(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, U.S.A.
pratish.datta@ntt-research.com,
[2] Indian Institute of Technology Kharagpur, Kharagpur, West Bengal 721302, India
tapas.pal@iitkgp.ac.in,

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.

Keywords: functional encryption, attribute-weighted sums, adaptive simulation security

1 Introduction

Functional Encryption: *Functional encryption* (FE), formally introduced by Boneh et al. [9] and O'Neill [26], 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 $\mathbb{M}$ 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 [3] 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) [1,7]. 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). The indices $i \in [N]$ are referred to as the *slots*. 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. [3] 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 [3] 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. further considered the notion of *unbounded-slot* FE scheme for attribute-weighted sum where 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 [3] 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 [3] 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 [26,9], 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 [3] 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 possibly supporting an 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 [28,18,17] 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 scal-

ing with that upper bound [24,27]). Recently Kowalczyk and Wee [16] and Lin and Luo [19] 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. [3]:

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 [9]. Since simulation-based security also implies indistinguishability-based security and indistinguishability-based security against single and multiple ciphertexts are equivalent [9,26], 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. [3], 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) [12]. 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 [5,30]. Unlike [3], our one-slot construction is semi-generic and is built upon two cryptographic building blocks, namely a slotted inner product functional encryption (IPFE) [20,19], 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) [14,19]. For bootstrapping from one-slot to unbounded-slot construction we make use of the same semi-generic transformation proposed in [3], 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 [19]. The techniques of [19] 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 [19] 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 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 [19] 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 [26,10], and carefully exploiting the structures of the underlying building blocks, namely AKGS and slotted IPFE.

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 1-key 1-ciphertext secure 1-slot FE and unbounded-key secure 1-slot FE for attribute-weighted sums in Sections 4 and 5 respectively. The details of security reductions are given in the full version. Next, in Section 6, we provide an extended version of our 1-slot FE scheme, on which we apply the bootstrapping transformation from [3] leading to our unbounded-slot scheme. The formal security analysis of the scheme is deferred to the full version as well. Further,

[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.

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 [15]	$\phi_{\boldsymbol{y}\in\mathbb{Z}_p^n} : \mathbb{Z}_p^n \rightarrow \{0,1\}, \phi_{\boldsymbol{y}}(\boldsymbol{z}) = (\boldsymbol{z}^\top \boldsymbol{y} \stackrel{?}{=} 0)$	1	$(-, \mathsf{poly}, \mathsf{poly})$-AD	$\times$	$O(\lvert\boldsymbol{z}\rvert)$	2 non-standard assumptions
OT12 [22]	$\phi_{\boldsymbol{y}\in\mathbb{Z}_p^n} : \mathbb{Z}_p^n \rightarrow \{0,1\}, \phi_{\boldsymbol{y}}(\boldsymbol{z}) = (\boldsymbol{z}^\top \boldsymbol{y} \stackrel{?}{=} 0)$	1	$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-AD	$\times$	$O(\lvert\boldsymbol{z}\rvert)$	DLIN
ABDCP15 [1]	$\phi_{\boldsymbol{y}\in\mathbb{Z}_p^n} : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p, \phi_{\boldsymbol{y}}(\boldsymbol{z}) = \boldsymbol{z}^\top \boldsymbol{y}$	1	$(-, \mathsf{poly}, \mathsf{poly})$-Sel	$\times$	$O(\lvert\boldsymbol{z}\rvert)$	DDH, LWE
ALS16, ALMT20 [7,6]	$\phi_{\boldsymbol{y}\in\mathbb{Z}_p^n} : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p, \phi_{\boldsymbol{y}}(\boldsymbol{z}) = \boldsymbol{z}^\top \boldsymbol{y}$	1	$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-AD	$(\mathsf{poly}, \mathsf{bdd}, \mathsf{poly})$-Sel	$O(\lvert\boldsymbol{z}\rvert)$	DDH, DCR, LWE
Agr17 [4]	$\phi_{f\in\mathsf{GC}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \rightarrow \{0,1\}, \phi_f(\boldsymbol{x}, \boldsymbol{z}) = (f(\boldsymbol{x})^\top \boldsymbol{z} \stackrel{?}{=} 0)$	1	$(-, \mathsf{poly}, \mathsf{bdd})$-S-AD	$(-, 1, \mathsf{bdd})$-S-AD	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert)$	LWE
Wee17 [29]	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \rightarrow \{0,1\}, \phi_f(\boldsymbol{x}, \boldsymbol{z}) = (f(\boldsymbol{x})^\top \boldsymbol{z} \stackrel{?}{=} 0)$	1	$(-, \mathsf{poly}, \mathsf{poly})$-S-AD	$(-, 1, \mathsf{poly})$-S-AD	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert)$	k-Lin
DOT18 [10]	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \rightarrow \{0,1\}, \phi_f(\boldsymbol{x}, \boldsymbol{z}) = (f(\boldsymbol{x})^\top \boldsymbol{z} \stackrel{?}{=} 0)$	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 [2]	$\phi_{(f\in(\mathsf{NC}^1)^{(n)}, \boldsymbol{y}\in\mathbb{Z}_p^{n'})} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \rightarrow \mathbb{Z}_p, \phi_{(f,\boldsymbol{y})}(\boldsymbol{x}, \boldsymbol{z}) = (f(\boldsymbol{x}) \stackrel{?}{=} 0) \cdot \boldsymbol{z}^\top \boldsymbol{y}$	1	$(\mathsf{poly}, \mathsf{poly}, \mathsf{poly})$-AD	$\times$	$O(\lvert\boldsymbol{x}\rvert + \lvert\boldsymbol{z}\rvert)$	SXDH
AGW20 [3]	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \rightarrow \mathbb{Z}_p, \phi_f(\boldsymbol{x}, \boldsymbol{z}) = f(\boldsymbol{x})^\top \boldsymbol{z}$	unbounded	$(-, \mathsf{poly}, \mathsf{poly})$-AD	$(-, \mathsf{bdd}, \mathsf{poly})$-S-AD	$O(\lvert\boldsymbol{z}\rvert)$	k-Lin
Wee20 [30]	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n_1 n_2)}} : \mathbb{Z}_p^n \times (\mathbb{Z}_p^{n_1} \times \mathbb{Z}_p^{n_2}) \rightarrow \mathbb{Z}_p, \phi_f(\boldsymbol{x}, (\boldsymbol{z}_1, \boldsymbol{z}_2)) = f(\boldsymbol{x})^\top (\boldsymbol{z}_1 \otimes \boldsymbol{z}_2)$	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	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \rightarrow \mathbb{Z}_p, \phi_f(\boldsymbol{x}, \boldsymbol{z}) = f(\boldsymbol{x})^\top \boldsymbol{z}$	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	$\phi_{f\in\mathcal{F}_{\mathsf{ABP}}^{(n,n')}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'} \rightarrow \mathbb{Z}_p, \phi_f(\boldsymbol{x}, \boldsymbol{z}) = f(\boldsymbol{x})^\top \boldsymbol{z}$	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
- SA: Semi-adaptive security
- Sel: Selective security
- poly: Arbitrary polynomial in the security parameter
- bdd: A-priori bounded by the public parameters
- $|x|$: 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.

the formal definition of bilinear maps, related hardness assumptions, syntax and security definition of slotted IPFE, and the details of special piecewise security of AKGS are provided in the full version. For security analysis of our 1-slot extended FE, we construct a 1-key 1-ciphertext secure 1-slot extended FE scheme which is also available in the full version. Finally, in the full version, we present our formal analysis of the bootstrapping transformation from [3].

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 [3] 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 [3]. However, [3] 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 [3]. 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 [3] applied on our one-slot extended FE 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 FE Scheme

Abdalla et al. [3] built their one-slot FE scheme for attribute-weighted sums by extending the techniques devised by Wee [29] 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 [3,29] 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 [3,29] does not seem to extend to the adaptive setting. Datta et al. [11] designed an adaptively simulation secure predicate encryption scheme for the same class of predicates as [29], 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 [19] developed new techniques to obtain adaptive indistinguishability secure ABE with compact ciphertexts for the class of predicates expressed as ABPs. [19] 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 [19] 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 [19] into our setting of simulation-based security for FE without the one-use restriction. However, a straight-forward adaptation of the [19] 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 [14] and recently it was further refined by [19] 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 [14]. 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, [19] 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 [14] directly implies a piecewise secure AKGS for ABPs. (See Section 3.3 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 [20,19] 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 $\boldsymbol{u}_{\mathsf{priv}} = \mathbf{0}$. 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 [19] and Our Ideas
We now briefly explain at a high level, the main challenges in adapting the [19] technique into our setting and our ideas to overcome those challenges.

1. To handle the pre-challenge secret-key queries, [19] 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 $\boldsymbol{z}[t] f_t(\boldsymbol{x}) + \beta_t$ completely random. In order to handle this, we carefully integrate the techniques of *pre-image*

sampleability [26,11] with the reverse sampling and marginal randomness properties of AKGS to handle the pre-challenge queries.

2. The security proof of [19] implements a version of the dual system encryption methodology [28,18,17] 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 [28,18,17,24,23,25,11,19]. 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 [19] 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 $\sum_{t=[n']} \beta_t = 0 \mod p$). 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 [19] 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 [14] 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 $(\boldsymbol{\ell}_{1,t}^{(\iota)}, \ldots, \boldsymbol{\ell}_{m+1,t}^{(\iota)}) \leftarrow \mathsf{Garble}(\boldsymbol{z}[t]\boldsymbol{\alpha}[\iota] f_t(\boldsymbol{x}) + \boldsymbol{\beta}_t[\iota]; \boldsymbol{r}_t^{(\iota)})$ for $\iota \in [k]$ where $\boldsymbol{r}_t^{(\iota)} \leftarrow \mathbb{Z}_p^m$. Using the instantiation of AKGS given by [14], we have that the $(m+1)$-th label functions $L_{m+1,t}^{(\iota)}$ take the form $L_{m+1,t}^{(\iota)}(\boldsymbol{z}[t]) = \boldsymbol{z}[t]\boldsymbol{\alpha}[\iota] - r_t^{(\iota)}[m]$ with $\boldsymbol{\alpha}[\iota]$ a constant.
- Compute the IPFE secret keys

$$\mathsf{IPFE.SK} = \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\alpha}, \mathbf{0}_{kn} \parallel 0, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2)$$

$$\mathsf{IPFE.SK}_{j,t} = \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\ell}_{j,t}^{(1)}, \ldots, \boldsymbol{\ell}_{j,t}^{(k)} \parallel 0, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2) \text{ for } j \in [m]$$

$$\widehat{\mathsf{IPFE.SK}}_{m+1,t} = \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{r}_t^{(1)}[m], \ldots, \boldsymbol{r}_t^{(k)}[m], \boldsymbol{\alpha} \parallel 0, 0, 0, 0, 0, 0, 0]\!]_2)$$

- 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

$$\mathsf{IPFE.CT} = \mathsf{IPFE.SlotEnc}(\mathsf{IPFE.MSK}, [\![\boldsymbol{s}, \boldsymbol{s} \otimes \boldsymbol{x}]\!]_1)$$

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

  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

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

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 $f_q = (f_{q,1}, \ldots, f_{q,n'})$, the simulator sets public slots of all the vectors $\boldsymbol{v}_q, \boldsymbol{v}_{q,j,t}$ for $j \in \{1, \ldots, m_q + 1\}$ 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

$$\mathsf{IPFE.SK}_q = \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\alpha}[\iota], \mathbf{0}_{kn} \parallel \widetilde{\alpha}_q, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2)$$

$$\mathsf{IPFE.SK}_{q,j,t} = \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\ell}^{(1)}_{q,j,t}, \ldots, \boldsymbol{\ell}^{(k)}_{q,j,t} \parallel \widetilde{\boldsymbol{\ell}}_{q,j,t}, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2) \quad \text{for } j \in [m_q]$$

$$\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} = \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{r}^{(1)}_t[m_q], \ldots, \boldsymbol{r}^{(k)}_t[m_q], \boldsymbol{\alpha} \parallel 0, 0, \widetilde{\boldsymbol{r}}_{q,t}[m_q], \widetilde{\alpha}_q, 0, 0, 0]\!]_2)$$

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 [26]. The simulated ciphertext becomes

$$\mathsf{IPFE.CT} = \mathsf{IPFE.Enc}(\mathsf{IPFE.MSK}, [\![\mathbf{0}_k, \mathbf{0}_{kn} \parallel 1, \boldsymbol{x}^*, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_1)$$

$$\widehat{\mathsf{IPFE.CT}}_t = \mathsf{IPFE.Enc}(\widehat{\mathsf{IPFE.MSK}}, [\![\mathbf{0}_k, \mathbf{0}_k \parallel 1, 0, -1, \boldsymbol{d}[t], 0, 0, 0]\!]_1)$$

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:

$$(\widehat{\ell}_{q,1,1}, \ldots, \widehat{\ell}_{q,m_q+1,1}) \leftarrow \mathsf{SimGarble}(f_{q,1}, \boldsymbol{x}^*, \widetilde{\alpha}_q \cdot f_q(\boldsymbol{x}^*)^\top \boldsymbol{z}^* + \beta_{q,1}) \tag{5}$$

$$(\widehat{\ell}_{q,1,t}, \ldots, \widehat{\ell}_{q,m_q+1,t}) \leftarrow \mathsf{SimGarble}(f_{q,t}, \boldsymbol{x}^*, \beta_{q,t}) \quad \text{for } 1 < t \leq n' \tag{6}$$

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

$$\mathsf{IPFE.SK}_q = \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\alpha}, \mathbf{0}_{kn} \parallel \widetilde{\alpha}_q, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2)$$

$$\mathsf{IPFE.SK}_{q,j,t} = \mathsf{IPFE.KeyGen}(\mathsf{IPFE.MSK}, [\![\boldsymbol{\ell}^{(1)}_{j,t}, \ldots, \boldsymbol{\ell}^{(k)}_{j,t} \parallel \ell_{q,j,t}, \mathbf{0}_n, \mathbf{0}_{n'}, \mathbf{0}_{n'}]\!]_2) \quad \text{for } j \in [m_q]$$

$$\widehat{\mathsf{IPFE.SK}}_{q,m_q+1,t} = \mathsf{IPFE.KeyGen}(\widehat{\mathsf{IPFE.MSK}}, [\![\boldsymbol{r}^{(1)}_t[m_q], \ldots, \boldsymbol{r}^{(k)}_t[m_q], \boldsymbol{\alpha} \parallel \ell_{q,m_q+1,t}, 0, 0, 0, 0, 0, 0]\!]_2)$$

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

$$\boldsymbol{u} = (\boldsymbol{s}, \boldsymbol{s} \otimes \boldsymbol{x}^* \parallel \boxed{0}, \boxed{\mathbf{0}_n}, \boxed{\mathbf{0}_{n'}}, \boxed{\mathbf{0}_{n'}}),$$

$$\boldsymbol{h}_t = (-\boldsymbol{s}, \boldsymbol{s} \cdot \boldsymbol{z}^*[t] \parallel \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}, \boxed{0}).$$

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

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

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

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

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 [26] and execute our *three slot* dual system encryption variant devised by extending the framework of [19].

Step 2

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

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

Hybrid $\mathsf{H_{5,q}}(\boldsymbol{q} \in [\boldsymbol{Q}_{\mathsf{pre}}])$: 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.

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

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

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

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

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

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 [22,10]. Built on the techniques of [19], 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.

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

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 [3] 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.

$$\begin{aligned}
\boldsymbol{v}_{q,j,t} &= (\cdots\ \|\ \boxed{\widetilde{\ell}_{q,j,t}}, \boxed{\mathbf{0}_n}, \mathbf{0}_{n'}, \mathbf{0}_{n'}) && \text{for } j \in [m_q], q > Q_{\mathsf{pre}}\\
\boldsymbol{v}_{q,m_q+1,t} &= (\ \cdots\ \|\ \boxed{\widetilde{\ell}_{q,m_q+1,t}},\ \boxed{0},\ 0,\ 0,\ 0,\ 0,\ 0\ ) && \text{for } q > Q_{\mathsf{pre}}\\
\boldsymbol{h}_t &= (\ \cdots\ \|\ \boxed{1},\ \boxed{0},\ -1,\ \boldsymbol{d}[t],\ 0,\ 0,\ 0\ )
\end{aligned}$$

Hybrid H_8: Indistinguishable from H_7 by simulation security of AKGS.

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

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 H_9: Statistically close to H_8

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

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 $1 < t \leq n'$, 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 (extFE) scheme which is required for applying the compiler of [3] to bootstrap to the unbounded-slot 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. [3] 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 [3] 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 [3] 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 $\sum_{i \in [N]} \boldsymbol{w}_i = \boldsymbol{0} \mod p$. 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. [3] 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 [3], 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 [12,5,30] 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 [3] 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. Please refer to the full version of the paper for a detailed formal exposure of the modifications and our analysis of the bootstrapping transformation.

3 Preliminaries

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 $\mathsf{Expt}^{\mathsf{C}}_{\mathcal{A}}$ is a function of λ and it is parametrized by an adversary $\mathcal{A}$ and a cryptographic protocol C. Let $\mathsf{Expt}^{\mathsf{C},0}_{\mathcal{A}}$ and $\mathsf{Expt}^{\mathsf{C},1}_{\mathcal{A}}$ 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 $\lambda \in \mathbb{N}$,

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

We write $\mathsf{Expt}^{\mathsf{C},0}_{\mathcal{A}} \overset{c}{\approx} \mathsf{Expt}^{\mathsf{C},1}_{\mathcal{A}}$ if they are *computationally indistinguishable* (or simply *indistinguishable*). Similarly, $\mathsf{Expt}^{\mathsf{C},0}_{\mathcal{A}} \overset{s}{\approx} \mathsf{Expt}^{\mathsf{C},1}_{\mathcal{A}}$ means *statistically indistinguishable* and $\mathsf{Expt}^{\mathsf{C},0}_{\mathcal{A}} \equiv \mathsf{Expt}^{\mathsf{C},1}_{\mathcal{A}}$ 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 $f(\boldsymbol{x}) = \mathbf{f}[\mathsf{const}] + \sum_{i \in [n]} \mathbf{f}[\mathsf{coef}_i]\boldsymbol{x}[i] \in \mathbb{Z}_p$.

3.1 Arithmetic Branching Program

Arithmetic Branching Program (ABP) is a computational model [21] 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 $\mathcal{F}_{\mathsf{ABP}}^{(n)}$ the class of ABPs over $\mathbb{Z}_p^n$:

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

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 1 *[13] 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 $f(\boldsymbol{x}) = \det(\mathbf{M})$.

3.2 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:

$\mathsf{Setup}(1^\lambda, 1^n, 1^{n'})$: 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.

$\mathsf{KeyGen}(\mathsf{MSK}, f)$: 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.

$\mathsf{Enc}(\mathsf{MPK}, (\boldsymbol{x}_i, \boldsymbol{z}_i)_{i\in[N]})$: 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.

$\mathsf{Dec}((\mathsf{SK}_f, f), (\mathsf{CT}, (\boldsymbol{x}_i)_{i\in[N]}))$: 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 $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ 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 adaptively simulation secure if $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real},\mathsf{ubdFE}}(1^\lambda) \overset{c}{\approx} \mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal},\mathsf{ubdFE}}(1^\lambda)$, where the experiments are defined as follows:

$\underline{\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real},\mathsf{ubdFE}}(1^\lambda)}$

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 $\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}}(\mathsf{MPK}, \mathsf{CT}^*)$

$\underline{\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal},\mathsf{ubdFE}}(1^\lambda)}$

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 $\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}_1^*(\mathsf{MSK}^*,(\boldsymbol{x}_i^*)_{i\in[N]},\cdot,\cdot)}}(\mathsf{MPK}, \mathsf{CT}^*)$

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

1. input: f
2. output: SK_f

$\underline{\mathcal{O}_{\mathsf{KeyGen}_0^*(\mathsf{MSK}^*,\cdot)}}$

1. input: f_q for $q \in [Q_{\mathsf{pre}}]$
2. output: $\mathsf{SK}_{f_q}^*$

$\underline{\mathsf{Enc}^*(\mathsf{MPK}, \mathsf{MSK}^*, (\boldsymbol{x}_i^*)_{i\in[N]}, \cdot)}$

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: CT^*

$\underline{\mathcal{O}_{\mathsf{KeyGen}_1^*(\mathsf{MSK}^*,(\boldsymbol{x}_i^*)_{i\in[N]},\cdot,\cdot)}}$

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

3.3 Arithmetic Key Garbling Scheme

Lin and Luo [19] introduced arithmetic key garbling scheme (AKGS). The notion of AKGS is an information theoretic primitive, inspired by randomized encodings [8] and partial garbling schemes [14]. 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 5 (Arithmetic Key Garbling Scheme (AKGS), [14,19]) 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 $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{m+1}$.

$\mathsf{Eval}(f, \boldsymbol{x}, \boldsymbol{\ell_1}, \ldots, \boldsymbol{\ell_{m+1}})$: 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 has *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 [14].

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 [3], however it is not sufficient for achieving adaptive security. We consider the *piecewise* security of AKGS proposed by Lin and Luo [19] where they used it to get adaptive security for ABE.

Instantiation of AKGS: ([14,19]). We now discuss an instantiation of AKGS = (Garble, Eval) for the function class $\mathcal{F} = \mathcal{F}_{\mathsf{ABP}}^{(n,1)}$ following [14,19].

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

1. Using Lemma 1, 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
$$\widehat{\mathbf{M}} = \mathbf{M}'\mathbf{N} = \begin{pmatrix} \mathbf{M} & \mathbf{M}\boldsymbol{r} + \boldsymbol{m}_1 \\ \boldsymbol{m}_2^\top & \boldsymbol{m}_2^\top \boldsymbol{r} + z \end{pmatrix} = \left(\begin{array}{ccccc|c} & & & & & L_1(\boldsymbol{x}) \\ & & & & & L_2(\boldsymbol{x}) \\ & & \mathbf{M} & & & \vdots \\ & & & & & L_m(\boldsymbol{x}) \\ \hline 0 & 0 & \cdots & 0 & -1 & L_{m+1}(z) \end{array}\right)$$
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 $i \in [n]$.

Eval$(f, \boldsymbol{x}, \ell_1, \ldots, \ell_m)$: 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 1 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
$$\mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1}) = A_1\ell_1 + A_2\ell_2 + \cdots + A_{m+1}\ell_{m+1} \tag{7}$$
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
$$\delta + \mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1}) = \mathsf{Eval}(f, \boldsymbol{x}, \delta, 0, \ldots, 0) + \mathsf{Eval}(f, \boldsymbol{x}, \ell_1, \ldots, \ell_{m+1}) \tag{8}$$
$$= \mathsf{Eval}(f, \boldsymbol{x}, \ell_1 + \delta, \ell_2, \ldots, \ell_{m+1}) \tag{9}$$
where equation 8 holds due to equation 7 and $A_1 = 1$; and equation 9 holds by the linearity of Eval. We will utilize equation 9 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$(f, \boldsymbol{x}, zf(\boldsymbol{x}) + \beta)$: 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 1, 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 [14]. 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 2 ([19]) *The above construction of* $\mathsf{AKGS} = (\mathsf{Garble}, \mathsf{Eval})$ *is secure. Moreover, it is special piecewise secure.*

4 Our 1-Key 1-Ciphertext Secure 1-Slot FE

In this section, we describe our 1-slot FE scheme for the attribute-weighted sum functionality secure against a single ciphertext and secret key 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, and $(\mathsf{SK\text{-}IPFE.Setup}.\mathsf{SK\text{-}IPFE.KeyGen}, \mathsf{SK\text{-}IPFE.Enc}, \mathsf{SK\text{-}IPFE.Dec})$ a secret-key function-hiding $\mathsf{SK\text{-}IPFE}$ based on G.

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

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

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

$\mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{f})$: 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 3.3. 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 $\mathsf{SK}_f = (\{\mathsf{IPFE.SK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.SK}}_{m+1,t}\}_{t\in[n']})$.

$\mathbf{Enc}(\mathbf{MSK}, \boldsymbol{x} \in \mathbb{Z}_{\boldsymbol{p}}^{\boldsymbol{n}}, \boldsymbol{z} \in \mathbb{Z}_{\boldsymbol{p}}^{\boldsymbol{n'}})$: 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 $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

$\mathbf{Dec}((\mathbf{SK}_{\boldsymbol{f}}, \boldsymbol{f}), (\mathbf{CT}, \boldsymbol{x}))$: 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 SK-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

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

Finally, it returns a value ρ by solving a discrete logarithm problem. Similar to [3], 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

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

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

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

Theorem 2 *The 1-FE scheme for attribute-weighted sum is adaptively simulation secure against a single ciphertext and secret key queries assuming the* AKGS *is piecewise secure and the* IPFE *is function hiding.*

5 Our 1-Slot FE for Attribute-Weighted Sums

In this section, we describe our 1-slot FE scheme Π_{one} for the attribute-weighted sum functionality. 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 piece-wise 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 the MDDH_k assumption 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 $\mathbb{M} = \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$.

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

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

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 $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widehat{\mathsf{IPFE.MPK}})$.

$\mathsf{KeyGen}(\mathsf{MSK}, \boldsymbol{f})$: 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

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

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

$\mathsf{Enc}(\mathsf{MPK}, \boldsymbol{x} \in \mathbb{Z}_{\boldsymbol{p}}^{\boldsymbol{n}}, \boldsymbol{z} \in \mathbb{Z}_{\boldsymbol{p}}^{\boldsymbol{n'}})$: 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 $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$.

Dec$((\mathsf{SK}_f, f), (\mathsf{CT}, \boldsymbol{x}))$: It parses $\mathsf{SK}_f = (\mathsf{IPFE.MSK}, \{\mathsf{IPFE.MSK}_{j,t}\}_{j\in[m],t\in[n']}, \{\widehat{\mathsf{IPFE.MSK}}_{m+1,t}\}_{t\in[n']})$ and $\mathsf{CT} = (\mathsf{IPFE.CT}, \{\widehat{\mathsf{IPFE.CT}}_t\}_{t\in[n']})$. 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 $\sum_{t\in[n']} \boldsymbol{\beta}_t[\iota] = 0 \mod p$ 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 1-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 scheme 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 the master public key, secret-keys, and ciphertext would grow linearly with q_{CT}. A similar strategy can be followed to convert our extended 1-slot FE scheme (of Section 6) that only supports a single ciphertext query to one that is secure for any a-priori bounded number of ciphertext queries.

Theorem 3 *The 1-slot* FE *scheme Π_{one} for attribute-weighted sums is adaptively simulation-secure assuming the* AKGS *is piece-wise secure, the* MDDH_k *assumption holds in group $\mathbb{G}_2$, and the slotted* IPFE *is function hiding.*

6 Our 1-Slot Extended FE for Attribute-Weighted Sums

In this section, we describe our 1-slot extFE scheme Π_{extOne} for the attribute-weighted sum functionality. 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 assumption 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 $\mathbb{M} = \mathbb{Z}_p^n \times \mathbb{Z}_p^{n'}$.

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

$$S_{\mathsf{pub}} = \Big\{\{\mathsf{const}^{(\iota)}\}_{\iota\in[k]}, \{\mathsf{coef}_i^{(\iota)}\}_{\iota\in[k],i\in[n]}, \{\mathsf{extnd}_\kappa^{(\iota)}\}_{\iota,\kappa\in[k]}\Big\},\ \widehat{S}_{\mathsf{pub}} = \{\widehat{\mathsf{const}}^{(\iota)}, \widehat{\mathsf{coef}}^{(\iota)}\}_{\iota\in[k]},$$
$$S_{\mathsf{priv}} = \Big\{\mathsf{const}, \{\mathsf{coef}_i\}_{i\in[n]}, \{\mathsf{extnd}_{\kappa,1}, \mathsf{extnd}_{\kappa,2}, \mathsf{extnd}_\kappa\}_{\kappa\in[k]}, \{\mathsf{query}_\eta\}_{\eta\in[B]}, \{\mathsf{sim}_\tau, \mathsf{sim}_\tau^*\}_{\tau\in[n']}\Big\},$$
$$\widehat{S}_{\mathsf{priv}} = \{\widehat{\mathsf{const}}_1, \widehat{\mathsf{coef}}_1, \widehat{\mathsf{const}}_2, \widehat{\mathsf{coef}}_2, \widehat{\mathsf{const}}, \widehat{\mathsf{coef}}, \widehat{\mathsf{sim}}^*\}$$

where B denotes a bound on the number of pre-ciphertext 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 $\mathsf{MPK} = (\mathsf{IPFE.MPK}, \widehat{\mathsf{IPFE.MPK}})$.

$\mathsf{KeyGen}(\mathsf{MSK}, (\boldsymbol{f}, \boldsymbol{y}))$: 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

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

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 3.3. 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

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

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 $\mathsf{IPFE.KeyGen}$ 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 is crucially while describing our unbounded-slot FE in the full version.

$\mathsf{Enc}(\mathsf{MPK}, (\boldsymbol{x}, \boldsymbol{z}||\boldsymbol{w}) \in \mathbb{Z}_p^{\boldsymbol{n}} \times \mathbb{Z}_p^{\boldsymbol{n'+k}})$: 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

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

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

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

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

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 $[\![\mu]\!]_T = [\![f(\boldsymbol{x})^\top \boldsymbol{z} + \boldsymbol{y}^\top \boldsymbol{w}]\!]_T$.

Theorem 4 *The extended one slot FE scheme Π_{extOne} for attribute-weighted sum is adaptively simulation-secure against adversaries making at most B pre-ciphertext secret key queries and an arbitrary polynomial number of post-ciphertext secret key queries assuming the AKGS is piecewise-secure, the MDDH_k assumption holds in group $\mathbb{G}_2$, and the slotted IPFE is function hiding.*

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: PKC 2015. pp. 733–751. Springer
2. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. IACR Cryptology ePrint Archive, Report 2020/577
3. Abdalla, M., Gong, J., Wee, H.: Functional encryption for attribute-weighted sums from k-Lin. In: CRYPTO 2020. pp. 685–716. Springer
4. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: CRYPTO 2017. pp. 3–35. Springer
5. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. IACR Cryptology ePrint Archive, Report 2020/1285
6. Agrawal, S., Libert, B., Maitra, M., Titiu, R.: Adaptive simulation security for inner product functional encryption. In: PKC 2020. pp. 34–64. Springer
7. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: CRYPTO 2016. pp. 333–362. Springer
8. Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: FOCS 2011. pp. 120–129. IEEE Computer Society
9. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: TCC 2011. pp. 253–273. Springer
10. Datta, P., Okamoto, T., Takashima, K.: Adaptively simulation-secure attribute-hiding predicate encryption. In: ASIACRYPT 2018. pp. 640–672. Springer

11. Datta, P., Okamoto, T., Takashima, K.: Adaptively simulation-secure attribute-hiding predicate encryption. IEICE TRANSACTIONS on Information and Systems **103**(7), 1556–1597 (2020)
12. Escala, A., Herold, G., Kiltz, E., Rafols, C., Villar, J.: An algebraic framework for diffie–hellman assumptions. Journal of cryptology **30**(1), 242–288 (2017)
13. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: ICALP 2002. pp. 244–256. Springer
14. Ishai, Y., Wee, H.: Partial garbling schemes and their applications. In: ICALP 2014. pp. 650–662. Springer
15. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: EUROCRYPT 2008. pp. 146–162. Springer
16. Kowalczyk, L., Wee, H.: Compact adaptively secure ABE for NC^1 from k-Lin. Journal of Cryptology pp. 1–49 (2019)
17. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: EUROCRYPT 2010. pp. 62–91. Springer
18. Lewko, A.B., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: TCC 2010. pp. 455–479. Springer
19. Lin, H., Luo, J.: Compact adaptively secure abe from k-Lin: Beyond NC^1 and towards NL. In: EUROCRYPT 2020. pp. 247–277. Springer
20. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: FOCS 2016. pp. 11–20. IEEE
21. Nisan, N.: Lower bounds for non-commutative computation (extended abstract). In: STOC 1991. pp. 410–418. ACM
22. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: EUROCRYPT 2012. pp. 591–608. Springer
23. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: ASIACRYPT 2012. pp. 349–366. Springer
24. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: CRYPTO 2010. pp. 191–208. Springer (2010)
25. Okamoto, T., Takashima, K.: 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)
26. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptology ePrint Archive, Report 2010/556
27. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: PKC 2011. pp. 53–70. Springer
28. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: CRYPTO 2009. pp. 619–636. Springer
29. Wee, H.: Attribute-hiding predicate encryption in bilinear groups, revisited. In: TCC 2017. pp. 206–233. Springer
30. Wee, H.: Functional encryption for quadratic functions from k-Lin, revisited. In: TCC 2020. pp. 210–228. Springer