldsymbol{v}_{m+1,t}[\widehat{\mathsf{rand}}]$. The positions $\boldsymbol{v}_{m+1,t}[\widehat{\mathsf{sim}}^*]$ and $\boldsymbol{h}_t[\widehat{\mathsf{sim}}^*]$ are set to zero. The changes are indicated as follows.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\quad 0,\quad 0,\ \delta_{t\tau},\ 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t},\quad 0,\quad 0,\ \ 0,\ \ 0\ ) \qquad \text{for } 1 < j \leq m, \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i], 0, \widetilde{\ell}_{1,\tau}, 0\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{\ell'_{m+1,t}}, 0, \boxed{0}\ ) \\
\boldsymbol{h}_t &= (\quad 1, \quad 0, \boxed{0}\ )
\end{aligned}$$

Observe that the inner products $\boldsymbol{v}_{m+1,t} \cdot \boldsymbol{h}_t$ for all $t \in [n']$ are unchanged as in H_5. Hence, the function-hiding security of IPFE ensures the indistinguishability between the hybrids H_5 and H_6.

Hybrid $\mathsf{H_7}$: It is analogous to H_6 except that the value $f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau]$ is removed from $\widetilde{\ell}_{1,\tau}$ for all $1 < \tau \leq n'$ and the value $f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^*$ is directly encoded into the label $\widetilde{\ell}_{1,1}$. To make this change, we replace the random elements β_τ by $\beta'_\tau = \beta_\tau - f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau] - \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{w}^*$ for all $1 < \tau \leq n'$ and change the element β_1 with $\beta'_1 = \beta_1 - (f_1(\boldsymbol{x}^*)\boldsymbol{z}^*[1] + \nu_1 \cdot \boldsymbol{y}^\top \boldsymbol{w}^*) + f(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}^\top \boldsymbol{w}^*$. Note that, the distributions

$$\{\beta_\tau \leftarrow \mathbb{Z}_p : \sum_{\tau \in [n']} \beta_\tau = 0 \mod p\} \text{ and } \{\beta'_\tau : \sum_{\tau \in [n']} \beta_\tau = 0 \mod p\}$$

are statistically close since β'_τ is also uniform over $\mathbb{Z}_p$ and $\sum_{\tau\in[n']}\beta'_\tau = 0 \mod p$. Thus the vectors associated to the challenge ciphertext become

$$\boldsymbol{u} = (\,1, \boldsymbol{x}^*[i], 0, \boxed{\widetilde{\ell}_{1,\tau}}, 0\,), \quad \boldsymbol{h}_t = (\,1, 0, 0\,)$$

where the labels $\widetilde{\ell}_{1,\tau}$ are given by

$$\begin{aligned}\widetilde{\ell}_{1,1} &\leftarrow \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f_1(\boldsymbol{x}^*)\boldsymbol{z}^*[1] + \nu_1\cdot\boldsymbol{y}^\top\boldsymbol{w}^* + \beta'_1, \ell'_{2,1}, \ldots, \ell'_{m+1,1})\\ &= \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f(\boldsymbol{x}^*)^\top\boldsymbol{z}^* + \boldsymbol{y}^\top\boldsymbol{w}^* + \beta_1, \ell'_{2,1}, \ldots, \ell'_{m+1,1})\\ \widetilde{\ell}_{1,\tau} &\leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, f_\tau(\boldsymbol{x}^*)\boldsymbol{z}^*[\tau] + \nu_\tau\cdot\boldsymbol{y}^\top\boldsymbol{w}^* + \beta'_\tau, \ell'_{2,\tau}, \ldots, \ell'_{m+1,\tau})\\ &= \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, \beta_\tau, \ell'_{2,\tau}, \ldots, \ell'_{m+1,\tau}) \quad \text{for } 1 < \tau \le n'\end{aligned}$$

Thus, H_6 and H_7 are indistinguishable from the adversary's view as they are statistically close. As discussed in the remark of H_2, the challenger can also simulate this hybrid when $[\![\boldsymbol{y}]\!]_1$ is known instead of $\boldsymbol{y}$.

Hybrid $\mathbf{H_8}$: This hybrid is exactly the same as H_7 except that we use a dummy vector $(\boldsymbol{d}_1 \parallel \boldsymbol{d}_2) \in \mathbb{Z}_p^{n'+k}$ in place of $(\boldsymbol{z}^* \parallel \boldsymbol{w}^*)$ while computing $\widetilde{\ell}_{1,1}$ where it holds that $\mu = f(\boldsymbol{x}^*)^\top\boldsymbol{z}^* + \boldsymbol{y}^\top\boldsymbol{w}^* = f(\boldsymbol{x}^*)^\top\boldsymbol{d}_1 + \boldsymbol{y}^\top\boldsymbol{d}_2$. The vector $\boldsymbol{u}$ is now defined as

$$\boldsymbol{u} = (\,1, \overbrace{\boldsymbol{x}^*[1], \ldots, \boldsymbol{x}^*[n]}^{\mathsf{coef}_i}, \overbrace{0, \ldots, 0}^{\mathsf{extnd}_\kappa}, \overbrace{\boxed{\widetilde{\ell}_{1,1}}, \widetilde{\ell}_{1,2} \ldots, \widetilde{\ell}_{1,n'}}^{\mathsf{sim}_\tau}, \overbrace{0, \ldots, 0}^{\mathsf{sim}^*_\tau}\,)$$

where the labels are computed as

$$\begin{aligned}\widetilde{\ell}_{1,1} &\leftarrow \mathsf{RevSamp}(f_1, \boldsymbol{x}^*, f(\boldsymbol{x}^*)^\top\boldsymbol{d}_1 + \boldsymbol{y}^\top\boldsymbol{d}_2 + \beta_1, \ell'_{2,1}, \ldots, \ell'_{m+1,1})\\ \widetilde{\ell}_{1,\tau} &\leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, \beta_\tau, \ell'_{2,\tau}, \ldots, \ell'_{m+1,\tau}) \quad \text{for } 1 < \tau \le n'\end{aligned}$$

Above, we write the full expression of the vector $\boldsymbol{u}$ as opposed to its compressed expression used so far in order to highlight the change. Since the inner product $\boldsymbol{v}_{j,t}\cdot\boldsymbol{u}$ for each $j \in [m], t \in [n']$ are unaltered between the two hybrids, the function-hiding security of IPFE preserved the indistinguishability of the hybrids H_7 and H_8.

Hybrid $\mathbf{H_9}$: The following sequence of hybrids is basically the reverse of the previous hybrids with $(\boldsymbol{z}^* \parallel \boldsymbol{w}^*)$ replaced with $(\boldsymbol{d}_1 \parallel \boldsymbol{d}_2)$. In this hybrid, we change the distribution of β_τ similar to what we did in H_7. In particular, β_τ is replaced with $\beta'_\tau = \beta_\tau + f_\tau(\boldsymbol{x}^*)\boldsymbol{d}_1[\tau] + \nu_\tau\cdot\boldsymbol{y}^\top\boldsymbol{d}_2$ and β_1 is replaced with $\beta'_1 = \beta_1 + f_1(\boldsymbol{x}^*)\boldsymbol{d}_1[1] + \nu_1\cdot\boldsymbol{y}^\top\boldsymbol{d}_2 - (f(\boldsymbol{x}^*)^\top\boldsymbol{d}_1 + \boldsymbol{y}^\top\boldsymbol{d}_2)$. So, the vectors associated with challenge ciphertext are distributed as

$$\boldsymbol{u} = (\,1, \boldsymbol{x}^*[i], 0, \boxed{\widetilde{\ell}_{1,\tau}}, 0\,), \quad \boldsymbol{h}_t = (\,1, 0, 0\,)$$

where $\widetilde{\ell}_{1,\tau} \leftarrow \mathsf{RevSamp}(f_\tau, \boldsymbol{x}^*, f_\tau(\boldsymbol{x}^*)\boldsymbol{d}_1[\tau] + \nu_\tau\cdot\boldsymbol{y}^\top\boldsymbol{d}_2 + \beta_\tau, \ell'_{2,\tau}, \ldots, \ell'_{m+1,\tau})$ Note that, H_8 and H_9 are statistically close as $\{\beta_\tau : \tau \in [n']\}$ and $\{\beta'_\tau : \tau \in [n']\}$ are both uniform over $\mathbb{Z}_p$ with $\sum_{\tau\in[n']}\beta_\tau = \sum_{\tau\in[n']}\beta'_\tau = 0 \mod p$. Hence, hybrids H_8 and H_9 are indistinguishable.

Hybrid H_{10}: In this hybrid we change the vectors $\boldsymbol{v}_{m+1,t}$ and $\boldsymbol{h}_t$ as follows

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\quad 0,\quad 0,\ \delta_{t\tau},\ 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t},\quad 0,\quad 0,\ 0,\ 0\ ) \qquad \text{for } 1 < j \le m, \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i], 0, \widetilde{\ell}_{1,\tau}, 0\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{0}, 0,\quad \boxed{1}\quad) \\
\boldsymbol{h}_t &= (\ 1,\ 0, \boxed{\ell'_{m+1,t}}\ )
\end{aligned}$$

where $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$. The indistinguishability between the hybrids H_9 and H_{10} follows from the function-hiding security of IPFE.

Hybrid H_{11}: It is exactly the same as H_{10} except that the random values $\ell'_{m+1,t} \leftarrow \mathbb{Z}_p$ are changed to the actual label $\ell_{m+1,t} = \boldsymbol{d}_1[t] - \boldsymbol{r}_t[m]$. Then the vectors associated with the challenge ciphertext become

$$\boldsymbol{u} = (\ 1, \boldsymbol{x}^*[i], 0, \widetilde{\ell}_{1,\tau}, 0\ ), \qquad \boldsymbol{h}_t = (\ 1, 0, \boxed{\ell_{m+1,t}}\ )$$

The hybrids H_{11} and H_{12} are identical due to the marginal randomness property of AKGS.

Hybrid H_{12}: In this hybrid we change the vectors $\boldsymbol{v}_{m+1,t}$ and $\boldsymbol{h}_t$ as follows

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\ 0,\quad 0,\quad 0,\ \delta_{t\tau},\ 0\ ) \\
\boldsymbol{v}_{j,t} &= (\ \ell'_{j,t},\quad 0,\quad 0,\ 0,\ 0\ ) \qquad \text{for } 1 < j \le m, \\
\boldsymbol{u} &= (\ 1,\ \boldsymbol{x}^*[i], 0, \widetilde{\ell}_{1,\tau}, 0\ ) \\
\boldsymbol{v}_{m+1,t} &= (\ \boxed{\boldsymbol{r}_t[m]},\quad \boxed{1},\quad \boxed{0}\ ) \\
\boldsymbol{h}_t &= (\quad \boxed{-1},\quad \boxed{\boldsymbol{d}_1[t]}, \boxed{0}\ )
\end{aligned}$$

The indistinguishability between the hybrids H_{11} and H_{12} follows from the function-hiding security of IPFE.

Hybrid $\mathsf{H}_{13,m+1-j}$ ($j \in [m-1]$): It is analogous to H_{12} except the secret-key is modified as follows. For all j' such that $m+1-j \le j' < m+1$, the random value $\ell'_{j',t} \leftarrow \mathbb{Z}_p$ is discarded from $\boldsymbol{v}_{j',t}[\mathsf{const}]$ and the coefficient vector $\boldsymbol{\ell}_{j',t}$ is used in $\boldsymbol{v}_{j',t}$.

$$\begin{aligned}
\boldsymbol{v}_{1,t} &= (\qquad 0, \qquad\qquad 0, \qquad 0, \delta_{t\tau}, 0\ ) \\
\boldsymbol{v}_{j',t} &= (\qquad \ell'_{j',t}, \qquad\qquad 0, \qquad 0,\ 0,\ 0\ ) \text{ for } 1 < j' < m+1-j, \\
\boldsymbol{v}_{j',t} &= (\ \boxed{\boldsymbol{\ell}_{j',t}[\mathsf{const}]}, \boxed{\boldsymbol{\ell}_{j',t}[\mathsf{coef}_i]}, 0,\ 0,\ 0\ ) \text{ for } m+1-j \le j' < m+1, \\
\boldsymbol{v}_{m+1,t} &= (\ \boldsymbol{r}_t[m], 1, 0\ )
\end{aligned}$$

In this hybrid, the label $\widetilde{\ell}_{1,t}$ is reversely sampled using the random values $\ell'_{2,t}, \ldots, \ell'_{m+1-j,t}$ and the actual values $\ell_{m-j+2,t}, \ldots, \ell_{m+1,t}$ for each $t \in [n']$. The hybrids $\mathsf{H}_{13,m+1-(j-1)}$ and $\mathsf{H}_{13,m+1-j}$ can be shown to be indistinguishable via the following sequence of sub-hybrids, namely, $\{\mathsf{H}_{13,m+1-j,1}, \mathsf{H}_{13,m+1-j,2}, \mathsf{H}_{13,m+1-j,3}\}_{j\in[m-1]}$.

Hybrid $\mathsf{H}_{13,m+1-j,1}$ ($j \in [m-1]$): It proceeds exactly the same as $\mathsf{H}_{13,m+1-(j-1)}$ except that the random labels $\ell'_{m+1-j,t}$ are sifted from $\boldsymbol{v}_{m+1-j,t}[\mathsf{const}]$ to $\boldsymbol{u}[\mathsf{sim}^*_\tau]$. We modify the vectors associated with the secret-key and the challenge ciphertext as follows

$$\begin{array}{rll}
\boldsymbol{v}_{1,t} = (& 0, \quad 0, \quad 0, \; \delta_{t\tau}, \quad 0 &) \\
\boldsymbol{v}_{j',t} = (& \ell'_{j',t}, \quad 0, \quad 0, \; 0, \quad 0 &) \text{ for } 1 < j' < m+1-j, \\
\boldsymbol{v}_{m+1-j,t} = (& \boxed{0}, \quad 0, \quad 0, \; 0, \quad \boxed{\delta_{t\tau}} &) \\
\boldsymbol{v}_{j',t} = (& \boldsymbol{\ell}_{j',t}[\mathsf{const}], \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], 0, \; 0, \quad 0 &) \text{ for } m+1-j < j' < m+1, \\
\boldsymbol{u} = (& 1, \quad \boldsymbol{x}^*[i], \quad 0, \widetilde{\ell}_{1,\tau}, \boxed{\ell'_{m+1-j,\tau}} &)
\end{array}$$

$$\begin{array}{rl}
\boldsymbol{v}_{m+1,t} = & (\boldsymbol{r}_t[m], \quad 1, \quad 0) \\
\boldsymbol{h}_t = & (\; -1, \quad \boldsymbol{d}_1[t], 0)
\end{array}$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-(j-1)}$ and $\mathsf{H}_{13,m+1-j,1}$ follows from the function-hiding security of IPFE.

Hybrid $\mathsf{H}_{13,m+1-j,2}$ ($j \in [m-1]$): It is exactly same as $\mathsf{H}_{13,m+1-j,1}$ except that the random label $\ell'_{m+1-j,\tau} \leftarrow \mathbb{Z}_p$ at $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ are now replaced with the actual labels $\ell_{m+1-j,\tau} = \boldsymbol{\ell}_{m+1-j,\tau}(\boldsymbol{x}^*)$. The change in the vector $\boldsymbol{u}$ associated to the challenge ciphertext is indicated below.

$$\boldsymbol{u} = (1, \boldsymbol{x}^*[i], 0, \widetilde{\ell}_{1,\tau}, \boxed{\ell_{m+1-j,\tau}}), \quad \boldsymbol{h}_t = (-1, \boldsymbol{d}_1[t], 0)$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-j,1}$ and $\mathsf{H}_{13,m+1-j,2}$ follows from the marginal randomness property of AKGS.

Hybrid $\mathsf{H}_{13,m+1-j,3}$ ($j \in [m-1]$): It proceeds analogous to $\mathsf{H}_{13,m+1-j,2}$ except that the actual label $\ell_{m+1-j,\tau} = \boldsymbol{\ell}_{m+1-j,\tau}(\boldsymbol{x}^*)$ is removed from $\boldsymbol{u}[\mathsf{sim}^*_\tau]$ and the coefficient vector $\boldsymbol{\ell}_{m+1-j,t}$ is used to set $\boldsymbol{v}_{m+1-j,t}$. The inner product $\boldsymbol{v}_{m+1-j,t} \cdot \boldsymbol{u}$ is unaltered as in $\mathsf{H}_{13,m+1-j,2}$. The changes in the vectors associated to the secret-key and the challenge ciphertext are shown below.

$$\begin{array}{rll}
\boldsymbol{v}_{1,t} = (& 0, \quad 0, \quad 0, \; \delta_{t\tau}, \; 0 &) \\
\boldsymbol{v}_{j',t} = (& \ell'_{j',t}, \quad 0, \quad 0, \; 0, \; 0 &) \text{ for } 1 < j' < m+1-j, \\
\boldsymbol{v}_{m+1-j,t} = (& \boxed{\boldsymbol{\ell}_{m+1-j,t}[\mathsf{const}]}, \boxed{\boldsymbol{\ell}_{m+1-j,t}[\mathsf{coef}_i]}, 0, \; 0, \; \boxed{0} &) \\
\boldsymbol{v}_{j',t} = (& \boldsymbol{\ell}_{j',t}[\mathsf{const}], \quad \boldsymbol{\ell}_{j',t}[\mathsf{coef}_i], \quad 0, \; 0, \; 0 &) \text{ for } m+1-j < j' < m+1, \\
\boldsymbol{u} = (& 1, \quad \boldsymbol{x}^*[i], \quad 0, \widetilde{\ell}_{1,\tau}, \boxed{0} &)
\end{array}$$

$$\begin{array}{rl}
\boldsymbol{v}_{m+1,t} = & (\boldsymbol{r}_t[m], \quad 1, \quad 0) \\
\boldsymbol{h}_t = & (\; -1, \quad \boldsymbol{d}_1[t], 0)
\end{array}$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m+1-j,2}$ and $\mathsf{H}_{13,m+1-j,3}$ follows from the function-hiding security of IPFE. We observe that $\mathsf{H}_{13,m+1-j,3}$ is identical to $\mathsf{H}_{13,m+1-j}$ for all $j \in [m-1]$.

Hybrid H_{14}: It proceeds exactly the same as $\mathsf{H}_{13,2}$ except that the reversely sampled labels $\widetilde{\ell}_{1,\tau}$ are replaced with the actual labels $\ell_{1,\tau} + \psi_\tau = \boldsymbol{\ell}_{1,\tau}(\boldsymbol{x}^*) + \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{d}_2$ when setting $\boldsymbol{u}[\mathsf{sim}_\tau]$. The vectors associated with the challenge ciphertext are now written as

$$\boldsymbol{u} = (1, \boldsymbol{x}^*[i], 0, \boxed{\ell_{1,\tau} + \psi_\tau}, 0), \quad \boldsymbol{h}_t = (-1, \boldsymbol{d}_1[t], 0)$$

The indistinguishability between the hybrids $\mathsf{H}_{13,m}$ and H_{14} follows from the piecewise security of AKGS.

Hybrid H_{15}: It is analogous to H_{14} except that the actual label $\ell_{1,\tau} = \boldsymbol{\ell}_{1,\tau}(\boldsymbol{x}^*) + \nu_\tau \cdot \boldsymbol{y}^\top \boldsymbol{d}_2$ is removed from $\boldsymbol{u}[\mathsf{sim}_\tau]$ and the coefficient vectors $\boldsymbol{\ell}_{1,t}$ are utilized while setting the vectors $\boldsymbol{v}_{1,t}$ for all $t \in [n']$. Also, the positions $\boldsymbol{v}_{1,t}[\mathsf{extnd}_\kappa]$ and $\boldsymbol{u}[\mathsf{extnd}_\kappa]$ are set as $\boldsymbol{y}[\kappa]\nu_t$ and $\boldsymbol{d}_2[\kappa]$ respectively. The vectors associated with the secret-key and the challenge ciphertext are shown below.

$$\begin{array}{rlllll}
\boldsymbol{v}_{1,t} = (& \boxed{\boldsymbol{\ell}_{1,t}[\mathsf{const}]}, & \boxed{\boldsymbol{\ell}_{1,t}[\mathsf{coef}_i]}, & \boxed{\boldsymbol{y}[\kappa]\nu_t}, & 0, & 0\ ) \\
\boldsymbol{v}_{j,t} = (& \boldsymbol{\ell}_{j,t}[\mathsf{const}], & \boldsymbol{\ell}_{j,t}[\mathsf{coef}_i], & 0, & 0, & 0\ ) \quad \text{for } 1 < j \le m, \\
\boldsymbol{u} = (& 1, & \boldsymbol{x}^*[i], & \boxed{\boldsymbol{d}_2[\kappa]}, & \boxed{0}, & 0\ )
\end{array}$$

$$\begin{array}{rlll}
\boldsymbol{v}_{m+1,t} = (& \boldsymbol{r}_t[m], & 1, & 0\ ) \\
\boldsymbol{h}_t = (& -1, & \boldsymbol{d}_1[t], & 0\ )
\end{array}$$

Since the inner products $\boldsymbol{v}_{1,t} \cdot \boldsymbol{u} = \ell_{1,t} + \psi_t$, for all $t \in [n']$, remain the same as in H_{14}, the function-hiding security of IPFE ensures the indistinguishability between the hybrids H_{14} and H_{15}. This completes the security analysis as H_{15} is the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal},1-\mathsf{extFE}}(1^\lambda)$. $\square$

B Unbounded-Key One-Slot Extended FE for Attribute-Weighted Sums

In this section, we present a public-key one-slot FE scheme $\Pi^{\mathsf{ubd}}_{\mathsf{extOne}}$ for an extended attribute-weighted sum functionality. This scheme is proven adaptively simulation secure against one ciphertext query and an arbitrary polynomial number of secret key queries both before and after the ciphertext query. We describe the construction for any fixed value of the security parameter λ and suppress the appearance of λ for simplicity of notations. Let $(\mathsf{Garble}, \mathsf{Eval})$ be a special piecewise secure AKGS for a function class $\mathcal{F}^{(n,n')}_{\mathsf{ABP}}$, $\mathsf{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ a tuple of pairing groups of prime order p such that MDDH_k holds in $\mathbb{G}_2$, and ($\mathsf{IPFE.Setup}.\mathsf{IPFE.KeyGen}$, $\mathsf{IPFE.Enc}$, $\mathsf{IPFE.Dec}$) a slotted IPFE based on G. We construct an FE scheme for attribute-weighted sums with the message space $\mathbb{M} = \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$.

$\mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$: Defines the following index sets as follows

$$S_{\mathsf{pub}} = \left\{\{\mathsf{const}^{(\iota)}\}_{\iota\in[k]}, \{\mathsf{coef}_i^{(\iota)}\}_{\iota\in[k],i\in[n]}, \{\mathsf{extnd}_\kappa^{(\iota)}\}_{\iota,\kappa\in[k]}\right\},\ \widehat{S}_{\mathsf{pub}} = \{\widehat{\mathsf{const}}^{(\iota)}, \widehat{\mathsf{coef}}^{(\iota)}\}_{\iota\in[k]}$$
$$S_{\mathsf{priv}} = \left\{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \{\mathsf{extnd}_{\kappa,1}, \mathsf{extnd}_{\kappa,2}, \mathsf{extnd}_\kappa\}_{\kappa\in[k]}, \{\mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}_{\tau\in[n']}\right\},$$
$$\widehat{S}_{\mathsf{priv}} = \{\widehat{\mathsf{const}}_1, \widehat{\mathsf{coef}}_1, \widehat{\mathsf{const}}_2, \widehat{\mathsf{coef}}_2, \widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}$$

It generates two pair of IPFE keys $(\mathsf{IPFE.MSK}, \mathsf{IPFE.MPK}) \leftarrow \mathsf{IPFE.Setup}(S_{\mathsf{pub}}, S_{\mathsf{priv}})$ and $(\widehat{\mathsf{IPFE.MSK}}, \widehat{\mathsf{IPFE.MPK}}) \leftarrow \mathsf{IPFE.Setup}(\widehat{S}_{\mathsf{pub}}, \widehat{S}_{\mathsf{priv}})$. Finally, it returns the master secret-key as $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$ and master public-key as $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widehat{\mathsf{IPFE.MPK}})$.

$\mathsf{KeyGen}(\mathsf{MSK}, (\boldsymbol{f}, \boldsymbol{y}))$: Let $f = (f_1, \ldots, f_{n'}) \in \mathcal{F}^{(n,n')}_{\mathsf{ABP}}$ and $\boldsymbol{y} \in \mathbb{Z}_p^k$. It samples integers $\nu_t \leftarrow \mathbb{Z}_p$ and vectors $\boldsymbol{\alpha}, \boldsymbol{\beta}_t \leftarrow \mathbb{Z}_p^k$ for $t \in [n']$ such that

$$\sum_{t\in[n']} \nu_t = 1 \text{ and } \sum_{t\in[n']} \boldsymbol{\beta}_t[\iota] = 0 \bmod p \text{ for all } \iota \in [k]$$

Next, sample independent random vectors $\boldsymbol{r}_t^{(\iota)} \leftarrow \mathbb{Z}_p^m$ and computes

$$(\boldsymbol{\ell}^{(\iota)}_{1,t}, \ldots, \boldsymbol{\ell}^{(\iota)}_{m,t}, \boldsymbol{\ell}^{(\iota)}_{m+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}[\iota]\boldsymbol{z}[t]f_t(\boldsymbol{x}) + \boldsymbol{\beta}_t[\iota]; \boldsymbol{r}_t^{(\iota)})$$

for all $\iota \in [k], t \in [n']$. Here, we make use of the instantiation of the AKGS described in Section 3.6. From the description of that AKGS instantiation, we note that the $(m+1)$-th label function $\boldsymbol{\ell}_{m+1,t}^{(\iota)}$ would be of the form $\boldsymbol{\ell}_{m+1,t}^{(\iota)} = \boldsymbol{\alpha}[\iota]\boldsymbol{z}[t] - \boldsymbol{r}_t^{(\iota)}[m]$ where $\boldsymbol{\alpha}[\iota]$ is a constant. Also all the label functions $\boldsymbol{\ell}_{1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{m,t}^{(\iota)}$ involve only the variables $\boldsymbol{x}$ and not the variable $\boldsymbol{z}[t]$. Next, for all $j \in [2, m]$ and $t \in [n']$, it defines the vectors $\boldsymbol{v}_{j,t}$ corresponding to the label functions $\boldsymbol{\ell}_{j,t}$ obtained from the partial garbling above and the vector $\boldsymbol{y}$ as

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	$\mathsf{extnd}_\kappa^{(\iota)}$	S_{priv}
$\boldsymbol{v}$	$\boldsymbol{\alpha}[\iota]$	0	0	0
$\boldsymbol{v}_{1,t}$	$\boldsymbol{\ell}_{j,t}^{(\iota)}[\mathsf{const}]$	$\boldsymbol{\ell}_{j,t}^{(\iota)}[\mathsf{coef}_i]$	$\boldsymbol{\alpha}[\iota]\boldsymbol{y}[\kappa]\nu_t$	0
$\boldsymbol{v}_{j,t}$	$\boldsymbol{\ell}_{j,t}^{(\iota)}[\mathsf{const}]$	$\boldsymbol{\ell}_{j,t}^{(\iota)}[\mathsf{coef}_i]$	0	0

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$	$\widehat{S}_{\mathsf{priv}}$
$\boldsymbol{v}_{m+1,t}$	$\boldsymbol{r}_t^{(\iota)}[m]$	$\boldsymbol{\alpha}[\iota]$	0

It generates the secret-keys as

$$\begin{aligned}
\mathsf{IPFE.SK} &\leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}]\!]_2)\\
\mathsf{IPFE.SK}_{j,t} &\leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{j,t}]\!]_2) \text{ for } j \in [m], t \in [n']\\
\widehat{\mathsf{IPFE.SK}}_{m+1,t} &\leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{m+1,t}]\!]_2) \text{ for } t \in [n']
\end{aligned}$$

Finally, it returns $\mathsf{SK}_{f,\boldsymbol{y}} = (\mathsf{IPFE.SK}, \{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ and $(f, \boldsymbol{y})$.

Enc(MPK, $(\boldsymbol{x}, \boldsymbol{z}||\boldsymbol{w}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$): It samples a random vector $\boldsymbol{s} \leftarrow \mathbb{Z}_p^k$ and sets the vectors

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	$\mathsf{extnd}_\kappa^{(\iota)}$
$\boldsymbol{u}$	$\boldsymbol{s}[\iota]$	$\boldsymbol{s}[\iota]\boldsymbol{x}[i]$	$\boldsymbol{s}[\iota]\boldsymbol{w}[\kappa]$

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$
$\boldsymbol{h}_t$	$-\boldsymbol{s}[\iota]$	$\boldsymbol{s}[\iota]\boldsymbol{z}[t]$

for all $t \in [n']$. It encrypts the vectors as

$$\begin{aligned}
\mathsf{IPFE.CT} &\leftarrow \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)\\
\widehat{\mathsf{IPFE.CT}}_t &\leftarrow \mathsf{IPFE.SlotEnc}(\widehat{\mathsf{IPFE.MPK}}, [\![\boldsymbol{h}_t]\!]_1) \text{ for } t \in [n']
\end{aligned}$$

and returns the ciphertext as $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$ and $\boldsymbol{x}$.

$\mathsf{Dec}((\mathsf{SK}_{f,y}, f), (\mathsf{CT}, x))$: It parses the secret-key as $\mathsf{SK}_f = (\mathsf{IPFE.SK}, \{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$ and the ciphertext as $\mathsf{CT}_{x,z} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$. It uses the decryption algorithm of IPFE to compute

$$
\begin{aligned}
[\![\rho]\!]_T &\leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}, \mathsf{IPFE.CT})\\
[\![\ell_{1,t} + \psi_t]\!]_T &\leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{1,t}, \mathsf{IPFE.CT})\\
[\![\ell_{j,t}]\!]_T &\leftarrow \mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{j,t}, \mathsf{IPFE.CT}) \text{ for } j \in [2, m], t \in [n']\\
[\![\ell_{m+1,t}]\!]_T &\leftarrow \mathsf{IPFE.Dec}(\widehat{\mathsf{IPFE.SK}}_{m+1,t}, \widehat{\mathsf{IPFE.CT}}_t) \text{ for } t \in [n']
\end{aligned}
$$

where $\psi_t = \sum_{\iota=1}^{k} \boldsymbol{\alpha}[\iota]\boldsymbol{s}[\iota] \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w} = \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}$. Next, it utilizes the evaluation procedure of AKGS and obtain a combined value

$$[\![\zeta]\!]_T = \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T).$$

Finally, it returns a value $[\![\mu]\!]_T = [\![\zeta]\!]_T \cdot [\![\rho]\!]_T^{-1} \in \mathbb{G}_T$.

Correctness: First, the IPFE correctness implies $\mathsf{IPFE.Dec}(\mathsf{IPFE.SK}_{1,t}, \mathsf{IPFE.CT}) = [\![\ell_{1,t} + \psi_t]\!]$ where $\psi_t = \sum_{\iota=1}^{k} \boldsymbol{\alpha}[\iota]\boldsymbol{s}[\iota] \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w} = \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}$. Next, by the correctness of IPFE, AKGS we have

$$
\begin{aligned}
&\mathsf{Eval}(f_t, \boldsymbol{x}, \ell_{1,t} + \psi_t, \ldots, \ell_{m+1,t})\\
&= \mathsf{Eval}(f_t, \boldsymbol{x}, \ell_{1,t}, \ldots, \ell_{m+1,t}) + \mathsf{Eval}(f_t, \boldsymbol{x}, \psi_t, 0, \ldots, 0)\\
&= \mathsf{Eval}(f_t, \boldsymbol{x}, \ell_{1,t}, \ldots, \ell_{m+1,t}) + \psi_t\\
&= \sum_{\iota=1}^{k} (\boldsymbol{\alpha}[\iota]\boldsymbol{s}[\iota] \cdot \boldsymbol{z}[t] f_t(\boldsymbol{x}) + \boldsymbol{\beta}_t[\iota]\boldsymbol{s}[\iota]) + \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}\\
&= \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot (\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}) + \boldsymbol{\beta}_t \cdot \boldsymbol{s}
\end{aligned}
$$

The first equality follows from the linearity of Eval algorithm. Therefore, multiplying all the evaluated values we have

$$
\begin{aligned}
[\![\zeta]\!]_T &= \prod_{t\in[n']} \mathsf{Eval}(f_t, \boldsymbol{x}, [\![\ell_{1,t} + \psi_t]\!]_T, \ldots, [\![\ell_{m+1,t}]\!]_T)\\
&= [\![\sum_{t=1}^{n'} \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot (\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \nu_t \cdot \boldsymbol{y}^\top \boldsymbol{w}) + \boldsymbol{\beta}_t \cdot \boldsymbol{s}]\!]_T\\
&= [\![\boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot (f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w})]\!]_T
\end{aligned}
$$

where the last equality follows from the fact that $\sum_{t\in n'} \nu_t = 1$ and $\sum_{t\in[n']} \boldsymbol{\beta}_t[\iota] = 0$ for all $\iota \in [k]$. Also, by the correctness of IPFE we see that $[\![\rho]\!]_T = [\![\boldsymbol{\alpha} \cdot \boldsymbol{s}]\!]_T$ and hence $[\![\mu]\!]_T = [\![f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w}]\!]_T$.

B.1 Security Analysis

Theorem 8 *The extended one slot FE scheme $\Pi^{\mathsf{ubd}}_{\mathsf{extOne}}$ for attribute-weighted sum is adaptively simulation-secure assuming the* AKGS *is piecewise-secure as per Definition 7, the* MDDH_k *assumption holds in group $\mathbb{G}_2$, and the slotted* IPFE *is function hiding as per Definition 5.*

The Simulator We describe the simulator for the extended one slot FE scheme $\Pi^{\mathsf{ubd}}_{\mathsf{extOne}}$. The simulated setup algorithm is the same setup of the original scheme. Let $(\mathsf{MSK}, \mathsf{MPK}) \leftarrow \mathsf{Setup}^*(1^\lambda, 1^n, 1^{n'}) = \mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$ where $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$ and $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widehat{\mathsf{IPFE.MPK}})$.

$\mathsf{KeyGen}^*_0(\mathsf{MSK}, (\boldsymbol{f_q}, \boldsymbol{y_q}))$: On input MSK, a function $f_q = (f_{q,1}, \ldots f_{q,n'}) \in \mathcal{F}^{(n,n')}_{\mathsf{ABP}}$ and a vector $\boldsymbol{y}_q \in \mathbb{Z}_p^k$ the simulator proceeds as follows:

Setting Public Positions: The public positions are set as in the original scheme.

1. It first samples $\boldsymbol{\beta}_{q,t} = (\boldsymbol{\beta}_{q,t}[1], \ldots, \boldsymbol{\beta}_{q,t}[k]) \leftarrow \mathbb{Z}_p^k, \nu_{q,t} \leftarrow \mathbb{Z}_p$ for $t \in [n']$, and $\boldsymbol{r}^{(\iota)}_{q,t} = (\boldsymbol{r}^{(\iota)}_{q,t}[1], \ldots, \boldsymbol{r}^{(\iota)}_{q,t}[m_q]) \leftarrow \mathbb{Z}_p^{m_q}$ where it holds that
$$\sum_{t\in[n']} \boldsymbol{\beta}_{q,t}[\iota] = 0 \text{ for all } \iota \in [k] \text{ and } \sum_{t\in[n']} \nu_{q,t} = 1.$$
2. Next, it computes the coefficient vectors for the label functions as
$$(\boldsymbol{\ell}^{(\iota)}_{q,1,t}, \ldots, \boldsymbol{\ell}^{(\iota)}_{q,m_q,t}, \boldsymbol{\ell}^{(\iota)}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t]f_{q,t}(\boldsymbol{x}^*) + \boldsymbol{\beta}_{q,t}[\iota]; \boldsymbol{r}^{(\iota)}_{q,t})$$
for all $\iota \in [k], t \in [n']$. From the description of AKGS, we note that the $(m_q + 1)$-th label function $\boldsymbol{\ell}^{(\iota)}_{q,m_q+1,t}$ would be of the form $\boldsymbol{\ell}^{(\iota)}_{q,m_q+1,t} = \boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t] - \boldsymbol{r}^{(\iota)}_{q,t}[m_q]$.
3. It picks $\boldsymbol{\alpha}_q \leftarrow \mathbb{Z}_p^k$ and sets the public positions at the indexes in $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ of following vectors

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}^{(\iota)}_i$	$\mathsf{extnd}^{(\iota)}_\kappa$
$\boldsymbol{v}_q$	$\boldsymbol{\alpha}_q[\iota]$	0	0
$\boldsymbol{v}_{q,1,t}$	$\boldsymbol{\ell}^{(\iota)}_{q,1,t}[\mathsf{const}]$	$\boldsymbol{\ell}^{(\iota)}_{q,1,t}[\mathsf{coef}_i]$	$\boldsymbol{\alpha}_q[\iota]\boldsymbol{y}_q[\kappa]\nu_{q,t}$
$\boldsymbol{v}_{q,j,t}$	$\boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{const}]$	$\boldsymbol{\ell}^{(\iota)}_{q,j,t}[\mathsf{coef}_i]$	0

for all $j \in [2, m_q]$ and $t \in [n']$. It also sets the following vectors for all $t \in [n']$.

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$
$\boldsymbol{v}_{q,m_q+1,t}$	$\boldsymbol{r}^{(\iota)}_{q,t}[m_q]$	$\boldsymbol{\alpha}_q[\iota]$

Setting Private Positions: It now fills the private indices as follows.

4. It samples $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ for $t \in [n']$ satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0$.
5. Next, it picks $\widetilde{\boldsymbol{r}}_{q,t} \leftarrow \mathbb{Z}_p^{m_q}$ and computes the coefficient vectors for the label functions as
$$(\widetilde{\boldsymbol{\ell}}_{q,1,t}, \ldots, \widetilde{\boldsymbol{\ell}}_{q,m_q,t}, \widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\widetilde{\alpha}_q\boldsymbol{z}^*[t]f_{q,t}(\boldsymbol{x}^*) + \widetilde{\beta}_{q,t}; \widetilde{\boldsymbol{r}}_{q,t}).$$
for all $t \in [n']$. From the description of AKGS, we note that the $(m_q + 1)$-th label function $\widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}$ would be of the form $\widetilde{\boldsymbol{\ell}}_{q,m_q+1,t} = \widetilde{\alpha}_q\boldsymbol{z}^*[t] - \widetilde{\boldsymbol{r}}_{q,t}[m_q]$.
6. Now, it fills the private positions at the indexes in $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ as follows

vector	const	coef_i	$\mathsf{extnd}_{\kappa,1}$	$\mathsf{extnd}_{\kappa,2}$	extnd_κ	sim_τ	sim^*_τ
$\boldsymbol{v}_q$	$\widetilde{\alpha}_q$	0	0	0	0	0	0
$\boldsymbol{v}_{q,1,t}$	$\widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}]$	$\widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i]$	0	$\widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}$	0	0	0
$\boldsymbol{v}_{q,j,t}$	$\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]$	$\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]$	0	0	0	0	0

for all $j \in [2, m_q]$ and $t \in [n']$; and for all $t \in [n']$

vector	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{q,m_q+1,t}$	0	0	$\widetilde{\boldsymbol{r}}_{q,t}[m_q]$	$\widetilde{\boldsymbol{\alpha}}_q$	0	0	0

7. It generates the IPFE secret-keys

$$\mathsf{IPFE.SK}_q \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_q]\!]_2)$$
$$\mathsf{IPFE.SK}_{q,j,t} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{q,j,t}]\!]_2) \text{ for } j \in [m_q], t \in [n']$$
$$\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} \leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{q,m_q+1,t}]\!]_2) \text{ for } t \in [n']$$

8. Finally, it returns $\mathsf{SK}_{f_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$.

Let Q_{pre} be the total number of secret-key queries made before the challenge query.

$\mathsf{Enc}^*(\mathsf{MPK}, \mathsf{MSK}, \boldsymbol{x}^*, \mathcal{V})$: On input $\mathsf{MPK}, \mathsf{MSK}$, a vector $\boldsymbol{x}^* \in \mathbb{Z}_p^n$ and a set $\mathcal{V} = \{(f_q, f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) : q \in [Q_{\mathsf{pre}}]\}$ the simulator executes the following steps:

1. It samples a dummy vector $(\boldsymbol{d}_1 || \boldsymbol{d}_2) \in \mathbb{Z}_p^{n'+k}$ from the set
$$\mathcal{D} = \{(\boldsymbol{d}_1||\boldsymbol{d}_2) \in \mathbb{Z}_p^{n'+k} : f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2 = \mu_q \text{ for all } q \in [Q_{\mathsf{pre}}]\}$$
where $\mu_q = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*$. Since the inner product functionality is *pre-image sampleable*, there exists an efficient algorithm (proposed by O'Neill [O'N10]) which on input $(f_{q,1}(\boldsymbol{x}^*), \ldots, f_{q,n'}(\boldsymbol{x}^*), \boldsymbol{y}_q, \mu_q)$ samples a vector $(\boldsymbol{d}_1||\boldsymbol{d}_2) \in \mathbb{Z}_p^{n'+k}$ such that $(f_{q,1}(\boldsymbol{x}^*), \ldots, f_{q,n'}(\boldsymbol{x}^*)) \cdot \boldsymbol{d}_1 + \boldsymbol{y}_q \cdot \boldsymbol{d}_2 = f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2 = \mu_q$ for all $q \in [Q_{\mathsf{pre}}]$.
2. Next, it sets the following vectors:

vector	$\mathsf{const}^{(\iota)}$	$\mathsf{coef}_i^{(\iota)}$	$\mathsf{extnd}_\kappa^{(\iota)}$	const	coef_i	$\mathsf{extnd}_{\kappa,1}$	$\mathsf{extnd}_{\kappa,2}$	extnd_κ	sim_τ	sim^*_τ
$\boldsymbol{u}$	0	0	0	1	$\boldsymbol{x}^*[i]$	0	$\boldsymbol{d}_2[\kappa]$	0	0	0

and for all $t \in [n']$

vector	$\widehat{\mathsf{const}}^{(\iota)}$	$\widehat{\mathsf{coef}}^{(\iota)}$	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{h}_t$	0	0	1	0	-1	$\boldsymbol{d}_1[t]$	0	0	0

3. It encrypts the vectors as
$$\mathsf{IPFE.CT} \leftarrow \mathsf{IPFE.Enc}(\mathsf{IPFE.MPK}, [\![\boldsymbol{u}]\!]_1)$$
$$\widehat{\mathsf{IPFE.CT}}_t \leftarrow \mathsf{IPFE.Enc}(\widehat{\mathsf{IPFE.MPK}}, [\![\boldsymbol{h}_t]\!]_1) \text{ for } t \in [n']$$
4. It returns the ciphertext as $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

$\mathbf{KeyGen}_1^*(\mathbf{MSK}^*, \boldsymbol{x}^*, (\boldsymbol{f_q}, \boldsymbol{y_q}), \boldsymbol{f_q}(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y_q}^\top \boldsymbol{w}^*)$: On input MSK^*, $\boldsymbol{x}^* \in \mathbb{Z}_p^n$, a function $f_q = (f_{q,1}, \dots, f_{q,n'}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')}$, a vector $\boldsymbol{y}_q \in \mathbb{Z}_p^k$ for $q \in [Q_{\mathsf{pre}}+1, Q]$ and $(f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) \in \mathbb{Z}_p$ the simulator proceeds as follows:

Setting Public Positions:

1. The simulator sets the public positions at the indexes in $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ analogous to $\mathsf{KeyGen}_0^*(\mathsf{MSK}^*, (f_q, \boldsymbol{y}_q))$.

Setting Private Positions:

2. First, it samples a random element $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$, for $t \in [n']$, satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0$ and then runs the simulator of the AKGS to obtain

$$(\widehat{\ell}_{q,1,1}, \dots, \widehat{\ell}_{q,m_q,1}, \widehat{\ell}_{q,m_q+1,1}) \leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, \widetilde{\alpha}_q \cdot (f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) + \widetilde{\beta}_{q,1})$$
$$(\widehat{\ell}_{q,1,t}, \dots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \widetilde{\beta}_{q,t}) \quad \text{for } 1 < t \le n'.$$

3. Next, it fills the private positions at the indices in $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ as follows

vector	const	coef_i	$\mathsf{extnd}_{\kappa,1}$	$\mathsf{extnd}_{\kappa,2}$	extnd_κ	sim_τ	sim_τ^*
$\boldsymbol{v}_q$	$\widetilde{\alpha}_q$	0	0	0	0	0	0
$\boldsymbol{v}_{q,j,t}$	$\widehat{\ell}_{q,j,t}$	0	0	0	0	0	0

for all $j \in [m_q]$ and $t \in [n']$; and

vector	$\widehat{\mathsf{const}}_1$	$\widehat{\mathsf{coef}}_1$	$\widehat{\mathsf{const}}_2$	$\widehat{\mathsf{coef}}_2$	$\widehat{\mathsf{const}}$	$\widehat{\mathsf{coef}}$	$\widehat{\mathsf{sim}}^*$
$\boldsymbol{v}_{q,m_q+1,t}$	$\widehat{\ell}_{q,m_q+1,t}$	0	0	0	0	0	0

for all $t \in [n']$.

4. It generates the IPFE secret-keys

$$\mathsf{IPFE.SK}_q \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_q]\!]_2)$$
$$\mathsf{IPFE.SK}_{q,j,t} \leftarrow \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{v}_{q,j,t}]\!]_2) \text{ for } j \in [m_q], t \in [n']$$
$$\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} \leftarrow \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{v}_{q,m_q+1,t}]\!]_2) \text{ for } t \in [n']$$

5. It outputs $\mathsf{SK}_{f_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$.

Hybrids and Reductions

Proof. We use a sequence of hybrid experiments to establish the indistinguishability between the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real,extFE}}(1^\lambda)$ and the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,extFE}}(1^\lambda)$ where $\mathcal{A}$ is any PPT adversary. In each experiment, $\mathcal{A}$ can query a polynomial number of secret-key queries for pairs $(f, \boldsymbol{y}) \in \mathcal{F}_{\mathsf{ABP}}^{(n,n')} \times \mathbb{Z}_p^k$, both before and after submitting the challenge message $(\boldsymbol{x}^*, \boldsymbol{z}^* || \boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'+k}$. Let Q be the total number of secret-key queries and Q_{pre} $(\le Q)$ be the number of secret-keys queried before submitting the challenge message. We denote the q-th secret-key by $\mathsf{SK}_{f_q,\boldsymbol{y}_q}$ corresponding to a function f_q and a vector $\boldsymbol{y}_q$. For the ease of presentation, we write the vector elements sitting in the public slots $S_{\mathsf{pub}}, \widehat{S}_{\mathsf{pub}}$ in *blue* color and the vector elements sitting in the private slots $S_{\mathsf{priv}}, \widehat{S}_{\mathsf{priv}}$ in *red* color. More precisely, we do this so that while describing the hybrid games, we sometimes omit the public parts of the vectors and write down only the private parts when the changes occur only in the private parts. Now, we describe the hybrids as follows:

Hybrid H_0: : This is the real experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real},\mathsf{extFE}}(1^\lambda)$ defined in Definition 4 (with single slot, i.e., $N = 1$). For each $q \in [Q]$, the q-th secret-key $\mathsf{SK}_{f_q,\boldsymbol{y}_q} = (\mathsf{IPFE.SK}_q, \{\mathsf{IPFE.SK}_{q,j,t}\}_{j\in[m_q],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t}\}_{t\in[n']})$ is computed using the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ given by

$$\begin{aligned}
\boldsymbol{v}_q &= (\quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad 0, \quad 0,0,0,0,0,0,0),\\
\boldsymbol{v}_{q,1,t} &= (\boldsymbol{\ell}_{q,1,t}^{(\iota)}[\mathsf{const}], \boldsymbol{\ell}_{q,1,t}^{(\iota)}[\mathsf{coef}_i], \boldsymbol{\alpha}_q[\iota]\boldsymbol{y}_q[\kappa]\nu_{q,t}, 0,0,0,0,0,0,0),\\
\boldsymbol{v}_{q,j,t} &= (\boldsymbol{\ell}_{q,j,t}^{(\iota)}[\mathsf{const}], \boldsymbol{\ell}_{q,j,t}^{(\iota)}[\mathsf{coef}_i], \quad 0, \quad 0,0,0,0,0,0,0),\\
\boldsymbol{v}_{q,m_q+1,t} &= (\quad \boldsymbol{r}_{q,t}^{(\iota)}[m_q], \quad \boldsymbol{\alpha}_q[\iota], \quad 0, \quad 0,0,0,0,0,0 \;)
\end{aligned}$$

for $j \in [2, m_q]$ and $t \in [n']$. Note that $\boldsymbol{\alpha}_q$ and $\boldsymbol{r}_{q,t}^{(\iota)}$ are random vectors sampled from $\mathbb{Z}_p^k$ and $\mathbb{Z}_p^{m_q}$ respectively. The integers $\nu_{q,t}$ for $t \in [n']$ is picked randomly from $\mathbb{Z}_p$ such that $\sum_{t\in[n']} \nu_{q,t} = 1$. For all $t \in [n']$, the garblings are computed as

$$(\boldsymbol{\ell}_{q,1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{q,m_q,t}^{(\iota)}, \boldsymbol{\ell}_{q,m_q+1,t}^{(\iota)}) \leftarrow \mathsf{Garble}(\boldsymbol{\alpha}_q[\iota]\boldsymbol{z}^*[t]f_{q,t}(\boldsymbol{x}^*) + \boldsymbol{\beta}_{q,t}[\iota]; \boldsymbol{r}_{q,t}^{(\iota)})$$

where $f_q = (f_{q,1}, \ldots, f_{q,n'})$ and $\boldsymbol{\beta}_{q,t} \leftarrow \mathbb{Z}_p^k$ with $\sum_{t\in[n']} \boldsymbol{\beta}_{q,t}[\iota] = 0 \; \forall \iota \in [k]$. The challenge ciphertext $\mathsf{CT}^* = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$ corresponds to the challenge vectors $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$ is computed using the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ given by

$$\begin{aligned}
\boldsymbol{u} &= (\; \boldsymbol{s}[\iota], \; \boldsymbol{s}[\iota]\boldsymbol{x}^*[i], \boldsymbol{s}[\iota]\boldsymbol{w}^*[\kappa], \perp,\perp,\perp,\perp,\perp,\perp,\perp),\\
\boldsymbol{h}_t &= (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t], \quad \perp, \quad \perp,\perp,\perp,\perp,\perp,\perp \;)
\end{aligned}$$

for $t \in [n']$ and $\boldsymbol{s} \leftarrow \mathbb{Z}_p^k$. Note that, in real experiment CT^* is computed using $\mathsf{IPFE.SlotEnc}$ and therefore the elements sitting at the indices in S_{priv} are set as $\perp$ for the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$.

Hybrid H_1: It is exactly the same as hybrid H_0 except the fact that instead of using $\mathsf{IPFE.SlotEnc}$, here the challenge ciphertext CT^* is generated applying $\mathsf{IPFE.Enc}$ which uses $\mathsf{MSK} = (\mathsf{IPFE.MSK}, \widehat{\mathsf{IPFE.MSK}})$ to encrypt the vectors. We indicate this change by changing the private positions of $\boldsymbol{u}$ and $\boldsymbol{h}_t$ from $\perp$ to 0. Thus the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ become

$$\begin{aligned}
\boldsymbol{u} &= (\; \boldsymbol{s}[\iota], \; \boldsymbol{s}[\iota]\boldsymbol{x}^*[i], \boldsymbol{s}[\iota]\boldsymbol{w}^*[\kappa], \boxed{0},\boxed{0},\boxed{0},\boxed{0},\boxed{0},\boxed{0},\boxed{0}),\\
\boldsymbol{h}_t &= (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t], \quad \boxed{0}, \quad \boxed{0},\boxed{0},\boxed{0},\boxed{0},\boxed{0},\boxed{0} \;).
\end{aligned}$$

The slot-mode correctness of IPFE guarantees that the two hybrids H_0 and H_1 are identically distributed.

Hybrid H_2: This hybrid is similar to H_1 except that in the private slots of the vectors $\boldsymbol{v}_{q,j,t}$ we put a garbling that linearly combines k garblings (of the public slots) with weight vector $\boldsymbol{s} \in \mathbb{Z}_p^k$ and in the private slots of the vector $\boldsymbol{v}_q$ we use a single random element combining the weight vector $\boldsymbol{s}$. Accordingly, we modify the challenge ciphertext CT^* by omitting the weight vector $\boldsymbol{s}$ and setting the public slots of the vectors $\boldsymbol{u}, \boldsymbol{h}_t$ to zero so that the inner products computed at the time of decryption remains the same in the previous hybrids.

In H_1, the public slots of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ are occupied by vectors $\boldsymbol{\alpha}_q \in \mathbb{Z}_p^k, \nu_{q,t} \in \mathbb{Z}_p$ for $t \in [n']$ and the garblings $\boldsymbol{\ell}_{q,j,t}^{(\iota)}$ computed using randomness $\boldsymbol{r}_{q,t}^{(\iota)} \in \mathbb{Z}_p^{m_q}$. In the public slots of the vectors $\boldsymbol{u}, \boldsymbol{h}_t$, we use $(\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{x}^*[i]), (-\boldsymbol{s}[\iota], \boldsymbol{s}[\iota]\boldsymbol{z}^*[t])$ respectively. Therefore, at the time of decryption we recover $[\![\rho_q]\!]_T, [\![\ell_{q,j,t}]\!]_T$ such that

$$\begin{aligned}
\rho_q &= \boldsymbol{\alpha}_q \cdot \boldsymbol{s} = \overline{\alpha}_q \text{ (say)},\\
\ell_{q,1,t} &= (\boldsymbol{\ell}^{(1)}_{q,1,t}, \ldots, \boldsymbol{\ell}^{(k)}_{q,1,t}) \cdot (\boldsymbol{s}[1](1,\boldsymbol{x}^*), \ldots, \boldsymbol{s}[k](1,\boldsymbol{x}^*)) + \boldsymbol{\alpha} \cdot \boldsymbol{s} \cdot \boldsymbol{y}^\top \boldsymbol{w} \cdot \nu_{q,t}\\
&= (\boldsymbol{s}[1]\boldsymbol{\ell}^{(1)}_{q,1,t}, \ldots, \boldsymbol{s}[k]\boldsymbol{\ell}^{(k)}_{q,1,t}) \cdot ((1,\boldsymbol{x}^*), \ldots, (1,\boldsymbol{x}^*)) + \overline{\alpha}_q \cdot \boldsymbol{y}^\top \boldsymbol{w} \cdot \nu_{q,t}\\
&= \overline{\boldsymbol{\ell}}_{q,1,t} \cdot (1,\boldsymbol{x}^*) + \overline{\alpha}_q \cdot \boldsymbol{y}^\top \boldsymbol{w} \cdot \nu_{q,t}\\
\ell_{q,j,t} &= (\boldsymbol{\ell}^{(1)}_{q,j,t}, \ldots, \boldsymbol{\ell}^{(k)}_{q,j,t}) \cdot (\boldsymbol{s}[1](1,\boldsymbol{x}^*), \ldots, \boldsymbol{s}[k](1,\boldsymbol{x}^*))\\
&= \overline{\boldsymbol{\ell}}_{q,j,t} \cdot (1,\boldsymbol{x}^*)
\end{aligned}$$

where $\overline{\boldsymbol{\ell}}_{q,j,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{\ell}^{(\iota)}_{q,j,t}$ for all $j \in [2, m_q]$ and $t \in [n']$. Similarly, the $m_q + 1$-the garbling returns

$$\begin{aligned}
\ell_{q,m_q+1,t} &= ((\boldsymbol{r}^{(1)}_{q,t}[m_q], \boldsymbol{\alpha}_q[1]), \ldots, (\boldsymbol{r}^{(k)}_{q,t}[m_q], \boldsymbol{\alpha}_q[k])) \cdot (\boldsymbol{s}[1](-1, \boldsymbol{z}^*[t]), \ldots, \boldsymbol{s}[k](-1, \boldsymbol{z}^*[t]))\\
&= (\boldsymbol{s}[1](\boldsymbol{r}^{(1)}_{q,t}[m_q], \boldsymbol{\alpha}_q[1]), \ldots, \boldsymbol{s}[k](\boldsymbol{r}^{(k)}_{q,t}[m_q], \boldsymbol{\alpha}_q[k])) \cdot ((-1, \boldsymbol{z}^*[t]), \ldots, (-1, \boldsymbol{z}^*[t]))\\
&= (\overline{\boldsymbol{r}}_{q,t}[m_q], \overline{\alpha}_q) \cdot (-1, \boldsymbol{z}^*[t])
\end{aligned}$$

where $\overline{\boldsymbol{r}}_{q,t}[m_q] = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{r}^{(\iota)}_{q,t}[m_q]$. In H_2, we use $\overline{\alpha}_q, \overline{\boldsymbol{\ell}}_{q,j,t}$ and $\overline{\boldsymbol{r}}_{q,t}[m_q]$ in the private slots of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ as described below

$$\begin{aligned}
\boldsymbol{v}_q &= (\ \boxed{\overline{\alpha}_q},\ 0,\ 0,\ 0, 0, 0, 0\ ),\\
\boldsymbol{v}_{q,1,t} &= (\ \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]}, \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]}, \boxed{\overline{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}}, 0, 0, 0, 0\ ),\\
\boldsymbol{v}_{q,j,t} &= (\ \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]}, \boxed{\overline{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]},\ 0,\ 0, 0, 0, 0\ ) \quad \text{for } j \in [2, m_q],\\
\boldsymbol{v}_{q,m_q+1,t} &= (\ \boxed{\overline{\boldsymbol{r}}^{q,t}[m_q]}, \boxed{\overline{\alpha}_q},\ 0,\ 0, 0, 0, 0\ )
\end{aligned}$$

Since the weight vector $\boldsymbol{s}$ is not required to generate the challenge ciphertext CT^*, we omit using it in the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$. Moreover, the public slots of $\boldsymbol{u}$ and $\boldsymbol{h}_t$ are set to zero as the inner product is computed through the private slots only. We describe the changes below.

$$\begin{aligned}
\boldsymbol{u} &= (\ \boxed{0}, \boxed{0}, \boxed{0}, \boxed{1}, \boxed{\boldsymbol{x}^*[i]}, \boxed{\boldsymbol{w}^*[\kappa]}, 0, 0, 0, 0\ ),\\
\boldsymbol{h}_t &= (\ \boxed{0}, \boxed{0}, \boxed{-1}, \boxed{\boldsymbol{z}^*[t]}, 0, 0, 0, 0, 0\ )
\end{aligned}$$

Finally, we observe that the inner products $\boldsymbol{v}_q \cdot \boldsymbol{u}, \boldsymbol{v}_{q,j,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ remain the same as in H_1. Thus, the function hiding property of IPFE preserves the indistinguishability between the hybrids H_1 and H_2.

Note that, in this hybrid we pick $\boldsymbol{\alpha}_q, \boldsymbol{\beta}_{q,t}, \boldsymbol{s} \leftarrow \mathbb{Z}_p^k, \nu_{q,t} \leftarrow \mathbb{Z}_p$ and $\boldsymbol{r}^{(\iota)}_{q,t} \leftarrow \mathbb{Z}_p^{m_q}$ for all $t \in [n'], \iota \in [k]$ satisfying $\sum_{t\in[n']} \boldsymbol{\beta}_{q,t}[\iota] = 0$ for each $\iota \in [k]$ and $\sum_{t\in[n']} \nu_{q,t} = 1$. Then, the linearity of the Garble algorithm allows us to write

$$(\overline{\boldsymbol{\ell}}_{q,1,t}, \ldots, \overline{\boldsymbol{\ell}}_{q,m_q,t}, \overline{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\overline{\alpha}_q \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \overline{\beta}_{q,t}; \overline{\boldsymbol{r}}_{q,t})$$

where $\overline{\boldsymbol{\ell}}_{q,j,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{\ell}^{(\iota)}_{q,j,t}, \overline{\boldsymbol{r}}_{q,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{r}^{(\iota)}_{q,t}$ and $\overline{\beta}_{q,t} = \boldsymbol{\beta}_{q,t} \cdot \boldsymbol{s}$.

From the next hybrid onward the public slots of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ are unaltered for all $q \in [Q], j \in [k]$ and $t \in [n']$. Therefore, we only write the components sitting in the private slots of the vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q,j,t}$ assuming that the components of public slots are the same as in the real experiment. We denote the private slots of the vectors by $\boldsymbol{v}_q|_{S_{\mathsf{priv}}}, \boldsymbol{v}_{q,j,t}|_{S_{\mathsf{priv}}}$ and $\boldsymbol{v}_{q,m_q+1,t}|_{\widehat{S}_{\mathsf{priv}}}$.

Hybrid H_3: It is analogous to H_2 except the liner combinations $\overline{\alpha}_q, \overline{\boldsymbol{\ell}}_{q,j,t}, \overline{\boldsymbol{r}}_{q,t}$ in the private slots of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}, \boldsymbol{v}_{q,m_q+1,t}$ are replaced with freshly and independently generated random values and garblings $\widetilde{\alpha}_q, \widetilde{\boldsymbol{\ell}}_{q,j,t}, \widetilde{\boldsymbol{r}}_{q,t}$. More specifically, we sample random elements $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t} \leftarrow \mathbb{Z}_p$ for all $t \in [n']$ such that $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0$ and a vector $\overline{\boldsymbol{r}}_{q,t} \leftarrow \mathbb{Z}_p^{m_q}$. Then, the garblings are computed as

$$(\widetilde{\boldsymbol{\ell}}_{q,1,t}, \ldots, \widetilde{\boldsymbol{\ell}}_{q,m_q,t}, \widetilde{\boldsymbol{\ell}}_{q,m_q+1,t}) \leftarrow \mathsf{Garble}(\widetilde{\alpha}_q \boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\beta}_{q,t}; \widetilde{\boldsymbol{r}}_{q,t})$$

for all $t \in [n']$. The vectors involved in the computation of $\mathsf{SK}_{f_q,\boldsymbol{y}_q}$ are as follows:

$$\begin{aligned}
\boldsymbol{v}_q &= (\ \boxed{\widetilde{\alpha}_q},\ 0,\ 0,\ 0, 0, 0, 0\ ),\\
\boldsymbol{v}_{q,1,t} &= (\ \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]}, \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]}, \boxed{\widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}}, 0, 0, 0, 0\ ),\\
\boldsymbol{v}_{q,j,t} &= (\ \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}]}, \boxed{\widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i]},\ 0,\ 0, 0, 0, 0\ ) \quad \text{for } j \in [2, m_q],\\
\boldsymbol{v}_{q,m_q+1,t} &= (\ \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]},\ \boxed{\widetilde{\alpha}_q},\ 0,\ 0, 0, 0, 0\ )
\end{aligned}$$

Recall that in H_2, the following linear combinations

$$\overline{\alpha}_q = \boldsymbol{\alpha}_q \cdot \boldsymbol{s},\ \ \overline{\beta}_{q,t} = \boldsymbol{\beta}_{q,t} \cdot \boldsymbol{s},\ \ \overline{\boldsymbol{r}}_{q,t} = \sum_{\iota\in[k]} \boldsymbol{s}[\iota]\boldsymbol{r}_{q,t}^{(\iota)}$$

with a common weight vector $\boldsymbol{s}$ has been used to set $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$. On the other hand, in H_3 fresh and independent random elements $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t}, \widetilde{\boldsymbol{r}}_{q,t}$ are used to compute $\mathsf{SK}_{f_q,\boldsymbol{y}_q}$. Note that the elements of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ are only used in the exponent of the source group $\mathbb{G}_2$ while generating the IPFE secret-keys. Let us consider the matrix $\mathbf{A}_{q,t} = (\boldsymbol{\alpha}_q|\boldsymbol{\beta}_{q,t}|(\mathbf{R}_{q,t})^\top) \in \mathbb{Z}_p^{k\times(m_q+1)}$ where $\mathbf{R}_{q,t} = (\boldsymbol{r}_{q,t}^{(1)}|\ldots|\boldsymbol{r}_{q,t}^{(k)}) \in \mathbb{Z}_p^{m\times k}$. Since the matrix $\mathbf{A}_{q,t}$ is uniformly chosen from $\mathbb{Z}_p^{k\times(m_q+1)}$ and $\boldsymbol{s}$ is uniform over $\mathbb{Z}_p^k$, by the MDDH_k assumption in group $\mathbb{G}_2$ we have

$$\underbrace{([\![\mathbf{A}_{q,t}]\!]_2, [\![\mathbf{A}_{q,t}^\top \boldsymbol{s}]\!])}_{\text{in } \mathsf{H}_2} \approx \underbrace{([\![\mathbf{A}_{q,t}]\!]_2, [\![(\widetilde{\alpha}_q, \widetilde{\beta}_{q,t}, \widetilde{\boldsymbol{r}}_{q,t})]\!]_2)}_{\text{in } \mathsf{H}_3}$$

holds for all $q \in [Q]$ and $t \in [n']$. Hence, the two hybrids H_2 and H_3 are indistinguishable under the MDDH_k assumption.

We have completed the first phase of our security analysis as we see that the private slots of the vectors associated to secret-keys and the challenge ciphertext are now computed similar to our extended 1-FE scheme. From the next hybrid, we modify the vectors in such a way that all the pre-challenge secret-key queries decrypt the challenge ciphertext without using the slots of $\boldsymbol{u}$ and $\boldsymbol{h}_t$ where the challenge massage $(\boldsymbol{x}^*, \boldsymbol{z}^*||\boldsymbol{w}^*)$ are used.

Hybrid H_4: It proceeds similar to hybrid H_3 except we change the vectors $\boldsymbol{u}$ and $\boldsymbol{h}_t$ for all $t \in [n']$ which are used in the computation of the challenge ciphertext. After all the pre-challenge secret-key queries made by $\mathcal{A}$, a dummy vector $(\boldsymbol{d}_1||\boldsymbol{d}_2) \in \mathbb{Z}_p^{n'+k}$ is picked from the set

$$\mathcal{D} = \{(\boldsymbol{d}_1||\boldsymbol{d}_2) \in \mathbb{Z}_p^{n'+k} : f_q(\boldsymbol{x}^*)^\top \boldsymbol{d}_1 + \boldsymbol{y}_q^\top \boldsymbol{d}_2 = \mu_q \text{ for all } q \in [Q_{\mathsf{pre}}]\}$$

where $\mu_q = f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*$. The sampling procedure is as described in the algorithm $\mathsf{Enc}^*(\cdot)$. Then the vectors $\boldsymbol{u}, \boldsymbol{h}_t$ are defined as below.

$$\begin{aligned}
\boldsymbol{u} &= (\ 0, 0,\ 0,\ 1,\ \boldsymbol{x}^*[i], \boldsymbol{w}^*[\kappa], \boxed{\boldsymbol{d}_2[\kappa]}, \boxed{\boldsymbol{w}^*[\kappa]}, 0, 0\ ),\\
\boldsymbol{h}_t &= (\ 0, 0, -1, \boldsymbol{z}^*[t], \boxed{-1}, \boxed{\boldsymbol{d}_1[t]}, \boxed{-1}, \boxed{\boldsymbol{z}^*[t]},\ 0\ )
\end{aligned}$$

Note that, these changes in $\boldsymbol{u}$ and $\boldsymbol{h}_t$ have no effect in the final inner product values of $\boldsymbol{v}_q \cdot \boldsymbol{u}, \boldsymbol{v}_{q,j,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$. This is because the elements at the slots $(\mathsf{extnd}_{\kappa,2}, \mathsf{extnd}_\kappa)$ of the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ $\boldsymbol{h}_t$ and the elements at the slots $(\widehat{\mathsf{const}}_2, \widehat{\mathsf{coef}}_2, \widehat{\mathsf{const}}, \widehat{\mathsf{coef}})$ of the vector $\boldsymbol{v}_{q,m_q+1,t}$ (where the changes take place in $\boldsymbol{u}, \boldsymbol{h}_t$) are all zero. Therefore, by the function hiding property of IPFE the hybrids H_3 and H_4 remain indistinguishable to the adversary.

Hybrid $\mathsf{H}_{5,q}$ ($q \in [Q_{\mathsf{pre}}]$): It proceeds similar to H_4 except that for each $1 \le q' \le q$, we modify the vectors $\boldsymbol{v}_{q,1,t}$ and $\boldsymbol{v}_{q,m_q+1,t}$ as described below.

$$\begin{aligned}
\boldsymbol{v}_{q',1,t} &= (\,\widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], \boxed{0}, \boxed{\widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}}, 0, 0, 0\,) && \text{for } 1 \le q' < q,\\
\boldsymbol{v}_{q',1,t} &= (\,\widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, 0, 0, 0, 0\,) && \text{for } q < q' \le Q_{\mathsf{pre}},\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\,0, 0, \boxed{\widetilde{\boldsymbol{r}}_{q',t}[m_{q'}]}, \boxed{\widetilde{\alpha}_{q'}}, 0, 0, 0\,) && \text{for } 1 \le q' < q,\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\,\widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \widetilde{\alpha}_{q'}, 0, 0, 0, 0, 0\,) && \text{for } q < q' \le Q_{\mathsf{pre}}
\end{aligned}$$

Note that, the post-challenge secret-key queries are still answered according to H_4. Observe that $\mathsf{H}_{5,0}$ coincides with H_4. We will prove that $\mathsf{H}_{5,(q-1)}$ and $\mathsf{H}_{5,q}$ are indistinguishable via the following sequence of sub-hybrids, namely $\{\mathsf{H}_{5,q,1}, \mathsf{H}_{5,q,2}, \mathsf{H}_{5,q,3}\}$.

Hybrid $\mathsf{H}_{5,q,1}$ ($q \in [Q_{\mathsf{pre}}]$): It is analogous to $\mathsf{H}_{5,(q-1)}$ except that in the qth secret-key query the vector $\boldsymbol{v}_{q,m_q+1,t}$ is modified as follows. The element $\widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}$ is shifted from $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_{\kappa,1}]$ to $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_\kappa]$ and the elements $\widetilde{\boldsymbol{r}}_{q,t}[m_q], \widetilde{\alpha}_q$ are shifted from $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}_1], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}_1]$ to $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}]$ respectively.

$$\begin{aligned}
\boldsymbol{v}_{q',1,t} &= (\,\widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], 0, \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, 0, 0, 0\,)\\
&\qquad \text{for } 1 \le q' < q,\\
\boldsymbol{v}_{q,1,t} &= (\,\widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i], \boxed{0}, 0, \boxed{\widetilde{\alpha}_q\boldsymbol{y}_q[\kappa]\nu_{q,t}}, 0, 0\,),\\
\boldsymbol{v}_{q',1,t} &= (\,\widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, 0, 0, 0, 0\,)\\
&\qquad \text{for } q < q' \le Q_{\mathsf{pre}},\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\,0, 0, \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \widetilde{\alpha}_{q'}, 0, 0, 0\,)\\
&\qquad \text{for } 1 \le q' < q,\\
\boldsymbol{v}_{q,m_q+1,t} &= (\,\boxed{0}, \boxed{0}, 0, 0, \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, \boxed{\widetilde{\alpha}_q}, 0\,),\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\,\widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \widetilde{\alpha}_{q'}, 0, 0, 0, 0, 0\,)\\
&\qquad \text{for } q < q' \le Q_{\mathsf{pre}}
\end{aligned}$$

We observe that the inner products $\boldsymbol{v}_{q,1,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ are unchanged due to the modification occurred in $\boldsymbol{v}_{q,1,t}$ and $\boldsymbol{v}_{q,m_q+1,t}$. Therefore, the function hiding security of IPFE ensures that the hybrids $\mathsf{H}_{5,(q-1)}$ and $\mathsf{H}_{5,q,1}$ are indistinguishable.

In this hybrid, the components of $\boldsymbol{v}_{q,j,t}$ corresponding to the slots $\{\mathsf{const}, \mathsf{coef}_i, \mathsf{extnd}_\kappa, \mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}$ and the components of $\boldsymbol{v}_{q,m_q+1,t}$ corresponding to the slots $\{\widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}$ are exactly the same as in the secret-key of our extended 1-FE scheme. Similarly, in case of the challenge ciphertext, the components of $\boldsymbol{u}$ at the positions $\{\mathsf{const}, \mathsf{coef}_i, \mathsf{extnd}_\kappa, \mathsf{sim}_\tau, \mathsf{sim}^*_\tau\}$ and the components of $\boldsymbol{h}_t$ at the positions $\{\widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}$ are also identical to the ciphertext of our extended 1-FE scheme.

Hybrid $\mathsf{H}_{5,q,2}$ ($q \in [Q_{\mathsf{pre}}]$): It is exactly the same as $\mathsf{H}_{5,q,1}$ except that the components $\boldsymbol{u}[\mathsf{extnd}_\kappa]$ and $\boldsymbol{h}_t[\widehat{\mathsf{coef}}]$ are changed from $\boldsymbol{z}^*[t], \boldsymbol{w}^*[\kappa]$ to $\boldsymbol{d}_1[t], \boldsymbol{d}_2[\kappa]$ respective. Thus, the vectors $\boldsymbol{u}, \boldsymbol{h}_t$ become

$$\begin{aligned}
\boldsymbol{u} &= (\,0, 0,\ \ 0, \quad\ 1, \quad \boldsymbol{x}^*[i], \boldsymbol{w}^*[\kappa], \boldsymbol{d}_2[\kappa], \boxed{\boldsymbol{d}_2[\kappa]}, 0, 0\,),\\
\boldsymbol{h}_t &= (\,0, 0, -1, \boldsymbol{z}^*[t], \ -1, \ \ \boldsymbol{d}_1[t], \quad -1, \ \ \boxed{\boldsymbol{d}_1[t]}, \ 0\ \,)
\end{aligned}$$

All the secret-keys are answered as in the previous hybrid. The indistinguishability follows from the security of our 1-FE scheme. We note that the security of our extended 1-FE scheme which relies on the function hiding security of IPFE and the security of AKGS. In particular, we use the security of IPFE and AKGS to reversely sample the first label and make all the other labels random as shown below

$$\begin{aligned}
&\widetilde{\ell}_{q,1,1} \leftarrow \mathsf{RevSamp}(f_{q,1}, \boldsymbol{x}^*, \widetilde{\alpha}_q f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^* + \widetilde{\beta}_{q,1}, \ell_{q,2,1}, \ldots, \ell_{q,m_q,1})\\
&\widetilde{\ell}_{q,1,\tau} \leftarrow \mathsf{RevSamp}(f_{q,\tau}, \boldsymbol{x}^*, \widetilde{\beta}_{q,\tau}, \ell_{q,2,\tau}, \ldots, \ell_{q,m_q,\tau}) \qquad \text{for } 1 < \tau < n',
\end{aligned}$$

where $\sum_{\tau\in[n']} \widetilde{\beta}_{q,\tau} = 0$ and $\ell_{q,j,\tau}$ is picked randomly for all $j \in [2, m_q]$. Then, the dummy vector $(\boldsymbol{d}_1 || \boldsymbol{d}_2)$ replaces the challenge message $(\boldsymbol{z}^* || \boldsymbol{w}^*)$ while computing $\widetilde{\ell}_{q,1,1}$. Finally, we move in the reverse direction so that the vectors $\boldsymbol{v}_{q,j,t}$ for all $j \in [m_q]$ and $t \in [n']$ are back in form as they were in $\mathsf{H}_{5,q,1}$ and $\boldsymbol{d}_1[t], \boldsymbol{d}_2[\kappa]$ are placed at $\boldsymbol{h}_t[\widehat{\mathsf{coef}}], \boldsymbol{u}[\mathsf{extnd}_\kappa]$ respectively. Note that, the hybrids involved in our extended 1-FE scheme uses the positions $\mathsf{sim}_\tau, \mathsf{sim}^*_\tau, \widehat{\mathsf{sim}}^*$ of the vectors $\boldsymbol{v}_{q,j,t}, \boldsymbol{u}$ and $\boldsymbol{h}_t$, which does not effect the decryption using any post-challenge secret-key.

Hybrid $\mathsf{H}_{5,q,3}$ ($q \in [Q_{\mathsf{pre}}]$): It proceeds analogous to $\mathsf{H}_{5,q,2}$ except that we change $\boldsymbol{v}_{q,m_q+1,t}$ and $\boldsymbol{h}_t$ as below. The element $\widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}$ is shifted from $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_\kappa]$ to $\boldsymbol{v}_{q,1,t}[\mathsf{extnd}_{\kappa,2}]$ and the elements $\widetilde{\boldsymbol{r}}_{q,t}[m_q], \widetilde{\alpha}_q$ are shifted from $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}]$ to $\boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{const}}_2], \boldsymbol{v}_{q,m_q+1,t}[\widehat{\mathsf{coef}}_2]$ respectively.

$$\begin{aligned}
\boldsymbol{v}_{q',1,t} &= (\,\widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], \quad 0, \quad \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, \ 0, \ 0, \ 0\,) && \text{for } 1 \le q' < q,\\
\boldsymbol{v}_{q,1,t} &= (\,\widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i], \quad 0, \quad \boxed{\widetilde{\alpha}_q\boldsymbol{y}_q[\kappa]\nu_{q,t}}, \boxed{0}, \ 0, \ 0\,),\\
\boldsymbol{v}_{q',1,t} &= (\,\widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q',1,t}[\mathsf{coef}_i], \widetilde{\alpha}_{q'}\boldsymbol{y}_{q'}[\kappa]\nu_{q',t}, \quad 0, \quad 0, \ 0, \ 0\,) && \text{for } q < q' \le Q_{\mathsf{pre}},\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\quad 0, \quad 0, \quad \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \quad \widetilde{\alpha}_{q'}, \quad 0, \ 0, \ 0\,) && \text{for } 1 \le q' < q,\\
\boldsymbol{v}_{q,m_q+1,t} &= (\quad 0, \quad 0, \quad \boxed{\widetilde{\boldsymbol{r}}_{q,t}[m_q]}, \quad \boxed{\widetilde{\alpha}_q}, \quad \boxed{0}, \boxed{0}, 0\,),\\
\boldsymbol{v}_{q',m_{q'}+1,t} &= (\ \ \widetilde{\boldsymbol{r}}_{q',t}[m_{q'}], \quad \widetilde{\alpha}_{q'}, \quad 0, \quad 0, \quad 0, \ 0, \ 0\,) && \text{for } q < q' \le Q_{\mathsf{pre}},
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{u} &= (\,0, 0,\ \ 0, \quad\ 1, \quad \boldsymbol{x}^*[i], \boldsymbol{w}^*[\kappa], \boldsymbol{d}_2[\kappa], \boxed{\boldsymbol{w}^*[\kappa]}, 0, 0\,),\\
\boldsymbol{h}_t &= (\,0, 0, -1, \boldsymbol{z}^*[t], \ -1, \ \ \boldsymbol{d}_1[t], \quad -1, \ \ \boxed{\boldsymbol{z}^*[t]}, \ 0\ \,)
\end{aligned}$$

Note that the inner products $\boldsymbol{v}_{q,1,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ remains the same as in $\mathsf{H}_{5,q,2}$. Therefore, the hybrids $\mathsf{H}_{5,q,2}$ and $\mathsf{H}_{5,q,3}$ are indistinguishable due to the function hiding security of IPFE. We observe that $\mathsf{H}_{5,q,3}$ is identical to $\mathsf{H}_{5,q}$ for all $q \in [Q_{\mathsf{pre}}]$.

Hybrid H_6: It is exactly the same as $\mathsf{H}_{5,Q_{\mathsf{pre}},4}$ except that the elements $\boldsymbol{u}[\mathsf{extnd}_\kappa]$, $\boldsymbol{h}_t[\widehat{\mathsf{const}}]$ and $\boldsymbol{h}_t[\widehat{\mathsf{coef}}]$ are set to zero. We describe the vectors associated to secret-key queries and the challenge ciphertext below. Note that the post-challenge secret-key queries are released in the same way as in H_4 (or in $\mathsf{H}_{5,Q_{\mathsf{pre}}}$).

$$1 \le q \le Q_{\mathsf{pre}} \begin{cases} \boldsymbol{v}_q = (\ \widetilde{\alpha}_q, \ 0, \ 0, \ 0, \ 0, 0, 0\ ), \\ \boldsymbol{v}_{q,1,t} = (\ \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i], \ 0, \ \widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}, 0, 0, 0\ ), \\ \boldsymbol{v}_{q,j,t} = (\ \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i], \ 0, \ 0, \ 0, 0, 0\ ) \quad \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t} = (\ 0, \ 0, \ \widetilde{\boldsymbol{r}}_{q,t}[m_q], \ \widetilde{\alpha}_q, \ 0, 0, 0\ ), \end{cases}$$

$$\boldsymbol{u} = (\ 0, 0, \ 0, \ 1, \ \boldsymbol{x}^*[i], \boldsymbol{w}^*[\kappa], \boldsymbol{d}_2[\kappa], \boxed{0}, 0, 0\ ),$$
$$\boldsymbol{h}_t = (\ 0, 0, -1, \boldsymbol{z}^*[t], \ -1, \ \boldsymbol{d}_1[t], \ \boxed{0}, \ \boxed{0}, \ 0\ )$$

$$Q_{\mathsf{pre}} < q \le Q \begin{cases} \boldsymbol{v}_q = (\ \widetilde{\alpha}_q, \ 0, \ 0, \ 0, 0, 0, 0\ ), \\ \boldsymbol{v}_{q,1,t} = (\ \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,1,t}[\mathsf{coef}_i], \widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}, 0, 0, 0, 0\ ), \\ \boldsymbol{v}_{q,j,t} = (\ \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{const}], \widetilde{\boldsymbol{\ell}}_{q,j,t}[\mathsf{coef}_i], \ 0, \ 0, 0, 0, 0\ ) \quad \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t} = (\ \widetilde{\boldsymbol{r}}_{q,t}[m_q], \ \widetilde{\alpha}_q, \ 0, \ 0, 0, 0, 0\ ) \end{cases}$$

Since the inner products $\boldsymbol{v}_{q,1,t} \cdot \boldsymbol{u}$ and $\boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ is unaltered due to the modification in this hybrid, the function hiding security of IPFE ensures indistinguishability between the hybrids $\mathsf{H}_{5,Q_{\mathsf{pre}},4}$ and H_6.

The second part of the proof is completed as all the pre-challenge secret-keys are now able to decrypt the challenge ciphertext without the components of $\boldsymbol{u}, \boldsymbol{h}_t$ that make use of $\boldsymbol{z}^*$ and $\boldsymbol{w}^*$. Note that, $\boldsymbol{u}[\mathsf{extnd}_{\kappa,1}] = \boldsymbol{w}^*[\kappa]$ and $\boldsymbol{h}_t[\widehat{\mathsf{coef}_1}] = \boldsymbol{z}^*[t]$ are only needed for the successful decryption of the challenge ciphertext by post-challenge secret-keys. From the next hybrid we change the computation of post-challenge secret-keys so that the challenge ciphertext can be simulated without using $(\boldsymbol{z}^*||\boldsymbol{w}^*)$.

Hybrid H_7: This hybrid proceeds exactly similar to H_6 except that we use the honest levels $\widetilde{\ell}_{q,1,t} = \widetilde{\boldsymbol{\ell}}_{q,1,t}(\boldsymbol{x}^*)$, $\widetilde{\ell}_{q,j,t} = \widetilde{\boldsymbol{\ell}}_{q,j,t}(\boldsymbol{x}^*)$ for $j \in [m_q]$ and $\widetilde{\ell}_{q,m_q+1,t} = -\widetilde{\boldsymbol{r}}_{q,t}[m_q] + \widetilde{\alpha}_q \boldsymbol{z}^*[t]$ while defining the vectors $\boldsymbol{v}_{q,j,t}$ in all the *post-challenge* secret-key queries. Moreover, all the other private components $\boldsymbol{v}_{q,j,t}[\mathsf{coef}_i]$ and $\boldsymbol{v}_{q,j,t}[\mathsf{extnd}_{\kappa,1}]$ are zero for all $j \in [m_q]$. We also modify $\boldsymbol{u}$ and $\boldsymbol{h}_t$ of the challenge ciphertext as shown below.

$$\boldsymbol{u} = (\ 0, 0, \ 0, \ 1, \ \boldsymbol{x}^*[i], \ \boxed{0}, \ \boldsymbol{d}_2[\kappa], 0, 0, 0\ ),$$
$$\boldsymbol{h}_t = (\ 0, 0, \boxed{1}, \boxed{0}, \ -1, \ \boldsymbol{d}_1[t], \ 0, \ 0, 0\ ),$$

$$Q_{\mathsf{pre}} < q \le Q \begin{cases} \boldsymbol{v}_q = (\ \widetilde{\alpha}_q, \ 0, \ 0, \ 0, 0, 0, 0\ ), \\ \boldsymbol{v}_{q,1,t} = (\ \boxed{\widetilde{\ell}_{q,1,t} + \widetilde{\alpha}_q \boldsymbol{y}_q[\kappa]\nu_{q,t}}, \boxed{0}, \boxed{0}, 0, 0, 0, 0\ ) \\ \boldsymbol{v}_{q,j,t} = (\ \boxed{\widetilde{\ell}_{q,j,t}}, \ \boxed{0}, \ 0, \ 0, 0, 0, 0\ ) \quad \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t} = (\ \boxed{\widetilde{\ell}_{q,m_q+1,t}}, \ \boxed{0}, \ 0, \ 0, 0, 0, 0\ ) \end{cases}$$

Since the inner products $\boldsymbol{v}_{q,j,t} \cdot \boldsymbol{u}, \boldsymbol{v}_{q,m_q+1,t} \cdot \boldsymbol{h}_t$ for all $q \in [Q_{\mathsf{pre}} + 1, Q]$ are the same as in the previous hybrid, the function hiding property of IPFE ensures that the hybrids H_6 and H_7 are indistinguishable.

Hybrid H_8: : This hybrid proceeds analogous to H_7 except that the post-challenge secret-key queries use the simulated garblings instead of the honest garblings. More specifically, we sample $\widetilde{\alpha}_q, \widetilde{\beta}_{q,t}, \widetilde{\nu}_{q,t} \leftarrow \mathbb{Z}_p$ satisfying $\sum_{t\in[n']} \widetilde{\beta}_{q,t} = 0, \sum_{t\in[n']} \widetilde{\nu}_{q,t} = 1$ and compute the simulated garblings

$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \widetilde{\alpha}_q \cdot (\boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\nu}_{q,t} \cdot \boldsymbol{y}_q^\top \boldsymbol{w}^*) + \widetilde{\beta}_{q,t})$$

for all $q \in [Q_{\mathsf{pre}} + 1, Q]$ and $t \in [n']$. Then, the post-challenge secret-keys are generated using the vectors described below.

$$Q_{\mathsf{pre}} < q \leq Q \begin{cases} \boldsymbol{v}_q = (& \widetilde{\alpha}_q, & 0,0,0,0,0,0\ ), \\ \boldsymbol{v}_{q,1,t} = (& \boxed{\widehat{\ell}_{q,1,t}}, & 0,0,0,0,0,0\ ) \\ \boldsymbol{v}_{q,j,t} = (& \boxed{\widehat{\ell}_{q,j,t}}, & 0,0,0,0,0,0\ ) & \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t} = (& \boxed{\widehat{\ell}_{q,m_q+1,t}}, & 0,0,0,0,0,0\ ) \end{cases}$$

The simulated levels of AKGS is used in place of actual garblings. The simulation security of AKGS implies that the hybrids H_7 and H_8 are indistinguishable.

Hybrid H_9: : This proceeds exactly the same as H_8 except that the distribution of $\{\widetilde{\beta}_{q,t}\}_{t\in[n']}$ is changed. We replace $\widetilde{\beta}_{q,t}$ by $\widetilde{\beta}'_{q,t} = \widetilde{\beta}_{q,t} - \widetilde{\alpha}_q \cdot (\boldsymbol{z}^*[t] f_{q,t}(\boldsymbol{x}^*) + \widetilde{\nu}_{q,t} \cdot \boldsymbol{y}_q^\top \boldsymbol{w}^*)$ for all $1 < t \leq n'$ and replace the element $\widetilde{\beta}_{q,1}$ by $\widetilde{\beta}'_{q,1} = \widetilde{\beta}_{q,1} - \widetilde{\alpha}_q \cdot (\boldsymbol{z}^*[1] f_{q,1}(\boldsymbol{x}^*) + \widetilde{\nu}_{q,1} \cdot \boldsymbol{y}_q^\top \boldsymbol{w}^*) + \widetilde{\alpha}_q \cdot (f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*)$. Note that, the distributions

$$\{\widetilde{\beta}_{t,q} \leftarrow \mathbb{Z}_p : \sum_{t\in[n']} \widetilde{\beta}_{t,q} = 0\} \text{ and } \{\widetilde{\beta}'_{t,q} : \sum_{t\in[n']} \widetilde{\beta}_{t,q} = 0\}$$

are statistically close since $\{\widetilde{\beta}'_{q,t}\}_{t\in[n']}$ are also uniform over $\mathbb{Z}_p$ and $\sum_{t\in[n']} \widetilde{\beta}'_{q,t} = 0$. Finally, the vectors associated to the post-challenge secret-keys are given by

$$Q_{\mathsf{pre}} < q \leq Q \begin{cases} \boldsymbol{v}_q = (& \widetilde{\alpha}_q, & 0,0,0,0,0,0\ ), \\ \boldsymbol{v}_{q,1,t} = (& \boxed{\widehat{\ell}_{q,1,t}}, & 0,0,0,0,0,0\ ) \\ \boldsymbol{v}_{q,j,t} = (& \boxed{\widehat{\ell}_{q,j,t}}, & 0,0,0,0,0,0\ ) & \text{for } j \in [2, m_q], \\ \boldsymbol{v}_{q,m_q+1,t} = (& \boxed{\widehat{\ell}_{q,m_q+1,t}}, & 0,0,0,0,0,0\ ) \end{cases}$$

where the simulated garblings take the form

$$(\widehat{\ell}_{q,1,1}, \ldots, \widehat{\ell}_{q,m_q,1}, \widehat{\ell}_{q,m_q+1,1}) \leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, \boxed{\widetilde{\alpha}_q \cdot (f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \boldsymbol{y}_q^\top \boldsymbol{w}^*) + \widetilde{\beta}_{q,1}})$$
$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q,t}, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \boxed{\widetilde{\beta}_{q,t}}) \quad \text{for } 1 < t \leq n'.$$

Observe that H_9 is the same as the ideal experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal,extFE}}(1^\lambda)$. This completes the security proof.

□